HEAD-TAIL SAMPLED SOFTMAX FOR SCALING VOCABULARIES IN LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Training a language model with softmax cross-entropy scores every vocabulary token at every position, though the loss needs them only for one normalizing sum. On a compact backbone with a large vocabulary, the output layer takes close to a third of a step's FLOPs, 28.9% in Qwen3.5-0.8B. Sampled softmax estimates that sum from a sample, but matches cross-entropy only if each draw is weighted by the inverse of how often it is drawn. Earlier language models trained with in-batch negatives are a sampled softmax without that weight, their scores deflated by token frequency, largely explaining why their direct readout reached 5× cross-entropy's perplexity. We propose *head-tail sampled softmax* (HTSS), which sums the K most frequent tokens exactly, where sampling is noisiest, and weights the sampled tail. The estimate of that sum stays unbiased at every K, so the loss and gradient consistently estimate those of cross-entropy. HTSS changes only the training loss, so the model and its inference are unchanged, while a step scores a third of the vocabulary. On GPT-2 (124M), weighting alone leaves a 2.1% gap to cross-entropy at matched steps, and at the same cost the exact head cuts it to 1.3%. At equal training compute, HTSS reaches 2.2–2.9% lower perplexity than cross-entropy, and overtakes it on a 304M multilingual backbone from V=128K. At 256K it reaches 4.5% lower perplexity on 46% less peak training memory, never forming the logit matrix. Finetuning base Qwen3.5-0.8B and Llama 3.2 1B, HTSS lands within 0.34% of cross-entropy's perplexity on 12–20% fewer training FLOPs.

1 INTRODUCTION

Much of the efficiency work in language model training has focused on the backbone. Attention has been made linear in memory (Dao et al., 2022) and in time (Yang et al., 2025), and sparse feed-forward layers add parameters at constant FLOPs per token (Fedus et al., 2022). None of those savings reach the output layer, which scores every vocabulary token at every position, although the loss only needs that work for a single normalizing sum. On a compact backbone, the layer therefore takes a substantial share of a training step. For example, it accounts for 27.1% of the FLOPs in a GPT-2 (124M) step, 17.5% in Llama 3.2 (1B), and 28.9% in Qwen3.5 (0.8B).

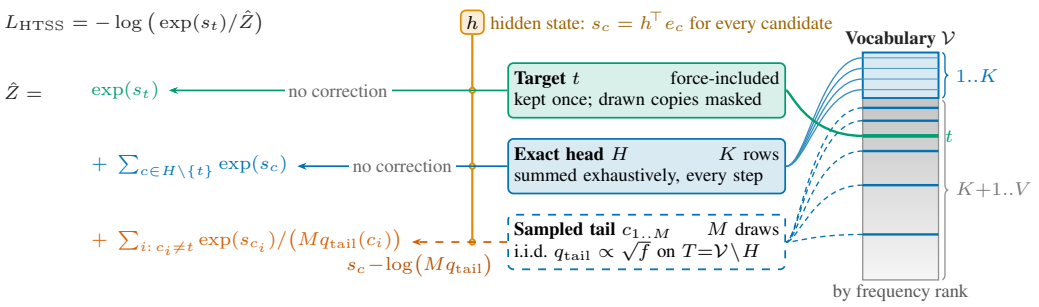

Figure 1: **HTSS splits the vocabulary into an exact head and a sampled tail.** One set of candidates per micro-batch; only sampled scores are corrected by the log of their expected count.

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.

The output layer's share of a step depends primarily on the vocabulary size relative to the backbone size. Vocabulary sizes have grown steadily, from 50,257 tokens in GPT-2 to 128,256 in Llama 3.2 and 248,320 in Qwen3.5, and Tao et al. (2024) argue that larger models warrant even larger ones. A larger vocabulary increases the output layer's share, while a larger backbone reduces it. Llama 3.2's transformer layers, about eleven times larger than GPT-2's, more than offset its larger vocabulary, and with a 70B shape, even the compute-optimal vocabulary leaves the share under 3% (Tao et al., 2024). We study the regime in which this share is large: compact backbones carrying large vocabularies, as in multilingual, code, and on-device models.

We build on headless language models (Godey et al., 2024), which train the embedding matrix E with an in-batch contrastive objective called Contrastive Weight Tying (CWT). Under weight tying (Press & Wolf, 2017; Inan et al., 2017), the classifier is E, so its scores can be read out through the ordinary softmax. Godey et al. (2024) evaluate it, report that it "yields poor performance", and repair the model by finetuning it under cross-entropy, which restores the dense softmax. Their repaired models match the downstream baseline. This suggests that the objective trains the backbone and the embeddings well, and that the failure lies in the readout.

We trace that failure to the estimator. We show that CWT is a *self-normalized sampled softmax* whose proposal is the in-batch token distribution, and that it omits the $\log q$ correction that sampled softmax applies to its negatives (Jean et al., 2015; Yi et al., 2019). Its minimizer is the density ratio of InfoNCE (van den Oord et al., 2019), the data's log-probability $\log p^*$ deflated by the unigram (§3.1), an offset that cross-entropy re-training absorbs. We find that adding the correction at inference, with no finetuning, closes about 80% of the readout's loss gap to cross-entropy but leaves it 42% above in perplexity, and that training with the correction on the same in-batch pool does no better. HTSS, which also replaces that pool, closes most of the rest (§4.2).

We propose *head-tail sampled softmax* (HTSS), which provides that correction while training instead of after it. It sums the K most frequent tokens exactly, where frequency sampling adds most noise, and importance-samples only the rare tail with M draws, extending complementary-sum sampling (Botev et al., 2017). We build the correction table from the tail proposal alone, so that each sampled score is debiased by the log of its expected count among the draws (Figure 1). The estimate remains unbiased at every K; summing the head exactly never increases its variance with an equal number of draws; and the representation gradient remains a consistent estimator of the full-softmax gradient, approaching it as more tokens are drawn (§3.3).

HTSS only changes the training loss, so the model keeps its output layer unchanged, and inference uses the standard full softmax. The saving comes in training, where each step scores only the candidates and never forms the dense logit vector. The output layer's FLOPs then scale with the number of candidates scored, $K + M$, which we set to about a third of the vocabulary (§3.4).

Contributions. (i) We show that omitting the $\log q$ correction accounts for about 80% of the loss gap between CWT's readout and cross-entropy (§3.2). (ii) We adapt complementary-sum sampling (Botev et al., 2017) to language-model pretraining as HTSS, with an exact frequent-token head, a static $\sqrt{f}$ tail proposal, and a fused kernel; its representation gradient is consistent (§3.3). (iii) HTSS is a drop-in replacement for cross-entropy that reaches lower loss at equal FLOPs once the output layer's share is large enough (in our pretraining, from a FLOP multiplier $g{\approx}1.2$), and, with K fixed, one coverage parameter sets M (§3.4).

2 Related Work

The output layer in the large-vocabulary regime. Scaling studies usually hold the vocabulary fixed, so the output layer's share of a step is rarely treated as a variable. Tao et al. (2024) derives a compute-optimal vocabulary that grows sublinearly with non-embedding parameters and finds frontier models under-vocabularized, and Huang et al. (2025) finds gains from scaling the input vocabulary. Neither prices the output layer's training cost on a compact backbone, where it is expensive.

Sampled softmax in language-model pretraining. Sampled softmax estimates the partition function from draws under a proposal distribution (Bengio & Senecal, 2008; Jean et al., 2015). Chen et al. (2016) compares large-vocabulary strategies, and Botev et al. (2017) sums a class set containing the target exactly and samples the rest. Lin et al. (2025) shows that no estimator seeing a sampled subset

of classes can be exactly unbiased for the softmax. In practice, language-model pretraining computes the full cross-entropy, and the approximate output layers used in practice, hierarchical (Morin & Bengio, 2005) and adaptive softmax (Grave et al., 2017), still normalize exactly. Both restructure the output layer into a tree or clusters with their own parameters, whereas HTSS leaves it unchanged. Ji et al. (2016) weight tempered-unigram negatives, as our tail does, without an exact head.

Optimizing the output layer, or replacing it. Cut Cross-Entropy (CCE) (Wijmans et al., 2025) computes the cross-entropy objective through a fused kernel that never materializes the full logit matrix. Chunked kernels (Hsu et al., 2025) do the same, exactly with fp32 accumulation. All still perform the dense projection, saving memory but not FLOPs. We include CCE as a baseline (§4.2). A separate line accelerates or compresses an already-trained output layer (Gou et al., 2025; Liu et al., 2026; Tranheden et al., 2026; Ben-Artzy & Schwartz, 2025), and a third changes the prediction target, using byte-level patches (Pagnoni et al., 2025), energy-based (Dehmamy et al., 2026) and continuous next-vector (Shao et al., 2026) objectives (see Wyatt et al., 2025).

Contrastive objectives. InfoNCE (van den Oord et al., 2019; Poole et al., 2019), in its in-batch form (Chen et al., 2020), underlies dense retrieval (Karpukhin et al., 2020) and contrastive sentence representations (Gao et al., 2021). Godey et al. (2024) pretrain a decoder with such a loss alone, scoring each position against the tied embeddings of its target and the other tokens in the batch, and report that, after a cross-entropy finetune, the model matches the baseline downstream. As InfoNCE with negatives drawn from the unigram, the loss is a sampled softmax without the $\log q$ correction, so its optimum is the density ratio p^*/q (van den Oord et al., 2019) (§3.1). Noise-Contrastive Estimation (Gutmann & Hyvärinen, 2010) trains an unnormalized model on noisy samples, and in language modeling it self-normalizes empirically (Mnih & Teh, 2012), as penalized log-linear models do (Devlin et al., 2014; Andreas & Klein, 2015). Ma & Collins (2018) proves the ranking-based conditional variant consistent for a well-specified model and i.i.d. full-support noise; HTSS's readout rests instead on Prop. 1. HTSS normalizes using an unbiased estimate of Z rather than relying on the normalization that emerges during training.

The $\log q$ sampling-bias correction. The correction subtracts from each sampled candidate's score the log-probability with which it was drawn. It has been part of sampled softmax since its introduction to neural language and translation models (Bengio & Senecal, 2008; Jean et al., 2015), and Yang et al. (2022) study it under self-normalized importance sampling for language modeling. In two-tower retrieval, Yi et al. (2019) applies it to in-batch negatives, where popular items are otherwise penalized too often, and Khrylchenko et al. (2025) refines it by dropping the always-present positive from the denominator, noting that keeping it there uncorrected does not follow from the standard derivation. HTSS applies the correction inside the complementary-sum estimator of Botev et al. (2017), to the sampled tail alone, whose independent draws, unlike in-batch negatives, make each weight exact, and keeps the positive uncorrected as a deterministic term of an unbiased $\hat{Z}$ (Prop. 1).

3 METHOD

Notation. Let $h \in \mathbb{R}^d$ be the backbone's contextual representation at a given position, $t \in \mathcal{V}$ the target token at that position, drawn from the vocabulary $\mathcal{V}$ of size $V = |\mathcal{V}|$, and $p^*(\cdot \mid h)$ the data's next-token distribution. Unless an index set is given, sums over c range over $\mathcal{V}$. The output side is a token-embedding matrix $E \in \mathbb{R}^{V \times d}$ with rows e_c, so token c scores $s_c = e_c^\top h$ with logits $\ell = Eh$, and the logit matrix over any set of positions has rank at most d. The softmax output layer takes $p = \mathrm{softmax}(\ell)$, $p_c = \exp(s_c)/Z$ with partition function $Z = \sum_c \exp(s_c)$, at $O(Vd)$ per token, and its gradient with respect to h,

$$\nabla_h = \textstyle\sum_c p_c e_c - e_t,$$

is a probability-mass-weighted pull toward the target embedding. The forward projection costs $2Vd$ FLOPs per token, and each objective below allows E to be tied to the input embedding or not.

Negative sets. The negatives scored at a position form a multi-set $\mathcal{C}$, with $N{=}|\mathcal{C}|$ counting repeated draws. We write $q(c)$ for the probability that token c is drawn as a negative under a generic proposal. The in-batch negatives come from the micro-batch's $n = b \cdot S$ targets, where b is the micro-batch size and S is the sequence length. We write M for the number of importance-sampled tail draws and K for the size of the exact, frequent-token head, so $N{=}n{-}1$ in-batch and $N{=}K{+}M$ for HTSS. A fresh tail is drawn per micro-batch, so $N_{\text{eff}} = M \cdot \mathrm{ga} \cdot \mathrm{world}$ independent draws stand behind one optimizer step, summed over its ga gradient-accumulation micro-steps and world data-parallel

replicas, while each position still scores its target and N candidates. We choose M from the coverage $(K+M)/V$ (§3.4); the draw count N_{eff} has little effect in our setting (App. §C.5).

3.1 THREE ESTIMATORS OF ONE LOSS

(a) Cross-entropy (CE) is the exact objective, $\mathcal{L}_{\text{CE}} = -\log p_t = -s_t + \log Z$, with Z summed over all of $\mathcal{V}$ at $O(Vd)$ per token.

(b) Sampled softmax draws $c_1, \ldots, c_M \sim q$ and replaces Z by the importance-sampling estimate $\hat{Z} = \exp(s_t) + \sum_{i:\, c_i \neq t} \exp(s_{c_i})/(Mq(c_i))$, i.e. subtracts $\log(Mq(c))$ from each sampled score and masks drawn copies of the target (Bengio & Senecal, 2008; Jean et al., 2015). Since $\hat{Z}$ is unbiased with $O(1/M)$ variance, the loss is consistent at $O(Md)$.

