

Training with Subword Variants Delays Overfitting in Repeated-Data Language Modeling

Adrian Cosma¹, Stefan Ruseti², Emilian Radoi², Mihai Dascalu²

¹Dalle Molle Institute for Artificial Intelligence Research (IDSIA)

²National University of Science and Technology POLITEHNICA Bucharest (UNSTPB)

adrian.cosma@idsia.ch, {stefan.ruseti, emilian.radoi, mihai.dascalu}@upb.ro

Abstract

Subword tokenization introduces a computational layer in language models, where many distinct token sequences decode to the same surface form while preserving meaning, yet induce different internal computations. Despite this non-uniqueness, language models are typically trained using a single canonical deterministic tokenization. In this work, we use non-canonical subword variants as a strictly meaning-preserving form of data augmentation. We introduce a lightweight training architecture that conditions canonical next-token prediction on sampled subword variants via an auxiliary causal encoder and block-causal cross-attention, without modifying the training objective or token interface. In data-constrained pretraining, augmentations with subword variants consistently delay overfitting under repeated data exposure and reduce held-out next-token loss across evaluation datasets. In multilingual fine-tuning, we find that the effectiveness of training with auxiliary subwords depends on tokenizer quality: gains are strongest when canonical tokens are highly compressed and diminish when the tokenizer already over-fragments the input.

1 Introduction

The relationship between form and meaning is not arbitrary. The standard hierarchy of linguistic analysis (from phonology and morphology to semantics and pragmatics) describes increasing levels of abstraction from surface form to communicative intent (Jurafsky and Martin, 2025). Classical linguistic theory places words at a privileged level in this hierarchy: lexical items mediate between form and meaning, while lower-level variations (e.g., phonetic realizations) are typically treated as semantically irrelevant. Modern Large Language Models

(LLMs), however, introduce an additional computational layer that is largely invisible to human linguistic analysis: subword tokenization (Kudo and Richardson, 2018; Shibata et al., 1999).

Subword tokenization algorithms such as Byte Pair Encoding (BPE) (Shibata et al., 1999) segment text into discrete units that are neither pure characters nor stable lexical items. These tokens sit at an intermediate level between characters, morphemes, and words, and are defined not by linguistic principles but by compression and frequency statistics (Shibata et al., 1999). In practice, LLMs are trained and evaluated almost exclusively on deterministic canonical tokenizations, even though alternative segmentations of the same word are equally valid from a decoding perspective.

The token is the fundamental unit of computation in contemporary LLMs: it determines sequence length, attention patterns, and, ultimately, the model’s internal activation dynamics. However, prior work has shown that training LLMs with BPE tokenizations is suboptimal (Bostrom and Durrett, 2020), and that finding an optimal tokenization (in terms of maximizing text compression) is computationally infeasible (Whittington et al., 2025). As a result, tokenization is typically treated as a fixed preprocessing step rather than an object of learning or perturbation.

There is conflicting evidence on whether LLMs are actually robust to tokenizations. For example, Zheng et al. (2025) showed that LLMs are robust to non-canonical tokenizations of the input, but a plethora of other works show that LLMs are severely bottlenecked by the underlying tokenization in multilingual settings (Petrov et al., 2023; Haslett, 2025; Wang et al., 2024). LLMs are blind to the character composition of words (Shin and Kaneko, 2024) and require a significant amount

of data to overcome this limitation (Cosma et al., 2025).

A natural response to tokenization sensitivity is to modify or replace the tokenizer itself. However, training better tokenizers or abandoning tokenization altogether often incurs substantial computational cost, architectural complexity, or technical debt (Pepe et al., 2024). In this work, we take a different approach: we treat the tokenizer as given and instead ask whether models can be trained to become invariant to tokenization choices and, by extension, to extract more signal out of the training dataset.

We use non-canonical subword segmentations¹, which prior work has studied under subword regularization (Kudo, 2018), as auxiliary inputs aligned to the canonical token stream. While subword segmentations differ at the level of internal computation, producing distinct token sequences and activation patterns, they preserve the exact lexical identity and semantics of the text. This makes them a particularly attractive form of data augmentation as they alter the models internal representation without changing meaning.

Building on this observation, we propose training LLMs with both canonical deterministic tokenization and sampled non-canonical subword segmentations of the same input. We implement this using a lightweight auxiliary causal encoder that processes non-canonical subwords and injects their representations into the main decoder via block-causal cross-attention. Crucially, the standard next-token prediction objective and canonical token interface are preserved, avoiding ambiguity in the training signal. From a data augmentation perspective, these non-canonical subwords occupy a unique position. Unlike rephrasings or synonym replacements (Marivate and Sefara, 2020), which operate at the lexical or semantic level and often fail to preserve precise meaning (Lyons, 1995), subword augmentations act at a lower computational abstraction level while preserving exact meaning. They therefore satisfy a strict notion of label preservation appropriate for language model pretraining, while being computationally cheap and compatible with other augmentation or regularization techniques.

¹That is, different subword segmentations of the same underlying word that decode to the identical surface string.

Our contributions are as follows:

1. We revisit alternative valid subword segmentations as a form of string-preserving subword regularization, and study their use as auxiliary aligned context for causal language modeling. To accommodate this form of data augmentation, we propose a simple, modular architectural modification to the standard transformer that enables training LLMs with auxiliary subwords without altering the standard language modeling objective.
2. We empirically show that pretraining with auxiliary subwords delays overfitting and improves data efficiency by increasing variation in the training signal, while remaining compatible with other activation-level perturbation methods.
3. We establish that the effectiveness of our approach depends on tokenizer quality in multilingual fine-tuning scenarios: performance improvements are greater when the input text is less fragmented by the tokenizer.

2 Related Work

2.1 Impact of Tokenization in LLM Training

Subword tokenization (Shibata et al., 1999; Provilkov et al., 2020; Zouhar et al., 2023b,a) was developed to address the issue of out-of-vocabulary words when considering individual words as fundamental units of computation. Indeed, defining a fixed vocabulary size is bound to introduce out-of-vocabulary words, either due to misspellings, but also because the morphological derivational processes make the lexicon unbounded (Aronoff and Fudeman, 2022). Using raw bytes as input units is undesirable as it is computationally inefficient and removes necessary inductive biases that enable models to generalize (Rajaraman et al., 2024).

However, multiple studies have shown that subword tokenization impacts the multilingual performance of language models (Limisiewicz et al., 2023) and introduces unfairness across languages (Petrov et al., 2023) since most tokenizers are trained on English-centric corpora. Subsequently, Jabbar (2023) found that designing a linguistically informed tokenizer improves performance. The root cause of this effect is that language models

are blind to the character composition of their tokens (Shin and Kaneko, 2024; Cosma et al., 2025). This problem cannot be readily solved by simply scaling the training data (Cosma et al., 2025) or by training the tokenizer on more data (Goldman et al., 2024). For example, Reddy et al. (2025) found diminishing returns for increasing tokenizer training data, with a more pronounced effect on languages with extensive inflectional and derivational morphology (e.g., Finnish, Hungarian, and Romanian, *inter alia*).

Other approaches attempt to circumvent the need for tokenization by designing neural architectures that operate directly on bytes (Slagle, 2024; Ahia et al., 2024; Deiseroth et al., 2024; Nawrot et al., 2023; Pagnoni et al., 2025), but incur additional computational costs and technical debt (Pepe et al., 2024) by deviating from known and relatively understood architectural design choices of the modern Transformer (Vaswani et al., 2017).

