DISTILLSPEC: IMPROVING SPECULATIVE DECODING VIA KNOWLEDGE DISTILLATION

Yongchao Zhou[1,3]*, **Kaifeng Lyu**[1,4]*, **Ankit Singh Rawat**[1], **Aditya Krishna Menon**[1], **Afshin Rostamizadeh**[1], **Sanjiv Kumar**[1], **Jean-François Kagy**[1]†, **Rishabh Agarwal**[2,5]†
[1]Google Research [2]Google DeepMind [3]University of Toronto [4]Princeton University [5]Mila

ABSTRACT

Speculative decoding (SD) accelerates large language model inference by employing a faster *draft* model for generating multiple tokens, which are then verified in parallel by the larger *target* model, resulting in the text generated according to the target model distribution. However, identifying a compact draft model that is well-aligned with the target model is challenging. To tackle this issue, we propose *DistillSpec* that uses knowledge distillation to better align the draft model with the target model, before applying SD. DistillSpec makes two key design choices, which we demonstrate via systematic study to be crucial to improve the draft and target alignment: utilizing *on-policy* data generation from the draft model, and *tailoring the divergence function* to the task and decoding strategy. Notably, DistillSpec yields impressive $10-45\%$ speedups over standard SD on a range of standard benchmarks, using both greedy and non-greedy sampling. Furthermore, we combine DistillSpec with lossy SD to achieve fine-grained control over the latency vs. task performance trade-off. Finally, in practical scenarios with models of varying sizes, first using distillation to boost the performance of the target model and then applying DistillSpec to train a well-aligned draft model can reduce decoding latency by $6-10\times$ with minimal performance drop, compared to standard decoding without distillation.

1 INTRODUCTION

Large language models (LLMs) have revolutionized natural language understanding and generation across diverse applications (OpenAI, 2023; Anil et al., 2023). However, their autoregressive generation nature poses significant computational challenges, especially in real-time deployments with stringent latency constraints (Thoppilan et al., 2022; Pope et al., 2023). Conversely, smaller language models, while computationally efficient, often lack the expressive power of their larger counterparts and achieve subpar performance. While reducing the inference cost of larger models, e.g., via quantization or pruning, or improving the performance of the smaller models, e.g., via knowledge distillation (KD) (Hinton et al., 2015), constitute natural approaches to enable a favorable performance versus inference cost trade-off, these approaches frequently result in unacceptable performance gap compared to the high-quality large models. This has inspired a growing literature on designing mechanisms that combine both large and small models at inference to approximate the performance of the larger models without incurring their high computational cost.

Conventionally, model cascading approaches aim to identify easy instances where smaller models suffice to achieve good performance, thereby soliciting larger models only on a subset of hard instances (Rowley et al., 1998; Xu et al., 2014) or tasks (Cai et al., 2023b). Different from such task- or instance-level cascading, *speculative decoding* (SD) (Leviathan et al., 2023; Chen et al., 2023) aims to exploit the token-level variability in the computation demand during LLM inference by interactively invoking a small "draft" model and a large "target" model. At a given stage during inference, the draft model generates successive candidate tokens for multiple inference steps via autoregressive decoding. The target model then verifies the candidate tokens via parallel decoding, and employs rejection sampling to accept a subset of candidate tokens at contiguous positions.

The main objective of SD is to speed up text generation while guaranteeing that the decoded tokens follow the target model distribution. SD relies on the insight that the combined cost of autoregressive decoding with a small draft model followed by parallel decoding with the target model is lower than the cost of autoregressive decoding with the target model alone. However, the realized inference cost

*Student Researcher at Google Research. †Advising contribution. Corresponding authors: <yczhou@cs.toronto.edu>, <jfkagy@google.com>, and <rishabhagarwal@google.com>.