(c) CWT is (b) with the weights dropped. Godey et al. defines it as a dot-product InfoNCE loss,

$$\mathcal{L}_{\text{CWT}} = -\log \frac{\exp(s_t)}{\exp(s_t) + \sum_{c \in \mathcal{C}} \exp(s_c)}, \tag{1}$$

where the negatives $\mathcal{C}$ are the other targets in the micro-batch. Each token appears among a batch's targets at its corpus frequency, so $\mathcal{C}$ is drawn from the unigram q, as in (b), at $O(Nd)$ per token. In contrastive learning over distinct examples such as images, the in-batch marginal is uniform over the data, so $\log q$ is constant, and the missing weight has no effect. Over a vocabulary, the marginal is the unigram. Without the weights the sum no longer estimates Z, and the optimum becomes the density ratio of InfoNCE (van den Oord et al., 2019), $\exp(s_c) \propto p^*(c \mid h)/q(c)$, so

$$s_c \;=\; \log p^*(c \mid h) \;-\; \log q(c) \;+\; \kappa(h). \tag{2}$$

A softmax over these scores therefore returns $p^*(c \mid h)/q(c)$, renormalized, so the readout under-weights frequent tokens and over-weights rare ones. Cross-entropy re-training absorbs this offset. We score with raw inner products, as Godey et al. (2024) do. The cosine similarity with a temperature used by Chen et al. (2020) bounds the scores and so cannot reach the range of log-probabilities.

3.2 RESTORING THE CORRECTION

Equation (2) leaves an extra $-\log q(c)$ in every score. We remove that offset by subtracting from each candidate's score the log-probability $\log q(c)$ with which it is drawn (Bengio & Senecal, 2008; Jean et al., 2015; Yi et al., 2019). With negatives sampled and the positive drawn from the data, every candidate carries its own, and the corrected objective is

$$\mathcal{L} = -\log \frac{\exp(s_t - \log q(t))}{\exp(s_t - \log q(t)) + \sum_{c \in \mathcal{C}} \exp(s_c - \log q(c))}. \tag{3}$$

Equation (3) is the ranking-based conditional noise-contrastive estimation (NCE) of Ma & Collins (2018), whose consistency theorem assumes i.i.d. negatives from q. We estimate q once from Laplace-smoothed token counts on a prefix of the training stream. In-batch negatives violate the i.i.d. assumption: a document repeats its words, so they are correlated, and their expected counts equal $Nq(c)$ only as far as a batch's token mix matches that estimate. HTSS (Eq. (4)) instead draws its tail i.i.d. from the fixed q_{tail}, which makes its weights exact, and force-includes the target without a correction, since the target is present with probability 1 (Khrylchenko et al., 2025) (§3.3).

Why $\log q$ is dropped at inference. We evaluate every model through its full softmax over the raw scores, with no finetuning and no correction term, as a CE-trained model is evaluated. Training subtracts $\log q$, so one might expect this softmax to return $\propto p^*(c \mid h)\, q(c)$. It does not, because the correction is an importance weight inside the estimate of Z and leaves the scores' target unchanged. In (b) and in HTSS, it applies only to sampled candidates, where $1/(Mq(c))$, the inverse of a candidate's expected count among the draws, makes the sampled sum unbiased. The target and HTSS's exact head enter deterministically and uncorrected, so $\mathbb{E}[\hat{Z}] = Z$ (Prop. 1), and the loss $-s_t + \log \hat{Z}$ approximates the full-softmax cross-entropy $-s_t + \log Z$, in which q does not appear. That cross-entropy is minimized at $s_c = \log p^*(c \mid h) + \kappa(h)$, so the raw scores are a cross-entropy model's logits up to the $O(1/M)$ Jensen gap of $\log \hat{Z}$. Equation (3) corrects the target and keeps its drawn copies; up to a constant, it is $-s_t + \log \hat{Z}$ with a biased $\hat{Z}$, and is instead ranking-NCE consistent (Ma & Collins, 2018). Temperature and top-p then act as on CE logits.

3.3 SUMMING THE HEAD EXACTLY

Equation (3) leaves the proposal q free, and the choice governs the partition estimate's relative variance $\mathrm{Var}(\hat{Z})/Z^2 = \chi^2(p \,\|\, q)/M$ for M i.i.d. draws (Lemma 1). That variance is dominated by the frequent (head) tokens (Figure 4b), and the model's distribution drifts away from any static frequency-derived proposal as training proceeds (Chen et al., 2025). We remove it by summing the head exactly, taking the frequent tokens as the exact set of complementary-sum sampling (Botev et al., 2017), and sampling only the rare tail, whose small probabilities add little variance.

We fix three static objects from the same Laplace-smoothed unigram frequencies f: the *exact head* H of the K most frequent tokens, a *tail proposal* $q_{\mathrm{tail}} \propto \sqrt{f}$ over the tail $T = \mathcal{V} \setminus H$, and a *tail correction table* $\widetilde{\log q}$, zero on H and $\log\big(M\, q_{\mathrm{tail}}(c)\big)$ on the tail. At every micro-batch, the multiset $\mathcal{C} = H \uplus \{c_1, \ldots, c_M\}$ holds the head and the draws with their multiplicity, with $c_1, \ldots, c_M$ drawn i.i.d. from q_{tail} and shared by all its positions. We force-include each position's target and mask its other copies in that row (its head column or drawn copies), so only sampled negatives are corrected:

$$\mathcal{L}_{\mathrm{HTSS}} = -\log \frac{\exp(s_t)}{\hat{Z}}, \qquad \hat{Z} = \exp(s_t) + \underbrace{\sum_{c \in H \setminus \{t\}} \exp(s_c)}_{\text{exact head}} + \underbrace{\sum_{i:\, c_i \neq t} \frac{\exp(s_{c_i})}{M\, q_{\mathrm{tail}}(c_i)}}_{\text{sampled tail}}. \tag{4}$$

The head term is deterministic, so it contributes mass to $\hat{Z}$ but no variance (Figure 4). $\hat{Z}$ stays unbiased at every K, including with the target forced in and its drawn copies masked (Prop. 1). Summing the head is a *Rao-Blackwellization* (Liu et al., 2019): with $\pi \propto \sqrt{f}$ over $\mathcal{V}$ and q_{tail} its renormalized restriction to T, $\hat{Z}$ is the conditional expectation of the π-sampled estimate given, for each draw, the tail candidate it takes when it lands in T. Its variance at equal M is therefore never larger (Prop. 2), and the gain is large when H holds tokens with large p^2/π, shrinking the tail factor $m_T^2 \chi^2(p_T \| q_{\mathrm{tail}})$, with $m_T = \sum_{c \in T} p_c$. At $K{=}0$ the head is empty and Eq. (4) reduces to the sampled softmax of Bengio & Senecal (2008) and Jean et al. (2015) under q_{tail}, which we train as a baseline (§4.2). Head tokens receive a dense gradient at every step, with stochasticity only through $\hat{Z}$.

The correction falls on the sampled negatives alone, because a drawn candidate carries weight $1/(M\, q_{\mathrm{tail}}(c))$, the inverse of its expected count among the draws. A deterministic one, by contrast (every head token and the force-included target), carries weight 1 and correction $\log 1 = 0$, where Eq. (3)'s data-drawn positive carries $\log q(t)$. Correcting a tail target while masking its drawn copies would weight its term by $\omega_t = 1/(M\, q_{\mathrm{tail}}(t))$ and give $\mathbb{E}[\hat{Z}] = Z + (\omega_t - 1)\exp(s_t)$, which is unbiased only when $\omega_t = 1$ (the argument of Khrylchenko et al., 2025, applied to $\hat{Z}$). On the GPT-2 unigram at $M{=}16{,}384$, $\omega_t > 1$ for 97.5% of tail tokens, so that choice would bias $\hat{Z}$ at every M and, as $M \to \infty$, would not be consistent; Eq. (3) keeps the copies instead.

We pay $O((K{+}M)d)$ per position (Figure 6), with $K{=}2{,}048$ and M set by the coverage of §3.4. We fix the $\sqrt{f}$ tempering throughout, which we find to be an interior optimum (§4.2). Beyond it and the fixed counting prefix, (K, M) is the only hyperparameter we set per configuration, and §3.4 reduces it to one number.

The corrected gradient is consistent. As in Botev et al. (2017), the head-tail estimates of the partition function and of the unnormalized mass are exactly unbiased for every M. The representation gradient $\widehat{\nabla}_h$, which drives the backbone, is consistent, with bias and variance $O(1/M)$ that vanish when the head enumerates $\mathcal{V}$, and each embedding row follows at the same rate in M, with a constant $\propto 1/q_{\mathrm{tail}}(c)$ (Prop. 1, Cor. 1). No sampled estimator can be unbiased for the softmax (Lin et al., 2025), so HTSS settles for consistency. The tail draws are shared by a micro-batch's positions, so their noise does not average down over positions.

3.4 COST MODEL, AND CHOOSING K AND M

In the forward pass, the output layer costs $2Vd$ FLOPs per token under the softmax and $2Nd$ FLOPs per token under a sampled objective with N candidates. The backbone's L layers cost $C_{\mathrm{bb}} \approx 24Ld^2 + 4LdS$, for projections and MLP plus attention scores at sequence length S. The output layer's forward share is then $\eta = 2Vd/(C_{\mathrm{bb}} + 2Vd)$, or $V/(12Ld + V)$ without the attention term. Backward costs twice as much as forward in every layer, so this is also its share of a training

step. Even a free output layer would cut a step's cost by only η, so η bounds our savings, and we report it for every model. Both objectives keep the same $V \times d$ matrix E, so the parameter count and inference cost are unchanged, and only the training computation differs. The per-step cost ratio of HTSS to CE is

$$r \;=\; \frac{C_{\mathrm{bb}} + 2Nd}{C_{\mathrm{bb}} + 2Vd} \;=\; 1 - \eta\big(1 - \tfrac{N}{V}\big)\,, \tag{5}$$

so an equal FLOP budget allows HTSS $g \coloneqq 1/r$ times as many optimizer steps, up to a ceiling of $1/(1-\eta)$ at which the output layer is free. At $N{=}V$, $r{=}1$, and for $N{>}V$ the candidate set overcovers the vocabulary ($r > 1$), so our substitution costs more than it saves.

Choosing K and M by coverage. We hold $K{=}2{,}048$, large enough that the tokens dominating χ^2 are summed exactly, and choose M through the coverage $\rho = (K{+}M)/V$, the candidate count as a fraction of the vocabulary, counting repeated draws. A larger M shrinks the $O(1/M)$ Jensen bias and the gradient noise but makes each step more expensive, so fewer steps fit a fixed budget. At fixed compute, this trade has a break-even coverage below which HTSS loses to CE, and a flat optimum above it. A fixed-compute sweep on GPT-2 places break-even at $\rho{=}0.12$–0.20 and the optimum at $\rho{\approx}0.29$–0.53 (App. §C.5). Our GPT-2 runs use $M{=}16{,}384$ ($\rho{=}0.37$), inside the optimum; other settings use our *operating coverage* $\rho{\approx}0.3$, around its low edge.

When HTSS wins at equal compute. HTSS trades a small per-step deficit for g times as many steps. In pretraining, HTSS-mc trails CE at $g{=}1.05$ and 1.11 and leads it at $g{=}1.21$, 1.22, and 1.41 (§4), so the crossover lies near $g{\approx}1.2$; in finetuning, where the per-step deficit is smaller, it already leads at $g{=}1.14$. Eq. (5) gives g from η and ρ alone, so this can be checked before training.

4 Experiments

4.1 Protocol

Each setting fixes one model and recipe, such as tied GPT-2, and varies only the output objective and the training budget. CE sets the budget, HTSS runs CE's step count, and HTSS-mc (matched compute) spends CE's training FLOPs. Matched steps separates the cost of a step from its effect on quality; matched compute measures the trade, and is the comparison we treat as primary. HTSS-mc reuses CE's corpus and learning-rate schedule unchanged, which likely understates it (App. §B). Percentages are relative to each setting's CE. We report perplexity throughout, except for the vocabulary sweep (§4.3), whose runs differ in their tokenizers and are therefore compared in bits per byte (BPB), which does not depend on the tokenizer.

4.2 GPT-2 pretraining, tied and untied

Setup. We train the original GPT-2 (124M) (Radford et al., 2019) on FineWeb-Edu (Penedo et al., 2024) at GPT-2's own batch and a Chinchilla-20 budget (Hoffmann et al., 2022), tied and untied. Besides CE, HTSS and HTSS-mc, we train two baselines at CE's step count: CWT (Eq. (1)), the objective of Godey et al. (2024), and CE computed through the memory-efficient Cut Cross-Entropy kernel (CCE; §2). The output layer claims $\eta{=}27.1\%$ of a step, and our candidate set spans 37% of the vocabulary. HTSS can reclaim $\eta\,(1{-}N/V){=}17.2\%$ of a step.

Takeaway. At equal compute, HTSS-mc reaches 2.2–2.9% lower perplexity than CE on a step 1.23× faster with 11% less memory; at equal steps HTSS is 1.3–1.4% behind CE. CWT reaches 5× CE's perplexity, and adding $\log q$ back at inference closes about 80% of that gap in loss.

The missing correction explains most of CWT's gap. Our CWT control reproduces the failure Godey et al. (2024) report, reaching 5× CE's test perplexity. If CWT's scores are $\log p^*$ deflated by $\log q$ (Eq. (2)), adding $\log q(c)$ back before the softmax should recover $\log p^*$ without retraining, since $\kappa(h)$ cancels. On a trained CWT checkpoint, this shift lowers test perplexity from 123.1 to 33.9, against CE's 23.90 on the evaluation set of our fixed-compute sweep (App. §C.5), closing about 80% of the loss gap. It has no free parameter, and scaling it barely helps, so the offset is the unigram term the derivation predicts (App. §C.4). The remaining 42% perplexity gap persists when the correction is applied in training, so it lies in the in-batch pool, which covers only ≈6.8K distinct tokens against HTSS's ≈13K plus its head, and whose negatives are correlated within documents (§3.2).

