Sparse Token-Level Teacher Queries for Efficient On-Policy Distillation

Xiang Fu
Faculty of Computing and Data Sciences
Boston University
xfu@bu.edu

Abstract

On-policy knowledge distillation trains a compact student language model on its own generated trajectories, reducing train-inference mismatch, but it is expensive because the teacher must score every generated token. We propose FORKDISTILL, a sparse token supervision method that reduces teacher computation by querying the teacher only at a small subset of token positions. Given a student rollout, ForkDistill assigns each position an uncertainty score (predictive entropy or logit margin), selects approximately a target coverage of tokens, adds a small guard prefix and suffix, and applies inverse-probability weights to correct for token subsampling. We distill a Gemma 3 1B student from a Gemma 3 4B instruction-tuned teacher using LoRA and evaluate on short-answer question answering. In a coverage sweep on SciQ, ForkDistill reduces teacher-scored tokens from 128 to 12 per example at 10% coverage (10.7$\times$ fewer queries) while maintaining a competitive distillation objective. At an equal 10% query budget, entropy gating yields a 25% lower final loss than random selection, indicating that supervising high-uncertainty forking points is especially beneficial in the low-coverage regime. Overall, sparse token-level teacher queries provide a simple and effective knob for scaling on-policy distillation under constrained teacher compute.

1 Introduction

Large language models (LLMs) enable strong instruction following and reasoning, but their inference and training costs often prevent deployment in latency- and memory-constrained settings such as edge or on-device applications [1, 2, 3, 4, 5]. A common approach to close this capability gap is knowledge distillation (KD), where a smaller student model is trained to imitate a larger teacher [6, 7, 8, 9]. For LLMs, distillation has been explored in both black-box settings that rely only on teacher outputs and in more direct distribution-matching regimes that use teacher probabilities as soft supervision [10, 11, 12].

A major challenge in modern LLM distillation is teacher compute. Many effective recipes supervise the student at the token level, which can require teacher scoring for each generated token and can dominate runtime when the teacher is large [13, 12, 14]. This issue becomes particularly acute in on-policy distillation, where the student learns from its own generated trajectories to reduce training-inference mismatch and exposure bias [15]. On-policy supervision improves robustness, but it increases the number of teacher queries because teacher feedback must be produced on student-generated prefixes rather than a fixed dataset.

Recent work addresses this cost through more compute-aware distillation pipelines. Speculative knowledge distillation interleaves student sampling with teacher verification to reduce wasted teacher computation [16, 17]. Other approaches adaptively mix generation sources or supervision modes to trade off stability and cost [18, 19]. In parallel, a growing line of token-adaptive and selective distillation methods argues that not all tokens contribute equally to learning, motivating token reweighting, gating, or pruning mechanisms [20, 21, 22, 23, 24].

In this project, we ask: can we reduce teacher supervision to a small fraction of tokens while preserving distillation quality? We propose ForkDistill, a sparse token supervision method for on-policy distillation. ForkDistill selects a subset of generated tokens for teacher scoring using lightweight heuristics (uncertainty-based or random) and applies inverse-probability importance weights so that sparse supervision remains a principled estimator of dense supervision

[25, 26, 27]. Empirically, we show that supervising as little as 10% of tokens can match dense supervision on short-form question answering benchmark while substantially reducing the number of teacher-scored positions.

2 Background and Related Work

2.1 Knowledge distillation for language models

Distillation has a long history in NLP, including sequence-level distillation for generation [28] and transformer compression pipelines such as patient knowledge distillation and compact pretrained students [29, 30]. With the rise of LLMs, KD has been revisited as a central mechanism for compressing instruction-following models and transferring capabilities from large teachers to smaller students [6, 7, 8, 9]. Surveys highlight KD as complementary to other efficiency techniques such as pruning, quantization, and efficient inference systems [31, 14].

Modern LLM distillation spans several supervision modalities. Black-box distillation uses only teacher outputs or preferences, which is especially relevant when the teacher is accessible only through an API [10, 11]. Other methods emphasize representation transfer and feature alignment between teacher and student, seeking to match intermediate representations in addition to outputs [32, 33]. Distillation can also be performed across heterogeneous tokenizers or vocabularies, motivating cross-tokenizer and token-alignment approaches [34, 35, 36]. Multi-teacher and multilingual settings further extend the teacher-student framework [37, 38]. Industrial-scale practice reports describe end-to-end distillation recipes and deployment constraints [39].

2.2 Distillation objectives and KL divergence variants

A common family of objectives aligns teacher and student next-token distributions using KL divergence or closely related losses [12, 40]. Forward KL objectives encourage mode-covering behavior, while reverse KL encourages mode-seeking behavior, which can be beneficial for instruction-following or preference-aligned generation [41, 12]. Several works study stability and calibration of distillation for autoregressive LMs, motivating alternative targets or divergence control mechanisms [42, 43, 44]. Distillation objectives also interact with downstream generation regimes and evaluation metrics, including summarization-style settings where objective choice can influence factuality and coverage [45]. For heavily compressed students, self-distillation and low-bit regimes introduce additional constraints on objective design [46].

2.3 On-policy distillation and interactive teacher-student training

On-policy distillation addresses the mismatch between training trajectories and student inference trajectories by querying the teacher on student-generated outputs [15]. This framing naturally connects KD to imitation learning and policy optimization, where training signal is collected on trajectories induced by the current policy [26]. However, on-policy schemes can be expensive because the teacher may be queried repeatedly during generation.

To reduce this overhead, recent work explores interleaved sampling and verification strategies, including speculative KD and draft-model training for speculative decoding [16, 17]. Adaptive switching methods dynamically mix supervision sources or generation modes to balance quality and cost [18]. More broadly, collaborative edge-cloud training perspectives study how small on-device models can interact with cloud-scale teachers [47, 1, 2]. Interactive feedback and iterative refinement also appear in distillation for difficult reasoning tasks [48, 49, 50].

2.4 Reasoning transfer and chain-of-thought distillation

Distillation can transfer more than final answers. Chain-of-thought prompting shows that LLMs can produce intermediate reasoning steps that improve task performance [51]. A line of work distills such step-by-step rationales into smaller students, improving reasoning with less data or smaller model sizes [52, 53, 54]. Data-centric benchmark and reweighting methods further analyze which rationales are most helpful and how to control variance in rationale supervision [55, 56]. Beyond standalone QA, distillation has also been applied to sequential recommendation and other domains where rationales provide structured supervision [57, 58, 59]. Agentic and tool-augmented settings motivate distilling trajectories that mix reasoning with actions such as retrieval and code execution [60, 61, 62, 63].

