# BREAKING THE TOKEN CEILING: DISTILLING SMALLER, STRONGER BYTE MODELS

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

## 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 $\approx$ 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. `End-Of-Token-1B` is 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 $\approx$ 256 bytes instead of $\approx$ 100K tokens, it circumvents the need for top-$k$ truncation during logit dumping, while also reducing logit storage costs up 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 (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 models (Kaplan et al., 2020; Hoffmann et al., 2022; Henighan 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 (Pope et al., 2023; Dettmers & Zettlemoyer, 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 (Hinton et al., 2015; Beyer et al., 2022), and 2) overtrained to amortize inference costs over a model's lifetime (Sardana et al., 2023; Gadre et al., 2024). 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 $\approx$ 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: byte models learn relationships between bytes from fewer samples than token models require for tokens (Hestness et al., 2017), which makes them data efficient learners (Xue et al., 2022; Zheng



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

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 training 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 models with comparable sizes and what are the data and storage efficiency implications. This work addresses all five. We summarize our key contributions below:

**Token logits to Byte logits conversion in a single forward pass.** We introduce two single pass variants for converting token logits to byte logits by marginalizing over byte positions: 1) approximate: `Marginalize-It` renormalizes over the prefix matching tokens, while 2) exact: `End-Of-Token` adds an **<eot>** token to the vocabulary so no probability is lost.

**BPB vs FLOPs Power Laws Across Tokenization and Training Objectives.** We present a large scale study of overtraining decoder-only dense transformers (Vaswani et al., 2017) along two axes: the training objective and the tokenization scheme: sweeping models with 1.28 billion layer parameters up to 1 trillion bytes to show that distilled `End-Of-Token-1B` models show better asymptotic behavior compared to distilled `Token-1B` and `Bytes-1B` .

**Average Downstream Task Performance vs Training FLOPs Scaling Trends.** We find that `Token-1B` transformers 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 $\approx 256$ bytes.

**Comparing Distillation Scaling Trajectories Across Compute Budgets.** We fit downstream error vs. validation BPB scaling laws and by extrapolating them, we predict that distilled `End-Of-Token-1B` transformers raise the asymptotic ceiling of the averaged downstream task performance and 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` **:** We show that 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 $\approx 256$ 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 asymptotically surpasses 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.

## 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 H.1 and discuss special cases in Appendix H.2.

**Marginalize-It Logits.** To compute the first byte distribution of given BPE token logits, 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. While this approach always yields the exact distribution for the first byte, the distributions over subsequent bytes are approximate. Figure 1 (column 1) shows the `Marginalize-It` logit conversion process step by step. To predict the distribution of **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 an additional teacher inference over the sequence: **[<bos>, T, iram, 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 the remaining probability is redistributed among tokens **isu** and **isk**. During training, the byte sequence used

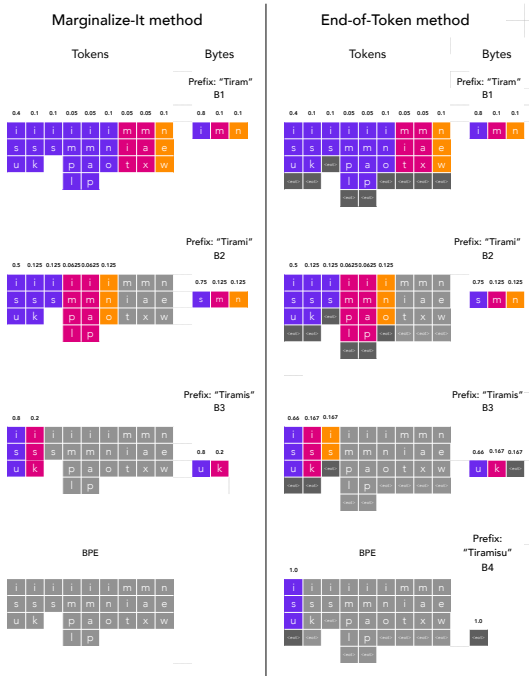


Figure 1: A step by step illustration of the two logit conversion methods. We show the formation process for both **Marginalize-It** and **End-Of-Token** . Consider the tokenized prompt **Tiramisu = [<bos>, T, iram, isu, <eos>]**, ;and in particular the next token distribution following the token **iram**, 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 probability shown at the top of the column where 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**.

is: **[<bos>,T,i,r,a,m,i,s,u,<eos>]**. We provide additional details in Appendix H.2. **Marginalize-It** serves as the main baseline throughout the study.

**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 **<eot>** token to mark the end of each BPE token as shown in Figure 1 (column 2). Second, we expand the byte vocabulary by one to store the **<eot>** probabilities and then apply marginalization over byte positions to form the byte logits. During training, we append **<eot>** after every BPE token in the pretraining dataset: **[<bos>,<eot>,T,<eot>,i,r,a,m,<eot>,i,s,u,<eot>,<eos>,<eot>]**. This resolves the missing probability problem of the **Marginalize-It** method. To compute **B3** distribution, unlike **Marginalize-It** , **End-Of-Token** considers all tokens with prefix **is**. In this way, the probability mass of the possible continuations of **is** is absorbed into the**<eot>** token. This approach is exact and does not require additional inferences over teacher. Inserting **<eot>** tokens also has an unintended but remarkable consequence: the $\approx$ **30.94%** additional compute it spends per unit of data (one **<eot>** every approx 4.5 bytes) turns out to further improve asymptotic performance relative to the **Marginalize-It** method.

## 3 Scaling Study Design

In this section we describe the model architecture, pretraining dataset and training details. We provide specifics of these settings in Appendix A.

**Training Dataset.** We use the Llama-2 (Meta AI, 2024) training mixture for our 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 analysis.

**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)

**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 in Table 1. The model size and FLOPs per unit values are obtained from Table 8.

**Benchmark Evaluations.** We use ARC-Easy (Clark et al., 2018), ARC-Challenge (Clark et al., 2018), HellaSwag (Zellers et al., 2019), and PIQA (Bisk et al., 2020) to evaluate Multiple Choice

| Experiment | Tokenization | Vocabulary size | Objective | Architecture | Model size | FLOPs/unit |
|---|---|---|---|---|---|---|
| Token Supervised | Tokens (Llama 3-8B) | 128256 | Cross-Entropy | Token-1B | 1.81B | $8.18 \times 10^9$ |
| Token Distilled | Tokens (Llama 3-8B) | 128256 | Distillation (top-$k$) | Token-1B | 1.81B | $8.18 \times 10^9$ |
| Bytes Supervised | Bytes (256 + 4 special) | 260 | Cross-Entropy | Bytes-1B | 1.28B | $8.81 \times 10^9$ |
| Marginalize-It Distilled | Bytes (256 + 4 special) | 260 | Distillation (approximate) | Bytes-1B | 1.28B | $8.81 \times 10^9$ |
| Bytes w/ <eot> Supervised | Bytes w/ <eot> (256 + 5 special) | 261 | Cross-Entropy | End-Of-Token-1B | 1.28B | $9.44 \times 10^9$ |
| End-Of-Token Distilled | Bytes w/ <eot> (256 + 5 special) | 261 | Distillation (Exact) | End-Of-Token-1B | 1.28B | $9.44 \times 10^9$ |

Table 1: Six experiments in our study across the training objective and tokenization scheme.

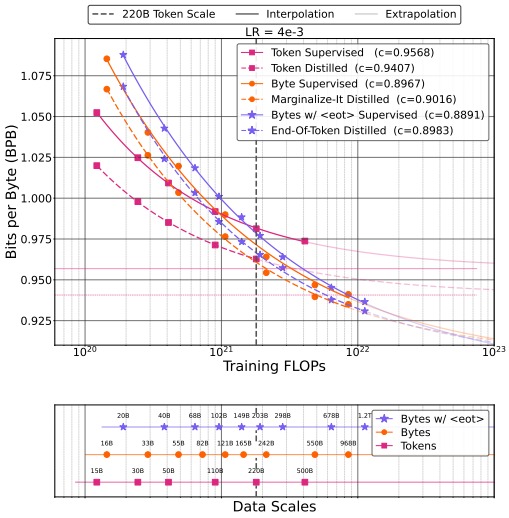


Figure 2: The figure shows our scaling curves for different data scales, for a learning rate of 4e-3 and a fixed layer parameter count of approximately 1.28 billion parameter transformer models (see Table 8 for details). Note that data scales plot serves as an alternative $x$ axis to training FLOPs. 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.

Question Answering capabilities. We also evaluate the language modeling capabilities of the byte and token models on MBPP (Austin et al., 2021) and Natural Questions (Kwiatkowski et al., 2019). 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. We plot the BPB of the model on a held out validation dataset. We fit the power law for validation BPB with respect to training FLOPs using equation $y = b \cdot x^a + c$ where $x$ is the training FLOPs and $y$ is the BPB at a given compute budget (Table 2). The $y$, asymptotically approaches c.

Table 2: We observe that the amount of FLOPs spent on a fixed amount of data (one BPE token in this case) affects the asymptotic BPB. Models appear follow a descending BPB trend as the total FLOPs per token increase. The FLOPs per unit values are obtained from Table 8. We note that **End-Of-Token-1B** uses $\approx$ **30.94%** extra FLOPs per token compared to **Bytes-1B** .

| | | FLOPs | Units | Total FLOPs | Power Law fit | | Asymptotic BPB |
|---|---|---|---|---|---|---|---|
| Experiment | Architecture | / Unit | Per Token | Per Token | Equation | $R^2$ | LR=4e-3 |
| Token Supervised | Token-1B | $8.18 \times 10^9$ | 1.0 | $8.18 \times 10^9$ | $y = 1.075 \times 10^9 \cdot x^{-0.500369} + 0.9568$ | 0.9999 | 0.9568 |
| Bytes Supervised | Bytes-1B | $8.81 \times 10^9$ | 4.5 | $39.65 \times 10^9$ | $y = 4.467 \times 10^6 \cdot x^{-0.365844} + 0.8967$ | 0.9980 | 0.8967 |
| Bytes w/ <eot> Supervised | End-Of-Token-1B | $9.44 \times 10^9$ | 5.5 | $51.92 \times 10^9$ | $y = 3.080 \times 10^6 \cdot x^{-0.354594} + 0.8891$ | 0.9982 | 0.8891 |
| Token Distilled | Token-1B | $8.18 \times 10^9$ | 1.0 | $8.18 \times 10^9$ | $y = 2.803 \times 10^8 \cdot x^{-0.475375} + 0.9407$ | 1.0000 | 0.9407 |
| Marginalize-It Distilled | Bytes-1B | $8.81 \times 10^9$ | 4.5 | $39.65 \times 10^9$ | $y = 2.706 \times 10^7 \cdot x^{-0.407427} + 0.9016$ | 0.9974 | 0.9016 |
| End-Of-Token Distilled | End-Of-Token-1B | $9.44 \times 10^9$ | 5.5 | $51.92 \times 10^9$ | $y = 2.734 \times 10^7 \cdot x^{-0.404740} + 0.8983$ | 0.9990 | 0.8983 |

We notice several trends from these plots. First, **Marginalize-It Distilled** method, despite being an approximation works very well in practice. **End-Of-Token Distilled** method is exact and yields consistent improvements over the supervised training. 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** models have the worst BPB to begin with yet yield the best asymptote in the infinite compute regime. **Bytes-1B** scaling curve, appears to be sandwiched between the other two scenarios at the beginning and the end of the scaling curve. From Table 2, we observe that asymptotically for both supervised and distilled models: 1) **End-Of-Token-1B** BPB

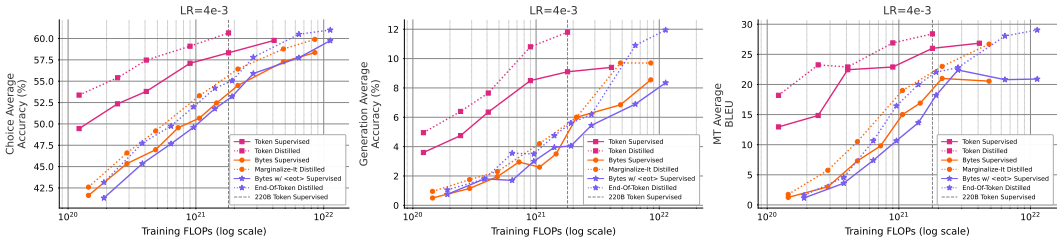


Figure 3: We plot the averaged benchmark performance vs Training FLOPs for our six experiments. We observe that the token models show better performance at low compute budgets and eventually plateau. **Bytes-1B** and **End-Of-Token-1B** start worse yet improve at a steeper rate and eventually match the **Token-1B** performance.

$<$ **Bytes-1B** BPB $<$ **Token-1B** BPB. Lastly, we also observe that the supervised and distilled models with the same tokenization scheme exhibits small differences in the asymptotic value. We also observe interesting trends across all scaling curves. At around $1.8 \times 10^{21}$ FLOPs, the **Token Supervised** model 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 with the benchmark evaluations. We evaluate our trained models on a total of eight benchmarks spanning three categories: Multiple choice question answering, Language Generation, and Machine Translation tasks (Figure 3).

**Results on Multiple Choice Question Answering Benchmarks.** We observe that **Token-1B** models outperform all of the byte curves at lower compute budgets. The byte curves start worse but appear to be improving faster than the token models, eventually catching up with them. If we compare within the four byte experiments **End-Of-Token-1B** start worse, and eventually surpass the **Bytes-1B** models. Lastly we observe that at 220B token supervised budget only the distilled **Token-1B** outperforms its supervised counterpart but **Bytes-1B** and **End-Of-Token-1B** models are much worse, which is counterintuitive to the key observation in 4.

**Results on Language Generation Benchmarks.** We observe that 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. In all cases, distilled models appear to be better for the same compute budget and the result about observation 4 at $1.8 \times 10^{21}$ FLOPs budget is observed in this case as well. Only the **Token Distilled** model has better performance while all other four byte scaling curves are much worse.

**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 show that: **Bytes-1B** and **End-Of-Token-1B** 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. Next, the results on these benchmarks contradict the observation 4 here as well. Lastly, **End-Of-Token Distilled** models by preserving the teacher distribution exactly show significant improvements over **Bytes w/ <eot> Supervised** models.

**Concluding Remarks.** For all of the three categories, 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.

## 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

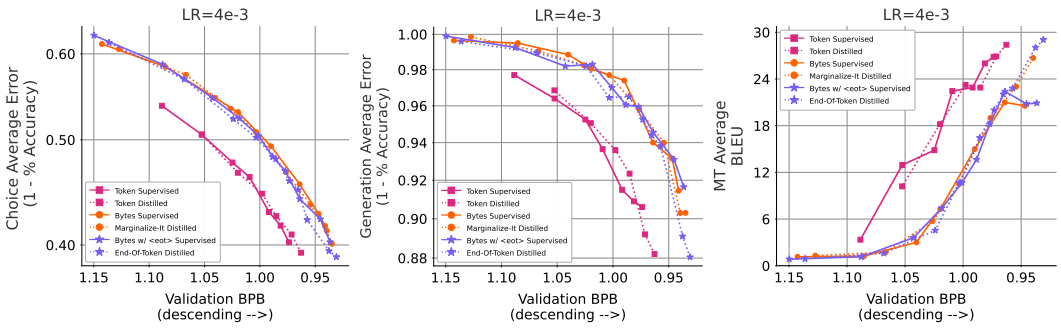


Figure 4: We visualize downstream task performance vs the validation BPB curves for our six experiments for LR=4e-3. Both choice average error as well as language generation appear to follow an exponential decay. The nature of this exponential decay also follow different trends for Multiple Choice and Language Generation benchmarks. **Token-1B** shows significantly different trend compared to **Bytes-1B** and **End-Of-Token-1B** .