Table 1: **Cost and quality on GPT-2 at matched steps and compute.** FineWeb-Edu, K=2,048, M=16,384. *Budget*: steps or FLOPs (*compute*) matched to CE; cost measured as in App. §B. Values in parentheses give the relative change from CE in the same group; bold is best per group.

		Cost				Quality		
Method	Budget	Steps	GPU-h ↓	Tok/s ↑	Peak GB ↓	Test ppl ↓	Top-1 ↑	Top-5 ↑
Tied — GPT-2, 124M params								
CE	ref.	4,748	8.94	77.4K	43.3	23.35	0.405	0.631
CE-CCE	steps	4,748	7.99	86.5K (+12%)	28.1 (−35%)	25.06 (+7.3%)	0.394	0.620
CWT	steps	4,748	7.94	87.1K (+13%)	45.2 (+4%)	119.08 (5.1×)	0.308	—
HTSS	steps	4,748	7.27	95.1K (+23%)	38.4 (−11%)	23.65 (+1.3%)	0.404	0.631
HTSS-mc	compute	5,732	8.78	95.0K (+23%)	38.4 (−11%)	**22.68** (−2.9%)	**0.408**	**0.636**
Untied — GPT-2, 163M params								
CE	ref.	6,220	11.72	77.3K	43.6	22.05	0.411	0.639
CE-CCE	steps	6,220	10.49	86.3K (+12%)	28.2 (−35%)	26.27 (+19.1%)	0.377	0.604
CWT	steps	6,220	10.40	87.1K (+13%)	45.6 (+5%)	109.89 (5.0×)	0.313	—
HTSS	steps	6,220	9.53	95.1K (+23%)	38.8 (−11%)	22.35 (+1.4%)	0.410	0.638
HTSS-mc	compute	7,509	11.51	95.0K (+23%)	38.8 (−11%)	**21.56** (−2.2%)	**0.414**	**0.643**

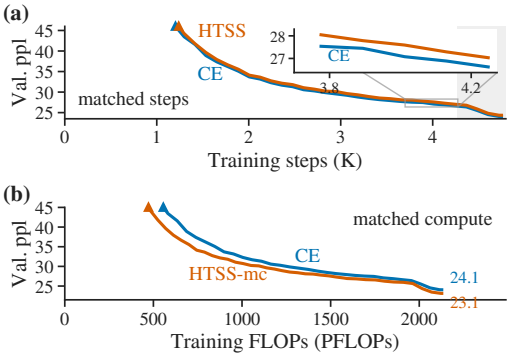

Figure 2: GPT-2 validation perplexity at (a) matched steps and (b) matched compute; each curve's ▲ marks where it enters the range.

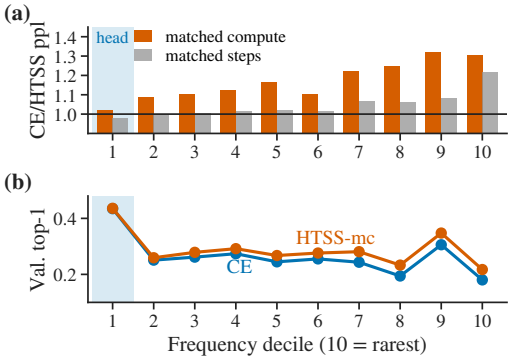

Figure 3: GPT-2 by frequency decile (validation). (a) CE/HTSS perplexity ratio, above 1 favors HTSS. (b) Top-1 accuracy.

At matched steps, HTSS is 1.3% behind CE on a cheaper step. When we run it for CE's own step count, HTSS comes out behind CE in test perplexity, 23.65 against 23.35 tied and 22.35 against 22.05 untied. Both gaps are 0.3 perplexity points, 1.3% tied and 1.4% untied. Sampled softmax at the same candidate count (K=0, tied) reaches 23.84, 2.1% behind CE, so the exact head closes about 40% of its gap (App. §C.5). In Figure 2, where both runs share a budget so their warmup–stable–decay (WSD) schedules decay at the same step, the plateau averages 1.5% behind CE. The step that produces it runs at 1.23× CE's throughput with 11% less peak training memory (38.4 GB against 43.3 GB), and the coverage choice of §3.4 accepts a deficit of this size by design. Untying leaves the step's cost unchanged, since the extra $V \times d$ table adds no output-layer work.

At equal compute, HTSS-mc reaches lower perplexity than CE. Spending CE's FLOPs instead of its steps allows HTSS-mc about 21% more optimizer steps, and it reaches 22.68 test perplexity against CE's 23.35 tied (−2.9%), and 3.8% lower on validation. Plotted against training FLOPs, its validation-perplexity curve lies below CE's at every evaluation both runs share, from the start of training (Figure 2b); against steps, the shared prefix trails CE (App. §C). Untied, we measure −2.2%, about three-quarters of the tied gain. At g=1.21, the extra steps outweigh the 1.3% per-step deficit.

The margin is largest on rare tokens. Aggregate perplexity is dominated by frequent tokens, so we also report it per frequency decile (Figure 3). HTSS-mc reaches lower validation perplexity than CE in all ten deciles, by 2.2% on the most frequent up to 23.5% on the rarest. Matched-steps HTSS shows the same tilt beneath its aggregate deficit: it trails CE by 1.8% on the first decile, which holds most token mass, and matches or beats CE on the rest. The exponent of the tail proposal controls

Table 2: **Cost and quality of a 304M backbone across vocabulary sizes.** HTSS on our fused kernel; cost measured as in App. §B. Values in parentheses give the relative change from CE at the same V; bold is best per row. **HTSS**-mc matches CE's training FLOPs and shares **HTSS**'s cost columns.

	Tok/s ↑		Peak GB ↓		Test BPB ↓		
V	CE	**HTSS**	CE	**HTSS**	CE	**HTSS**	**HTSS**-mc
32K	36.1K	37.3K (+3%)	19.0	16.7 (–12%)	**1.0353**	1.0504 (+1.46%)	1.0452 (+0.96%)
64K	34.2K	36.9K (+8%)	22.5	17.6 (–22%)	**1.0043**	1.0207 (+1.63%)	1.0109 (+0.66%)
128K	30.9K	36.5K (+18%)	29.5	19.6 (–34%)	0.9681	0.9801 (+1.24%)	**0.9639** (–0.43%)
256K	26.0K	35.3K (+36%)	43.4	23.4 (–46%)	0.9244	0.9328 (+0.91%)	**0.9105** (–1.50%)

the rare-token gain. At matched steps, $q_{\text{tail}} \propto f^{0.25}$ removes it, and $q_{\text{tail}} \propto f$ raises it to 39% on the rarest decile, costing about 3% on the three most frequent (App. §C); in aggregate test perplexity they trail $\sqrt{f}$ by 0.45% and 1.0%. The first-decile deficit persists at every exponent and at K=8,192, where the exact head covers that whole decile, so tail sampling of frequent tokens does not cause it. We suspect the sampled normalizer, which enters every token's gradient, but leave this open.

HTSS transfers zero-shot. Evaluated zero-shot with no finetuning, HTSS matches CE on the seven-task suite at both budgets, with small per-task gaps at a scale where the multiple-choice tasks are near chance (Table 4, App. §C). LAMBADA is the exception and the only free-form completion task in the suite, scoring a held-out token against the full vocabulary. That is where the rare-token gain above predicts HTSS should win. Its perplexity falls from CE's 242.6 to 221.7 at matched steps, and 191.7 at matched compute, with accuracy following ($0.206 \rightarrow 0.221 \rightarrow 0.232$), and only the matched-compute reduction clears lm-eval's stderr.

A fused CE kernel lowers memory and leaves FLOPs unchanged. We also train CE through the Cut Cross-Entropy kernel (Wijmans et al., 2025) with the same recipe. Because CCE never forms the logit matrix, it has the lowest memory usage of any run in Table 1 but retains CE's dense FLOPs. Table 1's HTSS rows are unfused; through our fused kernel, HTSS matches CCE's memory (28.1 GB tied) and runs 1.25× faster, since fusion accounts for the memory savings and sampling for the FLOP savings (App. §C.6). At released defaults, CCE reaches +7.3% and +19.1% test perplexity, which we attribute to gradient filtering and bf16 accumulation (App. §C), so we use it only as a memory baseline.

4.3 Scaling the vocabulary on a 304M backbone

Setup. We test the growth with V that §3.4 predicts, in tokenizer-fair BPB. Each run is a Llama-style model (RoPE, SwiGLU, RMSNorm; d=1,024, L=24) with 304M non-embedding parameters and untied embeddings, 369M–828M in total, trained on the FineWeb-2 multilingual mix at a global batch of 2.1M tokens, over $V \in \{32{,}000,\ 64{,}000,\ 128{,}000,\ 256{,}000\}$. Every run trains on 20 tokens per total parameter, so a larger vocabulary trains longer. The multilingual mix keeps the vocabulary in use as a production corpus does: 95.1% of a 256K vocabulary appears in held-out text, against 32.6% for English-only data (App. §D.2). The output layer's share rises from η=7.5% at V=32K to 39.4% at 256K. We keep K=2,048 and apply the operating coverage without retuning, $M = V/4$ rounded to a power of two (8,192 to 65,536), so ρ=0.32 to 0.26, near the low edge of GPT-2's optimum.

Takeaway. The gain grows with V. At equal compute, HTSS-mc is behind CE at 32K and 64K, overtakes it from V=128K, and leads by 1.50% BPB (4.5% perplexity) at 256K, on a step 1.36× faster with 46% less memory than CE with materialized logits.

The matched-steps deficit stays near 1% in BPB while the step gets cheaper. In Table 2, we find HTSS behind CE by 0.9–1.6% in test BPB (2.8–4.7% in perplexity, as $2^{\Delta\text{BPB}\cdot\text{bytes/token}}$ with each tokenizer's bytes per token on the test split) at all four vocabulary sizes, smallest at 256K. Its step gets cheaper: the FLOP multiplier $g = 1/r$ (Eq. (5)) rises from 1.05 at V=32K to 1.41 at V=256K. Peak memory falls further, from 12% below CE's at 32K to 46% below at 256K, since CE's $n \times V$ logit matrix grows with V; a fused CE kernel recovers the memory, not the FLOPs (§4.2).

At equal compute HTSS-mc leads CE from V=128K, and the margin grows with V. Reinvesting the savings moves HTSS-mc monotonically toward CE in BPB, from +0.96% at 32K, and past it

between 64K and 128K, to -1.50% at 256K. In perplexity, the same runs go from $+2.5\%$ at 32K to -1.3% at 128K and -4.5% at 256K. Below 128K, the extra steps are too few (g=1.05, 1.11). The corpus is fixed while budgets grow with V, so HTSS-mc at 256K reads it about 3.5 times and coverage falls from ρ=0.32 to 0.26, costing at most 0.6% in perplexity if GPT-2's response transfers; both handicaps grow with V, so neither explains the trend (App. §D).

4.4 Finetuning off-the-shelf base models

Setup. We finetune the base checkpoints of Qwen3.5-0.8B (V=248,320) and Llama 3.2 1B (Grattafiori et al., 2024) (V=128,256) on 10^8 tokens of SmolTalk (Ben Allal et al., 2025) (763 CE steps) at the operating coverage ρ=0.3 (recipe and downstream evaluations in App. §E).

Takeaway. HTSS drops into finetuning unchanged: at matched steps it trains 14% (Qwen) and 8% (Llama) faster on 23–36% less memory, stays within 0.34% of CE's perplexity, and shows no downstream difference beyond two standard errors.

At matched steps, the finetune costs 0.15–0.34% on a cheaper step. Run for CE's 763 steps (Table 3), HTSS trails CE in test perplexity by 0.15% on Qwen and 0.34% on Llama on full held-out sequences ($+0.17\%$ and $+0.36\%$ on assistant tokens; App. §E). Both are below the pretraining deficits of 1.3% (GPT-2) and 2.8–4.7% (304M), on steps $1.26\times$ and $1.14\times$ cheaper in FLOPs with 36% and 23% less peak memory.

At matched compute, HTSS-mc is ahead on both models. Reinvesting the savings at CE's training-FLOP budget allows HTSS-mc 964 steps on Qwen and 869 on Llama. It reaches 2.6230 test perplexity against CE's 2.6448 on Qwen, -0.82%, and 2.6595 against 2.6677 on Llama, -0.31%; Qwen has twice the vocabulary and a larger η (28.9% against 17.5%).

The finetuned models end close to CE's. We find that both objectives lift assistant-token perplexity equally (Qwen 2.777 $\rightarrow$ 2.245 under CE, 2.249 under HTSS; Llama 3.319 $\rightarrow$ 2.335 and 2.343) and add prompt-level IFEval points (CE 4.1 and 5.7, HTSS 3.3 and 3.9). Per-metric scores and standard errors are in App. §E.

Table 3: **SmolTalk finetuning.** Sequence length 4,096. **HTSS** runs CE's steps and **HTSS**-mc CE's training FLOPs, both on our fused kernel; values in parentheses are relative to CE.