2.5 Token-level supervision, selection, and efficiency

Token-level distillation can be more effective than sentence-level objectives in certain regimes, especially for smaller students and simpler text settings [13]. Many recent methods therefore study token-wise selection or weighting to

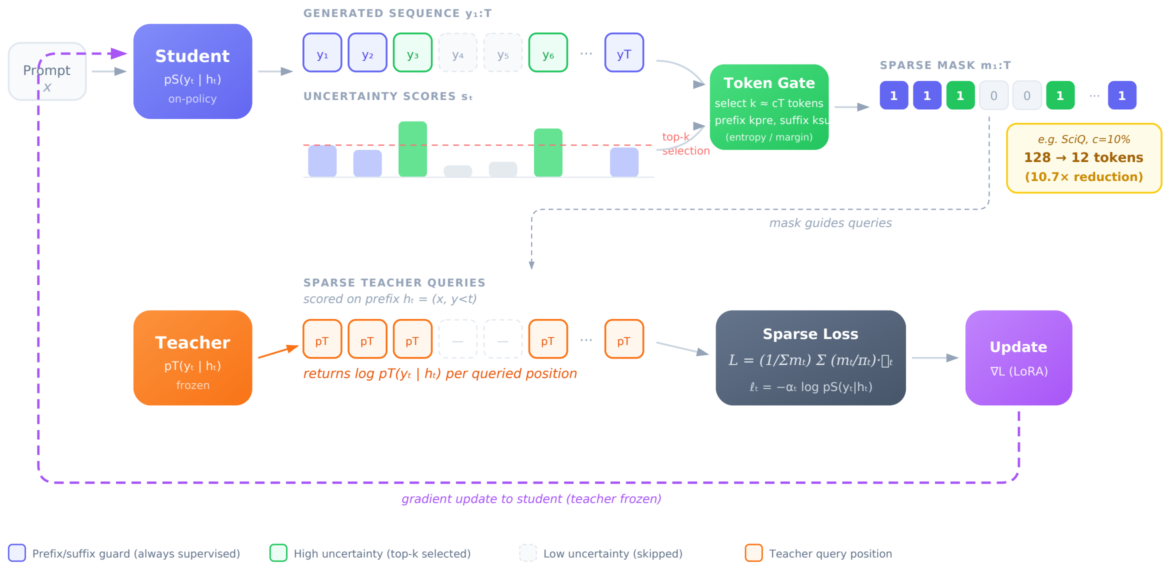
Prompt
x
Student
$pS(y_t \mid h_t)$
on-policy
GENERATED SEQUENCE $y_{1:T}$
y_1 y_2 y_3 y_4 y_5 y_6 ··· y_T
UNCERTAINTY SCORES s_t
top-k selection
Token Gate
select k ≈ cT tokens
prefix kpre, suffix ksu
(entropy / margin)
SPARSE MASK $m_{1:T}$
1 1 1 0 0 1 ··· 1
e.g. SciQ, c=10%
128 → 12 tokens
(10.7× reduction)
mask guides queries
SPARSE TEACHER QUERIES
scored on prefix $h_t = (x, y<t)$
Teacher
$pT(y_t \mid h_t)$
frozen
pT pT pT — — pT ··· pT
returns log $pT(y_t \mid h_t)$ per queried position
Sparse Loss
$L = (1/\Sigma m_t)\ \Sigma\ (m_t/\pi_t)\cdot\ell_t$
$\ell_t = -\alpha_t \log pS(y_t|h_t)$
Update
∇L (LoRA)
gradient update to student (teacher frozen)
Prefix/suffix guard (always supervised)
High uncertainty (top-k selected)
Low uncertainty (skipped)
Teacher query position

Figure 1: FORKDISTILL overview (sparse on-policy distillation). Given a prompt x, the student generates an on-policy continuation $y_{1:T}$ and computes token-wise uncertainty scores s_t from its predictive distribution. A token gate selects a sparse supervision mask $m_{1:T}$ by choosing approximately $k \approx cT$ high-uncertainty positions (entropy or margin), while always including a small prefix and suffix guard region. The teacher is queried only at positions with $m_t = 1$ to score the realized token under the corresponding prefix $h_t = (x, y_{<t})$ (e.g., returning $\log p_T(y_t \mid h_t)$). The student is updated using a masked objective (optionally inverse-probability weighted via m_t/π_t when gating is stochastic), and only the student LoRA parameters are trained while the teacher remains frozen; the callout highlights the resulting reduction in teacher-scored tokens at low coverage (e.g., 128→12 at c=10%).

prioritize informative positions and suppress redundant gradients. Examples include token-level relationship modeling [64], selective token-weighted KD [21], token-adaptive KD [20], calibration-driven gating [22], and multi-granularity semantic revision [19]. Related efficiency mechanisms appear in token routing and pruning, which skip computation on low-utility tokens during inference or training [65, 66, 67, 68]. Sampling and generation strategies also influence distillation data quality, including cautionary findings about degeneration and mitigation via sampling constraints [69, 70].

Token selection and prioritization are also studied in multimodal distillation and pretraining, where focusing on salient tokens can improve efficiency [71, 72, 73, 74]. Application-specific distillation frameworks incorporate explainability or structured signals, such as e-commerce relevance distillation [75]. Finally, domain- or language-specific pretraining choices can alter what knowledge is easiest to compress into small models [76].

2.6 Importance weighting and uncertainty-based token selection

ForkDistill is closely related to importance sampling and reweighting methods that reduce training cost while preserving correct expectations. Importance sampling has been used to accelerate deep learning by prioritizing high-utility samples [25] and appears broadly in stochastic optimization and policy learning [26, 27]. For language modeling, estimating or emphasizing rare or high-uncertainty events is often beneficial [77]. Entropy and uncertainty are frequently used as token-level difficulty signals, motivating weighted entropy fine-tuning and risk-aware selective language modeling [24, 23]. Other reweighting perspectives include token-level guidance for fine-tuning and group-based objectives for stable supervised adaptation [78, 79]. Entropy regularization is also a standard tool for encouraging exploration and controlling collapse in policy learning settings [80, 81].

3 Method

3.1 Setup and notation

Let x denote an input prompt (e.g., a question) and let $y_{1:T}$ denote the student-generated continuation of length T tokens, where $y_t \in \mathcal{V}$ and $\mathcal{V}$ is the vocabulary. We write the prefix at time t as $h_t \triangleq (x, y_{<t})$.