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

| Experiment | Equation $z(y)$ |
|---|---|
| Token Supervised | $0.7501 - 934.3212\, e^{-8.7782\, y}$ |
| Token Distilled | $0.7383 - 1292.0124\, e^{-9.2132\, y}$ |
| Bytes Supervised | $0.7686 - 1231.1241\, e^{-9.3532\, y}$ |
| Marginalize-It Distilled | $0.7721 - 1137.3004\, e^{-9.2265\, y}$ |
| Bytes w/ <eot> Supervised | $0.7972 - 185.6303\, e^{-7.1822\, y}$ |
| End-Of-Token Distilled | $0.7776 - 827.6076\, e^{-8.8125\, y}$ |

Table 4: Asymptotic BPB (c) and the average accuracy $a' = 1 - \%z'$ where $z'$ is obtained by: $z' = \epsilon - k \cdot e^{-\gamma c}$ , and $c$ is obtained from the power law equations in Table 3.

| Experiment | 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 |

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, the $(1 - \%$ Avg. Accuracy$)$ vs the validation BPB (Figure 4), for Multiple Choice Question Answering and Language Generation. Lastly, for Machine Translation we plot the BLEU score vs the validation BPB.

**Results on Multiple Choice Question Answering Benchmarks.** . In the case of Multiple Choice Questions, comparing performance at a fixed BPB reveals wildly different error rates. Second, the iso-error lines intersect with the **End-Of-Token** and **Token** curves at very different compute budgets. Third, 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. Lastly, the training objective (supervised as well as distillation) also appear to alter the scaling behavior.

**Results on Language Generation Benchmarks.** We observe that for language generation as well, the same BPB can mean different averaged downstream performance for different tokenization schemes ( **Token** , **Bytes** , **Bytes w/ <eot>** ) as well as the training objectives.

**Results on Machine Translation Benchmarks.** Lastly we plot BLEU score vs Validation BPB curves for Machine Translation tasks. We observe that: The **Token-1B** models, **Bytes-1B** models and **End-Of-Token-1B** models show different scaling behaviors. Overall, **End-Of-Token** distillation appears to exhibit monotonic trend compared to the noisy **Bytes w/ <eot> Supervised** curves.

**Fitting the Downstream Task Error vs Validation BPB Scaling Law.** We fit the equation $z(y) = \epsilon - k \cdot e^{-\gamma y}$ for our six experiments on Average top-1 error calculated over a total of four Multiple Choice and two Language Generation benchmarks (from 3) where $y$ denotes the BPB value, $z(y)$ is the Average Top-1 downstream task error, and $\epsilon$, k and $\gamma$ are fit using data. The scaling law equations are detailed in Table 3. The interpolations and extrapolations are visualized in Figure 5. The curves are bounded by the asymptotic BPB c, obtained from the power law equations in Table 2.

**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,

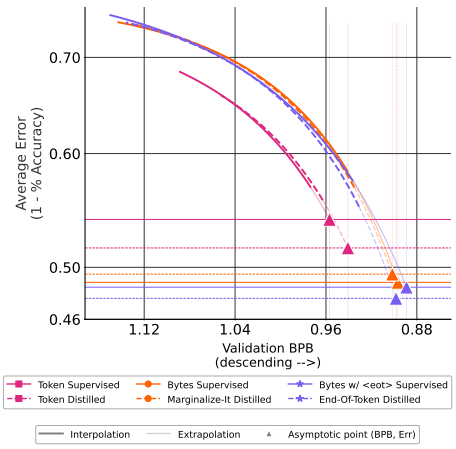


Figure 5: We fit the downstream error scaling laws with respect to Validation BPB. **Token-1B** curves show significantly different behavior compared to the **Bytes-1B** and **End-Of-Token-1B** curves. The training objective (distillation vs supervised (Cross Entropy) also affects the scaling laws.

Table 5: Training compute at which distilled **Bytes-1B** and **End-Of-Token-1B** match the asymptotic performance of **Token-1B** .

| Expt. | **Token** dist. BPB | Accuracy | Expt. BPB | Training FLOPs |
|---|---|---|---|---|
| **Marginalize-It** | 0.9407 | 48.4% | 0.9102 | $2.04 \times 10^{23}$ |
| **End-Of-Token** | 0.9407 | 48.4% | 0.9145 | $6.33 \times 10^{22}$ |

Table 6: Data and Storage Cost Analysis

| Metric | Formula | **Marginalize-It** | **End-Of-Token** |
|---|---|---|---|
| Training FLOPs | From Table 4) | $2.04 \times 10^{23}$ | $6.33 \times 10^{22}$ |
| $\text{FLOPs}_{\text{byte}}$ | FLOPs per byte unit (Table 8) | $8.81 \times 10^{9}$ | $9.44 \times 10^{9}$ |
| $\text{FLOPs}_{\text{token}}$ | FLOPs per token (Table 8) | $8.18 \times 10^{9}$ | $8.18 \times 10^{9}$ |
| Byte Units | Training $\text{FLOPs}/\text{FLOPs}_{\text{byte}}$ | $2.32 \times 10^{13}$ | $6.70 \times 10^{12}$ |
| Multiplier | (for actual data) | 1 | 4.5/5.5 |
| Bytes | Byte units × Multiplier | $2.32 \times 10^{13}$ | $5.48 \times 10^{12}$ |
| Tokens | Training $\text{FLOPs}/\text{FLOPs}_{\text{token}}$ | $2.50 \times 10^{13}$ | $7.73 \times 10^{12}$ |
| Token to Byte | Tokens × 4.5 | $1.12 \times 10^{14}$ | $3.48 \times 10^{13}$ |
| Data Savings | Token to Byte/Bytes | 4.85× | 6.35× |
| Storage Savings | Tokens/(Byte Units/4.5) | 4.85× | 5.19× |

Table 7: Predicted asymptotic downstream accuracy for our obtained from table 14, calculated by extrapolating the scaling trend.

| Model | Experiment | Asymptotic Average 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 (≈ 9T BPE tokens) |
| **Gemma-3-1B-pt** | Kamath et al. (2025) | 44.3 (≈ 2T BPE tokens) |
| **Gemma 2b** | Team et al. (2024) | 50.3 (≈ 3T BPE tokens) |

for each curve obtained from Table 2. We then compute the asymptotic performance at that BPB as shown in Figure 5. Table 3 shows the asymptotic BPB as well as the task performance of the six experiments. We notice that distilled **Token-1B** asymptotically outperforms over its supervised counterpart.**End-Of-Token Distilled** method outperforms both **Token Distilled** and **Bytes w/ <eot> Supervised** asymptotically. Distilled **Bytes-1B** performs 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. 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.

**When Lower BPB Does Not (Always) Imply Better Models.** Our observations show that the relationship between the BPB across the training objectives and the tokenization scheme (**Tokens** , **Bytes** , **Bytes w/ <eot>** ) experiments is nuanced in the context of benchmark evaluations. These disparities potentially arise from differences in the 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, they exhibit worse downstream task performance across benchmarks, falsifying observation 4, emphasizing the importance of this final calibration step while determining the quality of a language model before real world deployment. See section 9 and appendix I.1 for additional discussion.

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

To understand if the **End-Of-Token** logits are advantageous, it is essential to find out when they will if at all surpass the distilled **Token** models. To do so, we use the downstream error prediction scaling laws from 6. Given the validation BPB prediction and downstream error prediction scaling laws for each model variant we want to find out at what compute budget **Marginalize-It Distilled** and **End-Of-Token Distilled** models will give the same error as that of the

asymptotic downstream error of a **Token Distilled** model. We describe the process below: See appendix B for additional visualizations with the help of **Feather Plot** .

**Training compute required to achieve Token distillation performance.** To calculate the training compute required to achieve the asymptotic **Token** distillation performance, we first take **Token Distilled** asymptotic BPB (the $c$ term of its fitted power law (column Token Distillation BPB obtained from Table 2) and use the **Token Distilled** downstream error scaling law to obtain its asymptotic downstream accuracy (Table 4). Then for each method **Marginalize-It** and **End-Of-Token** we then invert its downstream task performance scaling law at that accuracy to get the BPB it must achieve. Lastly we invert its BPB vs training FLOPs power law at that BPB to get the required training compute. (Table 5).

**Data and storage cost savings analysis.** We summarize our results in Table 6. Both distilled **End-Of-Token-1B** and **Bytes-1B** models 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 the **End-Of-Token** uses the **<eot>** tokens every 4.5 bytes, the actual text data is less than the byte units used for training. To calculate storage savings we use our storage cost calculations from Table 9 where we equate the storage cost of one **Token** with 4.5 **Bytes** to obtain k (for top-$k$). Taking this into account, **End-Of-Token** 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 Llama3-8B tokens.

**Comparisons with Open-Weight models.** We now compare our asymptotic predictions for each of the six experiments described in section 3 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 7. We evaluate these models using (Gao et al., 2024). We Additional details for llama and gemma models are provided in table 17. We show that distilled **End-Of-Token-1B** asymptotically outperforms all other models, in particular the **Token-1B** by $4\%$.

## 8 Related Work

**Scaling Laws with Overtraining.** (Hestness et al., 2017) show that character models learn relationships between characters with fewer samples compared to word level language models learn from words. (Sardana et al., 2023) demonstrate that when the inference budget is large, 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 (10) and table (14). We show that a simple addition of an <eot> token every $\approx 4.5$ bytes can improve the asymptotic downstream task performance.

**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 BLEU score. Gadre et al. (2024) establish the relationship between the Average top-1 downstream error across 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) propose a two stage approach where they predict the intermediate task loss which is then mapped to the task performance. Lourie et al. (2025) conduct a metaanalysis 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. Our work studies how the training objective and the tokenization scheme can affect the downstream performance with training FLOPs.

**Token to Bytes transfer.** Minixhofer et al. (2025) present an approach for adapting pretrained token 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 finetuning stage. Hayase et al. (2025) and Phan et al.

(2024) study the exact BPE to bytes transfer to tackle the prompt boundary problem where a prompt, when ends in the middle of a BPE token, results into distorted next token distributions at inference time. These methods, however, require multiple inference passes over the teacher model to obtain exact conversions. We bypass this issue by 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.** Initial attempts at byte modeling include Al-Rfou et al. (2018), Choe et al. (2019), El Boukkouri et al. (2020), Clark et al. (2022). Xue et al. (2022) train an encoder-decoder model operating on bytes and show that byte level transformers are data efficient learners, though take longer to train and are expensive during inference. Yu et al. (2023), Nawrot et al. (2023),Pagnoni et al. (2025), Hwang et al. (2026) propose hierarchical architectures for byte level modeling. Zheng et al. (2025) trained byte level transformers with various tricks like multibyte 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` ). We show that `Token-1B` models ourperform `Bytes-1B` and `End-Of-Token-1B` in the low FLOP regime but eventually they plateau. `End-Of-Token-1B` despite having fewer total parameters improve at a steeper rate and asymptotically surpass the `Token-1B` models.

## 9 Discussion

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. While our scaling laws show that, distilling Llama 3-8B (Grattafiori et al., 2024) model into `End-Of-Token-1B` is asymptotically better than distilling into `Token-1B` , `End-Of-Token-1B` models remain significantly expensive at inference time. Comparing the asymptotic performance of a `Token` model against that of a `End-Of-Token` transformers with matched inference cost could reveal whether distilled `End-Of-Token-1B` transformers are also efficient at inference time. This analysis is beyond the scope of this study. 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; Veličković 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. 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 Shazeer et al. (2017) models affects these power laws. Next we report our findings based on eight benchmarks 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.

## 10 AI Disclosure Statement

In this work, we used generative AI tools for creating plots, formatting tables, minor writing edits and brainstorming an example in Appendix Table 23. We have not used generative AI tools to generate synthetic data sets, help develop theoretical models or conceptual frameworks, formulate mathematical claims, provide critical ingredients for proving mathematical claims, assist in the writing of proofs, propose or refine hypotheses, design or provide feedback on research methodology or experiments, implement methods, assist with translation, clean and reformat dataset, support qualitative and thematic data analysis, interpret results, summarize or analyze existing literature, discover research topics or identify gaps, brainstorming, sourcing/searching for information, identify relevant literature, format references, suggest a structure for a research paper. We have reviewed all AI-assisted work. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

## REFERENCES


Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://api.semanticscholar.org/CorpusID:52004855.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Yasaman Bahri, Ethan Dyer, Jared Kaplan, Jaehoon Lee, and Utkarsh Sharma. Explaining neural scaling laws. *Proceedings of the National Academy of Sciences*, 121(27):e2311878121, 2024.

Yamini Bansal, Behrooz Ghorbani, Ankush Garg, Biao Zhang, Colin Cherry, Behnam Neyshabur, and Orhan Firat. Data scaling laws in nmt: The effect of noise and architecture. In *International Conference on Machine Learning*, pp. 1466–1482. PMLR, 2022.

Zishuo Bao, Jiaqi Leng, Junxiong Wang, Bowen Peng, and Yucheng Lu. Distilling token-trained models into byte-level models. *arXiv preprint arXiv:2602.01007*, 2026.

Lucas Beyer, Xiaohua Zhai, Amélie Royer, Larisa Markeeva, Rohan Anil, and Alexander Kolesnikov. 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, 2022.

Akshita Bhagia, Jiacheng Liu, Alexander Wettig, David Heineman, Oyvind Tafjord, Ananya Harsh Jha, Luca Soldaini, Noah A Smith, Dirk Groeneveld, Pang Wei Koh, et al. Establishing task scaling laws via compute-efficient model ladders. *arXiv preprint arXiv:2412.04403*, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Dan Busbridge, Amitis Shidani, Floris Weers, Jason Ramapuram, Etai Littwin, and Russ Webb. Distillation scaling laws. *arXiv preprint arXiv:2502.08606*, 2025.

Dokook Choe, Rami Al-Rfou, Mandy Guo, Heeyoung Lee, and Noah Constant. Bridging the gap for tokenizer-free language models. *arXiv e-prints*, pp. arXiv–1908, 2019.

Jonathan H Clark, Dan Garrette, Iulia Turc, and John Wieting. Canine: Pre-training an efficient tokenization-free encoder for language representation. *Transactions of the Association for Computational Linguistics*, 10:73–91, 2022.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255, 2009. doi:10.1109/CVPR.2009.5206848.

Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. In *International Conference on Machine Learning*, pp. 7750–7774. PMLR, 2023.

Hicham El Boukkouri, Olivier Ferret, Thomas Lavergne, Hiroshi Noji, Pierre Zweigenbaum, and Jun'ichi Tsujii. 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, 2020.

Lizhe Fang, Yifei Wang, Zhaoyang Liu, Chenheng Zhang, Stefanie Jegelka, Jinyang Gao, Bolin Ding, and Yisen Wang. What is wrong with perplexity for long-context language modeling? In *International Conference on Learning Representations*, volume 2025, pp. 94541–94563, 2025.