	Tok/s ↑	GB ↓	Test ppl ↓
Qwen3.5-0.8B, V=248K			
CE	28.2K	54.3	2.6448
HTSS	32.0K (+14%)	34.8 (–36%)	2.6487 (+0.15%)
HTSS-mc	32.1K (+14%)	34.8 (–36%)	**2.6230** (–0.82%)
Llama 3.2 1B, V=128K			
CE	24.9K	43.0	2.6677
HTSS	26.9K (+8%)	33.1 (–23%)	2.6768 (+0.34%)
HTSS-mc	27.1K (+9%)	33.1 (–23%)	**2.6595** (–0.31%)

5 Conclusion

Contrastive objectives had failed as generative models mainly because CWT drops sampled softmax's importance weights: adding $\log q$ back closes about 80% of its loss gap, and HTSS's exact head over a weighted i.i.d. tail closes most of the rest, yielding an ordinary softmax LM trained without forming its logit matrix. Its cheaper steps repay a small per-step deficit when the output layer's share of the step is large. HTSS leads cross-entropy at equal compute on GPT-2 and in both finetunes, and on the 304M backbone, it trails at 32K and 64K but leads from V=128K, by 4.5% at 256K. The results suggest it pays when the vocabulary is large relative to the backbone, and we expect more so as sparse or linear-time layers make the backbone cheaper. In future work, we plan to optimize the fused kernel so that more of the FLOP saving shows up as wall-clock time, adapt the tail proposal during training, sample frequent tokens within the head, and tune HTSS-mc's schedule for its own budget.

Limitations. We pretrain at most 828M parameters and finetune at most 1B, and the output layer's share shrinks as the backbone grows. At matched compute, HTSS-mc pretraining takes up to 4% more wall-clock time than CE, which a better-optimized kernel could reduce. At matched steps, HTSS trails CE on the most frequent decile for reasons we have not identified, and on all eight downstream metrics, each within noise.

REPRODUCIBILITY STATEMENT

We specify the datasets, the run definitions and the conventions every setting is read under in §4.1 and App. §B, and the recipe of each experiment in a separate appendix: App. §C gives the hardware, budget, optimizer, schedule and precision behind the GPT-2 comparison, App. §D the same for the vocabulary sweep, with the tokenizer-training protocol and the occupancy and fertility measurements in App. §D.2, and App. §E for the finetuning suite. The training and evaluation code, including the fused HTSS kernel, is available with instructions for running all experiments at the anonymized repository `https://anonymous.4open.science/r/htss-7476` and in the supplementary zip. Upon publication, we will make the code public, together with the tokenizers, the trained checkpoints for every run in Tables 1, 2 and 3 with their configurations and training logs, and the output-layer share survey, which counts the η values we quote per layer from a model's configuration.

ETHICS STATEMENT

The work introduces no new data collection, involves no human subjects, and uses only publicly available corpora. Its societal impact profile is that of general language model efficiency research. Lowering the compute of large-vocabulary, and therefore multilingual, models can broaden language coverage. Because HTSS samples the rare tokens that lower-resource languages rely on, that benefit should be checked per language; we report only aggregate multilingual BPB and English downstream tasks. The work shares the dual-use risks common to all language-model capability research.

AI USE STATEMENT

We used LLMs for research ideation and execution, for drafting sections of the paper, and for polishing the writing. LLMs helped refine the method's design choices and formalize the proofs in App. §A, and wrote and debugged much of the experimental code, including the fused HTSS kernel and the plotting tooling. They also drafted parts of the manuscript from author outlines and results, and edited author-written text. The authors re-derived every proof, reviewed and tested all AI-written code, traced every reported number to the logged runs, and read every cited reference in its original form. We did not use LLMs to generate or process data. We take full responsibility for the content of this paper.

REFERENCES

Jacob Andreas and Dan Klein. When and why are log-linear models self-normalizing? In Rada Mihalcea, Joyce Chai, and Anoop Sarkar (eds.), *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 244–249, Denver, Colorado, May–June 2015. Association for Computational Linguistics. doi: 10.3115/v1/N15-1027. URL `https://aclanthology.org/N15-1027/`.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martin Blazquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Agustín Piqueres Lajarín, Hynek Kydlíček, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan Son Nguyen, Ben Burtenshaw, Clémentine Fourrier, Haojun Zhao, Hugo Larcher, Mathieu Morlon, Cyril Zakka, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Second Conference on Language Modeling*, 2025. URL `https://openreview.net/forum?id=3JiCl2A14H`.

Amit Ben-Artzy and Roy Schwartz. SpeLLM: Character-level multi-head decoding, 2025. URL `https://arxiv.org/abs/2507.16323`.

Yoshua Bengio and Jean-Sébastien Senecal. Adaptive importance sampling to accelerate training of a neural probabilistic language model. *IEEE Transactions on Neural Networks*, 19(4):713–722, April 2008. ISSN 1045-9227. doi: 10.1109/TNN.2007.912312. URL `https://doi.org/10.1109/TNN.2007.912312`.

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.

Aleksandar Botev, Bowen Zheng, and David Barber. Complementary sum sampling for likelihood approximation in large scale classification. In Aarti Singh and Jerry Zhu (eds.), *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, volume 54 of *Proceedings of Machine Learning Research*, pp. 1030–1038. PMLR, 20–22 Apr 2017. URL https://proceedings.mlr.press/v54/botev17a.html.

Changyou Chen, Jianyi Zhang, Yi Xu, Liqun Chen, Jiali Duan, Yiran Chen, Son Tran, Belinda Zeng, and Trishul Chilimbi. Why do we need large batchsizes in contrastive learning? A gradient-bias perspective. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 33860–33875. Curran Associates, Inc., 2022. doi: 10.52202/068431-2454. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/db174d373133dcc6bf83bc98e4b681f8-Paper-Conference.pdf.

Jin Chen, Jin Zhang, Xu Huang, Yi Yang, Defu Lian, and Enhong Chen. Adaptive sampled softmax with inverted multi-index: Methods, theory and applications, 2025. URL https://arxiv.org/abs/2501.08563.

Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

Wenlin Chen, David Grangier, and Michael Auli. Strategies for training large vocabulary neural language models. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1975–1985, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1186. URL https://aclanthology.org/P16-1186/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? Try ARC, the AI2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Nima Dehmamy, Benjamin Hoover, Bishwajit Saha, Leo Kozachkov, Jean-Jacques Slotine, and Dmitry Krotov. NRGPT: An energy-based alternative for GPT. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=B3Muyi2zgo.

Jacob Devlin, Rabih Zbib, Zhongqiang Huang, Thomas Lamar, Richard Schwartz, and John Makhoul. Fast and robust neural network joint models for statistical machine translation. In Kristina Toutanova and Hua Wu (eds.), *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1370–1380, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/P14-1129. URL https://aclanthology.org/P14-1129/.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(1):1–39, January 2022. ISSN 1532-4435.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. SimCSE: Simple contrastive learning of sentence embeddings. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 6894–6910, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.552. URL https://aclanthology.org/2021.emnlp-main.552/.

Nathan Godey, Éric Villemonte de la Clergerie, and Benoît Sagot. Headless language models: Learning without predicting with contrastive weight tying. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=ONPECq0Rk7.

Dayin Gou, Sanghyun Byun, Nilesh Malpeddi, Gabrielle De Micheli, Prathamesh Vaste, Jacob Song, and Woo Seong Chung. CARVQ: Corrective adaptor with group residual vector quantization for LLM embedding compression. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 18594–18604, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.1009. URL https://aclanthology.org/2025.findings-emnlp.1009/.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya

Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The Llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Édouard Grave, Armand Joulin, Moustapha Cissé, David Grangier, and Hervé Jégou. Efficient softmax approximation for GPUs. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 1302–1310. JMLR.org, 2017.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In Yee Whye Teh and Mike Titterington (eds.), *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of

Proceedings of Machine Learning Research, pp. 297–304, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR. URL https://proceedings.mlr.press/v9/gutmann10a.html.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Oriol Vinyals, Jack W. Rae, and Laurent Sifre. Training compute-optimal large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

Pin-Lun Hsu, Yun Dai, Vignesh Kothapalli, Qingquan Song, Shao Tang, Siyu Zhu, Steven Shimizu, Shivam Sahni, Haowen Ning, Yanning Chen, and Zhipeng Wang. Liger-Kernel: Efficient Triton kernels for LLM training. In *Championing Open-source DEvelopment in ML Workshop @ ICML25*, 2025. URL https://openreview.net/forum?id=36SjAIT42G.

Shengding Hu, Yuge Tu, Xu Han, Ganqu Cui, Chaoqun He, Weilin Zhao, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Xinrong Zhang, Zhen Leng Thai, Chongyi Wang, Yuan Yao, Chenyang Zhao, Jie Zhou, Jie Cai, Zhongwu Zhai, Ning Ding, Chao Jia, Guoyang Zeng, Dahai Li, Zhiyuan Liu, and Maosong Sun. MiniCPM: Unveiling the potential of small language models with scalable training strategies. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=3X2L2TFr0f.

Hongzhi Huang, Defa Zhu, Banggu Wu, Yutao Zeng, Ya Wang, Qiyang Min, and Zhou Xun. Overtokenized transformer: Vocabulary is generally worth scaling. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 26261–26282. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/huang25bb.html.

Hakan Inan, Khashayar Khosravi, and Richard Socher. Tying word vectors and word classifiers: A loss framework for language modeling. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=r1aPbsFle.

Sébastien Jean, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. On using very large target vocabulary for neural machine translation. In Chengqing Zong and Michael Strube (eds.), *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 1–10, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1001. URL https://aclanthology.org/P15-1001/.

Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, and Pradeep Dubey. BlackOut: Speeding up recurrent neural network language models with very large vocabularies, 2016. URL https://arxiv.org/abs/1511.06909.

Andrej Karpathy. llm.c: Large language model training in pure C/CUDA. https://github.com/karpathy/llm.c, 2024.

Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6769–6781, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.550. URL https://aclanthology.org/2020.emnlp-main.550/.

Kirill Khrylchenko, Vladimir Baikalov, Sergei Makeev, Artem Matveev, and Sergei Liamaev. Correcting the LogQ correction: Revisiting sampled softmax for large-scale retrieval. In *Proceedings of the Nineteenth ACM Conference on Recommender Systems*, RecSys '25, pp. 545–550, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713644. doi: 10.1145/3705328.3748033. URL https://doi.org/10.1145/3705328.3748033.

Li-Chung Lin, Yaxu Liu, and Chih-Jen Lin. Sampled estimators for softmax must be biased. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=xtKNbPTnMA.

Dong Liu, Shu Wang, Yanxuan Yu, Haisheng Wang, and Ben Lengerich. CSV-Decode: Certifiable sub-vocabulary decoding for efficient large language model inference, 2026. URL https://arxiv.org/abs/2511.21702.

Runjing Liu, Jeffrey Regier, Nilesh Tripuraneni, Michael Jordan, and Jon McAuliffe. Rao-Blackwellized stochastic gradients for discrete distributions. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 4023–4031. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/liu19c.html.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Bkg6RiCqY7.

Zhuang Ma and Michael Collins. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3698–3707, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1405. URL https://aclanthology.org/D18-1405/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017. URL https://openreview.net/forum?id=Byj72udxe.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, ICML'12, pp. 419–426, Madison, WI, USA, 2012. Omnipress. ISBN 9781450312851.

Frederic Morin and Yoshua Bengio. Hierarchical probabilistic neural network language model. In Robert G. Cowell and Zoubin Ghahramani (eds.), *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, volume R5 of *Proceedings of Machine Learning Research*, pp. 246–252. PMLR, 06–08 Jan 2005. URL https://proceedings.mlr.press/r5/morin05a.html. Reissued by PMLR on 30 March 2021.

Niklas Muennighoff, Alexander M. Rush, Boaz Barak, Teven Le Scao, Aleksandra Piktus, Nouamane Tazi, Sampo Pyysalo, Thomas Wolf, and Colin Raffel. Scaling data-constrained language models. *Journal of Machine Learning Research*, 26(53):1–66, 2025. URL http://jmlr.org/papers/v26/24-1000.html.

Artidoro Pagnoni, Ramakanth Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason E Weston, Luke Zettlemoyer, Gargi Ghosh, Mike Lewis, Ari Holtzman, and Srini Iyer. Byte latent transformer: Patches scale better than tokens. In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9238–9258, Vienna, Austria, July 2025. Association for Computational Linguistics. ISBN 979-8-89176-251-0. doi: 10.18653/v1/2025.acl-long.453. URL https://aclanthology.org/2025.acl-long.453/.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In Katrin Erk and Noah A. Smith (eds.), *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1525–1534, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1144. URL https://aclanthology.org/P16-1144/.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben Allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro von Werra, and Thomas Wolf. The FineWeb datasets: Decanting the web for the finest text data at scale. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=n6SCkn2QaG.

Guilherme Penedo, Hynek Kydlíček, Vinko Sabolčec, Bettina Messmer, Negar Foroutan, Amir Hossein Kargaran, Colin Raffel, Martin Jaggi, Leandro von Werra, and Thomas Wolf. FineWeb2: One pipeline to scale them all — adapting pre-training data processing to every language. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=jnRBe6zatP.

Ben Poole, Sherjil Ozair, Aaron van den Oord, Alex Alemi, and George Tucker. On variational bounds of mutual information. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5171–5180. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/poole19a.html.

Ofir Press and Lior Wolf. Using the output embedding to improve language models. In Mirella Lapata, Phil Blunsom, and Alexander Koller (eds.), *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pp. 157–163, Valencia, Spain, April 2017. Association for Computational Linguistics. URL https://aclanthology.org/E17-2025/.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, OpenAI, 2019. URL https://api.semanticscholar.org/CorpusID:160025533.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial Winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://doi.org/10.1145/3474381.

Chenze Shao, Darren Li, Fandong Meng, and Jie Zhou. Continuous autoregressive language models, 2026. URL https://openreview.net/forum?id=Ce6Mep9oie.