We consider a *student* language model with token-level conditional distribution

$$p_S(y_t \mid h_t) \triangleq p_S(y_t \mid x, y_{<t}), \tag{1}$$

and a fixed *teacher* model with

$$p_T(y_t \mid h_t) \triangleq p_T(y_t \mid x, y_{<t}). \tag{2}$$

On-policy distillation means the trajectory is generated by the student decoding policy,

$$y_t \sim \pi_S(\cdot \mid h_t), \tag{3}$$

where π_S may be sampling or greedy decoding derived from p_S. Throughout training, the teacher parameters remain frozen, while the student is updated via parameter-efficient fine-tuning (LoRA).

Teacher query cost. Our primary efficiency metric is the number of *teacher-scored tokens*, defined as the count of token positions for which we compute teacher supervision (either a scalar log-probability or a full logit distribution). Formally, for a binary supervision mask $m_{1:T} \in \{0,1\}^T$ (defined below), the teacher query cost for one generated sequence is

$$Q(x, y_{1:T}) \triangleq \sum_{t=1}^{T} m_t. \tag{4}$$

This metric directly reflects how many token positions require teacher computation and is the quantity we reduce. We report runtime only when explicitly measured.

3.2 Dense baseline objective

ForkDistill approximates a dense token-level distillation objective with sparse teacher supervision. We first define a per-token loss ℓ_t and then average it over all generated tokens.

(A) Realized-token scoring. In settings where the teacher interface naturally provides only the log-probability of the realized student token,

$$\log p_T(y_t \mid h_t), \tag{5}$$

we define a teacher-weighted negative log-likelihood objective on the student token:

$$\ell_t^{\text{SCORE}} \triangleq -\alpha_t \, \log p_S(y_t \mid h_t), \qquad \alpha_t \triangleq p_T(y_t \mid h_t). \tag{6}$$

Intuitively, the teacher confidence α_t modulates the magnitude of the student update, emphasizing positions where the teacher strongly endorses the realized token. This supervision requires only a single scalar from the teacher per queried position.

(B) Full distribution distillation. When teacher logits (or distributions) are available, we distill using a divergence between teacher and student token distributions at each prefix. Let $z_{S,t} \in \mathbb{R}^{|\mathcal{V}|}$ and $z_{T,t} \in \mathbb{R}^{|\mathcal{V}|}$ denote student and teacher logits at prefix h_t. With temperature $\tau > 0$, define softened distributions

$$p_S^{(\tau)}(\cdot \mid h_t) \triangleq \text{softmax}\Big(\frac{z_{S,t}}{\tau}\Big), \qquad p_T^{(\tau)}(\cdot \mid h_t) \triangleq \text{softmax}\Big(\frac{z_{T,t}}{\tau}\Big). \tag{7}$$

We consider the standard forward KL (teacher to student),

$$\ell_t^{\text{KL}} \triangleq \tau^2 \, \text{KL}\Big(p_T^{(\tau)}(\cdot \mid h_t) \,\|\, p_S^{(\tau)}(\cdot \mid h_t)\Big), \tag{8}$$

and we can analogously use reverse KL if desired by swapping the arguments. This supervision requires teacher logits at queried positions.

Dense objective. Given a choice of $\ell_t \in \{\ell_t^{\text{SCORE}}, \ell_t^{\text{KL}}\}$, the dense baseline loss over the student-generated continuation is

$$L_{\text{DENSE}}(x) \triangleq \frac{1}{T}\sum_{t=1}^{T} \ell_t. \tag{9}$$

3.3 ForkDistill: sparse token supervision with a mask

ForkDistill reduces teacher queries by supervising only a subset of token positions. Let $m_t \in \{0,1\}$ denote whether token position t is teacher-supervised:

$$m_t = \begin{cases} 1 & \text{query the teacher and include token } t \text{ in the loss,} \\ 0 & \text{skip teacher supervision at token } t. \end{cases} \tag{10}$$

We define a *target coverage* $c \in (0,1]$ and the *realized coverage*

$$\widehat{c}(x, y_{1:T}) \triangleq \frac{1}{T}\sum_{t=1}^{T} m_t, \tag{11}$$

which is the quantity we report. In practice, guard tokens (Section 3.5) can cause $\widehat{c}$ to slightly exceed the nominal target c.

3.4 Token selection strategies (gating)

ForkDistill constructs $m_{1:T}$ by first computing a scalar *score* s_t at each position from the student predictive distribution, then selecting approximately $k \approx cT$ positions for supervision (excluding guard tokens).

Let $p_S(\cdot \mid h_t)$ be the student distribution at prefix h_t and let $z_{S,t}$ be the student logits at that position. We consider the following scoring functions:

- **Entropy:**
$$s_t \triangleq H(p_S(\cdot \mid h_t)) = -\sum_{v\in\mathcal{V}} p_S(v \mid h_t)\, \log p_S(v \mid h_t). \tag{12}$$
High entropy indicates local uncertainty and corresponds to potential “forking points”.
- **Margin (logit ambiguity):** let $z_{S,t}^{(1)} \geq z_{S,t}^{(2)}$ be the top-1 and top-2 logits in $z_{S,t}$, then
$$s_t \triangleq -\left(z_{S,t}^{(1)} - z_{S,t}^{(2)}\right). \tag{13}$$
A small top-1 minus top-2 margin yields a high score, indicating ambiguity.
- **Random:**
$$s_t \sim \mathcal{U}(0,1), \tag{14}$$
which serves as a simple baseline for selection.

Mapping scores to a mask. Let $\mathcal{I}$ be the set of non-guard positions (Section 3.5). A common deterministic instantiation selects the top-k positions by score:

$$\mathcal{S} \triangleq \operatorname{TopK}\big(\{s_t\}_{t\in\mathcal{I}},\ k\big), \qquad m_t \leftarrow \mathbb{I}[t \in \mathcal{S}] \ \text{ for } t \in \mathcal{I}. \tag{15}$$

Alternatively, one may use stochastic sampling with per-token selection probability π_t (defined below), which enables inverse-probability weighting with exact unbiasedness guarantees.

3.5 Guard tokens

To stabilize generation and ensure that supervision covers crucial boundary regions, we always supervise a small number of prefix and suffix tokens in the generated continuation.

Let $k_{\text{PRE}} \geq 0$ and $k_{\text{SUF}} \geq 0$ denote the guard lengths. We set

$$m_t \leftarrow 1 \quad \text{for} \quad t \in \{1, \ldots, \min(k_{\text{PRE}}, T)\} \,\cup\, \{\max(1, T - k_{\text{SUF}} + 1), \ldots, T\}. \tag{16}$$

The prefix guard stabilizes early decoding decisions (where errors can cascade), and the suffix guard increases the chance that the final answer region is teacher-supervised, which is especially important for short-form QA outputs.