Samir Yitzhak Gadre, Georgios Smyrnis, Vaishaal Shankar, Suchin Gururangan, Mitchell Wortsman, Rulin Shao, Jean Mercat, Alex Fang, Jeffrey Li, Sedrick Keh, et al. Language models scale reliably with over-training and on downstream tasks. *arXiv preprint arXiv:2403.08540*, 2024.

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

Naman Goyal, Cynthia Gao, Vishrav Chaudhary, Peng-Jen Chen, Guillaume Wenzek, Da Ju, Sanjana Krishnan, Marc'Aurelio Ranzato, Francisco Guzmán, and Angela Fan. The flores-101 evaluation benchmark for low-resource and multilingual machine translation. *Transactions of the Association for Computational Linguistics*, 10:522–538, 2022.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Jonathan Hayase, Alisa Liu, Noah A Smith, and Sewoong Oh. Sampling from your language model one byte at a time. *arXiv preprint arXiv:2506.14123*, 2025.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Tom Henighan, Jared Kaplan, Mor Katz, Mark Chen, Christopher Hesse, Jacob Jackson, Heewoo Jun, Tom B Brown, Prafulla Dhariwal, Scott Gray, et al. Scaling laws for autoregressive generative modeling. *arXiv preprint arXiv:2010.14701*, 2020.

Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, DDL Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 10, 2022.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. In *International Conference on Learning Representations*, volume 2026, pp. 149273–149313, 2026.

Berivan Isik, Natalia Ponomareva, Hussein Hazimeh, Dimitris Paparas, Sergei Vassilvitskii, and Sanmi Koyejo. Scaling laws for downstream task performance in machine translation. In *International Conference on Learning Representations*, volume 2025, pp. 88769–88790, 2025.

Gemma Team Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, Alexandre Ram'e, Morgane Rivière, Louis Rouillard, Thomas Mesnard, Geoffrey Cideron, Jean-Bastien Grill, Sabela Ramos, Edouard Yvinec, Michelle Casbon, Etienne Pot, Ivo Penchev, Gael Liu, Francesco Visin, Kathleen Kenealy, Lucas Beyer, Xiaohai Zhai, Anton Tsitsulin, Róbert Istvan Busa-Fekete, Alex Feng, Noveen Sachdeva, Benjamin Coleman, Yi Gao, Basil Mustafa, Iain Barr, Emilio Parisotto, David Tian, Matan Eyal, Colin Cherry, Jan-Thorsten Peter, Danila Sinopalnikov, Surya Bhupatiraju, Rishabh Agarwal, Mehran Kazemi, Dan Malkin, Ravin Kumar, David Vilar, Idan Brusilovsky, Jiaming Luo, Andreas Steiner, Abe Friesen, Abhanshu Sharma, Abheesht Sharma, Adi Mayrav Gilady, Adrian Goedeckemeyer, Alaa Saade, Alexander Kolesnikov, Alexei Bendebury, Alvin Abdagic, Amit Vadi, Andr'as Gyorgy, André Susano Pinto, Anil Das, Ankur Bapna, Antoine Miech, Antoine Yang, Antonia Paterson, Ashish Shenoy, Ayan Chakrabarti, Bilal Piot, Boxi Wu, Bobak Shahriari, Bryce Petrini, Charlie Chen, Charline Le Lan, Christopher A. Choquette-Choo, Cj Carey, Cormac Brick, Daniel Deutsch, Danielle Eisenbud, Dee Cattle, Derek Cheng, Dimitris Paparas, Divyashree Shivakumar Sreepathihalli, Doug Reid, Dustin Tran, Dustin Zelle, Eric Noland, Erwin Huizenga, Eugene Kharitonov, Frederick Liu, Gagik Amirkhanyan, Glenn Cameron, Hadi

Hashemi, Hanna Klimczak-Pluci'nska, Harman Singh, Harsh Mehta, Harshal Tushar Lehri, Hussein Hazimeh, Ian Ballantyne, Idan Szpektor, Ivan Nardini, Jean Pouget-Abadie, Jetha Chan, Joe Stanton, J. Michael Wieting, Jonathan Lai, Jordi Orbay, Joe Fernandez, Joshua Newlan, Junsong Ji, Jyotinder Singh, Kat Black, Kathy Yu, Kevin Hui, Kiran Vodrahalli, Klaus Greff, Linhai Qiu, Marcella Valentine, Marina Coelho, Marvin Ritter, Matt Hoffman, Matthew Watson, Mayank Chaturvedi, Michael Moynihan, Min Ma, Nabila Babar, Natasha Noy, Nathan Byrd, Nick Roy, Nikola Momchev, Nilay Chauhan, Oskar Bunyan, Pankil Botarda, Paul Caron, Paul Kishan Rubenstein, Phil Culliton, Philipp Schmid, Pier Giuseppe Sessa, Ping mei Xu, Piotr Stańczyk, Pouya Dehghani Tafti, Rakesh Shivanna, Renjie Wu, Renke Pan, Reza Ardeshir Rokni, Rob Willoughby, Rohith Vallu, Ryan Mullins, Sammy Jerome, Sara Smoot, Sertan Girgin, Shariq Iqbal, Shashir Reddy, Shruti Sheth, Siim Põder, Sijal Bhatnagar, Sindhu Raghuram Panyam, Sivan Eiger, Susan Zhang, Tianqi Liu, Trevor Yacovone, Tyler Liechty, Uday Kalra, Utku Evci, Vedant Misra, Vincent Roseberry, Vladimir Feinberg, Vlad Kolesnikov, Woohyun Han, Woosuk Kwon, Xi Chen, Yinlam Chow, Yuvein Zhu, Zichuan Wei, Zoltan Egyed, Victor Cotruta, Minh Giang, Phoebe Kirk, Anand Rao, Jessica Lo, Erica Moreira, Luiz Gustavo Martins, Omar Sanseviero, Lucas Gonzalez, Zach Gleicher, Tris Warkentin, Vahab S. Mirrokni, Evan Senter, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, Yossi Matias, D. Sculley, Slav Petrov, Noah Fiedel, Noam Shazeer, Oriol Vinyals, Jeffrey Dean, Demis Hassabis, Koray Kavukcuoglu, Clément Farabet, Elena Buchatskaya, Jean-Baptiste Alayrac, Rohan Anil, Dmitry Lepikhin, Sebastian Borgeaud, Olivier Bachem, Armand Joulin, Alek Andreev, Cassidy Hardin, Robert Dadashi, and L'eonard Hussenot. Gemma 3 technical report. *ArXiv*, abs/2503.19786, 2025. URL https://api.semanticscholar.org/CorpusID:277313563.

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

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. URL https://aclanthology.org/Q19-1026/.

Celine Lee, Jing Nathan Yan, Chen Liang, Jiaxin Shi, Yin Zhang, Jeremiah Liu, Pengcheng Yin, Fernando Pereira, Ed Chi, Derek Cheng, et al. The efficiency gap in byte modeling. *arXiv preprint arXiv:2605.12928*, 2026.

Tomasz Limisiewicz, Artidoro Pagnoni, Srini Iyer, Mike Lewis, Sachin Mehta, Alisa Liu, Margaret Li, Gargi Ghosh, and Luke Zettlemoyer. Compute optimal tokenization. *arXiv preprint arXiv:2605.01188*, 2026.

Hong Liu, Sang Michael Xie, Zhiyuan Li, and Tengyu Ma. Same pre-training loss, better downstream: Implicit bias matters for language models. In *International Conference on Machine Learning*, pp. 22188–22214. PMLR, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Nicholas Lourie, Michael Y Hu, and Kyunghyun Cho. Scaling laws are unreliable for downstream tasks: A reality check. *arXiv preprint arXiv:2507.00885*, 2025.

Ian Magnusson, Nguyen Tai, Ben Bogin, David Heineman, Jena D Hwang, Luca Soldaini, Akshita Bhagia, Jiacheng Liu, Dirk Groeneveld, Oyvind Tafjord, et al. Datadecide: How to predict best pretraining data with small experiments. *arXiv preprint arXiv:2504.11393*, 2025.

Meta AI. Llama 3.2: Revolutionizing edge ai and vision, 2024. URL https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/. Accessed: 2024-09-25.

Benjamin Minixhofer, Tyler Murray, Tomasz Limisiewicz, Anna Korhonen, Luke Zettlemoyer, Noah A Smith, Edoardo M Ponti, Luca Soldaini, and Valentin Hofmann. Bolmo: Byteifying the next generation of language models. *arXiv preprint arXiv:2512.15586*, 2025.

Benjamin Minixhofer, Ivan Vulić, and Edoardo Maria Ponti. Universal cross-tokenizer distillation via approximate likelihood matching. *Advances in Neural Information Processing Systems*, 38: 79297–79326, 2026.

Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. Efficient transformers with dynamic token pooling. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 6403–6417, Toronto, Canada, July 2023. Association for Computational Linguistics. doi:10.18653/v1/2023.acl-long.353. URL https://aclanthology.org/2023.acl-long.353/.

David Owen. How predictable is language model benchmark performance? *arXiv preprint arXiv:2401.04757*, 2024.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, et al. 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, 2025.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pp. 311–318, USA, 2002. Association for Computational Linguistics. doi:10.3115/1073083.1073135. URL https://doi.org/10.3115/1073083.1073135.

Hao Peng, Xin Lv, Yushi Bai, Zijun Yao, Jiajie Zhang, Lei Hou, and Juanzi Li. 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, 2025.

Buu Phan, Brandon Amos, Itai Gat, Marton Havasi, Matthew Muckley, and Karen Ullrich. Exact byte-level probabilities from tokenized language models for fim-tasks and model ensembles. *arXiv preprint arXiv:2410.09303*, 2024.

Truong Buu Phan, Ashish Khisti, and Karen Ullrich. Cross-tokenizer likelihood scoring algorithms for language model distillation. In *International Conference on Learning Representations*, volume 2026, pp. 38410–38428, 2026.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of machine learning and systems*, 5:606–624, 2023.

Nikhil Sardana, Jacob Portes, Sasha Doubov, and Jonathan Frankle. Beyond chinchilla-optimal: Accounting for inference in language model scaling laws. *arXiv preprint arXiv:2401.00448*, 2023.

Rylan Schaeffer, Brando Miranda, and Sanmi Koyejo. Are emergent abilities of large language models a mirage? *Advances in neural information processing systems*, 36:55565–55581, 2023.

Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Petar Veličković, Federico Barbero, Christos Perivolaropoulos, Simon Osindero, and Razvan Pascanu. Perplexity cannot always tell right from wrong. *arXiv preprint arXiv:2601.22950*, 2026.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, Mihir Kale, Adam Roberts, and Colin Raffel. Byt5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions of the Association for Computational Linguistics*, 10:291–306, 2022.

Lili Yu, Dániel Simig, Colin Flaherty, Armen Aghajanyan, Luke Zettlemoyer, and Mike Lewis. Megabyte: Predicting million-byte sequences with multiscale transformers. *Advances in Neural Information Processing Systems*, 36:78808–78823, 2023.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Thomas T Zhang, Alok Shah, Yifei Zhang, Vincent Zhang, Nikolai Matni, and Max Simchowitz. Double preconditioning (dopr): Optimization for test-time performance, not validation loss. *arXiv preprint arXiv:2606.06418*, 2026.

Lin Zheng, Xueliang Zhao, Guangtao Wang, Chen Wu, David Dong, Angela Wang, Mingran Wang, Yun Du, Haige Bo, Amol Sharma, Bo Li, Kejie Zhang, Changran Hu, Urmish Thakker, and Lingpeng Kong. Evabyte: Efficient byte-level language models at scale, 2025. URL `https://hkunlp.github.io/blog/2025/evabyte`.

# CONTENTS

**1 Introduction** **1**

**2 Logit Conversion Methods** **2**

**3 Scaling Study Design** **3**

**4 Scaling Trends I: Validation BPB vs Training FLOPS** **4**

**5 Scaling Trends II: Task Performance vs Training FLOPs** **5**

**6 Scaling Trends III: Task Performance vs Validation BPB** **5**

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

**8 Related Work** **8**

**9 Discussion** **9**

**10 AI Disclosure statement** **9**

**A Experimental Setup Details** **17**

A.1 Model Architecture and FLOP Calculations. . . . . 17

A.2 Pretraining Dataset, Tokenization, and Data Scales . . . . . 17

A.3 Loss Function and Optimization Settings . . . . . 17

A.4 Sequence Length and Global Batch Size. . . . . 18

A.5 Bits Per Byte Calculation. . . . . 18

A.6 Storage Cost Calculations of Token and Byte Logits . . . . . 18

**B `Feather Plot` : Visualizing Iso-FLOP Point Connections** **20**

**C Additional Results using Validation Dataset 1** **20**

**D Comparisons with open-weight models: Additional details** **31**

**E Extended Related Work** **31**

**F On Scaling Laws Sensitivity** **33**

F.1 Leave One Out Error Analysis . . . . . 33

F.2 Residual Weighting Scheme Sensitivity Analysis . . . . . 33

F.3 Choice of the validation data . . . . . 34

**G Additional Results using Validation Dataset 2** **36**

G.1 BPB vs Training FLOPs power law equations . . . . . 37

G.2 Average top-1 Error Scaling Laws . . . . . 37
G.3 `Feather Plots` : Comparisons using validation dataset 2 . . . . . 39
G.4 Predicting the asymptotic Average Downstream task performance . . . . . 39

**H Distillation Methods: Additional Details** **40**
H.1 `Marginalize-It` and `End-Of-Token` Pseudocode . . . . . 40
H.2 Comparing `Marginalize-It` with `End-Of-Token` . . . . . 41

**I On BPB vs Benchmark Performance Discrepancy** **43**
I.1 Why the same BPB may imply different downstream performance? . . . . . 43

# 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 1.28 billion layer 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 and changes in the sequence length. 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 8: 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 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: $\alpha L_{KL}$, 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 1 as our loss function and use $\alpha$, the mixing coefficient to control our supervised and distillation runs.

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

$$L_{\text{KL}} = \frac{1}{n} \sum_{t=1}^{n} \sum_{c} p_T(c \mid x_{<t}) \log \frac{p_T(c \mid x_{<t})}{p_S(c \mid x_{<t})} \tag{2}$$

$$L_{\text{CE}} = -\frac{1}{n} \sum_{t=1}^{n} \sum_{c} y_{t,c} \log p_S(c \mid x_{<t}) \tag{3}$$

where:

1. $t$ is the position index in the sequence
2. $n$ is the total number of tokens in the sequence
3. $c$ is the token index over the vocabulary $V$
4. $y_{t,c} \in \{0, 1\}$ is a one-hot indicator that the ground truth token at position $t$ is $c$
5. $p_T(c \mid x_{<t})$ is the teacher's next token distribution given the context $x_{<t}$
6. $p_S(c \mid x_{<t})$ is the student's next token distribution given the context $x_{<t}$
7. $\alpha \in [0, 1]$ is the weight of the KL divergence in the overall loss

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 $\in$\{1e-3, 4e-3, 8e-3\}. We use independent weight decay (Loshchilov & Hutter, 2017) of 1e-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 (8K $\times$ 4 $\times$ 64 = 2M 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 $\approx$ 0.5M 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 4), 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.

$$\text{BPB} = \frac{L_{\text{CE}}(x) \cdot \log_2 e}{B} \tag{4}$$

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 compute 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 bypass 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 $\approx$ 260 bytes, trading vocabulary size for increased sequence length. Our controlled study is designed to characterize this tradeoff.