Lintang Sutawika, Hailey Schoelkopf, Leo Gao, Baber Abbasi, Stella Biderman, Jonathan Tow, ben fattori, Charles Lovering, farzanehnakhaee70, Jason Phang, Anish Thite, Fazz, Thomas Wang, Niklas, Aflah, sdtblck, nopperl, gakada, tttyuntian, researcher2, Julen Etxaniz, Chris, James A. Michaelov, Hanwool Albert Lee, Janna, Leonid Sinev, Khalid, Kiersten Stokes, Zdeněk Kasner, and KonradSzafer. EleutherAI/lm-evaluation-harness: v0.4.11, February 2026. URL https://doi.org/10.5281/zenodo.18636344.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In A. Globerson, L. Mackey, D. Belgrave, A. Fan, U. Paquet, J. Tomczak, and C. Zhang (eds.), *Advances in Neural Information Processing Systems*, volume 37, pp. 114147–114179. Curran Associates, Inc., 2024. doi: 10.52202/079017-3626. URL https://proceedings.neurips.cc/paper_files/paper/2024/file/cf5a019ae9c11b4be88213ce3f85d85c-Paper-Conference.pdf.

Wilhelm Tranheden, Shahnawaz Ahmed, Devdatt Dubhashi, Jonna Matthiesen, and Hannes von Essen. FlashHead: Efficient drop-in replacement for the classification head in language model inference, 2026. URL https://arxiv.org/abs/2603.14591.

Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding, 2019. URL https://arxiv.org/abs/1807.03748.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In Leon Derczynski, Wei Xu, Alan Ritter, and Tim Baldwin (eds.), *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-4413. URL https://aclanthology.org/W17-4413/.

Erik Wijmans, Brody Huval, Alexander Hertzberg, Vladlen Koltun, and Philipp Krähenbühl. Cut your losses in large-vocabulary language models. In Y. Yue, A. Garg, N. Peng, F. Sha, and R. Yu (eds.), *International Conference on Learning Representations*, volume 2025, pp. 68174–68193, 2025. URL https://proceedings.iclr.cc/paper_files/paper/2025/file/aa963ac256590bb7ad5fc26c68229a3a-Paper-Conference.pdf.

Charlie Wyatt, Aditya Joshi, and Flora Salim. Alternatives to next token prediction in text generation – a survey, 2025. URL https://arxiv.org/abs/2509.24435.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving Mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=r8H7xhYPwz.

Zijian Yang, Yingbo Gao, Alexander Gerstenberger, Jintao Jiang, Ralf Schlüter, and Hermann Ney. Self-normalized importance sampling for neural language modeling. In *Interspeech 2022*, pp. 3909–3913, 2022. doi: 10.21437/Interspeech.2022-10588.

Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, RecSys '19, pp. 269–277, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450362436. doi: 10.1145/3298689.3346996. URL https://doi.org/10.1145/3298689.3346996.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In Anna Korhonen, David Traum, and Lluís Màrquez (eds.), *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1472. URL https://aclanthology.org/P19-1472/.

Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models, 2023. URL https://arxiv.org/abs/2311.07911.

A THEORY

CE and HTSS train logits in the same rank-d family and differ per step only in the gradient: Eq. (4) replaces the sum over c by an importance-weighted sum over $\{t\} \cup \mathcal{C}$, whose bias we bound below. App. §A.1 shows that the head-tail estimates of the partition function and of the unnormalized mass are exactly unbiased at every M, and that the representation gradient is consistent with explicit $O(1/M)$ rates; App. §A.2 shows that summing the head exactly never increases the variance; App. §A.3 relates the scheme to NCE and to model-aware proposals.

The argument is the standard importance-sampling one: subtracting $\log(Mq(c))$ multiplies each exponential by the weight $1/(Mq(c))$ that makes a sum over M q-sampled draws unbiased, and a delta-method expansion bounds the ratio's bias through $\mathrm{Var}(\hat{Z})$ and $\mathrm{Cov}(\hat{\mu}, \hat{Z})$, both $O(1/M)$ (Bengio & Senecal, 2008; Botev et al., 2017; Liu et al., 2019). No sampled estimator removes the ratio's bias entirely (Lin et al., 2025; Chen et al., 2025; 2022); the $\log q$ correction reduces it from CWT's $O(1)$ to $O(1/M)$. Beyond compute, CE and HTSS differ only by that residual and by gradient noise of the same order (Corollary 1), whose joint effect §4.2 measures as the matched-steps deficit.

Standing assumptions and notation. Throughout, every proposal (q, π, and q_{tail} on T) has full support, guaranteed here by Laplace-smoothed counts, so every χ^2 below is finite. These are the conditions under which the delta-method expansion is valid. The condition such expansions usually add, that $\hat{Z}$ be bounded away from 0, is automatic for Eq. (4): all terms are non-negative, and the force-included positive gives $\hat{Z} \geq \exp(s_t) > 0$ deterministically. With $H \subseteq \mathcal{V}$ the exact head and $T = \mathcal{V} \setminus H$ the tail, we write $T_t = T \setminus \{t\}$, $m_{T_t} = \sum_{c \in T_t} p_c$ for the model's tail mass without the target, $p_{T_t} = p/m_{T_t}$ for its distribution restricted to T_t, and $\chi^2(p_{T_t} \| q_{\mathrm{tail}}) = \sum_{c \in T_t} p_{T_t,c}^2 / q_{\mathrm{tail}}(c) - 1$. When $t \in H$, $T_t = T$ and these are m_T, p_T and $\chi^2(p_T \| q_{\mathrm{tail}})$.

A.1 UNBIASEDNESS AND CONSISTENCY

With $c_1, \dots, c_M$ the i.i.d. tail draws from q_{tail}, Eq. (4) evaluates the partition function as

$$\hat{Z} = \exp(s_t) + \sum_{c \in H \setminus \{t\}} \exp(s_c) + \sum_{i:\, c_i \neq t} \frac{\exp(s_{c_i})}{M\, q_{\text{tail}}(c_i)}, \tag{6}$$

and $\widehat{\mu}$ is the same sum with $\exp(s_c)$ replaced by $\exp(s_c)\, e_c$. We first present the variance of $\hat{Z}$, as illustrated in Figure 4, and then the unbiasedness and consistency results.

Lemma 1 (variance as chi-square mismatch)**.** *For M i.i.d. negatives drawn from a generic full-support proposal q over $\mathcal{V}$, the importance-weighted partition estimator $\hat{Z} = \frac{1}{M}\sum_i \exp(s_{c_i})/q(c_i)$ is unbiased for $Z = \sum_c \exp(s_c)$, and its relative variance is*

$$\frac{\operatorname{Var}(\hat{Z})}{Z^2} = \frac{\chi^2(p\|q)}{M}, \qquad p_c = \frac{\exp(s_c)}{Z}, \quad \chi^2(p\|q) = \sum_c \frac{p_c^2}{q_c} - 1. \tag{7}$$

Under complementary-sum sampling, Eq. (6)*, with an exact head H, M tail draws from q_{tail} on T, and the target's drawn copies masked,*

$$\frac{\operatorname{Var}(\hat{Z})}{Z^2} = \frac{m_{T_t}^2}{M}\, \chi^2(p_{T_t}\|q_{\text{tail}}) \tag{8}$$

exactly. Since q_{tail} need not sum to 1 on T_t when $t \in T$, the quantity called χ^2 there is at least $q_{\text{tail}}(t)/(1 - q_{\text{tail}}(t)) \geq 0$ by Cauchy–Schwarz, a mismatch rather than strictly a divergence.

Proof. Generic proposal. $\mathbb{E}_q[\hat{Z}] = \sum_c q_c \exp(s_c)/q_c = Z$. With M i.i.d. draws,

$$\operatorname{Var}(\hat{Z}) = \frac{1}{M}\Big(\sum_c q_c\big(\exp(s_c)/q_c\big)^2 - Z^2\Big) = \frac{1}{M}\Big(\sum_c \frac{\exp(2s_c)}{q_c} - Z^2\Big),$$

and dividing by Z^2 and writing $p_c = \exp(s_c)/Z$ gives Eq. (7).

Head-tail. The head and force-included terms carry zero variance, leaving the masked tail sum of M i.i.d. terms

$$w_i = \mathbb{1}[c_i \neq t]\, \frac{\exp(s_{c_i})}{M\, q_{\text{tail}}(c_i)}, \qquad \mathbb{E}[w_i] = \frac{1}{M}\sum_{c \in T_t} \exp(s_c). \tag{9}$$

Hence

$$\operatorname{Var}(\hat{Z}) = M \operatorname{Var}(w_1) = \frac{1}{M}\Big(\sum_{c \in T_t} \frac{\exp(2s_c)}{q_{\text{tail}}(c)} - \Big(\sum_{c \in T_t} \exp(s_c)\Big)^2\Big),$$

and dividing by Z^2 gives

$$\frac{1}{M}\Big(\sum_{c \in T_t} \frac{p_c^2}{q_{\text{tail}}(c)} - m_{T_t}^2\Big) = \frac{m_{T_t}^2}{M}\, \chi^2(p_{T_t}\|q_{\text{tail}}),$$

which is Eq. (8). □

Proposition 1 (consistency of the HTSS gradient)**.** *Fix the exact head $H \subseteq \mathcal{V}$ and let the M tail negatives be drawn i.i.d. from q_{tail}, of full support on T. Let the target be force-included at weight $\exp(s_t)$ with its drawn copies masked, as in Eq.* (6)*. Then for* every $M \geq 1$ *the per-position estimators $\hat{Z}$ of the partition function $Z = \sum_c \exp(s_c)$ and $\widehat{\mu}$ of the unnormalized mass $\mu = \sum_c \exp(s_c)\, e_c$ are* exactly unbiased *for their full-vocabulary targets. Consequently, the HTSS representation-gradient $\widehat{\nabla}_h = \widehat{\mu}/\hat{Z} - e_t$ is a* consistent *estimator of the full-softmax gradient ∇_h, and $\widehat{\nabla}_h = \nabla_h$ identically when the exact head enumerates $\mathcal{V}$.*

Corollary 1 (rates)**.** *In the setting of Proposition 1, let $\bar{e} = \mu/Z$ and $B = \max_{c \in T_t} \|e_c - \bar{e}\|$. To leading order in $1/M$, the total variance of $\widehat{\nabla}_h$ (the trace of its covariance) and the norm of its bias satisfy*

$$\operatorname{tr}\operatorname{Cov}(\widehat{\nabla}_h) \;\leq\; \frac{B^2\, m_{T_t}^2\big(1 + \chi^2(p_{T_t}\|q_{\text{tail}})\big)}{M}, \tag{10a}$$

$$\big\|\operatorname{bias}(\widehat{\nabla}_h)\big\| \;\leq\; \frac{B\, m_{T_t}^2 \sqrt{\chi^2(p_{T_t}\|q_{\text{tail}})\big(1 + \chi^2(p_{T_t}\|q_{\text{tail}})\big)}}{M}. \tag{10b}$$

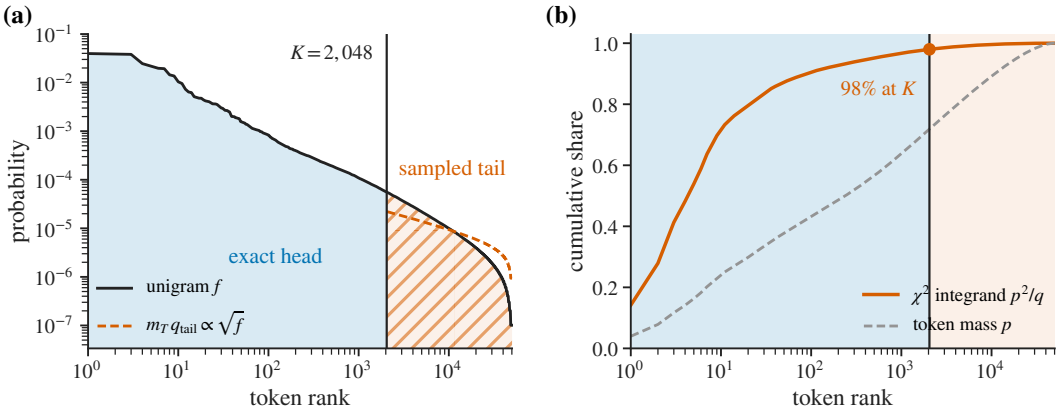

Figure 4: **Summing the head exactly removes most of the partition function's sampling variance.** (a) The GPT-2 unigram split at $K{=}2{,}048$: the head is summed exactly and carries no variance, holding 72% of the token mass from 4% of the types, while the tail is importance-sampled from $q_{\text{tail}} \propto \sqrt{f}$, flatter than f. (b) The χ^2 summand p^2/π of Lemma 1 concentrates on the frequent tokens the head absorbs. With the unigram as the model's conditional, $M\ \mathrm{Var}(\hat{Z})/Z^2$ falls from $\chi^2(p\|\pi){=}5.85$ at $K{=}0$ to $m_T^2\,\chi^2(p_T\|q_{\text{tail}}){=}0.0267$ at $K{=}2{,}048$, $219\times$ lower at equal M, and $195\times$ lower at an equal candidate count ($N{=}18{,}432$ draws at $K{=}0$).

Both are $O(1/M)$, both vanish as $M \to \infty$, and both are exactly zero when the exact head enumerates $\mathcal{V}$. Each row of the embedding gradient obeys the same rate in M, with a constant that grows as $1/q_{\text{tail}}(c)$ for rarely drawn tail rows.

Unbiasedness and consistency follow the argument of Botev et al. (2017) for a general exact set; we restate them for the head-tail estimator with the target forced in and its drawn copies masked, and add the explicit rates of Corollary 1.