Algorithm 1 ForkDistill (single training step)

```
Input: prompt $x$, target coverage $c$, guards $(k_{\text{PRE}}, k_{\text{SUF}})$, student $S$, teacher $T$
 1: Sample a student trajectory $y_{1:T} \sim \pi_S(\cdot \mid x)$          ▷ on-policy decoding
 2: for $t = 1$ to $T$ do
 3:     Form prefix $h_t \leftarrow (x, y_{<t})$
 4:     Compute student logits $z_{S,t}$ and distribution $p_S(\cdot \mid h_t)$
 5:     Compute score $s_t \leftarrow \mathrm{Score}(p_S(\cdot \mid h_t))$ (entropy, margin, or random)
 6: end for
 7: Initialize mask $m_{1:T} \leftarrow 0$
 8: Apply guard tokens $m_t \leftarrow 1$ for prefix and suffix positions as in Eq. (16)
 9: Select approximately $cT$ additional non-guard positions with highest $s_t$ and set $m_t \leftarrow 1$
10: for each $t$ with $m_t = 1$ do
11:     Query teacher to obtain supervision at $h_t$ (log-probs or logits)
12:     Compute per-token loss $\ell_t$ (e.g., Eq. (6) or Eq. (8))
13:     Set weight $w_t \leftarrow m_t/\pi_t$ (if stochastic gating; otherwise $w_t \leftarrow 1$)
14: end for
15: Form sparse objective $L_{\text{SPARSE}}(x)$ as in Eq. (18)
16: Update student parameters (LoRA) using $\nabla L_{\text{SPARSE}}(x)$; keep teacher parameters fixed
```

3.6 Importance weighting for subsampled tokens

Naively optimizing the masked objective $\sum_t m_t \ell_t$ biases learning toward the selected positions. To correct for this, we use inverse-probability weighting over token positions, a standard technique in importance sampling [25, 26].

Assume the mask is sampled such that each token position t has a known selection probability $\pi_t \in (0, 1]$ (conditioned on the realized sequence and gating procedure). Define the weight

$$w_t \triangleq \frac{m_t}{\pi_t}. \tag{17}$$

ForkDistill then optimizes the weighted sparse objective

$$L_{\text{SPARSE}}(x) \triangleq \frac{1}{\sum_{t=1}^{T} m_t} \sum_{t=1}^{T} w_t \, \ell_t. \tag{18}$$

When gating is stochastic and π_t matches the true sampling probabilities, the inverse-probability correction yields an unbiased estimate of the dense token-average up to the normalization constant (and becomes exactly unbiased if one replaces the denominator $\sum_t m_t$ with T). For deterministic top-k selection, π_t is not defined as a marginal probability, so the estimator is only approximate; in practice, we set $\pi_t = 1$ on selected positions and still observe stable learning.

3.7 ForkDistill algorithm

Algorithm 1 summarizes one training step.

3.8 Cost analysis

Let T be the number of generated tokens per example. Dense distillation requires teacher supervision at every position, giving $Q_{\text{DENSE}} = T$ teacher-scored tokens. ForkDistill reduces this to

$$Q_{\text{SPARSE}} = \sum_{t=1}^{T} m_t = \widehat{c}T, \tag{19}$$

so the teacher query reduction factor is approximately $T/Q_{\text{SPARSE}} \approx 1/\widehat{c}$. Crucially, the student forward pass and sequence generation still process all T tokens; ForkDistill specifically reduces the amount of teacher supervision required. Prompt processing overhead is separate from per-token teacher scoring and is unchanged by token masking.

Table 1: Dataset used in our evaluation.

Dataset	Output type	Notes
SciQ	short text	science QA with short factual answers

Table 2: Training configuration (main runs).

Setting	Value
Student / Teacher	Gemma 3 1B (PT) / Gemma 3 4B (IT)
Fine-tuning	LoRA ($r = 32$, $\alpha = 32$)
Optimizer	AdamW
Learning rate	2×10^{-4}
Steps	200
Batch size	1
Max new tokens $T_{\max}$	128
Gradient clipping	1.0

4 Experimental Setup

4.1 Models and parameter-efficient fine-tuning

We distill from a larger teacher language model into a smaller student model. Unless stated otherwise, our student is `Gemma 3 1B` (pretrained) and our teacher is `Gemma 3 4B` (instruction-tuned), both loaded in `bfloat16`. We keep the teacher frozen throughout training and only update a low-rank adaptation (LoRA) on top of the student. Following standard practice for transformer adaptation, we apply LoRA to the attention projection matrices and MLP projections (e.g., `q_proj,k_proj,v_proj,o_proj,gate_proj,up_proj,down_proj`) with rank $r = 32$ and scaling $\alpha = 32$.

4.2 Dataset (SciQ)

We evaluate FORKDISTILL on **SciQ**, a science question answering dataset with short factual answers. We treat SciQ as a collection of (question, answer) pairs. During training, we sample prompts from the SciQ training split for on-policy rollouts, and we use a held-out split for periodic monitoring.

4.3 Prompt format and decoding

We use a single prompt template:

```
Question: {question}
Answer:
```

During training, the student generates answers autoregressively using its decoding policy (on-policy rollouts). We cap generations to $T_{\max} = 128$ new tokens and enforce a small minimum generation length to reduce degenerate early termination. During evaluation/monitoring, we use deterministic greedy decoding (argmax) with the same maximum generation length.

4.4 Training procedure and hyperparameters

We train with on-policy rollouts: each step samples a SciQ example, runs student generation, and then computes a distillation loss using teacher supervision at selected token positions (Section 3). We optimize only the LoRA parameters using AdamW. In our main runs we use 200 optimization steps, batch size 1, and a learning rate of 2×10^{-4}. We clip the gradient norm to 1.0. We monitor training by tracking the distillation objective value (Loss) and teacher query cost, and we report final results in Section 5. All experiments use a single random seed under this small compute budget, so results should be interpreted as a proof of concept rather than a large-scale study.

4.5 ForkDistill settings, baselines, and ablations

We sweep target token coverage $c \in \{1.0, 0.5, 0.3, 0.2, 0.1\}$, where $c = 1.0$ corresponds to dense supervision. For sparse runs, we compare multiple gating strategies at matched coverage, including entropy-based selection and random

Table 3: Teacher query reduction via sparse token supervision. "Teacher Queries" counts teacher-scored token positions per example. "Loss" is the final distillation objective value (lower is better). "Reduction" is computed against the dense baseline.