We depart from these approaches and instead adopt a data-driven approach to increase the model’s perceptual range: the tokenizer is a given, and we encourage robustness to alternative tokenizations by exposing the model to non-canonical tokenizations of the same underlying text as a form of data augmentation.

2.2 Data Augmentation for Text

Data augmentation is widely used in other domains, such as Computer Vision (Yang et al., 2022b; Shorten and Khoshgoftaar, 2019), enabling much longer training runs, up to hundreds of epochs, especially when the dataset size is limited (Geiping et al., 2022; Villalobos et al., 2022). Data augmentation induces data-driven invariances in the model (Geiping et al., 2022; Balestrieri et al., 2022; Chen et al., 2020) and increases robustness to OOD data (Geiping et al., 2022). Most such augmentations for Computer Vision are cheaply generated using programmatic operations (e.g., random flipping, cropping, and color jitter). However, cheap data augmentations for text data have remained elusive. Chai et al. (2025) surveyed approaches to text data augmentation for large language models, but most require substantial computational overhead. For example, the Kimi-K2 (Team et al., 2025) model employed large-scale rephrasing with another pretrained model during the construction

of the pretraining dataset to increase the count of high-quality tokens.

In general, effective text augmentations have been limited to back-translations or synonym replacements (Marivate and Sefara, 2020). It is unclear, however, if such augmentations preserve the precise meaning and syntax of the original text – augmentations should be “label preserving” (Taylor and Nitschke, 2018). Nevertheless, in the case of language modeling, augmentation should be “meaning preserving”. Back-translations are bottlenecked by translation model performance and lexical-semantic gaps between languages (Pimentel et al., 2020). Furthermore, there are a few exact synonyms across languages (Edmonds and Hirst, 2002; Lyons, 1995; Quine, 2013), suggesting that augmentations that rely on synonym replacements subtly change the meaning of the original text.

Prior work has explored alternative subword segmentations through subword regularization (Kudo, 2018), BPE-dropout (Chai et al., 2024), and MaxMatch-style dropout (Hiraoka, 2022). These methods are most relevant to our setting because they modify the token sequence while preserving the decoded string. Our work differs not in observing that alternative segmentations exist, but in how they are used: rather than replacing the model input with a sampled segmentation, we keep the canonical causal LM stream and use non-canonical segmentations as aligned auxiliary context. This preserves the standard next-token label space and avoids redefining the language-modeling objective.

Further, the work from Prabhudesai et al. (2025) suggested that the training process of text diffusion models acts as a form of implicit data augmentation, which alleviates the diminishing returns of repeating data and enables longer training (Muenighoff et al., 2025).

3 Method

3.1 Augmentations across Levels of Linguistic Abstraction

As discussed in Section 2.2, augmentations in supervised tasks are perturbations of the input that preserve class labels. In Computer Vision, it is clear that pixel transformations (unless severe) do not change the class label. However, each linguistic text unit serves a definite purpose in communicat-

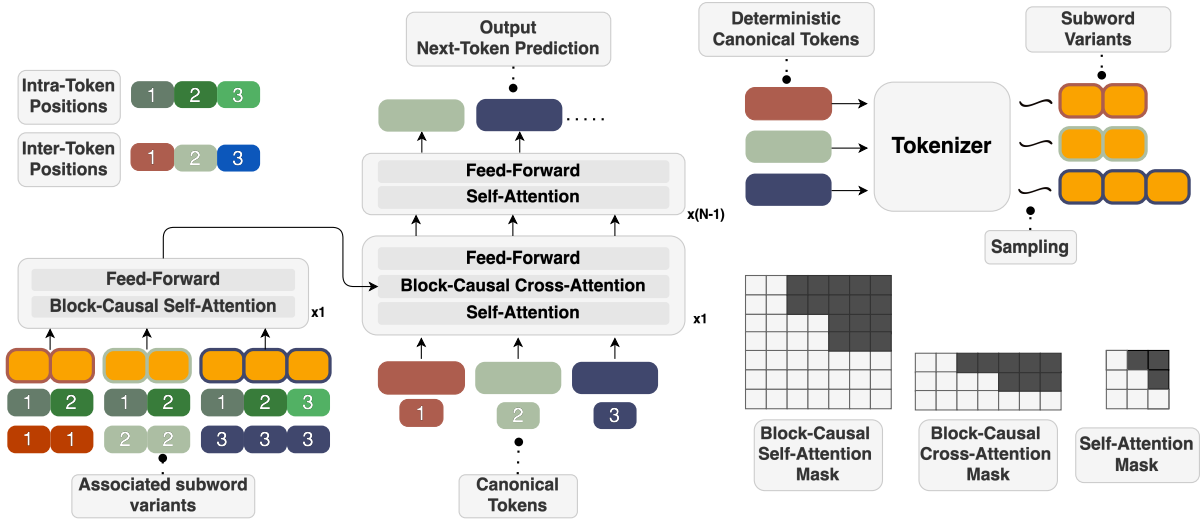


Figure 1: Diagram of our architecture. The main causal decoder operates on canonical BPE tokens, while a lightweight single-block causal encoder consumes sampled subword variants (e.g., dinosaur $\rightarrow$ dino + saur | d + inosaur) and injects them into the main trunk via a block-causal cross-attention operation.

ing intent and in defining semantics, calling into question the efficacy of current augmentation techniques. Intuitively, augmentations that have the highest impact on the model should be at higher levels of abstractions, to make the model invariant to surface forms. However, altering surface forms at a higher level of abstraction than words does change the underlying meaning of the text. In language model pretraining, supervision is implicit, and, in this case, augmentations should precisely preserve the meaning of the underlying unit, even if the surface form structure changes.

To formalize this intuition, we consider the different levels of linguistic abstractions (Jurafsky and Martin, 2025). Let $\mathcal{X}_0 \subset \mathcal{X}_1 \subset \dots \subset \mathcal{X}_L$ denote progressively higher linguistic levels (e.g., morphology $\rightarrow$ lexicality $\rightarrow$ semantics). Each level ℓ has an associated projection map $\pi_{\ell+1} : \mathcal{X}_\ell \rightarrow \mathcal{X}_{\ell+1}$ that identifies the linguistic unit at the next abstraction level.

Definition 3.1 (Text Augmentation). An augmentation at level ℓ is a transformation $a : \mathcal{X}_\ell \rightarrow \mathcal{X}_\ell$ that preserves the *supervisory signal* defined at level $\ell + 1$. Formally, if $x \in \mathcal{X}_\ell$, then $\pi_{\ell+1}(x) = \pi_{\ell+1}(a(x))$.

Applying the augmentation does not change the representation at the higher level. Given the limitations of existing methods, we claim that current text augmentations that operate on levels beyond the lexical level, such as rephrasings and synonym

replacements (Marivate and Sefara, 2020), violate Definition 3.1. To further support our claim from a linguistic perspective, we consider the classical distinctions in lexical semantics, namely that the underlying assumption of synonym replacements is absolute synonymy, which is known to be exceedingly rare except in technical registers (Lyons, 1995); in a given context, each word slightly changes the meaning and intent of the text. Furthermore, in practice, low-resource languages often lack comprehensive lexical databases (e.g., WordNet (Miller, 1994)) and exhibit poor back-translation quality, influencing downstream tasks (Agrawal et al., 2024).

3.2 Subword Variants as Meaning-Preserving Text Augmentations

In practice, language models use subword tokenization (Shibata et al., 1999), which introduces a computational layer between the orthographic surface and the lexical level. A single word w may admit multiple valid segmentations under a stochastic tokenizer. These segmentations differ in their subword decomposition but correspond to the same lexical item and the same meaning. This motivates the following definitions.