In table 9 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

Table 9: Storage cost analysis for top-$k$ logits per token. We solve for $k$ and found it to be $\approx 600$

| 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 | $260 \times 4.5$ | $4680 + 9$ |
| | Bytes | `int16` | 2 | 4.5 | |

distribution. By equating the total cost of BPE-tiktoken with bytes, we solve for $k = 4685/8$, which turns out to be $585.625 \approx 600$. Therefore for our controlled comparison we store 600 logits per BPE token and all of the byte logits i.e. 260 values for distillation.

## B FEATHER PLOT : VISUALIZING ISO-FLOP POINT CONNECTIONS

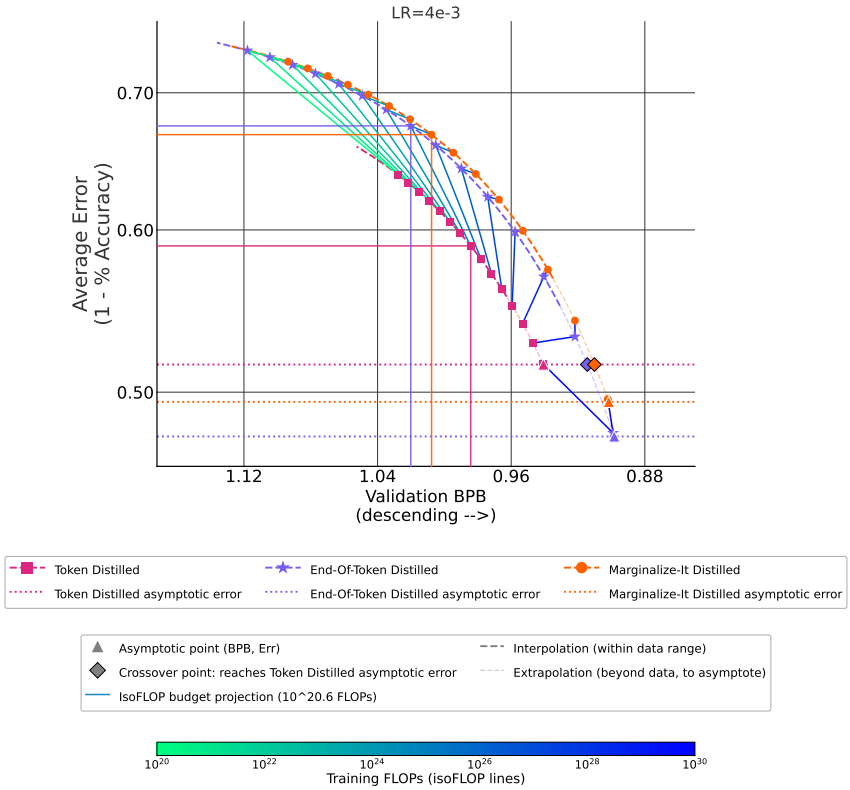


Figure 6: The figure shows a **Feather Plot** for a learning rate of 4e-3. We show that **End-Of-Token Distilled** transformer model has the lowest Average top-1 downstream task performance error asymptotically compared to the **Marginalize-It** and **Token** distillation. We note the differences in observed BPB value and the task performance by projecting iso-FLOP points on the X and Y axes for the three curves for a fixed compute budget value. 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 $f1$ to $f2$. For each sampled point $x$ we compute $y$ using equation $y = b \cdot x^a + c$. Next we use the obtained validation loss in terms of BPB $y$ to compute the downstream error $z(y)$ using equation $z(y) = \epsilon - k \cdot e^{-\gamma y}$. 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. At low compute budgets, distilled **Token-1B** performance leads significantly compared to **Bytes-1B** and **End-Of-Token-1B** models. 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 **Token-1B** too and continues to improve until the asymptotic value of the performance.

## C ADDITIONAL RESULTS USING VALIDATION DATASET 1

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

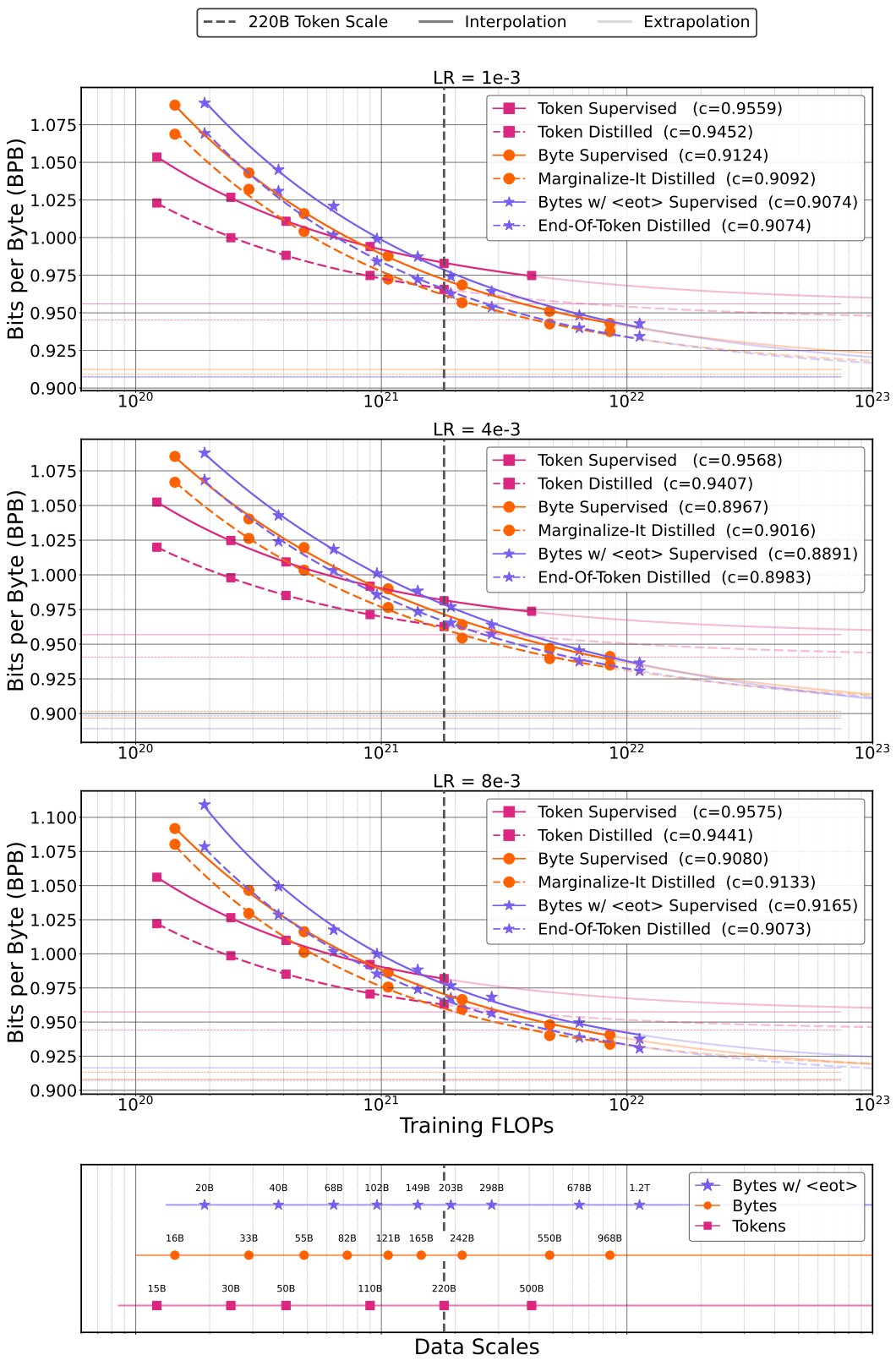


Figure 7: The figure shows our power law curves for different data scales and learning rates for a fixed parameter count of approximately 1 billion parameter transformer models(see Table 8 for details). The distilled models are trained using Llama 3-8B as a teacher model. The validation BPB of the teacher model is 0.85511.

Table 10: Power law equations $y = b \cdot x^a + c$ fitted to BPB vs. training FLOPs.

| LR | Experiment | Equation |
|---|---|---|
| $1 \times 10^{-3}$ | 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 | $y = 3.383 \times 10^8 \cdot x^{-0.4593} + 0.9074$ |
| $4 \times 10^{-3}$ | 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 | $y = 2.734 \times 10^7 \cdot x^{-0.4047} + 0.8983$ |
| $8 \times 10^{-3}$ | 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 11: 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 8.

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

Table 12: The coefficient of determination $R^2 = 1 - \mathrm{SS}_{\mathrm{res}}/\mathrm{SS}_{\mathrm{tot}}$ for the fit $y = b \cdot x^a + c$ of BPB versus training FLOPs where $\mathrm{SS}_{\mathrm{res}} = \sum (y_i - f_i)^2$ (the residual sum of squares) and $\mathrm{SS}_{\mathrm{tot}} = \sum (y_i - \bar{y})^2$ (the total sum of squares) where $y_i$ is the actual data, $f_i$ is the predicted value using the scaling law equation, and $\bar{y}$ is the mean of the actual data.

| Experiment | $R^2$ (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 |

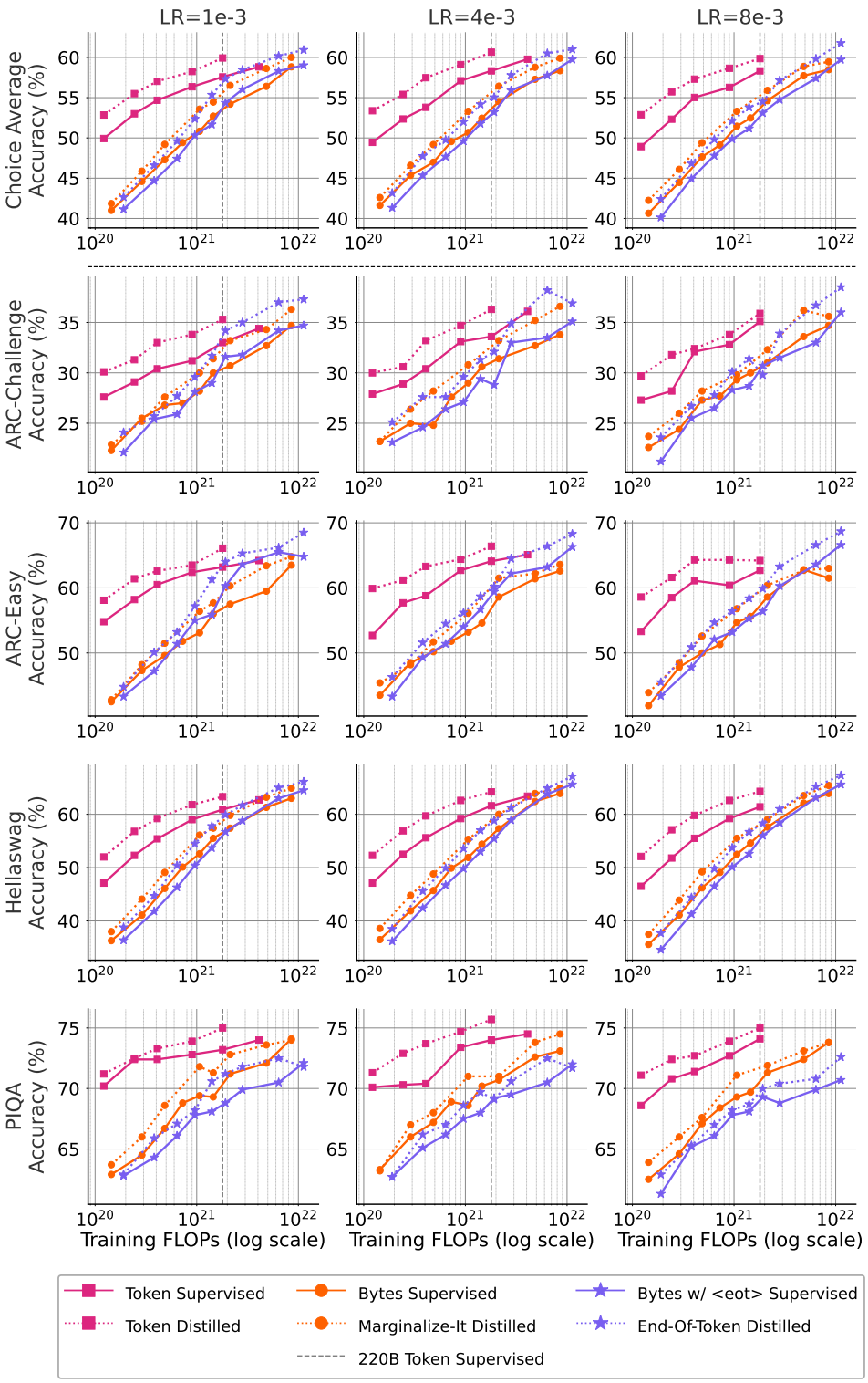


Figure 8: Multiple Choice Question Answering Benchmarks Evaluation: We plot the % Accuracy vs. training FLOP curves for all the six scenarios from section 3. For all of the Multiple Choice Question Answering Benchmarks, we observe that the token models show better performance at low compute budgets and eventually plateau. Byte models, start worse yet improve at a much faster rate.

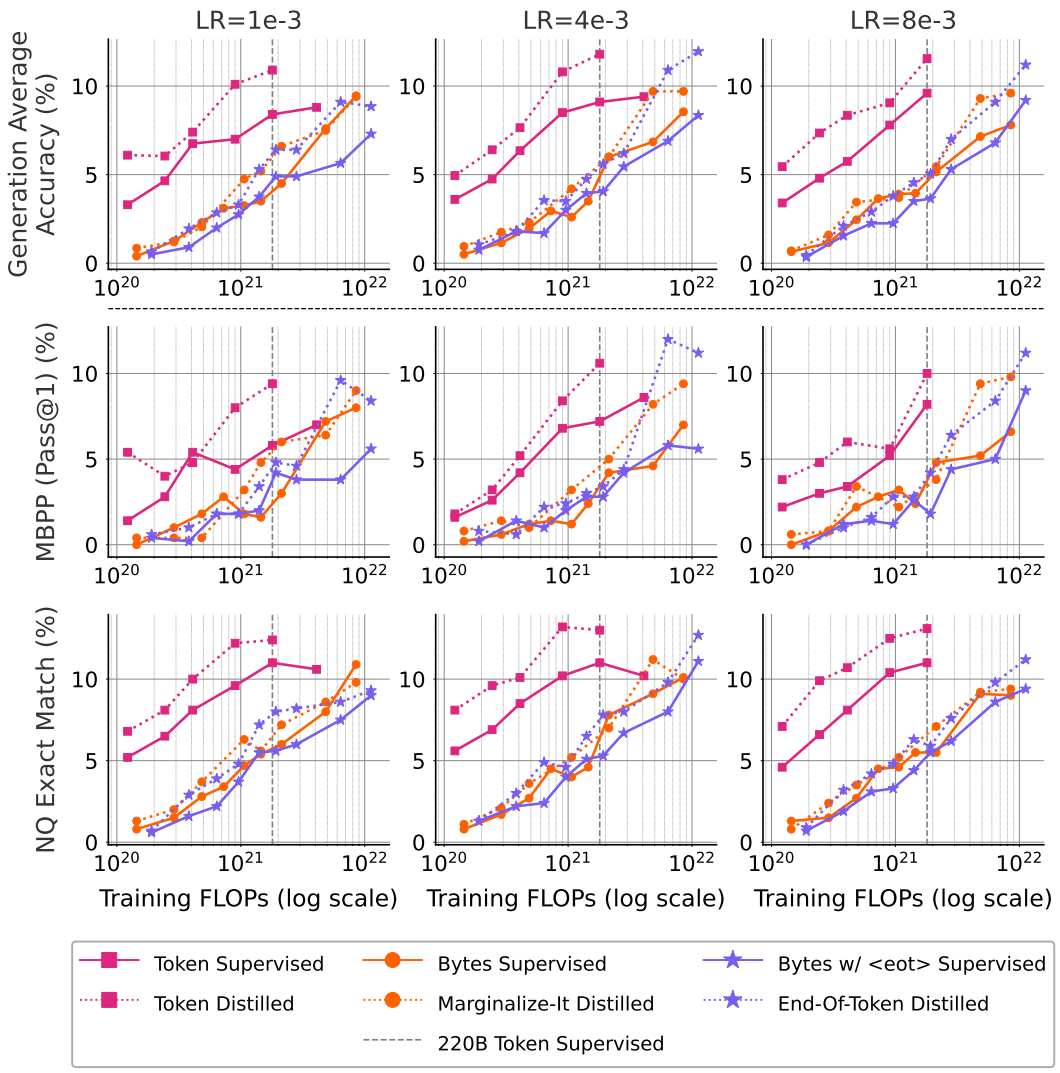


Figure 9: Language Generation Benchmarks Evaluation: We visualize the % Accuracy vs training FLOPs curves for language generation tasks. We find that token models achieve better performance at lower compute budgets and appear to plateau with the training FLOPs. **Bytes-1B** and **End-Of-Token-1B** models 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.