Method	Coverage (%)	Teacher Queries	Loss ↓	Reduction
Dense Baseline	100	128.0	4.037	1.00×
FORKDISTILL (entropy)	50	64.0	1.547	2.00×
FORKDISTILL (entropy)	30	38.0	5.999	3.37×
FORKDISTILL (entropy)	20	25.0	5.540	5.12×
FORKDISTILL (entropy)	10	12.0	3.214	10.67×
Random (10% coverage)	10	12.0	4.305	10.67×

selection. We include a small set of guard tokens by always supervising a prefix ($k_{\text{pre}} = 4$) and a suffix ($k_{\text{suf}} = 1$) of each generated response. Unless otherwise stated, our sweep uses deterministic top-k gating at each example (i.e., non-stochastic selection), so selected and guard tokens have $\pi_t = 1$ and inverse-probability weights reduce to $w_t = 1$ (Eq. (17)), yielding a self-normalized masked average. We enable inverse-probability weighting as a principled correction only when using stochastic gating with explicit per-token selection probabilities.

4.6 Metrics and efficiency accounting

Our primary quality metric in this report is the final value of the distillation objective (Loss), computed over teacher-supervised token positions as described in Section 3. For efficiency, we report teacher query cost as the number of teacher-scored token positions per rollout, along with realized coverage $\hat{c}$ (Eq. (11)), which accounts for guard tokens and rounding effects. (We also implemented answer-extraction evaluation for sanity checks, but we focus on the distillation objective here.)

5 Results

We present a proof-of-concept evaluation of FORKDISTILL that isolates the core claim: sparse, token-level teacher supervision can substantially reduce teacher query cost while preserving (and sometimes improving) the on-policy distillation objective. Across runs we keep the model pair, optimization settings, and training budget fixed, and vary only (i) the supervision coverage and (ii) the token selection rule. Unless stated otherwise, Loss refers to the realized-token distillation objective using teacher log-prob scoring (Eq. (6)) aggregated with the sparse masked average in Eq. (18); we use entropy gating with guards $(k_{\text{PRE}}, k_{\text{SUF}}) = (4, 1)$.

5.1 Main results: cost–quality tradeoff

Table 3 summarizes the teacher query cost and final distillation loss for a sweep over realized coverage $\hat{c}$. Here, *teacher queries* denotes the number of teacher-scored token positions per example, i.e., $\sum_{t=1}^{T} m_t$, averaged across the run. The *reduction* is computed relative to the dense baseline (100% coverage). Lower loss is better.

Two trends stand out. First, FORKDISTILL provides a direct and predictable handle on teacher cost: reducing coverage from 100% to 10% decreases teacher queries from 128 to 12 tokens per example, a $10.67\times$ reduction. Second, sparse supervision need not degrade optimization. At 10% coverage, entropy-gated FORKDISTILL attains a loss of 3.214, which is slightly lower than the dense baseline (4.037) despite using an order of magnitude fewer teacher-scored tokens.

5.2 Entropy gating matters at fixed teacher budget

A key question is whether improvements are purely a consequence of using fewer teacher-scored tokens, or whether *which* tokens are supervised matters. Table 3 includes a matched-budget comparison at 10% coverage: both entropy gating and random gating use the same teacher query count (12.0 tokens per example), but entropy gating achieves substantially lower loss (3.214 vs. 4.305). This corresponds to a relative reduction of approximately 25% in loss at equal teacher cost:

$$\frac{4.305 - 3.214}{4.305} \approx 0.253.$$

This gap supports the interpretation that sparse supervision is not only a compute-saving mechanism, but can also be a *sample-efficient* allocation of teacher signal. In particular, entropy gating preferentially spends teacher queries on positions where the student is most uncertain, which are also the positions most likely to benefit from corrective supervision.

5.3 Moderate sparsity can outperform dense supervision

Interestingly, 50% coverage yields the best loss in this sweep, improving substantially over the dense baseline (1.547 vs. 4.037) while halving teacher queries. One plausible explanation is that dense token-level supervision spends significant capacity on low-entropy positions where the student already agrees with the teacher, contributing little incremental learning signal. By concentrating supervision on higher-uncertainty positions (while still retaining guard tokens), FORKDISTILL emphasizes informative corrections and can reduce gradient noise from redundant tokens. We view this non-monotonic behavior as a promising signal that there exists a regime where sparse token supervision is not merely cheaper, but also better targeted.

5.4 Takeaway

Overall, these results validate the core premise of FORKDISTILL: substantial teacher query reductions are achievable without sacrificing the distillation objective, and entropy-based token selection provides a meaningful advantage over random selection at the same teacher budget. This evidence supports sparse token-level supervision as a practical knob for scaling on-policy distillation under constrained teacher compute.

6 Discussion and Limitations

6.1 Why sparse supervision works in our regime

ForkDistill is motivated by a simple inefficiency in on-policy distillation: dense token-level supervision repeatedly queries the teacher at positions where the student is already confident and locally correct, yielding little incremental learning signal per unit teacher compute. In short-answer question answering, many generated tokens correspond to boilerplate structure, predictable continuations, or high-probability surface forms. Supervising such tokens densely can be wasteful because their gradients are small and often redundant, while the optimization budget is dominated by a comparatively small set of high-impact decisions where the student is uncertain, ambiguous alternatives exist, or early mistakes would otherwise be reinforced by on-policy sampling. This perspective is consistent with prior observations that token difficulty is highly non-uniform, and that uniform allocation of supervision can dilute computation on low-signal positions [20].

Our empirical sweep supports this mechanism. When we reduce teacher supervision from dense scoring to sparse scoring, the number of teacher-scored token positions decreases proportionally with coverage, yet the distillation objective does not degrade monotonically and can even improve at intermediate sparsity. In particular, supervising an informative subset of tokens can act as a regularizer by avoiding over-constraining the student on easy positions, thereby concentrating updates on the parts of the trajectory that most strongly shape downstream behavior under on-policy generation. Guard tokens further stabilize this regime by ensuring that training retains consistent supervision near the beginning of generation, where distributional shift is most pronounced, and near the end of generation, where short-answer correctness is typically determined.

6.2 Entropy gating helps at low coverage

A central finding is that token selection matters most when the supervision budget is extremely constrained. At 10% coverage, entropy-based gating achieves a substantially lower final loss than random selection at the same teacher query budget, with losses of 3.214 and 4.305 respectively in our SciQ setting. This corresponds to approximately a 25% reduction in loss at fixed teacher-scored tokens, indicating that allocating supervision to high-uncertainty positions better concentrates learning on points where the student distribution diverges most from the teacher and where corrective gradients are largest.

We also observe that moderate sparsity can outperform dense supervision in terms of the distillation objective, for example when 50% coverage yields a lower loss than the dense baseline in our sweep. A plausible explanation is that dense token-level supervision overemphasizes low-entropy tokens, including formatting and locally deterministic continuations, which can inject optimization noise and reduce the effective step size devoted to genuinely informative corrections. Entropy gating implicitly prioritizes positions with higher expected gradient contribution, connecting to importance-aware training and selective sampling ideas studied in the importance sampling and reweighting literature [25, 26].