Definition 3.2 (Canonical and Non-Canonical Tokenizations). Let $w \in \Sigma^*$ be a string span. Let $\mathcal{T} : \Sigma^* \rightarrow V^*$ denote the deterministic tokenization procedure associated with a fixed pretrained

tokenizer, and $\hat{\mathcal{T}}$ a stochastic version of the same tokenizer which randomly splits the word into several tokens from the token vocabulary V . Then:

- The canonical tokenization is the deterministic mapping $x = \mathcal{T}(w)$ such that $\mathcal{T}^{-1}(x) = w$.
- A non-canonical tokenization is any $s = \hat{\mathcal{T}}(w)$, consisting of tokens $(s_1, \dots, s_k) \in V$ such that $\mathcal{T}^{-1}(s) = w$.

For BPE tokenizers, canonical tokenization is determined by the learned merge table; for WordPiece tokenizers (Devlin et al., 2019), it is commonly implemented through greedy longest-prefix matching. Our method does not rely on either property, only on the existence of a deterministic canonical tokenization and alternative valid segmentations that decode to the same string.

For example, the word "dinosaur", using a pre-trained GPT-2 tokenizer, is canonically tokenized as $\{d_{67}, inosaur_{21317}\}$, whereas a non-canonical tokenization would be: $\{d_{67}, ino_{2879}, s_{82}, aur_{2899}\}$, which is decoded into the same string. All tokens are part of the same tokenizer vocabulary.

Definition 3.3 (Subword Variants). Two sequences s and s' are said to be *subword variants* if they arise from different valid segmentations of the same word. That is, $\mathcal{T}^{-1}(s) = \mathcal{T}^{-1}(s')$.

Corollary 3.1. Stochastic tokenizations are a form of text augmentation.

Proof. Considering $\pi_{\ell+1} = \mathcal{T}^{-1}$, the transformation from the token level to the lexical level $\mathcal{X}_{\text{tok}} \rightarrow \mathcal{X}_{\text{lex}}$, from Definitions 3.1, 3.2 and 3.3 it is clear that $\hat{\mathcal{T}}$ is an augmentation. $\square$

Although non-canonical tokenizations modify the internal computational representation (changing the token sequence seen by the model), they preserve the *exact* word identity and thus, meaning of the text. Because tokenization determines the discrete units processed by an LLM, two segmentations of the same word produce different activation patterns, even though they correspond to the same lexical item and the same semantics.

3.3 Training with subword variants

Given a large corpus $\mathcal{D}$, the next-token-prediction modelling objective ubiquitously used in language

models implies sampling a random text $x \sim \mathcal{D}$ and applying the canonical tokenizer to obtain $s = \mathcal{T}(x) = (s_1, s_2, \dots, s_K)$, the individual canonical tokens of x , having the sequence length K . The objective is to model $P(s_{t+1}|s_1, s_2, \dots, s_t)$, $t < K$.

Consider the subword variants of each canonical token: $\hat{s} = (\hat{\mathcal{T}}(s_1), \dots, \hat{\mathcal{T}}(s_K))$, with $\hat{\mathcal{T}}(s_i) = (\hat{s}_{i1}, \dots, \hat{s}_{iL_i})$, having the larger context size $K \leq K' = \sum_{i=1}^K L_i$. Training with subword variants implies that the objective is to model $P(s_{t+1}|s_1, \dots, s_t; \hat{s}_1, \dots, \hat{s}_{tL_t})$. That is, estimating the probability of the next canonical token, while accounting for all previous canonical tokens and their sampled subword variants. We further describe the architectural details that best accommodate this setup.

Causal encoder and cross-attention. We decided to adopt an encoder-decoder style design (Cosma et al., 2025; Provilkov et al., 2020) tailored to causal language modeling, as shown in Figure 1. In Appendix A.1 we briefly discuss alternative designs that we considered, but discarded. The main trunk is a standard causal decoder operating on canonical tokens $(s_1, \dots, s_K)$. In parallel, a significantly smaller *causal encoder* branch processes the subword sequence $\hat{s}$. The augmentation branch consists of a single Transformer block, which encodes $\hat{s}$ under a block causal mask M . This causal mask is constructed as follows:

$$M = \begin{pmatrix} \mathbf{1}_{L_1 \times L_1} & 0 & \cdots & 0 \\ \mathbf{1}_{L_2 \times L_1} & \mathbf{1}_{L_2 \times L_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{1}_{L_K \times L_1} & \mathbf{1}_{L_K \times L_2} & \cdots & \mathbf{1}_{L_K \times L_K} \end{pmatrix}, \quad (1)$$

where K is the number of canonical tokens and each block $\mathbf{1}_{L_i \times L_j}$ denotes an all-ones matrix of shape $L_i \times L_j$, with L_i being the length of the subword variant i .

This mask ensures that all subtokens from a token s_t can attend only to subtokens of tokens $1, \dots, t$ (including themselves), preserving causality at the canonical token level.

The encoded subword variants representations are injected into the first block of the main trunk via cross-attention using a block-causal cross-attention

mask M_{cross} . This mask is constructed as:

$$M_{\text{cross}} = \begin{pmatrix} \mathbf{1}_{L_1} & 0 & \cdots & 0 \\ \mathbf{1}_{L_1} & \mathbf{1}_{L_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{1}_{L_1} & \mathbf{1}_{L_2} & \cdots & \mathbf{1}_{L_K} \end{pmatrix}, \quad (2)$$

where $\mathbf{1}_{L_i}$ is a shorthand for $\mathbf{1}_{1 \times L_i}$. Row t specifies which subtokens are visible to the canonical token s_t during cross-attention.

This yields a word-level causal structure: s_t can attend to the subtokens of tokens $1, \dots, t$, but not to those of future tokens. The tokenizer typically splits the canonical tokens into single characters/bytes; having the subtoken sequence consisting mostly of single characters increases the sequence length, which affects computational speed, but also prevents proper training of longer subtokens that are more likely to be canonical tokens in other pieces of text. Consequently, subword variants are constructed by randomly sampling from the top 5 longest subtokens by character count, excluding the canonical token, thereby balancing granularity and computational cost. If the canonical token cannot be subdivided, we use it as such. At inference time, we use the same procedure.

Since attention is permutation-invariant over its input set (Vaswani et al., 2017; Lee et al., 2019), we explicitly encode token order using learned positional embeddings: order within each subword variant sequence is encoded using *intra-token position embeddings*, and the correspondence between canonical tokens s_i and their subword variant sequences $(\hat{s}_{ij})_{j \leq L_i}$ is encoded using *inter-token position embeddings*. This design is inspired by Abacus position embeddings (McLeish et al., 2024) that explicitly encode the position of individual digits inside a number. RoPE (Su et al., 2024) cannot be used without some modifications for the subword variant branch, since it will induce a mismatch between subword variants positions and canonical token positions. This problem has been addressed, for example, by considering fractional position embeddings (Harzig et al., 2022; Gimeno-Gómez et al., 2024) that align the sampling rates of videos and associated audio. For a fair comparison, both the standard transformer used in our experiments and our variant use learned position embeddings, similar to GPT-2 (Radford et al., 2019).

Note that such subword variants augmentations are *stack-compatible* with other data- or model-level augmentations. Rephrasing, dropout, or other regularization schemes operate orthogonally to our subword variant branch; they can be applied in combination with our method without changing the model interface or loss definition.