Proof of Proposition 1 and Corollary 1. Unbiasedness. Write $\mathcal{V} = H \sqcup T$. The head term of Eq. (6) is deterministic. For the tail, full support of q_{tail} on T gives, for any $A \subseteq T$,

$$\mathbb{E}\Big[\sum_{i:\,c_i\in A} \frac{\exp(s_{c_i})}{M\,q_{\text{tail}}(c_i)}\Big] = \sum_{c\in A} M\,q_{\text{tail}}(c)\cdot\frac{\exp(s_c)}{M\,q_{\text{tail}}(c)} = \sum_{c\in A}\exp(s_c), \tag{11}$$

each of the M draws contributing $\sum_{c\in A} q_{\text{tail}}(c)\exp(s_c)/(Mq_{\text{tail}}(c))$. Masking takes $A = T\backslash\{t\}$, so the drawn copies of the target contribute nothing in expectation, and the force-included term restores $\exp(s_t)$ exactly once. That term carries *no* correction because it enters deterministically, with weight 1 and so correction $\log 1 = 0$. Summing the three pieces, $\mathbb{E}[\hat{Z}] = \sum_{c\in H}\exp(s_c) + \sum_{c\in T}\exp(s_c) = Z$ for every $M \geq 1$, whether $t \in H$ or $t \in T$. Replacing $\exp(s_c)$ by $\exp(s_c)\,e_c$ throughout gives $\mathbb{E}[\hat{\mu}] = \sum_c \exp(s_c)\,e_c$. Both identities are exact at finite M; correcting the force-included target instead would leave $\mathbb{E}[\hat{Z}] = Z + (\omega_t - 1)\exp(s_t)$ with $\omega_t = 1/(Mq_{\text{tail}}(t))$, which equals Z only at $\omega_t = 1$ (Khrylchenko et al., 2025).

Bias. $\widehat{\nabla}_h = \hat{\mu}/\hat{Z} - e_t$ is a ratio of unbiased estimators and is therefore biased at finite M, since $\mathbb{E}[X/Y] \neq \mathbb{E}[X]/\mathbb{E}[Y]$. Expanding to second order about $(\mathbb{E}[\hat{\mu}], Z)$,

$$\mathrm{bias}(\widehat{\nabla}_h) = -\frac{\mathrm{Cov}(\hat{\mu},\hat{Z})}{Z^2} + \frac{\mathbb{E}[\hat{\mu}]\ \mathrm{Var}(\hat{Z})}{Z^3} + o(1/M) = -\frac{\mathrm{Cov}(\hat{\mu}-\bar{e}\,\hat{Z},\ \hat{Z})}{Z^2} + o(1/M), \tag{12}$$

and Lemma 1 gives $\mathrm{Var}(\hat{Z})/Z^2$ as Eq. (8).

Variance. To first order, $\widehat{\nabla}_h - \nabla_h \approx (\hat{\mu} - \bar{e}\,\hat{Z})/Z$, a masked tail sum of M i.i.d. terms $(w_i/Z)(e_{c_i} - \bar{e})$, with w_i as in Eq. (9), plus a deterministic head-and-target term. Its total variance is

$$\frac{1}{M}\Big(\sum_{c\in T_t}\frac{p_c^2\,\|e_c-\bar{e}\|^2}{q_{\text{tail}}(c)} - \Big\|\sum_{c\in T_t} p_c(e_c-\bar{e})\Big\|^2\Big) \ \leq\ \frac{B^2\,m_{T_t}^2\big(1+\chi^2(p_{T_t}\|q_{\text{tail}})\big)}{M}, \tag{13}$$

which is Eq. (10a). This variance does not vanish even when $t \in H$ and $q_{\text{tail}} \propto p$ on T, where $\text{Var}(\hat{Z}) = 0$, since the embeddings still differ across the draws. Cauchy–Schwarz bounds the covariance in Eq. (12) by the square root of the product of the variances in Eqs. (8) and (13), which is Eq. (10b). All are finite because q_{tail} has full support on the finite T and the embeddings are bounded on any compact parameter set. Hence the bias and the variance of $\widehat{\nabla}_h$ are both $O(1/M)$, with a constant proportional to $m_{T_t}^2\big(1+\chi^2(p_{T_t}\|q_{\text{tail}})\big) = \sum_{c\in T_t} p_c^2/q_{\text{tail}}(c)$, and both vanish as $M \to \infty$, which is the consistency claimed. The same argument, applied to one row e_c of E at a time, gives the embedding gradient the same rate, with a constant that grows as $1/q_{\text{tail}}(c)$ for rarely drawn tail rows. Finally, if $H = \mathcal{V}$ the tail is empty, $\hat{Z} = Z$ and $\widehat{\mu} = \sum_c \exp(s_c)\, e_c$ identically, so $\widehat{\nabla}_h = \nabla_h$. □

A.2 The exact head never increases the variance

Proposition 1's unbiasedness is exact at every M; the bounds of Corollary 1 are leading-order in $1/M$ (we train with M=16,384 of N=18,432 candidates at GPT-2), accurate when $m_T^2\chi^2/M \ll 1$ for the model's per-context χ^2 (App. §C.5). The head adds candidates but no variance, so the rate is governed by M rather than $K+M$, and summing the top-K exactly never increases the tail factor $m_T^2\chi^2(p_T\|q_{\text{tail}})$ of Eq. (8).

Proposition 2 (the exact head never increases the variance)**.** *Let π be a full-support proposal over $\mathcal{V}$, $\emptyset \neq H \subsetneq \mathcal{V}$ with $Q_H = \pi(H)$, and $q_{\text{tail}} = \pi|_T/(1-Q_H)$. For the partition estimate without a forced target, M i.i.d. draws from π over $\mathcal{V}$ ($\hat{Z}_\pi$: Lemma 1 with $q=\pi$) and the head-tail estimate $\hat{Z}_{\text{HT}}$ with M draws from q_{tail} satisfy*

$$\frac{M\,\text{Var}(\hat{Z}_{\text{HT}})}{Z^2} \;\le\; \frac{M\,\text{Var}(\hat{Z}_\pi)}{Z^2}, \tag{14}$$

with equality iff $p = \pi$.

Proof. Write $m_H = 1 - m_T$. By Lemma 1, the difference of the two sides of Eq. (14) is

$$D = \sum_{c\in H}\frac{p_c^2}{\pi_c} + \sum_{c\in T}\frac{p_c^2}{\pi_c} - 1 - (1-Q_H)\sum_{c\in T}\frac{p_c^2}{\pi_c} + m_T^2 = \sum_{c\in H}\frac{p_c^2}{\pi_c} + Q_H\sum_{c\in T}\frac{p_c^2}{\pi_c} - 1 + m_T^2.$$

Cauchy–Schwarz gives $\sum_H p_c^2/\pi_c \ge m_H^2/Q_H$ and $\sum_T p_c^2/\pi_c \ge m_T^2/(1-Q_H)$, so

$$D \;\ge\; \frac{m_H^2}{Q_H} + \frac{Q_H(1-m_H)^2}{1-Q_H} - 1 + (1-m_H)^2.$$

The right side is convex in m_H, with derivative $2m_H/Q_H - 2(1-m_H)/(1-Q_H)$ vanishing at $m_H = Q_H$, where it equals 0. Hence $D \ge 0$. □

Remark 1 (Rao-Blackwell reading). Proposition 2 also follows from the Rao-Blackwell theorem: let each π-draw take a tail candidate $u_i \sim q_{\text{tail}}$ with probability $1-Q_H$ and a head token from $\pi|_H/Q_H$ otherwise; conditioning the π-sampled estimate on $u_1, \ldots, u_M$ gives Eq. (4) exactly, also with the target forced in and masked when the π-estimate does the same. Both hold for every p and every choice of H; heuristically, the gain is large when H holds the tokens with large p^2/π.

Remark 2 (what shrinks, and what need not). Moving a token into H shrinks m_T and never increases the product $m_T^2\chi^2(p_T\|q_{\text{tail}})$ (Prop. 2 applied to the tail with $\pi = q_{\text{tail}}$), but not necessarily the normalized $\chi^2(p_T\|q_{\text{tail}})$ alone: with $\pi = (0.5, 0.25, 0.25)$ and $p = (0.5, 0.3, 0.2)$, putting the first in H doubles χ^2 from 0.02 to 0.04 while $m_T^2\chi^2$ falls to 0.01.

A.3 Relation to NCE and to model-aware proposals

The head+tail scheme and NCE. Equation (3) is the ranking-based conditional NCE of Ma & Collins (2018) with noise q, whose consistency theorem gives the population minimizer $s_c = \log p^*(c\,|\,h) + \kappa(h)$ for any full-support proposal and any number of negatives, but requires the correction on the positive as well. It therefore certifies Eq. (3) as written, not the scheme we train, which force-includes the positive and masks its drawn copies. For that scheme the guarantee is Proposition 1: $\hat{Z}$ is unbiased for Z at every M, so $\mathbb{E}[\mathcal{L}_{\text{HTSS}}] \to \mathcal{L}_{\text{CE}}$ as $M \to \infty$ and the minimizer approaches $\log p^* + \kappa(h)$, with an $O(1/M)$ Jensen shift at finite M (§3.2). Because the positive

enters with its *raw* $\exp(s_t)$ and its drawn copies are masked, $\mathbb{E}[\hat{Z}] = Z$ holds at every M rather than only in the limit, and only the sampled negatives need a correction; the proof of Proposition 1 gives the bias of the alternative.

Why a static proposal. A per-context model-aware proposal (Chen et al., 2025) concentrates q on the tokens that scored high at its last refresh. That lowers χ^2 but makes the weights stale: a rare token whose score rises between refreshes gets a large weight $\exp(s_c)/q(c)$, a single draw can then dominate $\hat{Z}$, and gradient clipping bounds the step's *size* but not its heavy-tailed *direction*. HTSS has no refresh to go stale. The exact head carries no importance weight at all, and a static full-support q_{tail} fixes each tail weight's denominator, so a weight grows only with the token's own score. The $\sqrt{f}$ tempering gives rare tokens more proposal mass than the unigram does, which lowers their weights, though a rare token whose score rises still gets a large one.

B SHARED EXPERIMENTAL PROTOCOL

The per-experiment recipes are in Apps. §C, §D and §E.

B.1 DATASETS

Pretraining uses FineWeb-Edu (Penedo et al., 2024) for the GPT-2 comparison and, for the vocabulary sweep, a multilingual mixture of FineWeb-EN and 28 FineWeb-2 languages (Penedo et al., 2025), chosen so that its large vocabularies are occupied (App. §D.2, which also gives the tokenizer-training protocol for the vocabulary sweep). Finetuning uses SmolTalk (Ben Allal et al., 2025) at a pinned revision (App. §E). Vocabulary occupancy (Table 5) is also measured on FineWeb-EN and WikiText-103 (Merity et al., 2017). Corpora are shuffled before splitting, and validation and test splits are disjoint from training.

B.2 COST ACCOUNTING

We price the comparison in training FLOPs because wall-clock depends on how well each objective's kernels are optimized, and CE's have had far more engineering than any sampled objectives. A better HTSS kernel could turn more of the FLOP saving into wall-clock time. GPU-h counts training only (forward pass, backward pass, optimizer step). A *matched compute* row reports its *matched steps* counterpart's peak memory, since the two run the same step. At matched FLOPs, HTSS-mc's wall-clock is 0.98× CE's on GPT-2 (Table 1), 1.02–1.04× on the vocabulary sweep and 1.05–1.11× in finetuning (Tables 2 and 3), where our kernel turns less of the FLOP saving into throughput.

B.3 THE MATCHED-COMPUTE RUNS REUSE CE'S DATA AND SCHEDULE

Our protocol puts HTSS-mc at a slight disadvantage. Its corpus is a fixed token count sized for CE's budget, so the extra steps revisit data already seen (tied GPT-2: about 1.7 passes against CE's 1.4), which is worth less than fresh data (Muennighoff et al., 2025). And we set the learning rate and schedule once, for CE, each run annealing over a fixed final fraction of its own steps (10% for GPT-2, 20% in finetuning), so HTSS-mc runs CE's schedule stretched instead of one tuned for its budget. Neither was tuned for HTSS-mc. Every run uses AdamW (Loshchilov & Hutter, 2019) and a warmup–stable–decay (WSD) schedule (Hu et al., 2024).

B.4 THE MATCHED-COMPUTE HAND-OFF

The matched-compute runs are branch continuations. In every setting, HTSS-mc does not retrain the plateau it shares with HTSS: it resumes HTSS's checkpoint at a step τ before either run's WSD decay onset. Under a warmup–stable–decay schedule both runs have identical learning rates everywhere before that onset, which makes the prefix HTSS-mc's own first τ steps. Each run reports its full standalone cost. The consequence for statistical inference is that the two HTSS rows within a setting are not independent samples: they share a data order and a prefix.

C GPT-2 COMPARISON: HYPERPARAMETERS AND MEASUREMENT DETAIL

C.1 HYPERPARAMETERS

Hardware, budget and cost accounting. All ten runs of Table 1 used one 80 GB A100 at one micro-batch factorization ($32\times$grad-accum 16, seq 1,024, so the global batch is GPT-2's own 524,288 tokens), so the cost columns are comparable across both settings (App. §B). Each setting takes its own Chinchilla-optimal budget from the parameter count, and untied runs 6,220 CE steps against 4,748 for tied (2.49B tokens). The untied setting is a second measurement of the same effect. CE-CCE is the *same* objective as CE run through the released Cut Cross-Entropy implementation (Wijmans et al., 2025) with its default settings, at the same nominal FLOPs (its kernel also recomputes the logits in the backward pass), data order, budget, GPU and factorization. All runs train in bf16 mixed precision. At its defaults, CCE skips the classifier gradient for blocks whose softmax values all fall below 2^{-12} (gradient filtering) and accumulates each classifier row's gradient in a bf16 buffer, rounding once per token block, while CE and HTSS, eager and fused, accumulate in fp32 and round once. Wijmans et al. (2025) note that filtering the classifier gradient leaves tokens with little support in the training data without gradient, harmless in finetuning but not in pretraining, and provide a Kahan-summation variant without that filtering for pretraining, which we did not run. We attribute CE-CCE's perplexity gap to these defaults, untested; that the gap is larger untied (+19.1%) than tied (+7.3%) fits filtering, since an untied classifier row gets no gradient from the input side.