6.3 When token selection should matter more

While entropy gating already improves efficiency in short-answer settings, we expect its advantage to increase in regimes where informative tokens are rarer relative to sequence length. In long-form generation and multi-step reasoning, a small fraction of tokens often determines correctness, such as key intermediate commitments or final decisive statements, which makes targeted supervision more valuable. In code and math, errors can concentrate on sparse critical tokens including operators, variable identifiers, boundary conditions, or punctuation that alters semantics, so supervising high-uncertainty or high-sensitivity positions should deliver a larger gain per teacher query. Token selection should also matter more when the student is weaker or the teacher-student gap is larger, because uncertainty tends to correlate more strongly with genuine errors, and when coverage is pushed to very small values, since random selection becomes increasingly likely to miss the few tokens that govern downstream behavior. These hypotheses align with recent approaches that modulate supervision based on token difficulty, confidence, or adaptive criteria [44, 21].

6.4 Limitations and threats to validity

First, our primary efficiency metric is the number of teacher-scored token positions, which corresponds directly to query cost in settings where teacher scoring is implemented as an external service or black-box call per token position. However, teacher-scored tokens do not automatically translate into proportional wall-clock savings when the teacher runs locally and can exploit batching, caching, or fused kernels. The realized speedup depends on the deployment architecture, prompt reuse, and whether per-token scoring is the dominant cost.

Second, we use the distillation objective value as a primary proxy for learning progress in this proof-of-concept. Although lower loss is desirable and often correlates with improved student behavior, it is not guaranteed to translate monotonically into downstream accuracy on every task. Stronger claims require comprehensive benchmark evaluation, careful answer extraction, and statistical testing with sufficient sample sizes, particularly when measured differences are modest.

Third, our current experiments focus on short-answer QA with relatively short generations, which may hide failure modes that emerge in longer sequences, including error compounding, exposure bias under aggressive subsampling, and increased sensitivity to selection bias. In addition, inverse-probability weighting admits clean unbiasedness guarantees under stochastic sampling with correctly specified selection probabilities, whereas deterministic top-k selection only supports approximate interpretations. Self-normalized weighting can also introduce a bias-variance tradeoff, and extreme sparsity can increase gradient variance if selected positions are highly concentrated.

Finally, our results are based on a single student-teacher pair, a fixed LoRA configuration, and a particular decoding policy. Different model scales, alternative adapter targets, or changes in sampling temperature may shift the operating point and alter the relative benefit of entropy gating versus simpler strategies.

7 Conclusion

On-policy knowledge distillation is attractive because it trains the student on its own generated trajectories, but it is often prohibitively expensive because standard formulations require teacher supervision at every generated token position. ForkDistill reduces this cost by supervising only a subset of generated tokens, selected via a gating mechanism, while preserving stable learning through simple stabilizers and, when applicable, importance weighting to correct for subsampling. In our proof-of-concept experiments, sparse supervision yields large reductions in teacher-scored tokens and remains effective across a range of coverage levels, with entropy-based selection providing a clear advantage over random selection in the most constrained regime. These findings suggest that much of dense token-level supervision is redundant in short-answer settings, and that allocating limited teacher queries to high-uncertainty forking points can substantially improve the efficiency of on-policy distillation.

References

[1] Jiajun Xu, Zhiyuan Li, Wei Chen, Qun Wang, Xin Gao, Qi Cai, and Ziyuan Ling. On-device language models: A comprehensive review. *arXiv.org*, 2024.

[2] Yue Zheng, Yuhao Chen, Bin Qian, Xiufang Shi, Yuanchao Shu, and Jiming Chen. A review on edge large language models: Design, execution, and applications. *ACM Computing Surveys*, 2024.

[3] Yanjie Dong, Xiaoyi Fan, Fangxin Wang, Chengming Li, Victor C. M. Leung, and Xiping Hu. Fine-tuning and deploying large language models over edges: Issues and approaches. *arXiv.org*, 2024.

[4] Muskan Garg, Shaina Raza, Shebuti Rayana, Xingyi Liu, and Sunghwan Sohn. The rise of small language models in healthcare: A comprehensive survey. *arXiv.org*, 2025.

[5] Tanjil Hasan Sakib, Md. Tanzib Hosain, and Md. Kishor Morol. Small language models: Architectures, techniques, evaluation, problems and future adaptation. *arXiv.org*, 2025.

[6] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *International Conference on Learning Representations*, 2023.

[7] Chuanpeng Yang, Yao Zhu, Wang Lu, Yidong Wang, Qian Chen, Chenlong Gao, Bingjie Yan, and Yiqiang Chen. Survey on knowledge distillation for large language models: Methods, evaluation, and application. *ACM Transactions on Intelligent Systems and Technology*, 2024.

[8] Xunyu Zhu, Jian Li, Yong Liu, Can Ma, and Weiping Wang. A survey on model compression for large language models. *Transactions of the Association for Computational Linguistics*, 2023.

[9] Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. A survey on transformer compression. *arXiv.org*, 2024.

[10] Hongzhan Chen, Ruijun Chen, Yuqi Yi, Xiaojun Quan, Chenliang Li, Ming Yan, and Ji Zhang. Knowledge distillation of black-box large language models. 2024.

[11] Mohamad Ballout, U. Krumnack, Gunther Heidemann, and Kai-Uwe Kühnberger. Efficient knowledge distillation: Empowering small language models with teacher model insights. *International Conference on Applications of Natural Language to Data Bases*, 2024.

[12] Qihuang Zhong, Liang Ding, Li Shen, Juhua Liu, Bo Du, and D. Tao. Revisiting knowledge distillation for autoregressive language models. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[13] Jingxuan Wei, Linzhuang Sun, Yichong Leng, Xu Tan, Bihui Yu, and Ruifeng Guo. Sentence-level or token-level? a comprehensive study on knowledge distillation. *International Joint Conference on Artificial Intelligence*, 2024.

[14] Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, Shengen Yan, Guohao Dai, Xiao-Ping Zhang, Yuhan Dong, and Yu Wang. A survey on efficient inference for large language models. *arXiv.org*, 2024.

[15] Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, P. Stańczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. *International Conference on Learning Representations*, 2023.

[16] Wenda Xu, Rujun Han, Zifeng Wang, Long T. Le, Dhruv Madeka, Lei Li, William Yang Wang, Rishabh Agarwal, Chen-Yu Lee, and Tomas Pfister. Speculative knowledge distillation: Bridging the teacher-student gap through interleaved sampling. *arXiv.org*, 2024.