This design is suggestive of the cross-resolution fusion in Computer Vision architectures such as CrossViT (Chen et al., 2021b), where high-resolution features enrich coarse representations. This high-level design pattern has been used in NLP (He et al., 2024; Chalkidis et al., 2022; Wu et al., 2021; Cosma et al., 2025), but it is far more common in Computer Vision (Wang et al., 2021; Liu et al., 2021; Chu et al., 2021; Xu et al., 2021; Wang et al., 2023; Chen et al., 2021a). Furthermore, this can also be seen as an attention adapter (Han et al., 2024), which can presumably be used to retrofit existing pretrained models to enhance their sub-token perception capabilities.

4 Experimental Setup

We compared our architectural variant with a standard transformer decoder language model in data-constrained scenarios. Details about the training hyperparameters and evaluation protocol can be found in Appendices A.2 and 4.1.

Pretraining configuration. For pretraining, we used a subset of FineWeb-Edu (Penedo et al., 2024). We follow the setup by Muennighoff et al. (2025) to evaluate our model in data-constrained settings during pretraining. We increased the number of dataset repetitions R while simultaneously sampling $1/R$ of the dataset to maintain Chinchilla-optimality (Hoffmann et al., 2022) in terms of compute: the dataset size is approximately 20 tokens per model parameter, regardless of model size and number of repetitions. In particular, we train on 4B tokens for $\mu = 1$, and we train on 16B tokens for $\mu = 2$. Unless specified, we used the GPT-2 (Radford et al., 2019) tokenizer². Since μP only addresses model size and not token horizon, we also adjust the learning rate according to the dataset size (Bjorck et al., 2024): $lr_\mu = lr_{\mu=1} \times (N_{\text{params}}(\mu)/N_{\text{params}}(\mu=1))^{-0.32}$.

²hf.co/openai-community/gpt2, Accessed: 5 January 2026

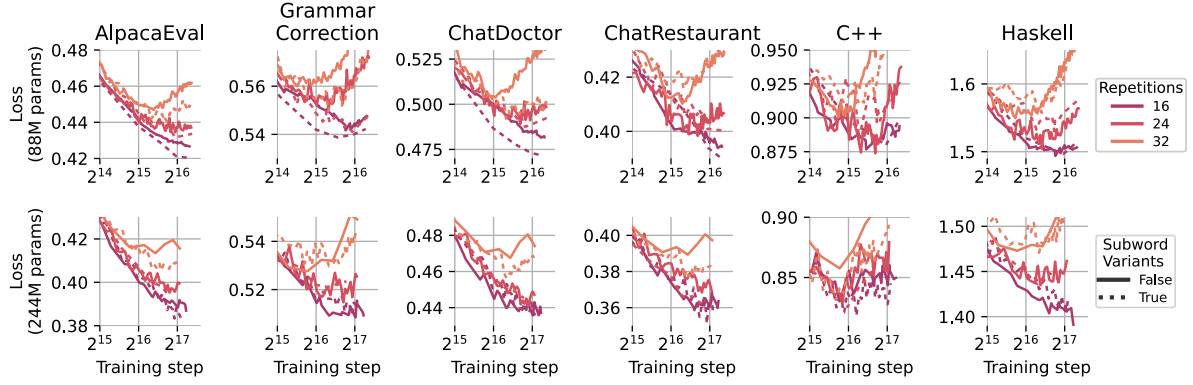


Figure 2: Evaluations on various chat datasets across pretraining duration for two model sizes: $\mu = 1$ - **top** and $\mu = 2$ - **bottom**. We show training runs with the number of data repetitions $R \in \{16, 24, 32, 64\}$ for clarity. Training with subword variants consistently delays overfitting in these scenarios.

Activation-level Perturbations. To compare with alternative activation-level augmentations, we pre-trained separate models using attention dropout at 10% and 20% and Gaussian noise injected at the token level with a standard deviation of 0.1. We used these perturbations for both the vanilla transformer and our variant for $R = 32$ and $\mu = 1$ using the GPT-2 tokenizer.

Multilingual Fine-tuning Configuration. A situation in which repeating data is common is instruction fine-tuning, and especially so in multilingual settings and when dealing with low-resource languages. Consequently, we pretrained each model variant on FineWeb-Edu using the GPT-2 tokenizer (vocabulary size $\sim 50k$) and the aya-23 multilingual tokenizer (vocabulary size $\sim 255k$) (Aryabumi et al., 2024) with one repetition of data and then finetuned for 32 epochs on a subset of 14 languages from the AYA dataset (Singh et al., 2024), containing instructions in multiple languages.

4.1 Evaluation Protocol.

During pretraining, test loss is computed on a fixed held-out subset of FineWeb-Edu. We also evaluated on several instruction datasets from heterogeneous domains, targeting general knowledge through Alpaca Instructions (Taori et al., 2023), lexical understanding through Grammar Correction³, code understanding through Haskell

³[hf.co/datasets/agentlans/grammar-correction](https://huggingface.co/datasets/agentlans/grammar-correction), Accessed: 5 January 2026

problems⁴, and C++ problems⁵ and two domain-specific chat datasets: ChatDoctor (Li et al., 2023) and ChatRestaurant⁶. Since the models are too small to generate meaningful text, we computed the average next-token prediction across concatenated question-answer pairs for each dataset.

5 Results

Pretraining Behaviour. In Figure 4 we show evaluation curves for runs trained using different amounts of data repetitions, in a similar setup to Muennighoff et al. (2025). We observe that when the data is repeated more than 16 times, training with subword variants offers clear advantages: the overfitting point is later in training, and the best loss is lower than that of the standard Transformer. For $R < 16$, training with subword variants does not affect final performance. For $\mu = 1$, the addition of subword variants seems somewhat detrimental at lower amounts of repetitions. Still, this effect is not held for $\mu = 2$, where both models have essentially the same evaluation curves for $R < 24$.

In Figure 2, we show the evolution of cross-domain LM evaluation losses during pretraining across several instruction-following and domain-

⁴[hf.co/datasets/finbarr/r1vr-code-data-haskell-edited](https://huggingface.co/datasets/finbarr/r1vr-code-data-haskell-edited), Accessed: 5 January 2026

⁵[hf.co/datasets/ReySajju742/synthetic-cpp](https://huggingface.co/datasets/ReySajju742/synthetic-cpp), Accessed: 5 January 2026

⁶[hf.co/datasets/tctsung/chat_restaurant_recommendation](https://huggingface.co/datasets/tctsung/chat_restaurant_recommendation), Accessed: 5 January 2026

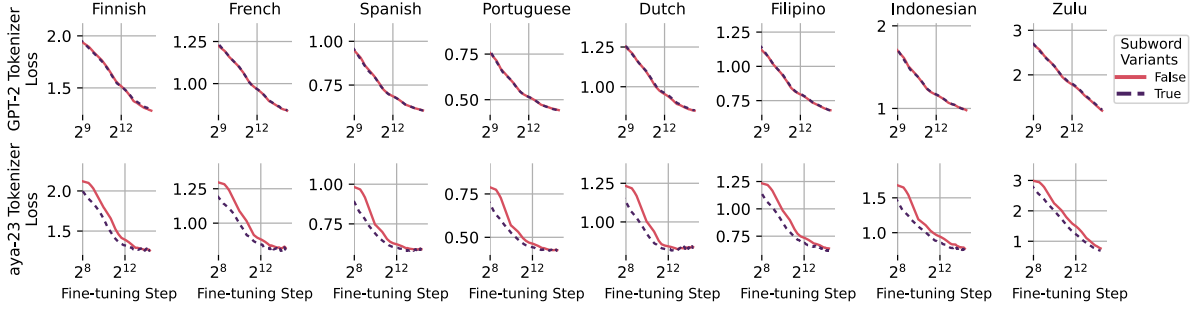


Figure 3: Performance across multilingual fine-tuning for $\mu = 1$ across 32 epochs. The model pretrained with the GPT-2 tokenizer (**top**) does not improve with auxiliary subword variants, whereas the variant trained with *aya-23* tokenizer (**bottom**) and subword variants consistently outperforms the standard Transformer decoder when repeating data.