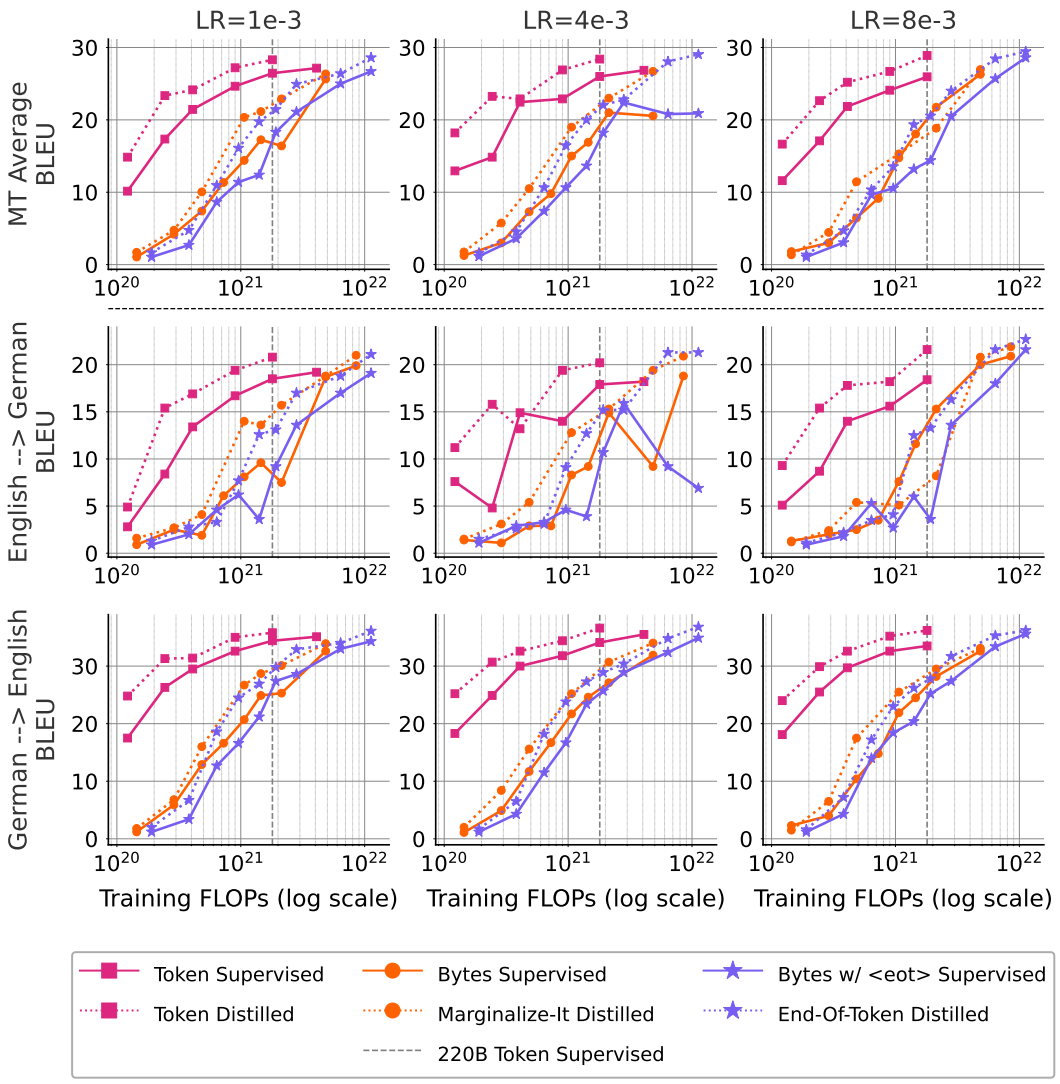


Figure 10: Machine Translation Benchmarks Evaluation: We plot BLEU score for the translation tasks for different training budgets. We observe clearly different scaling behaviors for token and byte models. Token models can achieve significantly better performance at a smaller compute budgets. The BLEU 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.

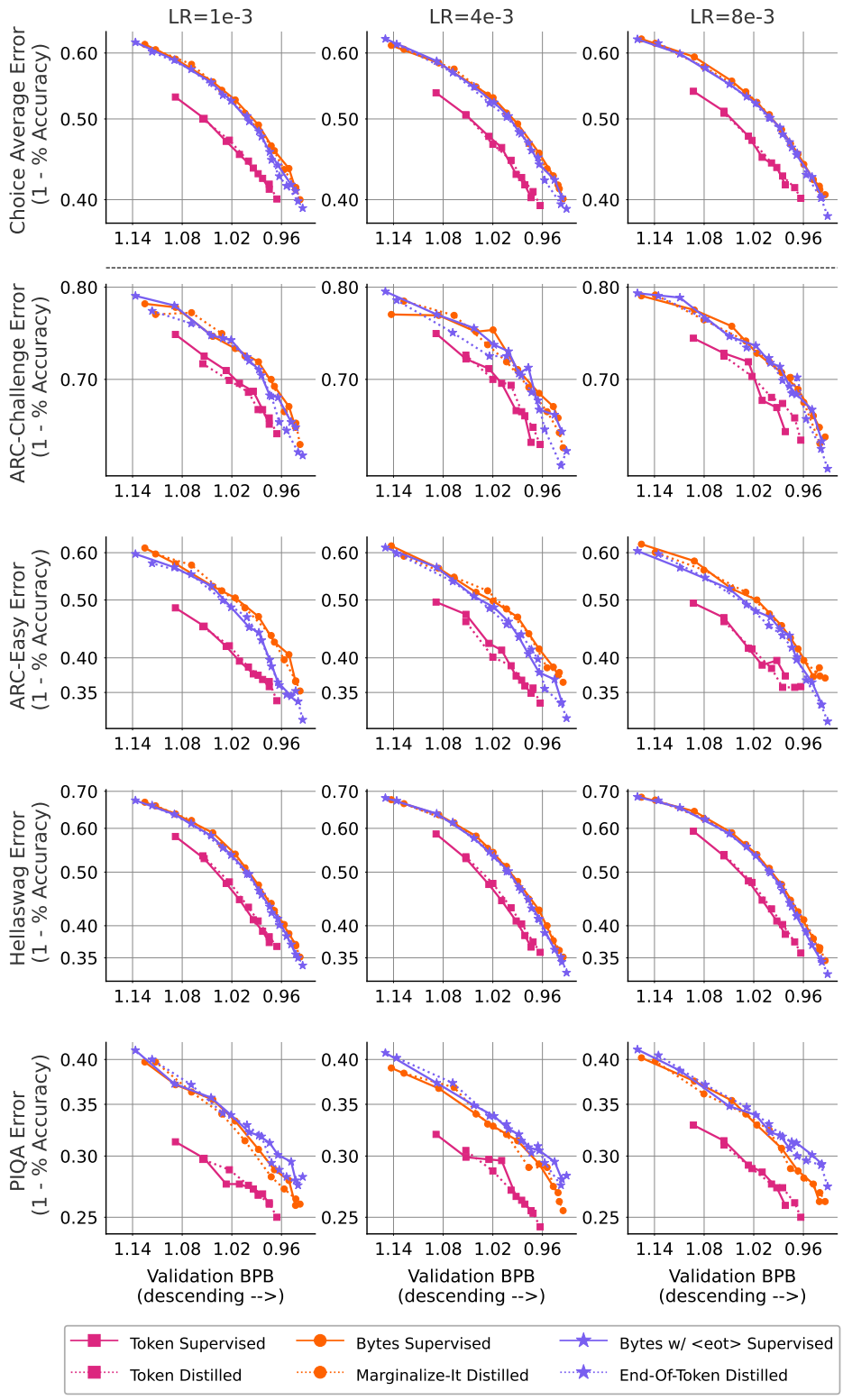


Figure 11: Multiple Choice QA downstream task performance scaling laws: We visualize Average top-1 error vs the validation BPB curves for the six scenarios for different learning rates and Multiple Choice Question Answering benchmarks.

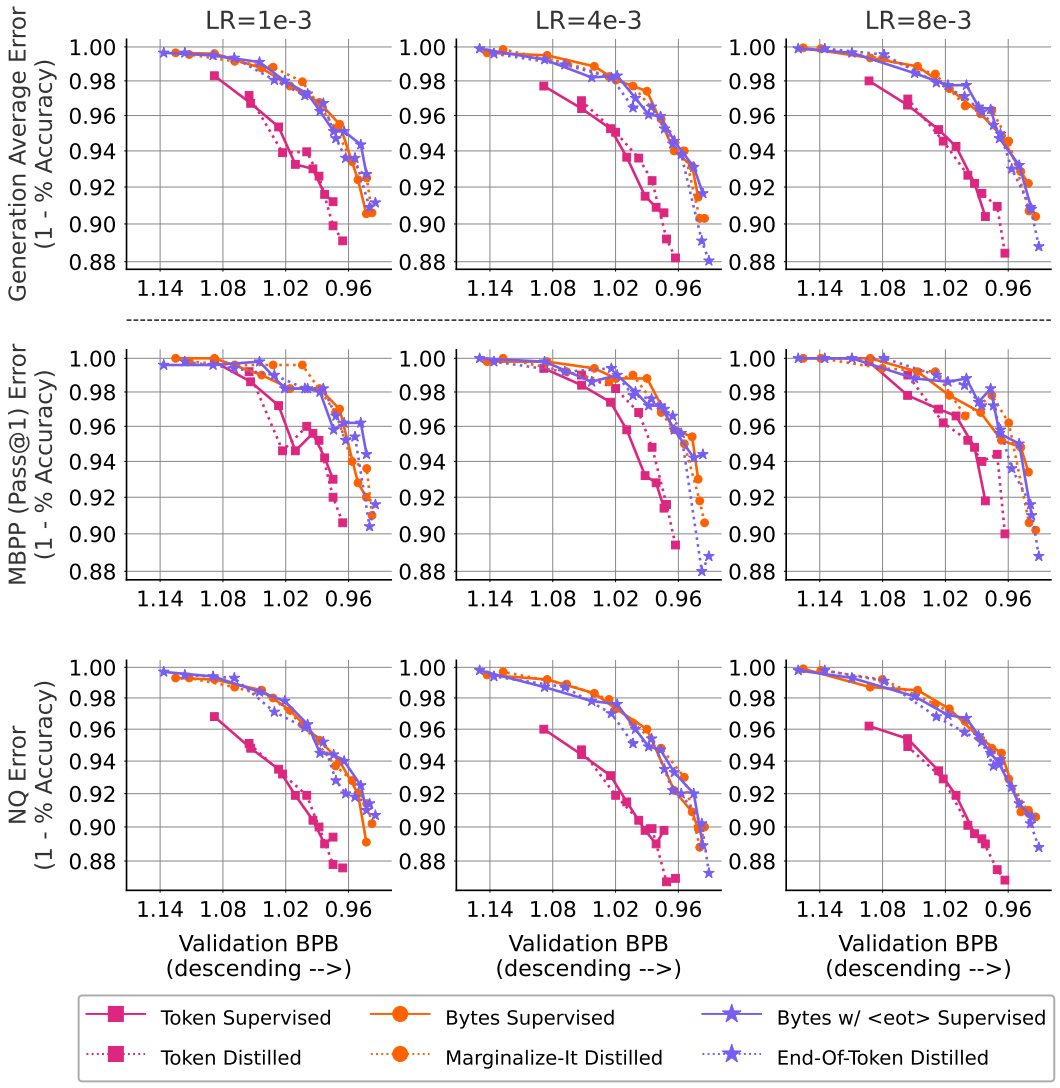


Figure 12: Language Generation downstream task performance scaling laws: We plot the Average top-1 error vs BPB curves for the MBPP and the NQ benchmarks. MBPP shows noisier behavior compared to NQ. We deduce that the same validation Bits Per Byte imply different downstream performance for different training objectives(distillation vs cross entropy) and tokenization schemes(**Tokens** , **Bytes** , **Bytes w/ <eot>** ). Distilled models as well demonstrate slightly different scaling behavior compared to the supervised counterparts.