Optimizer and schedule. All runs use AdamW at learning rate 6×10^{-4}, weight decay 0.1 applied to parameters with $\mathrm{ndim} \geq 2$, so norms and biases are exempt, gradient clipping at 1.0, bf16, and a warmup–stable–decay schedule with a 700-step linear warmup from ≈ 0 and a decay over the final 10% of steps. On a tied model, the weight decay also applies to E, the layer whose objective differs between runs. We hold it identical across runs: CE, CWT and both HTSS variants use one optimizer configuration, and the untied setting, where E is a separate decayed matrix, measures the same effect again. Gradient clipping, which keeps $\|E\|$ stable, is likewise shared by every run, so any difference in stability comes from the objective. The schedule is WSD instead of llm.c's cosine because the matched-compute hand-off (App. §B) requires a shared constant-learning-rate plateau; every other knob follows llm.c's GPT-2 recipe (Karpathy, 2024).

C.2 PER-MEASUREMENT DETAIL

The matched-steps deficit holds along the whole plateau. Within each run, the paired validation trajectory puts HTSS behind CE at 35 of 36 plateau evaluation points tied and 36 of 36 untied, with plateau means of +1.5% and +1.8% in validation perplexity, against final test-perplexity gaps of +1.3% (23.65 vs. CE's 23.35) and +1.4%. The consistent sign rules out an unlucky final checkpoint.

Accuracy at matched compute is flat across top-k. Tied, HTSS-mc gains +0.0032 at top-1, +0.0048 at top-5 and +0.0045 at top-10 over CE, while the matched-steps run is uniformly and marginally behind (−0.0012, −0.0006, −0.0008). The matched-compute perplexity margin is −2.9% on test and −3.8% on validation; untied the same margins are −2.2% on test and −1.6% on validation.

Frequency deciles: where the deficit falls, and where the gain does. Split into equal-count vocabulary deciles of 5,026 types, the matched-steps deficit is confined to the most frequent tokens: HTSS is 1.8% worse than CE in perplexity on the first decile, matches or edges ahead of it through the middle of the vocabulary, and leads by 6–18% on the rarest four (tied; the untied setting has the same shape). The first decile carries most of the token mass, and the aggregate is still behind.

The two ends respond to different parts of the estimator. The first-decile deficit is insensitive to the exact head: it persists at $K{=}8{,}192$, where the head holds every type in that decile, so it does not come from sampling head tokens. The sampled normalizer, shared by every token's gradient, is the remaining candidate; the N_{eff} pair of App. §C.5 detects no gradient-noise effect and cannot probe the Jensen term, so we leave the mechanism open. The tail advantage is set by the proposal exponent. Tempering to $f^{0.25}$ removes it, and $q_{\mathrm{tail}} \propto f$ deepens it to 39% on the rarest decile while costing $\approx 3\%$ on the three most frequent. In aggregate test perplexity they lose to $\sqrt{f}$ by 0.45% and 1.0%, so $\sqrt{f}$ balances the two ends. Since the importance weight keeps $\hat{Z}$ unbiased, the tilt is a systematic effect of finite M on the update, which the consistency result allows but does not characterize.

Table 4: **Zero-shot downstream evaluation of the GPT-2 checkpoints.** lm-eval (Sutawika et al., 2026) on the checkpoints of Table 1: ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), LAMBADA (Paperno et al., 2016), PIQA (Bisk et al., 2020), SciQ (Welbl et al., 2017) and WinoGrande (Sakaguchi et al., 2021). Accuracy is `acc_norm` where lm-eval reports it and `acc` otherwise; the last column is LAMBADA perplexity, excluded from Avg. Budget as in Table 1; bold is best of CE and the two HTSS rows per group.

		Accuracy ↑								Ppl ↓
Method	Budget	ARC-c	ARC-e	HellaSwag	LAMBADA	PIQA	SciQ	WinoG.	Avg	LAMBADA
Tied — GPT-2, 124M params										
CE	ref.	0.243	0.438	0.292	0.206	0.601	**0.652**	0.506	0.420	242.6
CE-CCE	steps	0.241	0.417	0.292	0.194	0.607	0.599	0.500	0.407	306.6
CWT	steps	0.255	0.361	0.284	0.161	0.585	0.505	0.517	0.381	826.5
HTSS	steps	0.252	0.438	0.294	0.221	**0.605**	0.631	**0.517**	0.423	221.7
HTSS-mc	compute	**0.256**	**0.441**	**0.296**	**0.232**	0.598	0.648	0.511	**0.426**	**191.7**
Untied — GPT-2, 163M params										
CE	ref.	0.252	0.443	0.299	0.208	0.603	0.644	0.513	0.423	218.9
CE-CCE	steps	0.235	0.421	0.294	0.194	0.589	0.598	0.528	0.408	307.5
CWT	steps	0.281	0.371	0.290	0.151	0.583	0.522	0.526	0.389	789.4
HTSS	steps	**0.253**	0.452	0.299	0.225	0.595	0.650	**0.523**	0.428	197.9
HTSS-mc	compute	0.249	**0.462**	**0.306**	**0.232**	**0.609**	**0.659**	0.521	**0.434**	**162.1**

C.3 Zero-shot evaluation

CE-CCE's test-perplexity gap carries over zero-shot. CE-CCE's LAMBADA perplexity is 306.6 tied and 307.5 untied, against CE's 242.6 and 218.9, and its average accuracy 0.407 and 0.408 against 0.420 and 0.423, in line with its +7.3% and +19.1% test perplexity in Table 1.

LAMBADA significance. LAMBADA is the only task in the suite scored in absolute terms. Because the other six rank a handful of supplied options, they see relative scores alone, whereas LAMBADA scores one held-out final token over the full vocabulary. That split is consistent with the CWT control failing more mildly on the ranking tasks than its $\approx 5\times$ test-perplexity gap suggests, and with LAMBADA being where a missing correction shows at full size. Against lm-eval's own stderr, the matched-compute perplexity reduction clears it ($z{=}-3.3$) while the matched-steps reduction alone does not (1.09×, $z{=}-1.3$; untied 1.11×, $z{=}-1.4$). On accuracy, the untied setting gives $z{=}2.1$ at matched steps and $z{=}3.0$ at matched compute; tied, the matched-steps run already improves accuracy at the edge of its interval ($z{=}1.9$) while sitting *behind* on aggregate perplexity, because its deficit is concentrated in the most frequent decile. lm-eval's stderr is sampling noise over the evaluation set alone, so these z-scores show only that the differences exceed evaluation noise.

C.4 The CWT control: its readout and its cost

The in-batch pool saves no memory at these shapes. CWT's in-batch pool ($n{=}32{,}768$ targets per micro-batch) gives an $n{\times}n$ similarity matrix of the same order as CE's $n{\times}V$ logit matrix at $V{=}50{,}257$ ($n/V{=}0.65$), against $(K{+}M)/V{=}0.37$ for HTSS (§3.4). Measured peak memory puts CWT 1.9–2.0 GB *above* CE in both settings (Table 1), where HTSS is 4.8–4.9 GB below: HTSS's saving comes from its smaller candidate set, which the contrastive form by itself does not supply.

Equation (2) predicts that CWT's scores are $\log p^*$ deflated by $\log q$. We test it on the held-out evaluation subset of the fixed-compute sweep (App. §C.5), where CE reaches 23.90 test perplexity. Absolute values there are higher than Table 1's. We compare runs only within that subset. *The inference-time shift.* We score a trained tied CWT checkpoint with $s_c + \beta \log q(c)$ and no retraining. At $\beta{=}1$, the value Eq. (2) predicts and the one we quote, test perplexity falls from 123.1 to 33.9. Tuning β on validation adds little, $\beta^*{=}0.9$ reading 32.8 with top-1 up from 0.306 to 0.394, which leaves the offset close to the value the derivation predicts; $\kappa(h)$ cancels in the softmax. The offset is therefore quantitatively the unigram term, which suggests that absorbing it is most of what the 10K-step cross-entropy stage of Godey et al. (2024) recovers. *The correction applied in training, at CWT's own pool.* CWT $+ \log q$, trained with Eq. (3) on the same in-batch negatives, reaches 34.3,

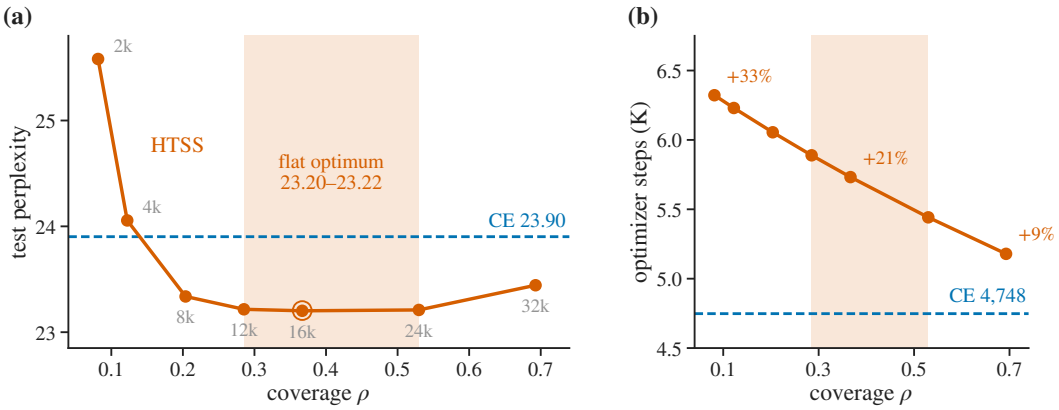

Figure 5: **Number of tail draws at fixed compute.** (a) Test perplexity, with M in grey; the open circle is a repeat run at M=16,384. (b) Optimizer steps the budget buys. Dashed: CE at the same budget. Shaded: the flat optimum.

no better than the free inference shift, and both are ≈42–44% above CE's 23.90 on this subset. The remaining gap therefore lies in the candidate set, which differs in scheme and coverage: against the in-batch pool's ≈6.8K distinct tokens of §4.2, HTSS's M=16,384 tail draws cover ≈13K distinct tokens besides its 2,048-token head (distinct coverage ≈0.30), both measured on FineWeb-Edu.

C.5 Choosing K and M: the error decomposition and the fixed-compute frontier

What each knob moves. Section 3.4's recipe rests on a decomposition of HTSS's matched-*steps* deficit relative to CE (in validation loss):

$$\Delta_{\text{step}} = \underbrace{\Delta_{\text{CE}_0}}_{\text{noise, mean 0}} + \underbrace{\gamma_{\text{bias}}\,\phi\big(m_T^2\chi^2/M\big)}_{\text{Jensen bias, } O(1/M)} + \underbrace{\Delta_{\text{traj}}(N_{\text{eff}})}_{\text{gradient noise, } \propto 1/N_{\text{eff}}}, \qquad N_{\text{eff}} = M{\cdot}\text{ga}{\cdot}\text{world}, \quad (15)$$

where m_T is the model's tail mass of §3.3 (0.293 on the trained GPT-2 checkpoints), χ^2 is Lemma 1's tail-restricted divergence $\chi^2(p_T\|q_{\text{tail}})$, and ϕ is the concave mean-of-log map, applied per context and then averaged; to first order in $1/M$ its per-context value is $m_T^2\chi^2/(2M)$ in magnitude, the second-order Jensen gap of $\log\hat{Z}$. Here χ^2 is taken per context, against the trained model's conditional $p(\cdot\,|\,h)$; it is far larger than the unigram value of Figure 4 (0.0267), which would put the Jensen gap near 10^{-6} nats and does not bound it. Equation (15) states the recipe's rationale—which term each knob moves—and stops short of a calibrated model: we fit neither γ_{bias} nor χ^2.

The two M-dependent terms respond to different knobs. Raising M shrinks the $O(1/M)$ Jensen bias, but it costs FLOPs, entering $K{+}M$ and lowering g. The product $\text{ga}\cdot\text{world}$ is the variance term's nominal knob and adds no candidate FLOPs, and a pair of runs isolates it. At GPT-2's fixed global batch and coverage, doubling the micro-batch from 32 to 64 sequences halves ga and so N_{eff}, and plateau validation perplexity moves by –0.7%, lower at 13 of 15 paired constant-learning-rate points. Halving the independent draws doing slightly better is the opposite of what a dominant variance term predicts, although the pair also changes how many positions share one draw. We detect no variance effect in aggregate perplexity, and we set M based solely on coverage. Averaging more draws cannot touch the bias in any case: each of the $\text{ga}\cdot\text{world}$ log-partition estimates keeps an $O(1/M)$ Jensen gap, and averaging them leaves it in place.

The fixed-compute sweep. Figure 5 reports seven tied GPT-2 runs at K=2,048, one per number of tail draws, each given CE's 2,127-PFLOP training budget. More tail draws leave fewer steps, and the comparison is matched compute by construction. We read the sweep alongside CE at the same budget (4,748 steps). The sweep is scored on a separate held-out subset, where CE reaches 23.90 against Table 1's 23.35; comparisons are again confined to it.