specific chat datasets, for two model sizes and varying numbers of data repetitions. Across most benchmarks with sufficiently high repetition factors ($R \geq 16$), models trained with subword variants exhibit delayed overfitting compared to the standard Transformer baseline. While both models initially improve at a similar rate, the models augmented with subword variants maintain lower evaluation loss for a more extended period of training, with the gap becoming more pronounced as repetition increases. This effect is observed across heterogeneous domains, suggesting that the benefit is not task-specific but instead reflects a more general regularization effect. These results are consistent with the hypothesis that exposing the model to multiple meaning-preserving tokenizations enables it to continue extracting useful signal from repeated data instead of overfitting to a single canonical segmentation.

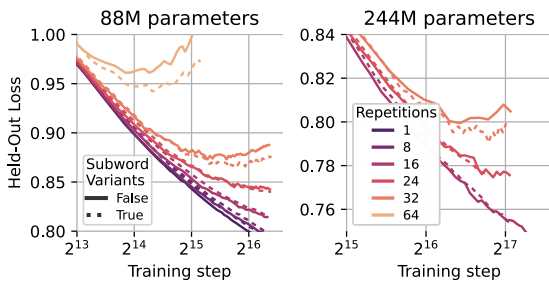


Figure 4: Pretraining curves for various numbers of data repetitions for an 88M parameter model ($\mu = 1$, **left**) and a 244M parameter model ($\mu = 2$, **right**). Training with subword variants delays overfitting for $R > 16$.

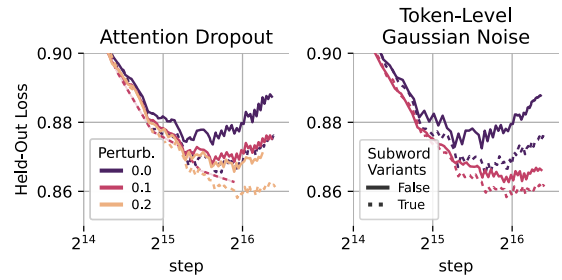


Figure 5: Loss during pretraining for runs using attention dropout and token-level Gaussian noise as augmentations for $R = 32$ and $\mu = 1$.

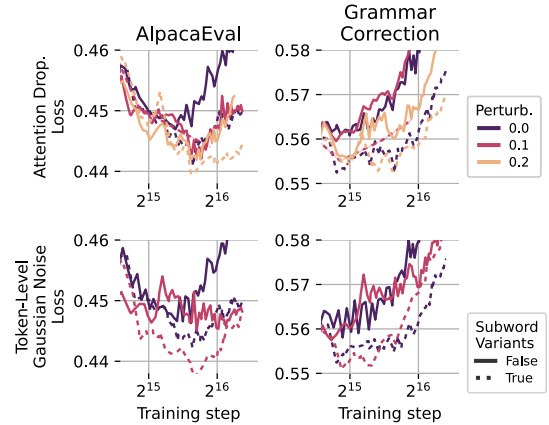


Figure 6: Loss across two benchmarks for runs using attention dropout (**top**) and token-level Gaussian noise (**bottom**) as augmentations for $R = 32$ and $\mu = 1$.

Comparison with Activation-level Perturbations. In Figure 5, we show the pretraining loss comparing our model with the standard Transformer under activation-level perturbations. Since

our architecture is stack-compatible with other augmentations, we can also apply these perturbations to it, further increasing the amount of data augmentation. In both cases, training with subword variants markedly improves performance. In Figure 6, we show performance on the AlpacaEval and GrammarCorrection datasets across training; training with auxiliary subword variants benefits model generalization beyond simple noise injection.

Multilingual Fine-tuning. In Figure 3, we show the performance across multilingual instruction fine-tuning for models using the GPT-2 and aya-23 tokenizer. These results highlight the fact that training with subword variants is not a free lunch in all situations. We hypothesize that the greatest benefits of our approach stem from tokenizers that heavily compress the input sequence, yielding large canonical tokens. This is because if a tokenizer is poorly trained or used in out-of-distribution settings, the resulting canonical tokens are already fragmented and cannot be divided further, making our causal encoder to essentially process the same sequence as the main trunk. In our case, training with auxiliary subword variants and with the GPT-2 tokenizer, which is geared towards English, does not benefit multilinguality. In contrast, using the aya-23 tokenizer, which is specifically designed to handle multiple languages, shows definite benefits in this setup, obtaining better performance across training (see A.3, Table 1 - for example, Finnish and Zulu show substantially higher tokens/word ratios). This shows that our method should not be used naively in multilingual settings, expecting improvements without addressing the underlying over-fragmentation issues with the tokenizer. Our method is most appropriately used to introduce more variation in the training data and is synergistic with current trends in scaling tokenizer vocabulary size (Huang et al., 2025).

6 Conclusions

We studied the effect of auxiliary subword regularization under repeated-data training. By exposing language models to multiple valid segmentations of the same lexical items, without altering the canonical next-token prediction objective, we provide evidence that aligned non-canonical segmentations can regularize repeated-data training;

whether this corresponds to representation-level tokenization invariance remains an open mechanistic question. Across data-constrained pretraining regimes, this approach consistently delays overfitting and reduces held-out next-token loss across evaluation datasets, particularly under repeated data exposure, while remaining compatible with standard architectures and existing regularization techniques. Our results suggest that tokenization should not be treated as a fixed preprocessing artifact, but as a source of structured variation that can be leveraged during training. More broadly, leveraging subword variants offers a practical path toward more robust and data-efficient language model training without abandoning established tokenization pipelines.

Limitations

As this study is intended as a proof of concept, its primary limitation is its limited scale due to hardware constraints. All experiments are conducted using relatively small language models and data-constrained training regimes. While this setup is well-suited for isolating overfitting behavior, we cannot claim that augmentations based on subword variants would produce comparable gains in large-scale ($> 1B$ parameter models) pretraining settings. Our reported evaluations use the auxiliary segmentation branch at both training and evaluation time. Thus, the method should be understood as auxiliary tokenization conditioning rather than train-only data augmentation. We do not claim that the main decoder fully internalizes the augmentation when the auxiliary branch is removed; evaluating train-only use is left for future work. The present contribution shows evidence for the value of aligned auxiliary segmentations in repeated-data training.

Acknowledgments

This work was supported by UBS Switzerland AG and its affiliates, and by the project Romanian Hub for Artificial Intelligence - HRIA, Smart Growth, Digitization and Financial Instruments Program, 2021-2027, MySMIS no. 351416.