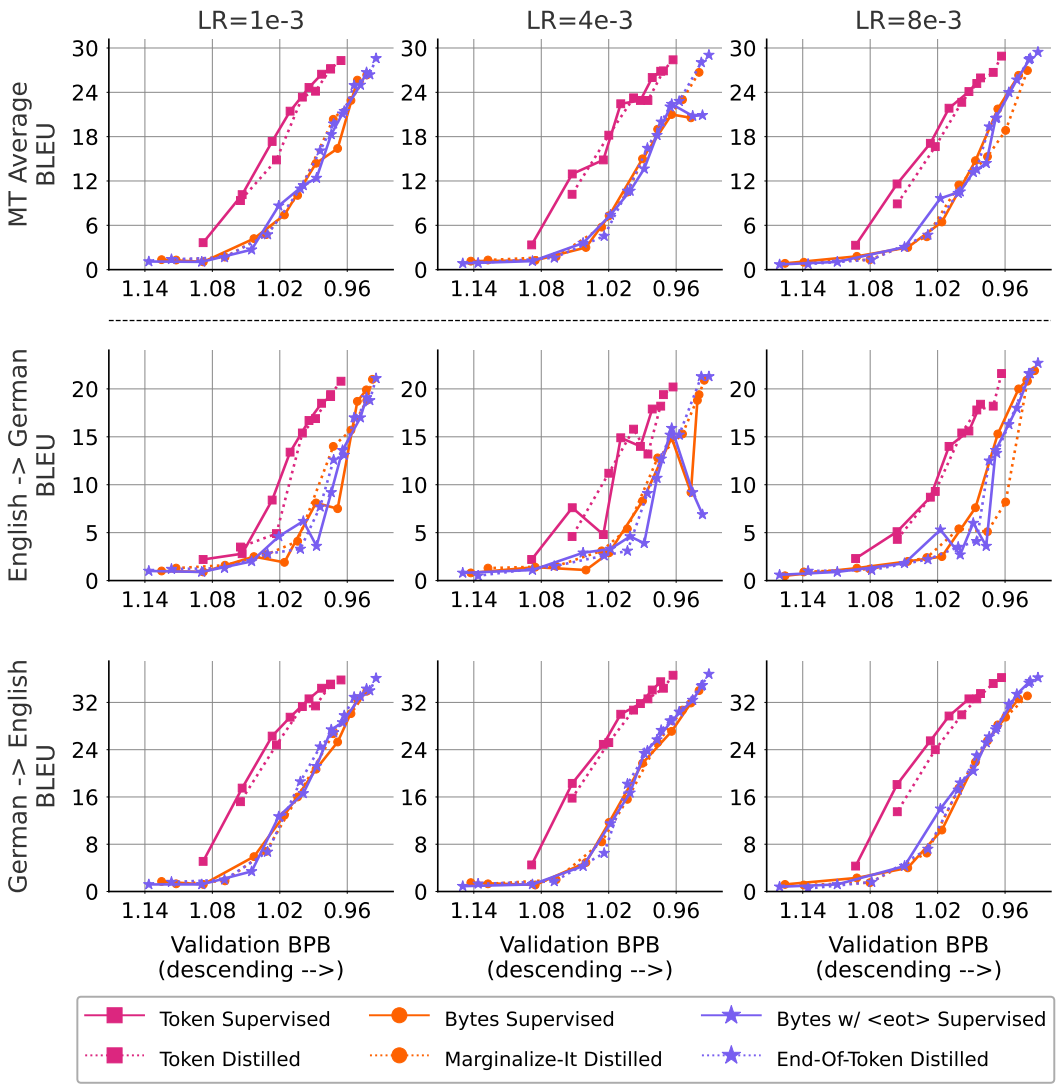


Figure 13: Machine Translation downstream task performance scaling laws: We plot BLEU score vs validation BPB performance for English to German translation, German to English translation, and on the averaged BLEU performance for the six scenarios for three learning rates. We notice the differences in 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.

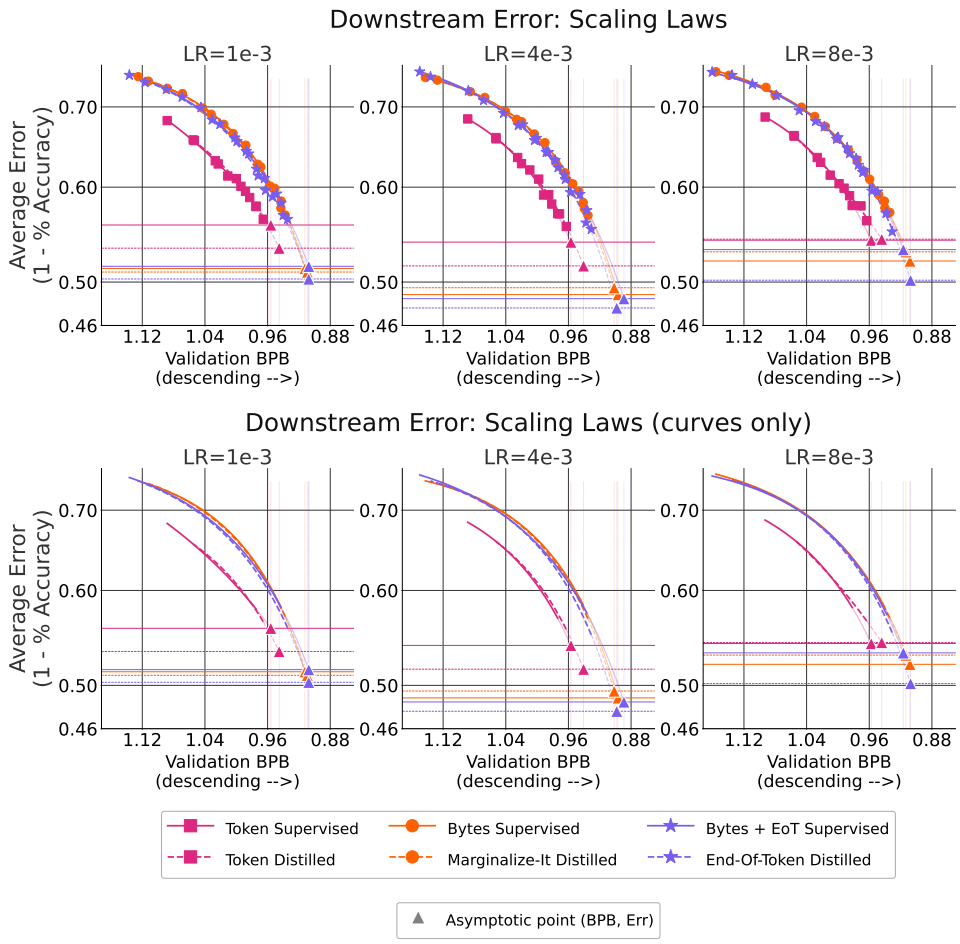


Figure 14: Average top-1 Error Scaling laws for validation dataset 1 and for three learning rates $\in$ {1e-3, 4e-3, 8e-3}

Table 13: Fitted downstream error scaling laws $z(y) = \epsilon - k \cdot e^{-\gamma y}$ for each experiment and learning rate $\in \{$1e-3, 4e-3, 8e-3$\}$, where $y$ is the validation BPB.

| Experiment | LR$= 1 \times 10^{-3}$ | LR$= 4 \times 10^{-3}$ | LR$= 8 \times 10^{-3}$ |
|---|---|---|---|
| `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 14: 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: $z' = \epsilon - k \cdot e^{-\gamma c}$ , where $c$ is obtained from the power law equations in Table 3 for each experiment and learning rate $\in \{$1e-3, 4e-3, 8e-3$\}$.

| | LR$= 1 \times 10^{-3}$ | | LR$= 4 \times 10^{-3}$ | | LR$= 8 \times 10^{-3}$ | |
|---|---|---|---|---|---|---|
| Experiment | 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 |

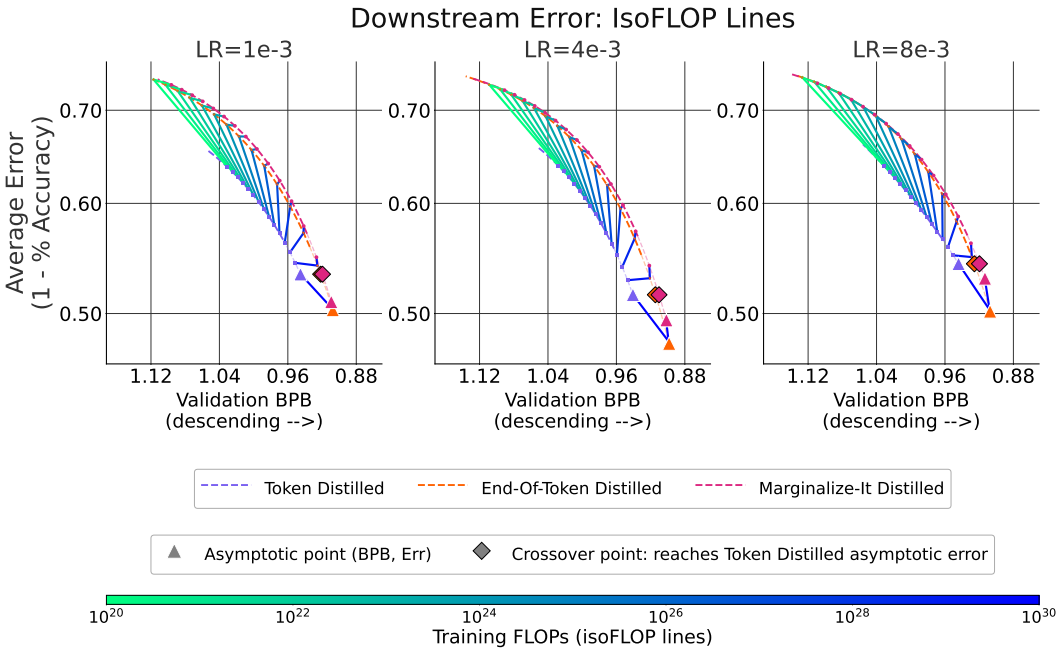


Figure 15: **Feather Plot** for learning rates $\in \{$1e-3, 4e-3, 8e-3$\}$ on validation dataset 1

Table 15: Learning rate wise training FLOPs calculations.

| LR | Method | Token Distillation BPB | Accuracy (%) | Method BPB | Method Training FLOPs |
|---|---|---|---|---|---|
| $1 \times 10^{-3}$ | Marginalize-It Distilled | 0.9452 | 46.6% | 0.9195 | $7.05 \times 10^{22}$ |
| | End-Of-Token Distilled | 0.9452 | 46.6% | 0.9216 | $3.91 \times 10^{22}$ |
| $4 \times 10^{-3}$ | Marginalize-It Distilled | 0.9407 | 48.4% | 0.9102 | $2.04 \times 10^{23}$ |
| | End-Of-Token Distilled | 0.9407 | 48.4% | 0.9145 | $6.33 \times 10^{22}$ |
| $8 \times 10^{-3}$ | Marginalize-It Distilled | 0.9441 | 45.7% | 0.9197 | $9.27 \times 10^{22}$ |
| | End-Of-Token Distilled | 0.9441 | 45.7% | 0.9256 | $2.13 \times 10^{22}$ |

Table 16: Storage and Data cost savings analysis calculation

| LR | Experiment | Training FLOPs | $\text{FLOPs}_{\text{byte}}$ | $\text{FLOPs}_{\text{token}}$ | Byte Units | Bytes | Tokens | Token to Byte | Data Savings | Storage Savings |
|---|---|---|---|---|---|---|---|---|---|---|
| $1 \times 10^{-3}$ | Marginalize-It Distilled | $7.05 \times 10^{22}$ | $8.81 \times 10^{9}$ | $8.18 \times 10^{9}$ | $8.00 \times 10^{12}$ | $8.00 \times 10^{12}$ | $8.61 \times 10^{12}$ | $3.87 \times 10^{13}$ | 4.85× | 4.85× |
| | End-Of-Token Distilled | $3.91 \times 10^{22}$ | $9.44 \times 10^{9}$ | $8.18 \times 10^{9}$ | $4.14 \times 10^{12}$ | $3.39 \times 10^{12}$ | $4.77 \times 10^{12}$ | $2.15 \times 10^{13}$ | 6.35× | 5.19× |
| $4 \times 10^{-3}$ | Marginalize-It Distilled | $2.04 \times 10^{23}$ | $8.81 \times 10^{9}$ | $8.18 \times 10^{9}$ | $2.32 \times 10^{13}$ | $2.32 \times 10^{13}$ | $2.50 \times 10^{13}$ | $1.12 \times 10^{14}$ | 4.85× | 4.85× |
| | End-Of-Token Distilled | $6.33 \times 10^{22}$ | $9.44 \times 10^{9}$ | $8.18 \times 10^{9}$ | $6.70 \times 10^{12}$ | $5.48 \times 10^{12}$ | $7.73 \times 10^{12}$ | $3.48 \times 10^{13}$ | 6.35× | 5.19× |
| $8 \times 10^{-3}$ | Marginalize-It Distilled | $9.27 \times 10^{22}$ | $8.81 \times 10^{9}$ | $8.18 \times 10^{9}$ | $1.05 \times 10^{13}$ | $1.05 \times 10^{13}$ | $1.13 \times 10^{13}$ | $5.10 \times 10^{13}$ | 4.85× | 4.85× |
| | End-Of-Token Distilled | $2.13 \times 10^{22}$ | $9.44 \times 10^{9}$ | $8.18 \times 10^{9}$ | $2.25 \times 10^{12}$ | $1.84 \times 10^{12}$ | $2.60 \times 10^{12}$ | $1.17 \times 10^{13}$ | 6.35× | 5.19× |

# D Comparisons with open-weight models: Additional details

Table 17: Downstream benchmark accuracy (%) for the LR=4e−3 learning rate runs at their highest observed data scale, compared against the Llama and Gemma baselines. Average is the mean reported over six benchmarks (four Multiple Choice QA and two language generation).

| 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** | 968B | 63.9 | 73.1 | 62.6 | 33.8 | 7.0 | 10.1 | 41.8 |
| **Marginalize-It Distilled** | 968B | 64.9 | 74.5 | 63.6 | 36.6 | 9.4 | 10.0 | 43.2 |
| **Bytes w/ <eot> Supervised** | 1.2T | 65.6 | 72.0 | 66.3 | 35.1 | 5.6 | 11.1 | 42.6 |
| **End-Of-Token Distilled** | 1.2T | 67.1 | 71.7 | 68.3 | 36.9 | 11.2 | 12.7 | 44.6 |

# E Extended Related Work

**Scaling Laws with Overtraining.** Which factors affect the power law exponent by varying data budgets 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. (Bansal et al., 2022; Bahri et al., 2024) 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, 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 (10) and table (14). We show that a simple addition of an <eot> token every $\approx 4.5$ bytes can improve the asymptotic downstream task 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 metaanalysis 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 models to operate on raw bytes. Minixhofer et al. (2026), Phan et al. (2026) and Bao et al. (2026) propose methods for cross tokenizer distillation in the finetuning 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 a prompt, when ends in the middle of a BPE token, results into distorted next token distributions at the 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. 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.

# F On Scaling Laws Sensitivity

We now characterize the sensitivity of downstream error scaling laws.

## F.1 Leave One Out Error Analysis

We analyze the leave one out sensitivity of the power laws. specifically for each of the six experiments we omit one data point from data scales shown in Figure 2 and fit the power laws and measure the c, the asymptotic value.