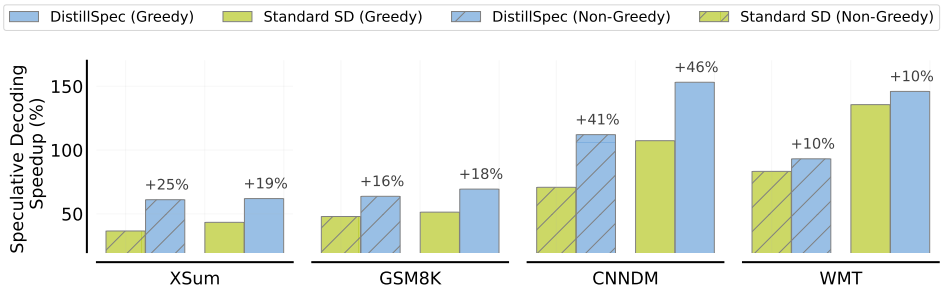


Figure 1: Performance comparison of standard speculative decoding (SD) vs. our proposed DistillSpec, with small and XL sized models from the T5 v1.1 family (Raffel et al., 2020) being utilized as the draft and the target models, respectively. DistillSpec enhances SD speed by better aligning the draft with the target via white-box knowledge distillation, resulting in a consistent $10-45\%$ improvement over Standard SD across various datasets. See § 5.1 for additional details.

reduction or latency improvement crucially depends on the *acceptance rate* of the draft-generated tokens by the target model, which can be shown to be directly tied to the alignment between the token distributions of the draft and target models. Thus, a successful application of SD hinges on identifying a compact draft model that simultaneously has small autoregressive decoding cost *and* is closely aligned with the target model.

In this work, we propose DistillSpec, a novel approach that relies on KD (Hinton et al., 2015) to obtain an effective draft model. Unlike the standard application of KD which primarily focuses on improving the task performance of a small student model, DistillSpec aims at aligning the student (draft) model with the teacher (target) model to enhance the acceptance rate during speculative decoding. This requires the student model to closely approximate the teacher distribution at the token and sequence level, even if it translates to suboptimal downstream task performance.

We undertake a comprehensive exploration of the distillation process for speeding up SD, considering several factors including the composition of training data, choice of divergence functions to define the training objective for KD, and decoding strategies. Notably, our findings underscore that using model-generated data is crucial for ensuring strong student-teacher alignment across various tasks via KD, and that the selection of the best-performing divergence function in DistillSpec is highly task-dependent and sensitive to the decoding strategy (i.e., greedy versus non-greedy). Furthermore, we explore the utility of DistillSpec for lossy SD (Leviathan et al., 2023) which allows for sampling away from the target model distribution. We show that combining DistillSpec with lossy SD enables a more fine-grained control over the latency versus task performance trade-off.

Finally, we carry out a systematic study of how to design an efficient inference scheme in a practical setting where one has access to multiple language models of increasing size and quality. Leveraging the insights that we have laid out in this paper about KD and SD, our study concludes that the most effective strategy involves first distilling a large model into a smaller one as the potential target model for performance optimization, followed by DistillSpec for distilling an even smaller model to be used as the draft model in SD. This approach results in a remarkable $6-10\times$ reduction in latency, compared to a standalone non-distilled target of same size, with minimal performance degradation.

Our key contributions are:

(i) We propose DistillSpec that uses KD to enhance draft model alignment with the target model (§4), and show that our method can improve SD speed by $10-45\%$ while preserving model performance across four diverse datasets with both greedy and non-greedy sampling (Figure 1).

(ii) We conduct an extensive analysis of the optimal distillation recipe (§5.2) for model alignment, encompassing factors such as training data generation and different divergences, and emphasizing the distinctions between standard KD and distillation tailored for SD.

(iii) We extend DistillSpec to lossy SD, enabling refined control over the quality-latency trade-off. Moreover, we offer insights for combining KD and SD when several models are available (§5.3).

2 Related Work

Speculative decoding (SD). Due to the inherent sequential nature of autoregressive decoding, the primary latency bottleneck in LLM inference arises from memory read/write operations rather than

arithmetic computations (Pope et al., 2023). Speculative decoding (Leviathan et al., 2023; Chen et al., 2023) (SD) addresses this challenge by utilizing a compact draft model to generate a batch of tokens sequentially, while validating them in parallel with a larger target model. Prior to SD, various parallel computing paradigms have been explored for autoregressive models, including block parallel sampling (Stern et al., 2018), shallow aggressive decoding (Sun et al., 2021), and aggressive decoding (Ge et al., 2022). However, these approaches are not readily adaptable to typical language models due to potential deviations from target model's distribution, strict input constraints, or limited support for general stochastic sampling. Notably, recent variants of SD have also incorporated parallel computation along the batch axis, sometimes combined with token tree verification, as seen in SpecTr (Sun et al., 2023), SpecInfer (Miao et al., 2023), and Medusa (Cai et al., 2023a). In contrast, our work focuses on enhancing SD by improving the alignment between the small draft model and the large target model through KD, which does not require any changes to serving infrastructures already implementing SD and is complementary to the recent variants of SD. Furthermore, we do a systematic study of lossy SD for providing nuanced control over the trade-off between quality and latency for specific serving models.