References

- Ashish Agrawal, Barah Fazili, and Preethi Jyothi. 2024. Translation errors significantly impact low-resource languages in cross-lingual learning. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 319–329.
- Orevaoghene Ahia, Sachin Kumar, Hila Gonen, Valentin Hofmann, Tomasz Limisiewicz, Yulia Tsvetkov, and Noah A Smith. 2024. Magnet: Improving the multilingual fairness of language models with adaptive gradient-based tokenization. *Advances in Neural Information Processing Systems*, 37:47790–47814.
- Mark Aronoff and Kirsten Fudeman. 2022. *What is morphology?* John Wiley & Sons.
- Viraat Aryabumi, John Dang, Dwarak Talupuru, Saurabh Dash, David Cairuz, Hangyu Lin, Bharat Venkitesh, Madeline Smith, Kelly Marchisio, Sebastian Ruder, Acyr Locatelli, Julia Kreutzer, Nick Frosst, Phil Blunsom, Marzieh Fadaee, Ahmet Üstün, and Sara Hooker. 2024. [Aya 23: Open weight releases to further multilingual progress](#). *Preprint*, arXiv:2405.15032.
- Randall Balestrieri, Ishan Misra, and Yann LeCun. 2022. [A data-augmentation is worth a thousand samples: Analytical moments and sampling-free training](#). In *Advances in Neural Information Processing Systems*, volume 35, pages 19631–19644. Curran Associates, Inc.
- Johan Bjorck, Alon Benhaim, Vishrav Chaudhary, Furu Wei, and Xia Song. 2024. Scaling optimal lr across token horizons. *arXiv preprint arXiv:2409.19913*.
- Kaj Bostrom and Greg Durrett. 2020. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*.
- Yaping Chai, Haoran Xie, and Joe S Qin. 2025. Text data augmentation for large language models: A comprehensive survey of methods, challenges, and opportunities. *arXiv preprint arXiv:2501.18845*.
- Yekun Chai, Yewei Fang, Qiwei Peng, and Xuhong Li. 2024. [Tokenization falling short: On subword robustness in large language models](#). In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 1582–1599, Miami, Florida, USA. Association for Computational Linguistics.
- Ilias Chalkidis, Xiang Dai, Manos Fergadiotis, Prodromos Malakasiotis, and Desmond Elliott. 2022. An exploration of hierarchical attention transformers for efficient long document classification. *arXiv preprint arXiv:2210.05529*.
- Chun-Fu Chen, Rameswar Panda, and Quanfu Fan. 2021a. Regionvit: Regional-to-local attention for vision transformers. *arXiv preprint arXiv:2106.02689*.
- Chun-Fu Richard Chen, Quanfu Fan, and Rameswar Panda. 2021b. Crossvit: Cross-attention multi-scale vision transformer for image classification. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 357–366.
- Shuxiao Chen, Edgar Dobriban, and Jane H Lee. 2020. A group-theoretic framework for data augmentation. *Journal of Machine Learning Research*, 21(245):1–71.
- Xiangxiang Chu, Zhi Tian, Yuqing Wang, Bo Zhang, Haibing Ren, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. 2021. Twins: Revisiting the design of spatial attention in vision transformers. *Advances in neural information processing systems*, 34:9355–9366.
- Adrian Cosma, Stefan Ruseti, Emilian Radoi, and Mihai Dascalu. 2025. [The strawberry problem: Emergence of character-level understanding in tokenized language models](#). In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 28240–28251, Suzhou, China. Association for Computational Linguistics.
- Björn Deiseroth, Manuel Brack, Patrick Schramowski, Kristian Kersting, and Samuel Weinbach. 2024. [T-FREE: Subword tokenizer-free generative LLMs via sparse representations for memory-efficient embeddings](#). In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21829–21851, Miami, Florida, USA. Association for Computational Linguistics.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.
- Philip Edmonds and Graeme Hirst. 2002. Near-synonymy and lexical choice. *Computational linguistics*, 28(2):105–144.
- Jonas Geiping, Micah Goldblum, Gowthami Somepalli, Ravid Shwartz-Ziv, Tom Goldstein, and Andrew Gordon Wilson. 2022. How much data are augmentations worth? an investigation into scaling laws, invariance, and implicit regularization. *arXiv preprint arXiv:2210.06441*.
- David Gimeno-Gómez, Ana-Maria Bucur, Adrian Cosma, Carlos-David Martínez-Hinarejos, and Paolo Rosso. 2024. Reading between the frames: Multimodal depression detection in videos from non-verbal cues. In *Advances in Information Retrieval*, pages 191–209, Cham. Springer Nature Switzerland.

- Omer Goldman, Avi Caciularu, Matan Eyal, Kris Cao, Idan Szpektor, and Reut Tsarfaty. 2024. Unpacking tokenization: Evaluating text compression and its correlation with model performance. *arXiv preprint arXiv:2403.06265*.
- Zeyu Han, Chao Gao, Jinyang Liu, Jeff Zhang, and Sai Qian Zhang. 2024. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*.
- Philipp Harzig, Moritz Einfalt, and Rainer Lienhart. 2022. Synchronized audio-visual frames with fractional positional encoding for transformers in video-to-text translation. In *2022 IEEE International Conference on Image Processing (ICIP)*, pages 2041–2045. IEEE.
- David A. Haslett. 2025. [Tokenization changes meaning in large language models: Evidence from chinese](#). *Computational Linguistics*, 51(3):785–814.
- Haoyu He, Markus Flicke, Jan Buchmann, Iryna Gurevych, and Andreas Geiger. 2024. [HDT: Hierarchical document transformer](#). In *First Conference on Language Modeling*.
- Tatsuya Hiraoka. 2022. Maxmatch-dropout: Subword regularization for wordpiece. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 4864–4872.
- Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.
- Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Ya Wang, Qiyang Min, and Xun Zhou. 2025. [Over-tokenized transformer: Vocabulary is generally worth scaling](#). *Preprint*, arXiv:2501.16975.
- Haris Jabbar. 2023. Morphpiece: A linguistic tokenizer for large language models. *arXiv preprint arXiv:2307.07262*.
- Daniel Jurafsky and James H. Martin. 2025. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, with Language Models*, 3rd edition. Online manuscript released August 24, 2025.
- Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.
- Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. *arXiv preprint arXiv:1804.10959*.
- Taku Kudo and John Richardson. 2018. [SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing](#). In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.
- Juho Lee, Yoonho Lee, Jungtaek Kim, Adam Kosiosek, Seungjin Choi, and Yee Whye Teh. 2019. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International conference on machine learning*, pages 3744–3753. PMLR.
- Yunxiang Li, Zihan Li, Kai Zhang, Ruilong Dan, Steve Jiang, and You Zhang. 2023. Chatdoctor: A medical chat model fine-tuned on a large language model meta-ai (llama) using medical domain knowledge. *Cureus*, 15(6).
- Tomasz Limisiewicz, Jiří Balhar, and David Mareček. 2023. Tokenization impacts multilingual language modeling: Assessing vocabulary allocation and overlap across languages. *arXiv preprint arXiv:2305.17179*.
- Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022.
- Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.
- John Lyons. 1995. *Linguistic Semantics: An Introduction*. Cambridge University Press.
- Vukosi Marivate and Tshephisho Sefara. 2020. Improving short text classification through global augmentation methods. In *International Cross-Domain Conference for Machine Learning and Knowledge Extraction*, pages 385–399. Springer.
- Sean Michael McLeish, Arpit Bansal, Alex Stein, Neel Jain, John Kirchenbauer, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and 1 others. 2024. [Transformers can do arithmetic with the right embeddings](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- George A. Miller. 1994. [WordNet: A lexical database for English](#). In *Human Language Technology: Proceedings of a Workshop held at Plainsboro, New Jersey, March 8-11, 1994*.
- Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. 2025. [Scaling data-constrained language models](#). *Preprint*, arXiv:2305.16264.