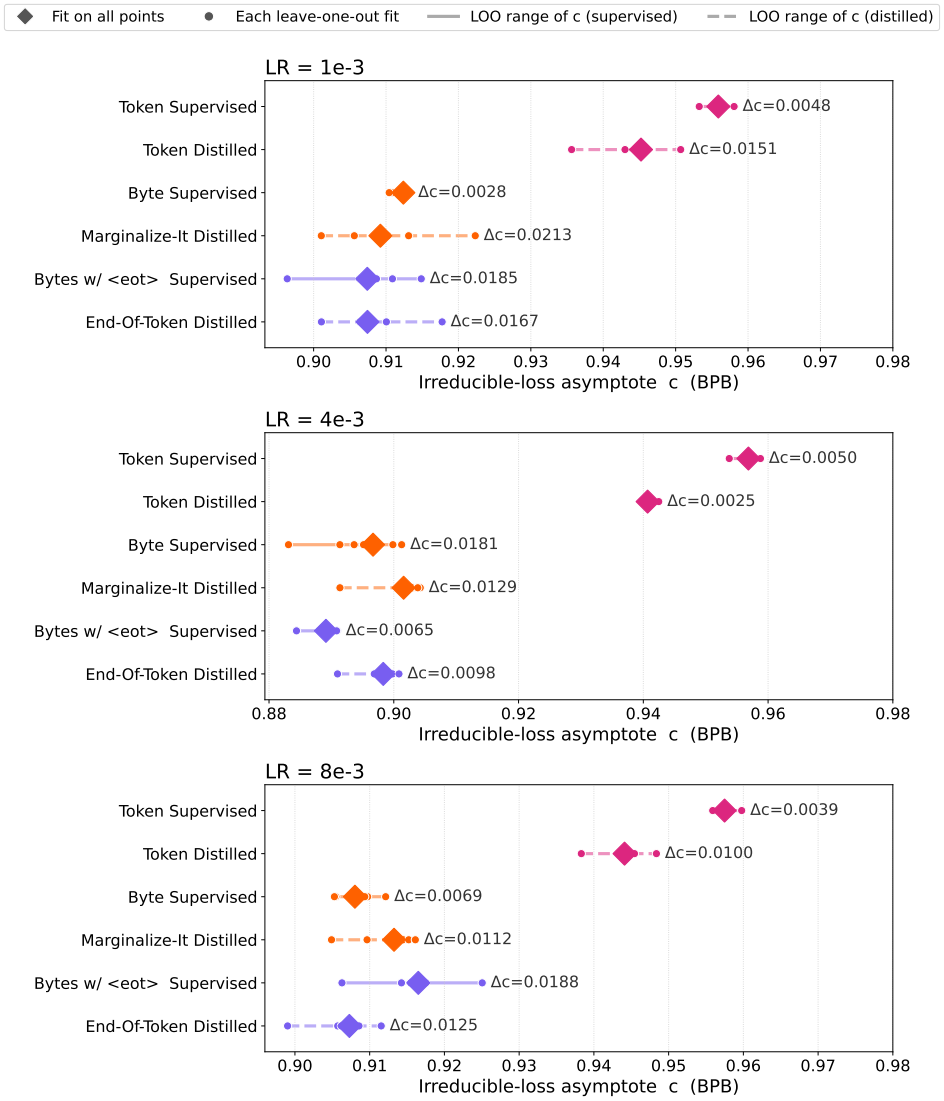


Figure 16: **Token** curves on average demonstrate little spread compared to **Bytes** and **Bytes w/ <eot>** methods.

## F.2 Residual Weighting Scheme Sensitivity Analysis

We use the `unweighted` weighting scheme method for fitting the power laws for all of our plots. In this section we analyze the effect of using alternative weighting schemes on c, the asymptotic value of the power laws. We note that other weighting schemes are more sensitive to **Bytes** and **Bytes w/ <eot>** tokenization.

Each a choice of the $\sigma$ vector in `Scipy curve fit`'s objective $\sum_i \left((\hat{y}(x_i) - y_i)/\sigma_i\right)^2$:

1. `unweighted`: Ordinary least squares.
2. `weighted y` : Down weighting high BPB low compute points.
3. `inverse x`: up weighting the large compute points.

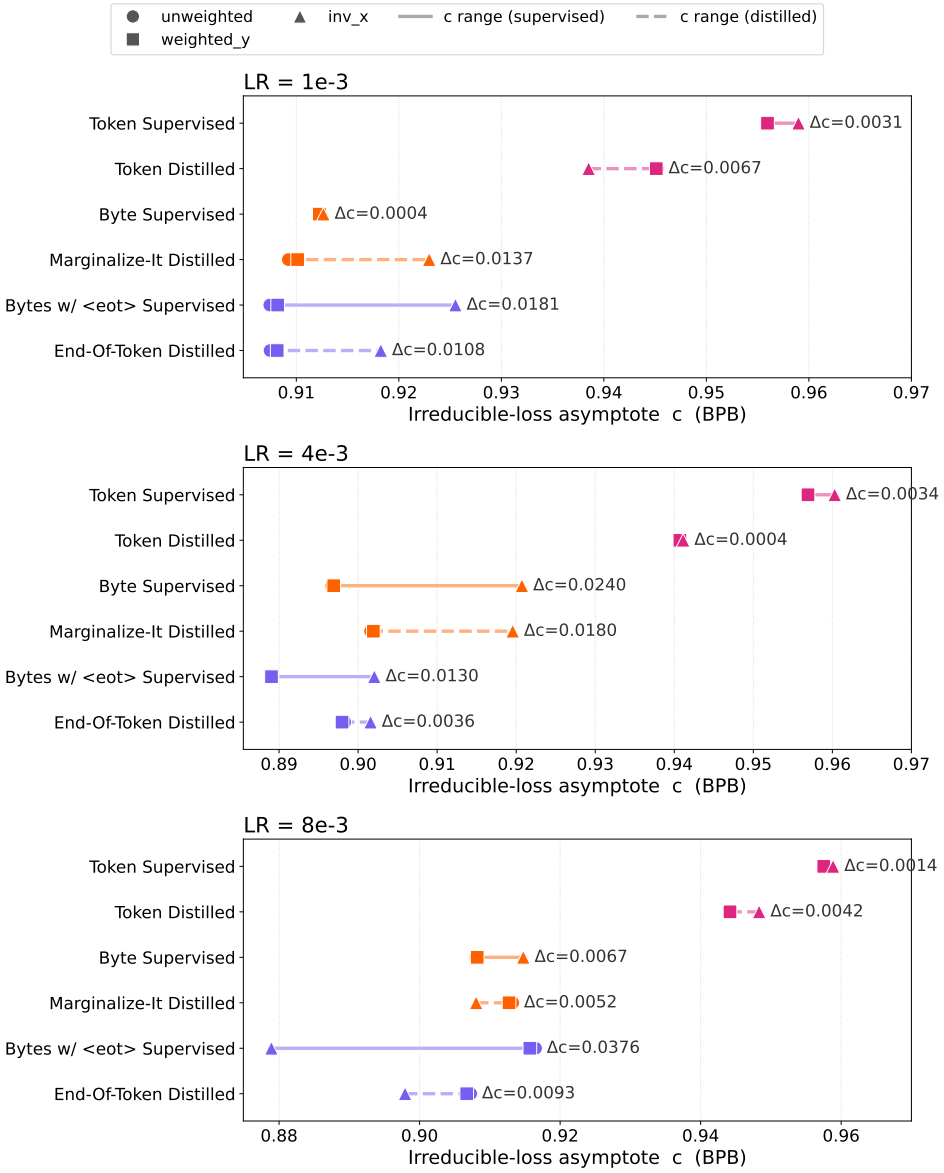


Figure 17: We visualize the sensitivity of the residual weighting scheme on the power laws.

### F.3 Choice of the validation data

We observe that the choice of the validation dataset can alter the crossover points of the scaling law curves and as a result the "perception" of a better model. Consider for example in Figure 18. 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.

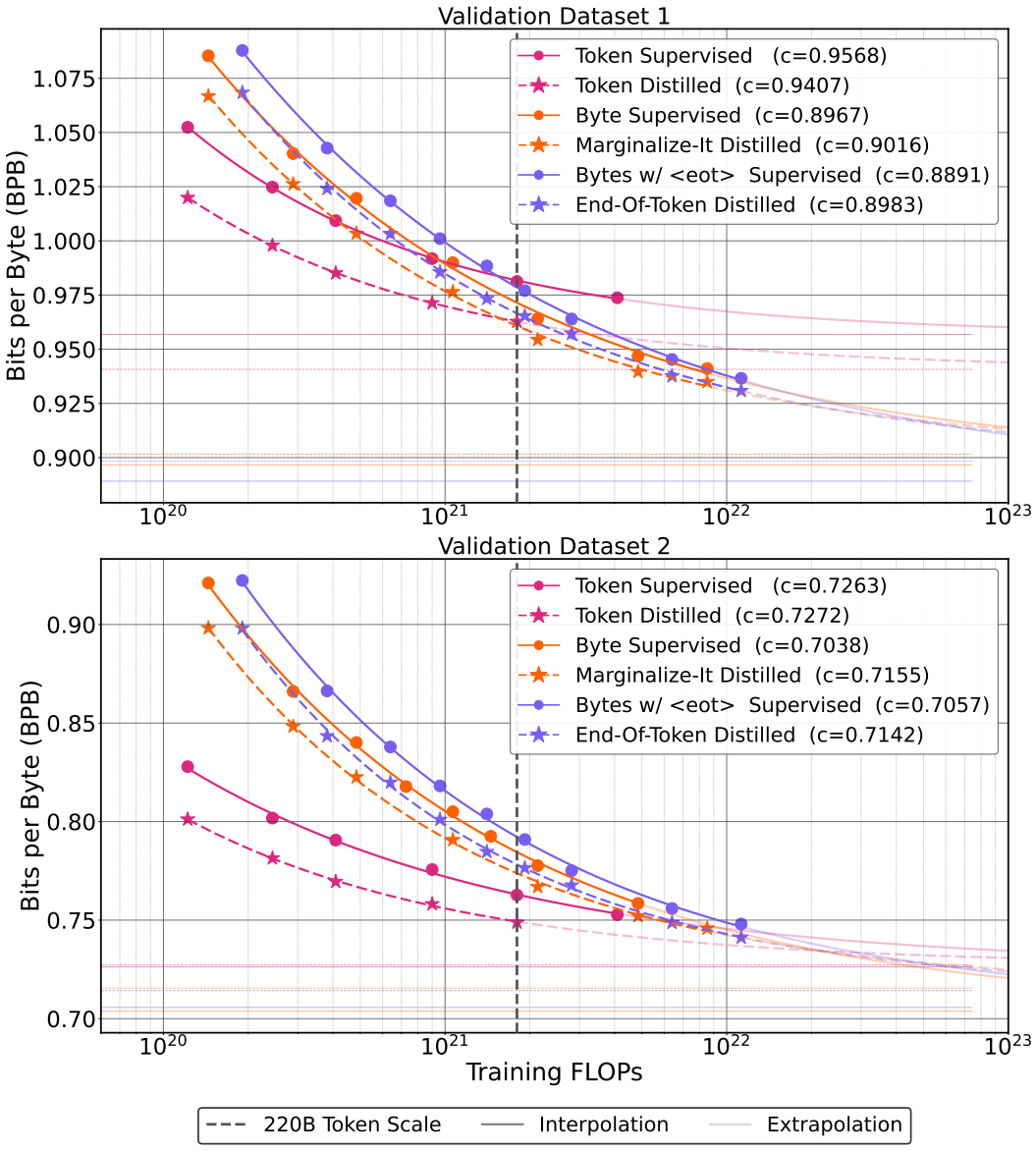


Figure 18: We demonstrate how changes in the validation dataset can affect the validation BPB vs training FLOPs scaling laws by comparing power laws for LR=4e-3. At the 220B **Token Supervised** FLOP budget, all of the other five models have a lower BPB for validation dataset 1. On the other hand, for 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 experiments.

# G ADDITIONAL RESULTS USING VALIDATION DATASET 2

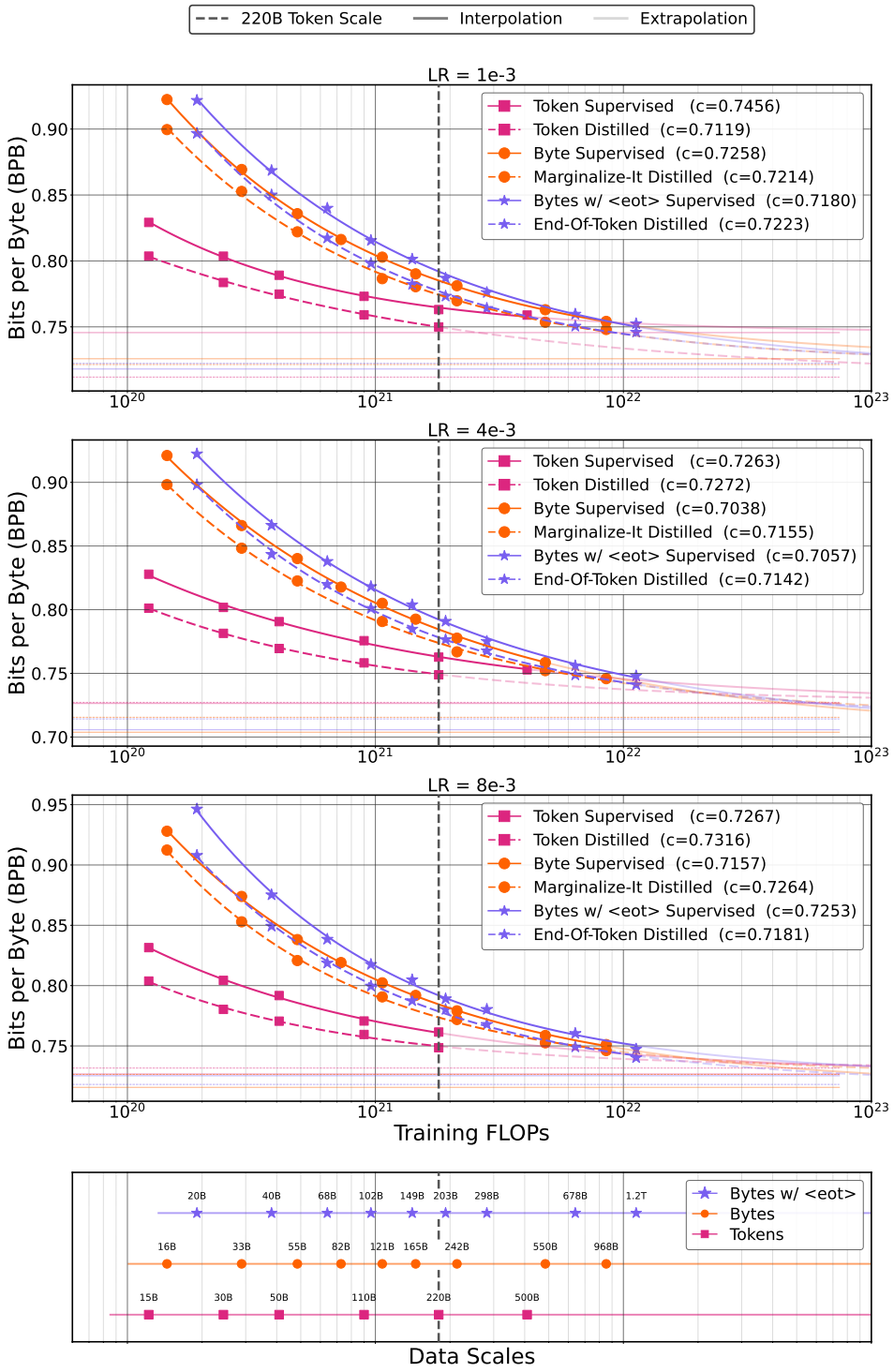


Figure 19: BPB vs FLOPs for validation dataset 2.