Break-even lies between ρ=0.12 and 0.20. At M=2,048 (ρ=0.08) HTSS is furthest behind CE, by about 7% in perplexity; at M=4,096 it is 0.6% behind; at M=8,192 it leads by 2.4%, both at CE's training FLOPs.

The optimum is a flat range, and M=16,384 is chosen on validation. On test, $M \in \{12{,}288, 16{,}384, 24{,}576\}$ (ρ=0.29–0.53) lie within 0.1% in perplexity of each other, while M=8,192 and M=32,768 are 0.6% and 1.0% above the minimum: past the optimum more tail draws lose more steps than their smaller estimator error is worth. On validation, the split we select on, M=16,384 is the best of the three. Its margin over CE at matched compute on this subset, −2.9%, is the same as Table 1's test margin on the main test set.

The classical sampled softmax (K=0), and why K is held fixed. K and M share one candidate count, and we set them by different arguments. M is set by the coverage of §3.4, whose value at fixed compute is the sweep above. K is set by Lemma 1: χ^2 is largest where p^2/q is, and a head of the few thousand most frequent tokens already contains the terms that dominate it, and we hold K=2,048 across every model and vocabulary in the paper instead of retuning it per configuration. To test the split itself, we hold the candidate count at $K+M$=18,432, which costs every run the same FLOPs and memory, and move it between head and tail (Table 1's recipe, 4,748 steps). Test perplexity is 23.84 at K=0, the classical sampled softmax, 23.65 at K=2,048 and 23.60 at K=8,192, against CE's 23.35: the exact head is the better use of a fixed candidate count, and moving the first 2,048 tokens into it cuts the gap to CE from 2.1% to 1.3% while the next 6,144 take it only to 1.1%. K=2,048 captures most of the gain, and at a fixed candidate count a larger head adds little more. Part of that gain may come from distinct coverage: with replacement, 28% of K=0's draws are repeats, against 14% at K=8,192, so a larger head scores more distinct tokens (13,279 against 17,014).

C.6 The fused HTSS kernel: tensors and memory

CCE fuses the reduction over V; our kernel fuses the smaller reduction over $K+M$. Table 1 runs HTSS unfused, which is the only reason its memory column is above CE-CCE's. Rerunning the GPT-2 HTSS configurations through the fused kernel that the vocabulary-sweep runs use brings peak training memory to 28.1 GB tied and 28.5 untied, level with CE-CCE's 28.1 and 28.2, with test perplexity +0.2% from the unfused HTSS run. Fusion alone makes CE 1.12× faster at the same nominal FLOPs (CE-CCE), since it avoids the materialized fp32 softmax; the fused HTSS kernel runs a further 1.25× faster than CE-CCE, about 1.4× CE's throughput.

D The vocabulary sweep

D.1 Hyperparameters

Each run is a Llama-style decoder (RoPE, SwiGLU, RMSNorm) with d=1,024, L=24, 16 attention heads, SwiGLU width 2,752 and untied input and output embeddings, trained at sequence 2,048 on the multilingual mixture of App. §B (FineWeb-EN and 28 FineWeb-2 languages), with only the tokenizer's vocabulary size moving across runs. The global batch is 2,097,152 tokens, held at every V. Each V sets its own budget at 20 tokens per total parameter, counting the input and output embeddings, so a larger vocabulary trains longer. AdamW at learning rate 3×10^{-4}, weight decay 0.1 on parameters with $\mathrm{ndim} \geq 2$, gradient clipping 1.0, bf16, and a warmup–stable–decay schedule whose ramp is set in tokens instead of steps (375M tokens, so 179 steps at this batch) so that the ramp is the same in every run although their step counts differ. K=2,048, $M=V/4$ rounded to a power of two, so coverage falls from ρ=0.32 at 32K to 0.26 at 256K: below GPT-2's flat optimum (ρ=0.29–0.53) but above the ρ=0.20 of M=8,192, which costs 0.6% there (App. §C.5). If GPT-2's coverage response transfers, the drift works against HTSS at large V by at most 0.6% in perplexity, so it cannot produce the trend with V. The corpus holds ≈6.7B unique tokens against per-run CE budgets from ≈7.4B tokens (3,509 steps) at V=32K to ≈16.5B (7,884 steps) at V=256K, so CE at 256K makes about 2.5 passes and HTSS-mc, at g=1.41, about 3.5. On four A100-64GB GPUs per run, HTSS uses our fused kernel, CE the materialized softmax, and HTSS-mc resumes HTSS's plateau (App. §B).

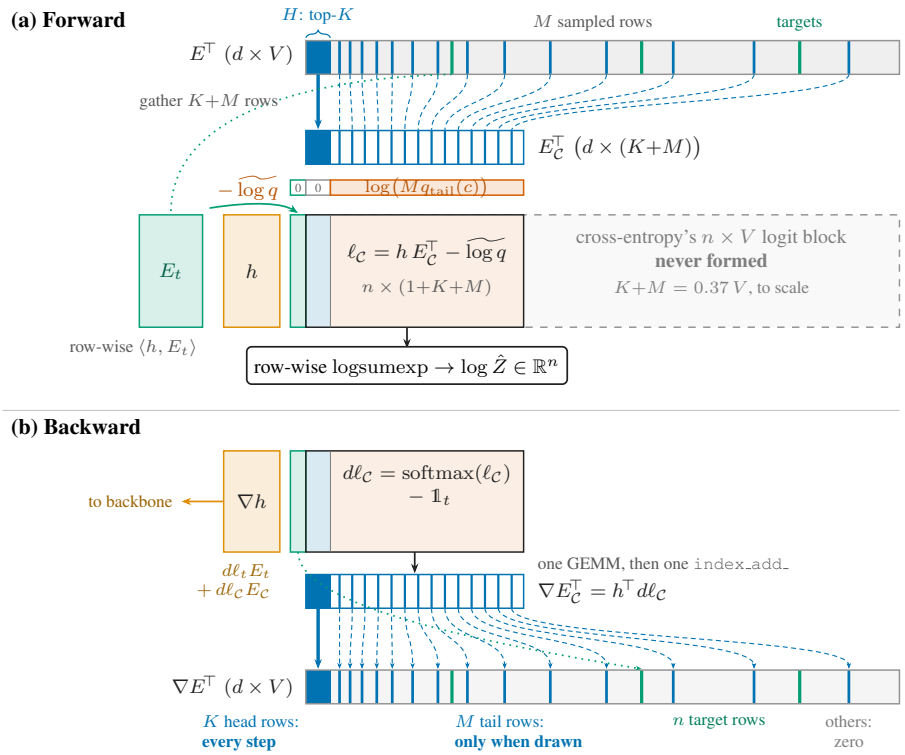

Figure 6: **One HTSS step in tensors.** The K head rows and M sampled rows of E are gathered once into $E_{\mathcal{C}}$ and shared by the micro-batch's n positions; each target row E_t gives s_t, and candidate columns equal to a row's target are masked. The $n \times (1{+}K{+}M)$ score block replaces the $n \times V$ logit matrix, so memory and FLOPs scale with $K{+}M$, not V. Backward, `index_add_` sums duplicate draws: every head row gets a gradient each step, a tail row only when drawn. Drawn at GPT-2's scale ($K{+}M{=}0.37\,V$), rows in frequency order.

Table 5: **Vocabulary occupancy and fertility.** Each corpus has its own byte-level BPE tokenizer at each V. Occupancy: fraction of the V types emitted at least once on held-out text. Fertility: mean subword tokens per whitespace word (lower is better). Bold: the $V{=}256$K values discussed in the text.

V	Mix (FineWeb-EN + FineWeb-2, 28 lang), held-out Occupancy	Fertility	FineWeb-EN Held-out occ.	WikiText-103 Held-out occ.
32K	99.7%	2.537	96.9%	68.3%
64K	99.5%	2.280	84.0%	42.5%
128K	98.3%	2.072	57.2%	22.6%
256K	**95.1%**	1.907	**32.6%**	**11.3%**

D.2 Vocabulary occupancy: why a multilingual corpus

Section 4.3 rests on the larger vocabularies being used—otherwise "scaling V" mostly adds untrained rows, and any objective insensitive to dead tokens looks good for an accounting reason. For each corpus, we train a byte-level BPE tokenizer (`min_frequency`=2) at each vocabulary size and measure **occupancy** (fraction of V emitted at least once) and **fertility** (mean subwords per whitespace word) on held-out documents of the same corpus, disjoint from those the tokenizer was trained on.

English-only saturates well before the largest vocabulary (Table 5): on web-English, only 32.6% of a 256K vocabulary recurs on held-out text, and the count of distinct tokens English reuses saturates: it grows only from ∼73K to ∼83K as V doubles from 128K to 256K on FineWeb-EN, and stays near ∼29K from 128K on WikiText-103, which is too small to form 256K types: BPE stops at 169K,

so Table 5's 11.3% is over the nominal 256K and ≈17% over the types that form. *Train* occupancy stays ≈99% (FineWeb-EN), so these are training-set-specific rarities the model would see a handful of times. An English-only scaling sweep would therefore measure how gracefully a method tolerates an increasingly *dead* vocabulary, leaving its scaling with output spaces in use unmeasured. The multilingual mix removes this confound: at 256K it is 95.1% occupied (≈243K live tokens) with fertility still falling. This is also the regime that motivates the method: the largest vocabularies in use, such as Qwen3.5's 248K, serve multilingual models. Hence, the sweep trains on the mix and reports **bits-per-byte**, comparable across V.

E SmolTalk finetuning: recipe and downstream evaluation

Shared recipe. Qwen3.5-0.8B uses hybrid gated-linear attention and Llama 3.2 1B standard attention. SmolTalk (Ben Allal et al., 2025)[1] in chat format, packed with document masking so no attention crosses a conversation boundary: 180,000 training conversations and 3,000 held-out ones split into validation and test. The budget is 10^8 packed tokens at 131,072 tokens per optimizer step. CE runs 763 steps. AdamW at 2×10^{-5}, weight decay 0, gradient clipping 1.0, a warmup–stable–decay schedule with a 25-step warmup and a decay over the final 20% of each run's own steps, bf16 autocast over fp32 master weights, and `sdpa` attention. HTSS holds K=2,048 and sets M per vocabulary at the operating coverage $\rho=(K+M)/V$=0.30, giving M=72,448 (Qwen) and M=36,428 (Llama). HTSS-mc resumes HTSS's WSD plateau as in App. §B. Our logged training FLOPs give g=1.26 and 1.14 (Eq. (5) with η=28.9% and 17.5% from §1 predicts 1.25 and 1.14), hence 964 and 869 HTSS-mc steps. We run fused HTSS without activation checkpointing (2×4,096 per micro-batch, grad-accum 16), each run on one H100-80GB GPU.

Table 6: **Downstream evaluation of the finetuned checkpoints.** Accuracy in % on IFEval (Zhou et al., 2023) (prompt and instruction level) and GSM8K (Cobbe et al., 2021) (strict and flexible), and SmolTalk assistant-token perplexity. *base* is the model before finetuning; bold is best of CE and the two HTSS rows per model.

	Qwen3.5-0.8B, V=248,320					*Llama 3.2 1B, V=128,256*				
	IFEval ↑		GSM8K ↑		Ppl ↓	IFEval ↑		GSM8K ↑		Ppl ↓
Method	prompt	instr.	strict	flex.	asst.	prompt	instr.	strict	flex.	asst.
base	35.49	47.60	15.85	47.84	2.777	21.63	35.13	4.32	4.62	3.319
CE	39.56	51.56	**48.67**	**48.90**	2.245	**27.36**	**39.69**	11.52	11.90	2.335
HTSS	38.82	51.08	47.92	47.99	2.249	25.51	38.97	10.46	10.69	2.343
HTSS-mc	**41.04**	**53.84**	47.84	47.92	**2.225**	26.62	**39.69**	**11.75**	**12.21**	**2.325**

Downstream evaluation. Table 6 evaluates these checkpoints zero-shot with lm-eval (Sutawika et al., 2026), alongside an assistant-token-only loss on the held-out SmolTalk test split (1,244,289 assistant tokens over 1,998 conversations for Qwen, 1,202,293 for Llama). Assistant perplexity preserves the ordering of Table 3. On assistant tokens alone, HTSS is +0.17% and +0.36% against CE, and HTSS-mc −0.87% and −0.40%, close to the full-sequence values of §4.4. CE gains 4.1 and 5.7 points of IFEval prompt-level strict accuracy over the base model, and HTSS 3.3 and 3.9. GSM8K strict-match accuracy rises from 15.9 to 48.7 (Qwen) and 4.3 to 11.5 (Llama), for Qwen, mostly formatting (its base scores 47.8 flexible-extract).

No difference reaches twice its standard error. We write z for a difference from CE over the pooled lm-eval standard errors of the two runs, which lm-eval does not report for instruction-level IFEval. All 12 comparisons that carry one lie within $|z| \le 0.98$; the largest $|z|$ is Llama's matched-steps HTSS on GSM8K flexible-extract (−1.21 points), and the largest gap, −1.85 on Llama's IFEval prompt level, has a smaller $|z|$. With pooled standard errors of 1.2 to 3.0 points (GSM8K to IFEval prompt level), a suite this size would not resolve a difference below about 2.5 to 6 points. Matched-steps HTSS trails CE on all eight metrics, about four independent comparisons since each model's metrics score two response sets twice; we report the pattern but do not treat it as significant.

[1]Hugging Face IDs: `Qwen/Qwen3.5-0.8B-Base`, `meta-llama/Llama-3.2-1B`, and `HuggingFaceTB/smoltalk` (config `all`, revision `5feaf2fd`).