- Piotr Nawrot, Jan Chorowski, Adrian Lancucki, and Edoardo Maria Ponti. 2023. Efficient transformers with dynamic token pooling. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6403–6417.
- Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, and 1 others. 2025. Byte latent transformer: Patches scale better than tokens. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9238–9258.
- Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin A Raffel, Leandro Von Werra, Thomas Wolf, and 1 others. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *Advances in Neural Information Processing Systems*, 37:30811–30849.
- Federica Pepe, Fiorella Zampetti, Antonio Mastropaolo, Gabriele Bavota, and Massimiliano Di Penta. 2024. A taxonomy of self-admitted technical debt in deep learning systems. In *2024 IEEE International Conference on Software Maintenance and Evolution (IC-SME)*, pages 388–399. IEEE.
- Aleksandar Petrov, Emanuele La Malfa, Philip Torr, and Adel Bibi. 2023. [Language model tokenizers introduce unfairness between languages](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 36963–36990. Curran Associates, Inc.
- Tiago Pimentel, Rowan Hall Maudslay, Damian Blasi, and Ryan Cotterell. 2020. Speakers fill lexical semantic gaps with context. *arXiv preprint arXiv:2010.02172*.
- Mihir Prabhudesai, Mengning Wu, Amir Zadeh, Kateřina Fragkiadaki, and Deepak Pathak. 2025. Diffusion beats autoregressive in data-constrained settings. *arXiv preprint arXiv:2507.15857*.
- Ivan Provilkov, Dmitrii Emelianenko, and Elena Voita. 2020. Bpe-dropout: Simple and effective subword regularization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1882–1892.
- Willard Van Orman Quine. 2013. *Word and object*. MIT press.
- Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, and 1 others. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.
- Nived Rajaraman, Jiantao Jiao, and Kannan Ramchandran. 2024. Toward a theory of tokenization in llms. *arXiv preprint arXiv:2404.08335*.
- Varshini Reddy, Craig W Schmidt, Yuval Pinter, and Chris Tanner. 2025. How much is enough? the diminishing returns of tokenization training data. *arXiv preprint arXiv:2502.20273*.
- Noam Shazeer. 2020. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*.
- Yusuxke Shibata, Takuya Kida, Shuichi Fukamachi, Masayuki Takeda, Ayumi Shinohara, Takeshi Shinohara, and Setsuo Arikawa. 1999. Byte pair encoding: A text compression scheme that accelerates pattern matching.
- Andrew Shin and Kunitake Kaneko. 2024. [Large language models lack understanding of character composition of words](#). In *ICML 2024 Workshop on LLMs and Cognition*.
- Connor Shorten and Taghi M. Khoshgoufar. 2019. [A survey on image data augmentation for deep learning](#). *Journal of Big Data*, 6(1):60.
- Shivalika Singh, Freddie Vargus, Daniel Dsouza, Börje F Karlsson, Abinaya Mahendiran, Wei-Yin Ko, Herumb Shandilya, Jay Patel, Deividas Mataciunas, Laura OMahony, and 1 others. 2024. Aya dataset: An open-access collection for multilingual instruction tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11521–11567.
- Kevin Slagle. 2024. [Spacebyte: Towards deleting tokenization from large language modeling](#). In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.
- Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. 2024. [Roformer: Enhanced transformer with rotary position embedding](#). *Neurocomput.*, 568(C).
- Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.
- Luke Taylor and Geoff Nitschke. 2018. Improving deep learning with generic data augmentation. In *2018 IEEE symposium series on computational intelligence (SSCI)*, pages 1542–1547. IEEE.
- Kimi Team, Yifan Bai, Yiping Bao, Guanduo Chen, Jiahao Chen, Ningxin Chen, Ruijie Chen, Yanru Chen, Yuankun Chen, Yutian Chen, and 1 others. 2025. Kimi k2: Open agentic intelligence. *arXiv preprint arXiv:2507.20534*.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all

- you need. *Advances in neural information processing systems*, 30.
- Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. 2022. Will we run out of data? limits of llm scaling based on human-generated data. *arXiv preprint arXiv:2211.04325*.
- Dixuan Wang, Yanda Li, Junyuan Jiang, Zepeng Ding, Ziqin Luo, Guochao Jiang, Jiaqing Liang, and Deqing Yang. 2024. Tokenization matters! degrading large language models through challenging their tokenization. *arXiv preprint arXiv:2405.17067*.
- Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. 2021. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578.
- Wenxiao Wang, Wei Chen, Qibo Qiu, Long Chen, Boxi Wu, Binbin Lin, Xiaofei He, and Wei Liu. 2023. Crossformer++: A versatile vision transformer hinging on cross-scale attention. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 46(5):3123–3136.
- Philip Whittington, Gregor Bachmann, and Tiago Pimentel. 2025. Tokenisation is np-complete. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 28133–28153.
- Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2021. [Hi-transformer: Hierarchical interactive transformer for efficient and effective long document modeling](#). In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 848–853.
- Weijian Xu, Yifan Xu, Tyler Chang, and Zhuowen Tu. 2021. Co-scale conv-attentional image transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9981–9990.
- Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. 2022a. [Tensor programs v: Tuning large neural networks via zero-shot hyperparameter transfer](#). *Preprint*, arXiv:2203.03466.
- Suorong Yang, Weikang Xiao, Mengchen Zhang, Suhan Guo, Jian Zhao, and Furao Shen. 2022b. Image data augmentation for deep learning: A survey. *arXiv preprint arXiv:2204.08610*.
- Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Advances in neural information processing systems*, 32.
- Brian Siyuan Zheng, Alisa Liu, Orevaoghene Ahia, Jonathan Hayase, Yejin Choi, and Noah A. Smith. 2025. [Broken tokens? your language model can secretly handle non-canonical tokenizations](#). *Preprint*, arXiv:2506.19004.
- Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Mrinmaya Sachan, and Ryan Cotterell. 2023a. Tokenization and the noiseless channel. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5184–5207.
- Vilém Zouhar, Clara Meister, Juan Gastaldi, Li Du, Tim Vieira, Mrinmaya Sachan, and Ryan Cotterell. 2023b. A formal perspective on byte-pair encoding. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 598–614.

A Appendix

A.1 Discarded Alternative Designs

Several alternative ways of incorporating alternative tokenizations were considered and discarded due to practical or conceptual limitations.

Direct stochastic tokenization for next-token prediction. [Chai et al. \(2024\)](#) use BPE-dropout in encoderdecoder machine translation, where the source and target play asymmetric roles and there is no causal label conflict. In contrast, we operate in a pure language modeling setting with next-token prediction. If the input stream itself were tokenized stochastically at training time, the notion of the *correct* next token becomes under-specified: should the label be the canonical token, one of many valid stochastic segmentations, or some mixture? This would require redefining the loss and the evaluation protocol. We instead keep the canonical interface as the only label space and treat subword variants as auxiliary, unlabeled augmentations.

Training directly on subtokens only. One possibility is to abandon canonical tokens entirely and train on subtoken sequences alone. However, this again raises the question of what constitutes the target: should the model predict the next subtoken, the completion of a sampled segmentation, or a canonical segmentation reconstructed from subtokens? Each choice bakes in a particular inductive bias about the "true" tokenization and complicates both the objective and deployment. Our design avoids this by keeping the main model and its loss

completely standard (canonical next-token prediction) and using subword variants only as additional context.

Multi-task prediction of subtoken structure.

Another alternative is to add an auxiliary task that predicts the subword variants decomposition of each canonical token (e.g., predicting s_t from x_t). This introduces additional labels and multi-task optimization challenges: the model must simultaneously learn language modeling and tokenizer reconstruction, potentially leading to conflicting gradients and unclear trade-offs. Moreover, the label space for the auxiliary task is itself ambiguous whenever multiple segmentations are valid. In contrast, our cross-attention-based design only conditions on sampled subword variants; it does not predict them, and thus avoids auxiliary supervision.