### G.1 BPB VS TRAINING FLOPS POWER LAW EQUATIONS

Table 18: Fitted BPB vs FLOPs power laws of the form $y = b\,x^a + c$, where $x$ is training FLOPs and $y$ is BPB on validation dataset 2. The asymptotic BPB values change significantly when we change the validation dataset

| LR | Experiment | Equation |
|---|---|---|
| $1 \times 10^{-3}$ | 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 | $y = 6.752 \times 10^{9}\, x^{-0.5218} + 0.7223$ |
| $4 \times 10^{-3}$ | 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 | $y = 5.648 \times 10^{8}\, x^{-0.4679} + 0.7142$ |
| $8 \times 10^{-3}$ | 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 | $y = 3.570 \times 10^{9}\, x^{-0.5068} + 0.7181$ |

### G.2 AVERAGE TOP-1 ERROR SCALING LAWS

Table 19: Downstream error scaling law equations $z(y) = \epsilon - k \cdot e^{-\gamma y}$ for each experiment and learning rate $\in$ { 1e-3, 4e-3, 8e-3 }, where $y$ is the validation BPB calibrated using Validation dataset 2 BPB values.

| **Model** | LR$= 1 \times 10^{-3}$ | LR$= 4 \times 10^{-3}$ | LR$= 8 \times 10^{-3}$ |
|---|---|---|---|
| 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 | $0.7726 - 110.6994\, e^{-8.3872\, y}$ | $0.7673 - 133.6625\, e^{-8.6766\, y}$ | $0.7728 - 102.4737\, e^{-8.3038\, y}$ |

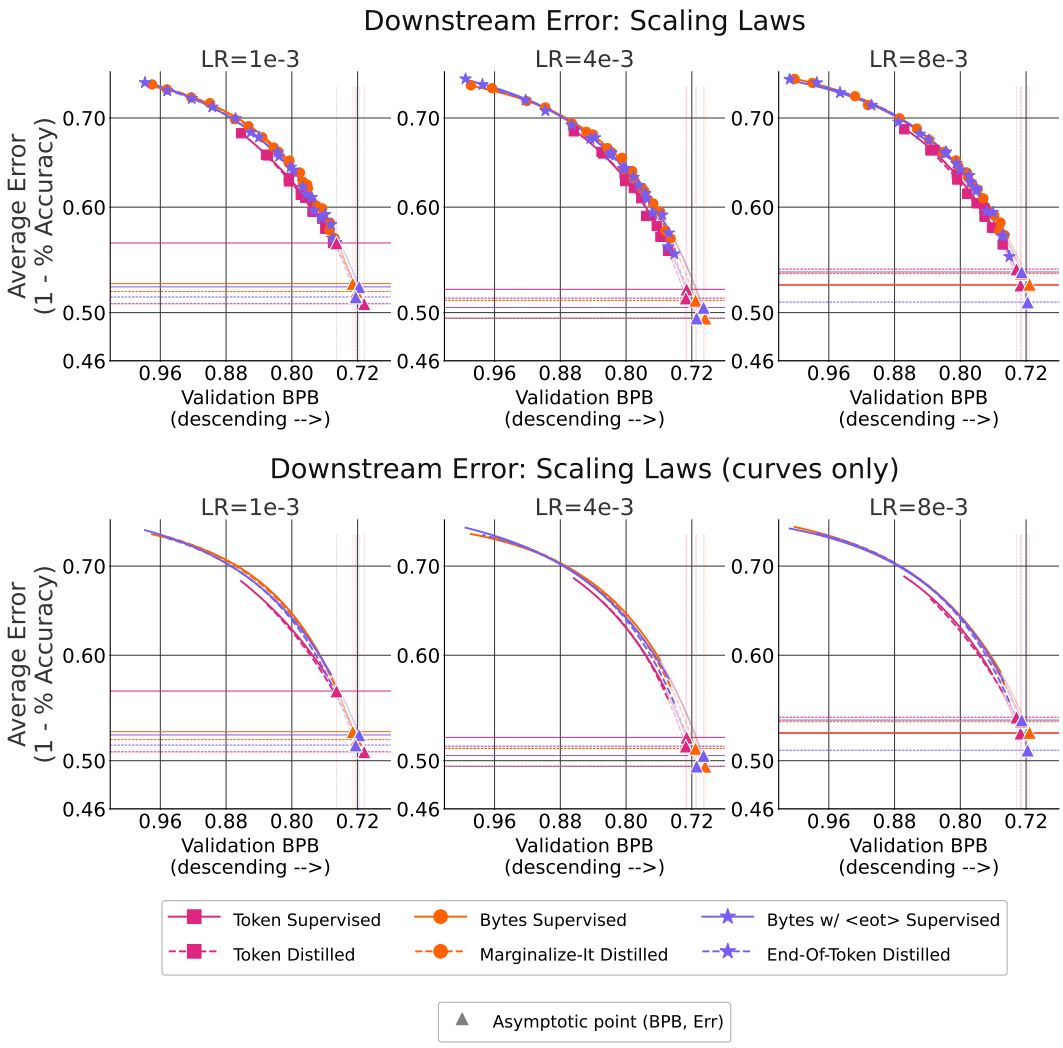


Figure 20: Average top-1 error scaling laws calibrated using Validation dataset 2 BPB values.

Table 20: Asymptotic validation 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.

| | LR$= 1 \times 10^{-3}$ | | LR$= 4 \times 10^{-3}$ | | LR$= 8 \times 10^{-3}$ | |
|---|---|---|---|---|---|---|
| **Model** | 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 |

### G.3 FEATHER PLOTS : COMPARISONS USING VALIDATION DATASET 2

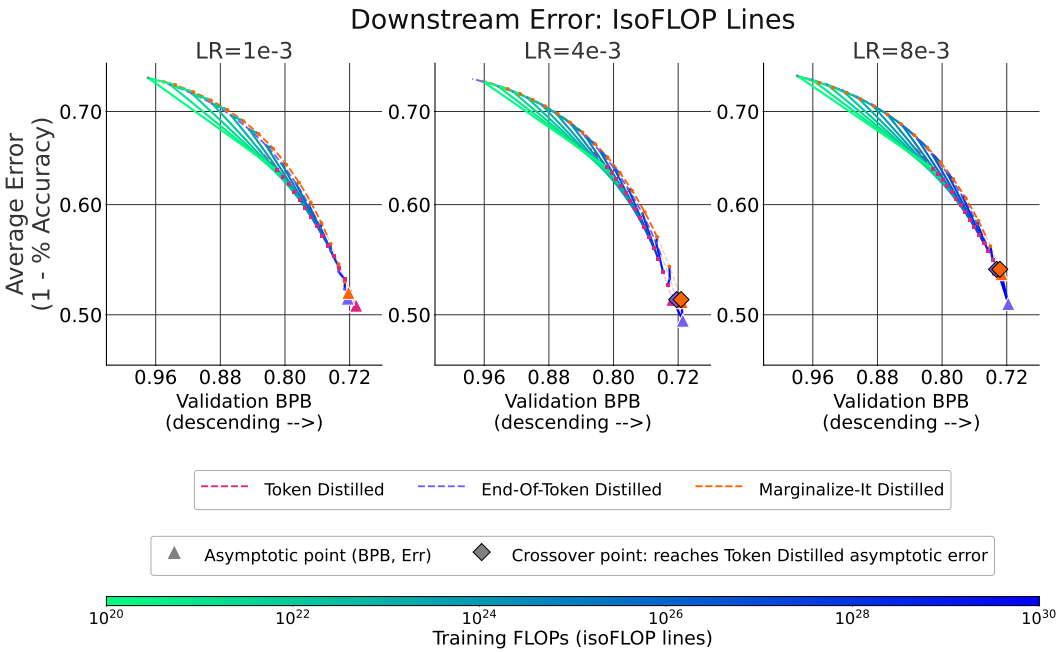


Figure 21: Feather plots calibrated using Validation dataset 2 BPB values

### G.4 PREDICTING THE ASYMPTOTIC AVERAGE DOWNSTREAM TASK PERFORMANCE

Table 21: Training compute at which **Marginalize-It Distilled** and **End-Of-Token Distilled** match the asymptotic downstream performance of a **Token** distilled model using Validation Dataset 2.

| LR | Experiment | Token Distillation BPB | Accuracy (%) | Experiment BPB | Training FLOPs |
|---|---|---|---|---|---|
| $1 \times 10^{-3}$ | Marginalize-It Distilled | — | — | — | — |
| | End-Of-Token Distilled | — | — | — | — |
| $4 \times 10^{-3}$ | Marginalize-It Distilled | 0.7272 | 48.7% | 0.7165 | $1.60 \times 10^{25}$ |
| | End-Of-Token Distilled | 0.7272 | 48.7% | 0.7219 | $1.69 \times 10^{23}$ |
| $8 \times 10^{-3}$ | Marginalize-It Distilled | 0.7316 | 46.1% | 0.7286 | $5.17 \times 10^{23}$ |
| | End-Of-Token Distilled | 0.7316 | 46.1% | 0.7326 | $3.01 \times 10^{22}$ |

Table 22: Data and Storage cost savings analysis

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

# H Distillation Methods: Additional Details

## H.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) els
    """
    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
```

### H.2 COMPARING MARGINALIZE-IT WITH END-OF-TOKEN

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

**Case I**: Predicting first byte distribution using token distribution (Figure 24):

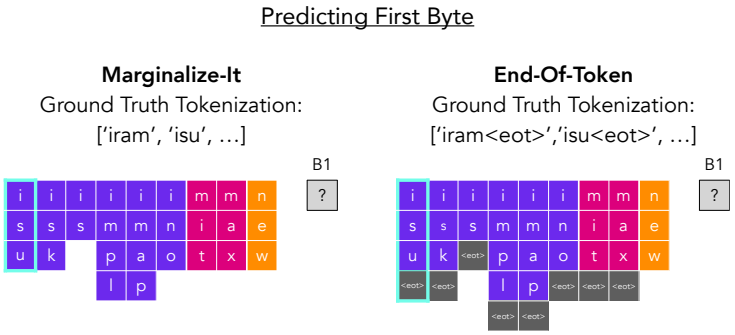


Figure 22: B1 is predicted directly by marginalization using next token distribution of token `iram` and is exact.

Figure 23: B1 is predicted directly by marginalization using next token distribution of token `iram<eot>` and is exact.

Figure 24: 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 intermediate byte distribution using token distribution when the ground truth token length matches with all other alternative tokens with non zero probability in the vocabulary (Figure 27): :

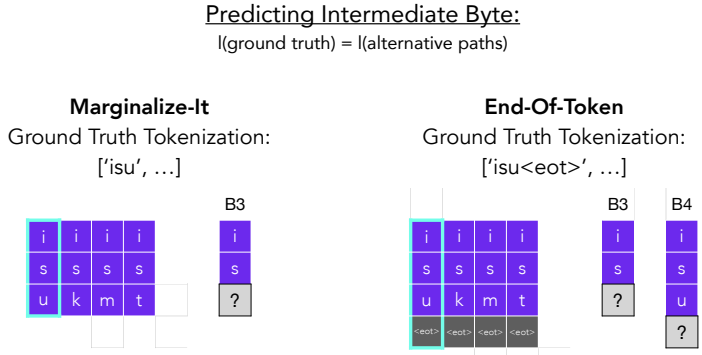


Figure 25: B3 prediction is based on tokens with prefix `is` with a length of at least 3 bytes. Other alternative paths (with non zero probability) have exactly 3 bytes length and can be marginalized exactly.

Figure 26: 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 27: 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 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 30):.

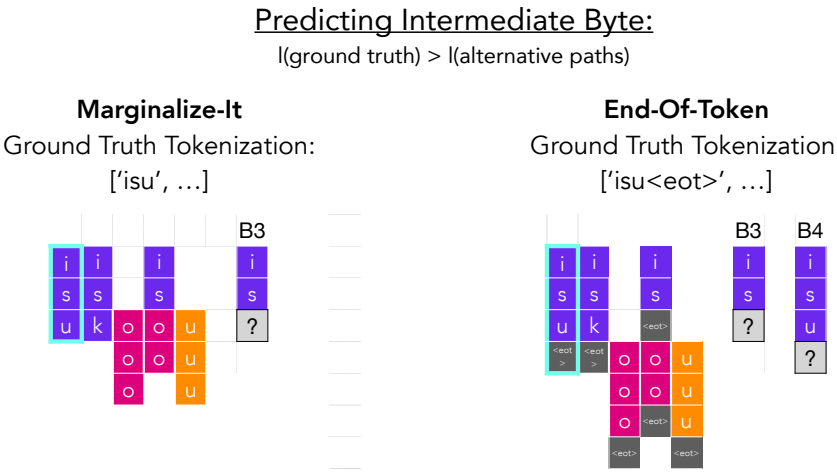


Figure 28: B3 prediction is based on tokens with prefix `is` with a length of at least 3 bytes. Therefore it misses other tokenization paths: [`is`, `ooo`] , [`is`, `oo`], [`is`, `uuu`]

Figure 29: 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 30: We demonstrate the intermediate byte prediction when there exits 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 solves this problem by adding the new `<eot>` token.

**Case IV**: Predicting 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 33):

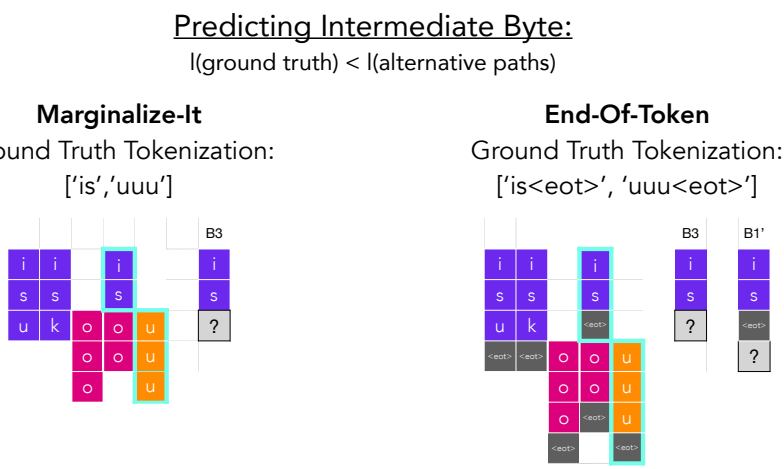


Figure 31: B3 prediction is based on the next token distribution of `is` and therefore it misses contributions from other tokenization paths formed by tokens `isu` and `isk`.

Figure 32: 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 33: 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.

# I On BPB vs Benchmark Performance Discrepancy

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

In this section we aim to provide some intuition on why two models can have exactly the same BPB yet can have different downstream task accuracy. Essentially, BPB is calculated using only one value: the probability the model predicts on the ground truth token, $P_{\text{true}}$. The per token loss is $-\log_2 P_{\text{true}}$, so once $P_{\text{true}}$ is fixed the loss is fixed. BPB does not consider how the remaining probability is distributed among the tokens in the vocabulary. Downstream task accuracy, however, depends on whether the ground truth is also the "argmax", i.e. on the ranking of the full probability distribution. Since the ranking also depends on the rest of the probability distribution (and not just the probability that a model places on the ground truth token), two models can have the same BPB while generating different text.

`Hypothetical Example:` Consider the ground truth text "`Where's my tiramisu?`", which, for example, the Llama3 8b tokenizer tokenizes into 7 tokens with 20 bytes:

`Where` | `'s` | `␣my` | `␣tir` | `am` | `isu` | `?`

Now consider a scenario where both models place probability $P_{\text{correct}} = 0.4$ on the correct token at every position. Then per token loss is $-\log_2 0.4 = 1.32$ bits, the total is $7 \times 1.32 = 9.24$ bits, and BPB = 9.24/20 = 0.462 for both models. The models differ only in the way they distribute the remaining probability of 0.6 among other tokens in the vocabulary. While calculating BPB, however, the remainder of the probability mass is never taken into account.

Table 23: An example of how the relative ranking of tokens in the vocabulary can lead to worse generation performance while maintaining the same BPB value.

| Token | Ground truth token | $P_{\text{true}}$ | Model A | | Model B | |
|---|---|---|---|---|---|---|
| | | | other tokens in the vocabulary | argmax | other tokens in the vocabulary | argmax |
| 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 | `!` |

As demonstrated in Table 23 two models have the exact same $P_{true}$ though Model A achieve 100% accuracy while Model B has a 0% accuracy. Equal BPB constrains the probability of the target but does not ensure the topmost rank in the vocabulary.