[17] Fenglu Hong, Ravi Raju, Jonathan Li, Bo Li, Urmish Thakker, Avinash Ravichandran, Swayambhoo Jain, and Changran Hu. Training domain draft models for speculative decoding: Best practices and insights. *arXiv.org*, 2025.

[18] Jingyu Peng, Maolin Wang, Hengyi Cai, Yuchen Li, Kai Zhang, Shuaiqiang Wang, Dawei Yin, and Xiangyu Zhao. Adaswitch: Adaptive switching generation for knowledge distillation. *arXiv.org*, 2025.

[19] Xiaoyu Liu, Yun feng Zhang, Wei Li, Simiao Li, Xu Huang, Hanting Chen, Yehui Tang, Jie Hu, Zhiwei Xiong, and Yunhe Wang. Multi-granularity semantic revision for large language model distillation. *arXiv.org*, 2024.

[20] Xurong Xie, Zhucun Xue, Jiafu Wu, Jian Li, Yabiao Wang, Xiaobin Hu, Yong Li, and Jiang-She Zhang. Llm-oriented token-adaptive knowledge distillation. *arXiv.org*, 2025.

[21] Haiduo Huang, Jiangcheng Song, Yadong Zhang, and Pengju Ren. Selectkd: Selective token-weighted knowledge distillation for llms. 2025.

[22] Dongkyu Lee, Zhiliang Tian, Ying Zhao, K. Cheung, and N. Zhang. Hard gate knowledge distillation - leverage calibration for robust and reliable language model. *Conference on Empirical Methods in Natural Language Processing*, 2022.

[23] Melis Ilayda Bal, V. Cevher, and Michael Muehlebach. Eslm: Risk-averse selective language modeling for efficient pretraining. *arXiv.org*, 2025.

[24] Guowei Xu, Wenxin Xu, Jiawang Zhao, and Kaisheng Ma. Weft: Weighted entropy-driven fine-tuning for dllms. *arXiv.org*, 2025.

[25] Angelos Katharopoulos and F. Fleuret. Not all samples are created equal: Deep learning with importance sampling. *International Conference on Machine Learning*, 2018.

[26] Alberto Maria Metelli, M. Papini, Francesco Faccio, and Marcello Restelli. Policy optimization via importance sampling. *Neural Information Processing Systems*, 2018.

[27] M. A. Nabian, R. J. Gladstone, and H. Meidani. Efficient training of physics-informed neural networks via importance sampling. *Comput. Aided Civ. Infrastructure Eng.*, 2021.

[28] Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. *Conference on Empirical Methods in Natural Language Processing*, 2016.

[29] S. Sun, Yu Cheng, Zhe Gan, and Jingjing Liu. Patient knowledge distillation for bert model compression. *Conference on Empirical Methods in Natural Language Processing*, 2019.

[30] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv.org*, 2019.

[31] Bohan Zhuang, Jing Liu, Zizheng Pan, Haoyu He, Yuetian Weng, and Chunhua Shen. A survey on efficient training of transformers. *International Joint Conference on Artificial Intelligence*, 2023.

[32] Junjie Yang, Jun-Jie Song, Xudong Han, Ziqian Bi, Tianyang Wang, C. Liang, Xinyuan Song, Yichao Zhang, Qian Niu, Benji Peng, Keyu Chen, and Ming Liu. Feature alignment and representation transfer in knowledge distillation for large language models. *arXiv.org*, 2025.

[33] Chen Liang, Simiao Zuo, Qingru Zhang, Pengcheng He, Weizhu Chen, and Tuo Zhao. Less is more: Task-aware layer-wise distillation for language model compression. *arXiv.org*, 2022.

[34] Xiao Cui, Mo Zhu, Yulei Qin, Liang Xie, Wen gang Zhou, and Houqiang Li. Multi-level optimal transport for universal cross-tokenizer knowledge distillation on language models. *AAAI Conference on Artificial Intelligence*, 2024.

[35] Songming Zhang, Xue Zhang, Zengkui Sun, Yufeng Chen, and Jinan Xu. Dual-space knowledge distillation for large language models. *Conference on Empirical Methods in Natural Language Processing*, 2024.

[36] Chong Li, Jiajun Zhang, and Chengqing Zong. Tokalign: Efficient vocabulary adaptation via token alignment. *Annual Meeting of the Association for Computational Linguistics*, 2025.

[37] Xu Tan, Yi Ren, Di He, Tao Qin, Zhou Zhao, and Tie-Yan Liu. Multilingual neural machine translation with knowledge distillation. *International Conference on Learning Representations*, 2019.

[38] Shuke Peng, Feng Ji, Zehao Lin, Shaobo Cui, Haiqing Chen, and Yin Zhang. Mtss: Learn from multiple domain teachers and become a multi-domain dialogue expert. *AAAI Conference on Artificial Intelligence*, 2020.

[39] Chengyu Wang, Junbing Yan, Yuanhao Yue, and Jun Huang. Distilqwen2.5: Industrial practices of training distilled open lightweight language models. *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 6: Industry Track)*, 2025.

[40] Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and SeYoung Yun. Distillm: Towards streamlined distillation for large language models. *International Conference on Machine Learning*, 2024.

[41] Shaozhen Shi, Yevgen Matusevych, and Malvina Nissim. Choosy babies need one coach: Inducing mode-seeking behavior in babyllama with reverse kl divergence. *arXiv.org*, 2024.

[42] Junhao Ruan, Abudukeyumu Abudula, Xinyu Liu, Bei Li, Yinqiao Li, Chenglong Wang, Yuchun Fan, Yuan Ge, Tong Xiao, and Jingbo Zhu. Ndp: Next distribution prediction as a more broad target. *arXiv.org*, 2024.

[43] Konstantin Dobler, Desmond Elliott, and Gerard de Melo. Token distillation: Attention-aware input embeddings for new tokens. 2025.

[44] Seongryong Jung, Suwan Yoon, DongGeon Kim, and Hwanhee Lee. Todi: Token-wise distillation via fine-grained divergence control. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[45] Shashi Narayan, Shay B. Cohen, and Mirella Lapata. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. *Conference on Empirical Methods in Natural Language Processing*, 2018.

[46] Dayou Du, Yijia Zhang, Shijie Cao, Jiaqi Guo, Ting Cao, Xiaowen Chu, and Ningyi Xu. Bitdistiller: Unleashing the potential of sub-4-bit llms via self-distillation. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[47] Chaoyue Niu, Yucheng Ding, Junhui Lu, Zhengxiang Huang, Hang Zeng, Yutong Dai, Xuezhen Tu, Chengfei Lv, Fan Wu, and Guihai Chen. Collaborative learning of on-device small model and cloud-based large model: Advances and future directions. *arXiv.org*, 2025.