Single-branch decoder with concatenated tokens and subtokens. One could concatenate canonical tokens and subword variants into a single long sequence and feed it to a larger causal decoder. This provides less control over how and where subword variants information enters the computation: every layer must process a longer sequence, increasing memory and compute without an explicit mechanism for aligning subtokens to their canonical parents. Our two-branch design isolates the augmentations in a small causal encoder and injects them only where needed, at a chosen depth in the main trunk.

Additive fusion of subword variants embeddings. A simpler fusion strategy would be to aggregate the subword variants embeddings belonging to token s_t (e.g., by averaging) and add them to the embedding of s_t . While cheap, this forces all subtokens to contribute uniformly and removes the model’s ability to focus on task-relevant segments or ignore noisy ones. Cross-attention, by contrast, lets the model learn task-dependent weights over the subword variants representations, deciding for each context how much to use or disregard each subtoken.

A.2 Hyperparameters

The standard transformer has 8 blocks with $d_{model} = 512 \times \mu$, parametrized by a width multiplier μ (Yang et al., 2022a), using SwiGLU

(Shazeer, 2020) and RMSNorm (Zhang and Sennrich, 2019). Our variant has a separate, one-block causal encoder, having a fixed $d_{model}^{enc} = 256$. We used a context size of 512 tokens for both models. We tested two model sizes, corresponding to $\mu \in \{1, 2\}$. The models have the number of parameters $N_{params}(\mu = 1) = 88M$ and $N_{params}(\mu = 2) = 244M$, when using the GPT-2 tokenizer vocabulary. The causal encoder has a fixed number of 1M parameters for all model sizes. Across experiments, we used AdamW (Kingma and Ba, 2014) optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.95$, $eps = 10^{-10}$, $wd = 0.01$, with a per-device batch size of 32, and a learning rate of 0.00001, annealed using a cosine decay scheduler (Loshchilov and Hutter, 2016). The learning rate was adjusted according to μP (Yang et al., 2022a). We trained all runs on 4 NVIDIA A100 GPUs. A training run lasted between 4 hours for the smallest model and 24 hours for the largest. No dropout or attention dropout was used during standard pretraining runs. **Parameter count.** For the encoder, we have a $d_{enc} \cdot \mu = d_{dec}$, with μ being the multiplicative factor for the model size. In practice, we use linear layers that function as adapters between the widths of the causal decoder and encoder. We have two such adapters: $L_{emb} : \mathbb{R}^{d_{dec}} \rightarrow \mathbb{R}^{d_{enc}}$ which translates the token embedding matrix into a smaller dimensionality, and $L_{aux} : \mathbb{R}^{d_{enc}} \rightarrow \mathbb{R}^{d_{dec}}$ which translates back the token embeddings to the dimensionality of the main trunk.

For example, under the usual Transformer parameter approximation $P_{enc}(d_{enc}) \approx c_{enc} d_{enc}^2$, the auxiliary branch scales as $P_{aux}(\mu) \approx (c_{enc} + 2\mu) d_{enc}^2$. Moving from $\mu = 1$ to $\mu = 2$ therefore increases the auxiliary branch from $(c_{enc} + 2) d_{enc}^2$ to $(c_{enc} + 4) d_{enc}^2$, rather than doubling the entire auxiliary module. Therefore, the total auxiliary branch grows from roughly 1.05M to 1.31M parameters between the two model scales. The auxiliary branch is thus not strictly parameter-constant, but most of its parameters come from the fixed-width encoder core; only the adapter projections scale with the decoder width.

A.3 Tokenizer statistics

In Table 1 we show statistics for the GPT-2 and AYA tokenizer across the datasets we used in this work.

Dataset	Lang.	Tokenizer	Canonical Seq. Len.	Aux Seq. Len.	Aux/Can Ratio	% Unsplit Tokens	Tok/Word	Tok/Byte
FineWeb-Edu		aya-23	145.89	303.64	2.08	17.13	1.27	0.21
		GPT-2	148.28	320.81	2.16	15.05	1.29	0.21
ChatDoctor		aya-23	86.58	178.73	2.06	18.00	1.34	0.22
		GPT-2	89.15	187.52	2.10	18.19	1.38	0.23
C++		aya-23	88.23	158.69	1.80	34.19	1.98	0.33
		GPT-2	93.07	167.09	1.80	35.68	2.09	0.35
Haskell		aya-23	184.80	357.61	1.94	25.05	1.58	0.26
		GPT-2	195.40	385.90	1.97	25.20	1.67	0.27
AlpacaEval		aya-23	76.20	150.84	1.98	24.22	1.31	0.22
		GPT-2	77.24	158.86	2.06	22.58	1.33	0.22
ChatRestaurant		aya-23	55.36	111.31	2.01	20.41	1.32	0.22
		GPT-2	57.89	120.20	2.08	19.54	1.39	0.23
AYA-23	eng	aya-23	100.28	202.97	2.02	21.48	1.36	0.22
		GPT-2	102.79	215.15	2.09	20.23	1.39	0.22
	fil	aya-23	116.98	236.59	2.02	19.56	1.91	0.32
		GPT-2	132.90	267.07	2.01	18.24	2.17	0.36
	fin	aya-23	189.36	389.46	2.06	15.22	3.32	0.37
		GPT-2	220.95	429.84	1.95	19.24	3.88	0.43
	fra	aya-23	117.14	239.94	2.05	21.25	1.53	0.24
		GPT-2	162.25	332.25	2.05	17.62	2.13	0.33
	ind	aya-23	123.70	255.06	2.06	21.82	1.68	0.23
		GPT-2	200.88	400.45	1.99	18.25	2.72	0.38
	nld	aya-23	114.60	233.00	2.03	22.84	1.60	0.26
		GPT-2	162.35	325.16	2.00	18.38	2.27	0.37
	nya	aya-23	170.20	342.98	2.02	18.89	2.64	0.37
		GPT-2	194.43	378.90	1.95	19.30	3.01	0.42
	plt	aya-23	99.70	196.91	1.98	22.20	2.48	0.36
		GPT-2	114.93	217.76	1.89	25.17	2.86	0.42
	por	aya-23	67.99	135.58	1.99	25.61	1.46	0.24
		GPT-2	101.33	200.03	1.97	22.00	2.18	0.35
	sna	aya-23	140.96	283.78	2.01	18.83	2.82	0.35
		GPT-2	161.57	314.33	1.95	20.61	3.24	0.40
	spa	aya-23	92.58	190.23	2.05	21.50	1.41	0.23
		GPT-2	136.95	277.62	2.03	19.43	2.09	0.34
	swe	aya-23	132.70	262.70	1.98	24.16	2.19	0.33
		GPT-2	149.76	299.15	2.00	17.58	2.47	0.37
	zsm	aya-23	94.53	196.88	2.08	19.47	1.73	0.24
		GPT-2	149.87	298.19	1.99	18.22	2.74	0.39
	zul	aya-23	234.76	470.74	2.01	18.50	3.33	0.39
		GPT-2	273.16	511.51	1.87	26.14	3.87	0.45

Table 1: Descriptive statistics for canonical (Can.) and auxiliary (Aux.) tokenizations across datasets, languages, and tokenizers. These statistics quantify both the additional sequence-level overhead introduced by the auxiliary branch and the amount of tokenization variation available to the method in each setting.