[48] Shivam Adarsh, Kumar Shridhar, Caglar Gulcehre, Nicholas Monath, and Mrinmaya Sachan. Siked: Self-guided iterative knowledge distillation for mathematical reasoning. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[49] Shreyansh Padarha. Enhancing reasoning capabilities in slms with reward guided dataset distillation. *arXiv.org*, 2025.

[50] Xiaofeng Zhou, Heyan Huang, and Lizi Liao. Debate, reflect, and distill: Multi-agent feedback with tree-structured preference optimization for efficient language model enhancement. *Annual Meeting of the Association for Computational Linguistics*, 2025.

[51] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, F. Xia, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. *Neural Information Processing Systems*, 2022.

[52] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and A. Severyn. Teaching small language models to reason. *Annual Meeting of the Association for Computational Linguistics*, 2022.

[53] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. *Annual Meeting of the Association for Computational Linguistics*, 2022.

[54] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander J. Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. *Annual Meeting of the Association for Computational Linguistics*, 2023.

[55] Ruichen Zhang, Rana Muhammad Shahroz Khan, Zhen Tan, Dawei Li, Song Wang, and Tianlong Chen. The quest for efficient reasoning: A data-centric benchmark to cot distillation. *arXiv.org*, 2025.

[56] Xuan Gong, Senmiao Wang, Hanbo Huang, Ruoyu Sun, and Shiyu Liang. Vcore: Variance-controlled optimization-based reweighting for chain-of-thought supervision. *arXiv.org*, 2025.

[57] Xinfeng Wang, Jin Cui, Yoshimi Suzuki, and Fumiyo Fukumoto. Rdrec: Rationale distillation for llm-based recommendation. *Annual Meeting of the Association for Computational Linguistics*, 2024.

[58] Yuling Wang, Changxin Tian, Binbin Hu, Yanhua Yu, Ziqi Liu, Zhiqiang Zhang, Jun Zhou, Liang Pang, and Xiao Wang. Can small language models be good reasoners for sequential recommendation? *The Web Conference*, 2024.

[59] Kaidong Feng, Zhu Sun, Jie Yang, Hui Fang, Xinghua Qu, and Wenyuan Liu. Does knowledge distillation matter for large language model based bundle generation? *arXiv.org*, 2025.

[60] Minki Kang, Jongwon Jeong, Seanie Lee, Jaewoong Cho, and Sung Ju Hwang. Distilling llm agent into small models with retrieval and code tools. *arXiv.org*, 2025.

[61] Sakhinana Sagar Srinivas and Venkataramana Runkana. Scaling test-time inference with policy-optimized, dynamic retrieval-augmented generation via kv caching and decoding. *arXiv.org*, 2025.

[62] Jujie He, Jiacai Liu, Chris Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner 1 technical report. *arXiv.org*, 2025.

[63] Zhenyu Lei, Zhen Tan, Song Wang, Yaochen Zhu, Zihan Chen, Yushun Dong, and Jundong Li. Learning from diverse reasoning paths with routing and collaboration. *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, 2025.

[64] Shuoxi Zhang, Hanpeng Liu, and Kun He. Knowledge distillation via token-level relationship graph. *arXiv.org*, 2023.

[65] Yeachan Kim, Junho Kim, Jun-Hyung Park, Mingyu Lee, and SangKeun Lee. Leap-of-thought: Accelerating transformers via dynamic token routing. *Conference on Empirical Methods in Natural Language Processing*, 2023.

[66] Mengzhou Xia, Zexuan Zhong, and Danqi Chen. Structured pruning learns compact and accurate models. *Annual Meeting of the Association for Computational Linguistics*, 2022.

[67] Zhuofan Zong, Kunchang Li, Guanglu Song, Yali Wang, Y. Qiao, B. Leng, and Yu Liu. Self-slimmed vision transformer. *European Conference on Computer Vision*, 2021.

[68] Jiahao Li, Quan Wang, Chiwei Zhu, Zhendong Mao, and Yongdong Zhang. Grammatical error correction via mixed-grained weighted training. *Conference on Empirical Methods in Natural Language Processing*, 2023.

[69] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *International Conference on Learning Representations*, 2019.

[70] Thomas Scialom, Paul-Alexis Dray, S. Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Coldgans: Taming language gans with cautious sampling strategies. *Neural Information Processing Systems*, 2020.

[71] Zhecan Wang, N. Codella, Yen-Chun Chen, Luowei Zhou, Jianwei Yang, Xiyang Dai, Bin Xiao, Haoxuan You, Shih-Fu Chang, and Lu Yuan. Clip-td: Clip targeted distillation for vision-language tasks. *arXiv.org*, 2022.

[72] Zhecan Wang, N. Codella, Yen-Chun Chen, Luowei Zhou, Xiyang Dai, Bin Xiao, Jianwei Yang, Haoxuan You, Kai-Wei Chang, Shih-Fu Chang, and Lu Yuan. Multimodal adaptive distillation for leveraging unimodal encoders for vision-language tasks. *arXiv.org*, 2022.

[73] Shilin Xu, Xiangtai Li, Haobo Yuan, Lu Qi, Yunhai Tong, and Ming-Hsuan Yang. Llavadi: What matters for multimodal large language models distillation. *arXiv.org*, 2024.

[74] Yangyi Chen, Hao Peng, Tong Zhang, and Heng Ji. Prioritizing image-related tokens enhances vision-language pre-training. *arXiv.org*, 2025.

[75] Gang Zhao, Ximing Zhang, Chenji Lu, Hui Zhao, Tianshu Wu, Pengjie Wang, Jian Xu, and Bo Zheng. Explainable llm-driven multi-dimensional distillation for e-commerce relevance learning. *The Web Conference*, 2024.

[76] Seungho Choi. Hanjabridge: Resolving semantic ambiguity in korean llms via hanja-augmented pre-training. *arXiv.org*, 2025.

[77] Gabriel Wu and Jacob Hilton. Estimating the probabilities of rare outputs in language models. *International Conference on Learning Representations*, 2024.

[78] Shentao Yang, Shujian Zhang, Congying Xia, Yihao Feng, Caiming Xiong, and Mi Zhou. Preference-grounded token-level guidance for language model fine-tuning. *Neural Information Processing Systems*, 2023.

[79] Gyuhak Kim, S. Thakur, Su Min Park, Wei Wei, and Yujia Bao. Sft-go: Supervised fine-tuning with group optimization for large language models. *arXiv.org*, 2025.

[80] Tuomas Haarnoja, Haoran Tang, P. Abbeel, and S. Levine. Reinforcement learning with deep energy-based policies. *International Conference on Machine Learning*, 2017.

[81] Tuomas Haarnoja, Aurick Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. *International Conference on Machine Learning*, 2018.