Knowledge distillation (KD) for LLMs. KD (Buciluǎ et al., 2006; Hinton et al., 2015), which trains high-quality smaller student models with the help of larger teacher models, has emerged as a vital technique for reducing the inference cost while maintaining performance quality across a range of domains. In the context of LLMs, prior uses of KD (Taori et al., 2023; Fu et al., 2023) have mostly focused on *black-box* KD, wherein only teacher's output generations, often via APIs, are accessible during student training. However, with the proliferation of open-source LLMs (Zhang et al., 2022; Touvron et al., 2023), there is a growing interest in *white-box* KD, where we have access to teacher weights and logits. White-box KD allows student models to benefit from richer supervision signals provided by white-box teacher models, leading to enhanced language abilities (Agarwal et al., 2023; Gu et al., 2023; Wen et al., 2023). Despite notable improvements in student quality, substantial performance gaps persist between large and small models (OpenAI, 2023; Anil et al., 2023), which may remain unbridgeable through distillation alone.

Unlike prior works focused on creating highly capable standalone student models, we harness KD to foster closer collaboration between smaller and larger models in SD, which may be particularly valuable when a small distilled model alone cannot meet stringent quality requirements. While Stern et al. (2018) use an black-box KD approach (SeqKD) to enhance blockwise parallel decoding, they use samples generated from the large target model, which is prohibitively expensive for LLMs. Furthermore, they ignore the teacher model's logits and train their draft model using only one-hot teacher labels – a reasonable choice for greedy decoding but a less effective one for non-greedy sampling (Figure 2). Relatedly, Rawat et al. (2021) leverage KD to improve model cascading. However, different from our efforts which focus on text generation with LLMs, their study focuses on classifications tasks in vision and NLP domains.

3 Background: Speculative Decoding

Notation. Given an input sequence x comprising tokens from a pre-defined vocabulary, a language model $\mathcal{M}$ provides a distribution over possible output sequences y. Suppose we employ SD with a **compact draft model** $\mathcal{M}_q$ to assist a **larger target model** $\mathcal{M}_p$. Let $p(y_t \mid x, y_{<t})$ and $q(y_t \mid x, y_{<t})$ represent the distributions governing next-token predictions at time step t for $\mathcal{M}_p$ and $\mathcal{M}_q$, respectively, given the context $\rho = \{x, y_{<t}\}$. Given input x as prefix, let $p_{\leq T}(y \mid x)$ and $q_{\leq T}(y \mid x)$ represent the distributions governing the sequence y sampled autoregressively from $\mathcal{M}_p$ and $\mathcal{M}_q$, respectively, where the generation stops either when an end-of-sequence token is sampled, or the maximum sequence length T is reached. For simplicity, we use $p(y_t)$ and $q(y_t)$ as shorthands for $p(y_t \mid x, y_{<t})$ and $q(y_t \mid x, y_{<t})$, whenever the context ρ is clear. Similarly, $p_{\leq T}(y)$ and $q_{\leq T}(y)$ serve as shorthands for $p_{\leq T}(y \mid x)$ and $q_{\leq T}(y \mid x)$, whenever the input x is clear.

Speculative sampling. Standard SD uses a procedure called *speculative sampling* to generate tokens from the draft model while maintaining the same output distribution as the target model. As detailed in Algorithm A.1 (Appendix), each step of SD works as follows. First, a *block* of γ tokens, denoted as $y_t, \ldots, y_{t+\gamma-1}$, is autoregressively sampled from $q(y_t), \ldots, q(y_{t+\gamma-1})$. Next, the γ tokens are verified in parallel by passing them to $\mathcal{M}_p$ as a whole block, which sequentially accepts token y_{t+i} with probability $\min\left(1, p(y_{t+i})/q(y_{t+i})\right)$. If any token y_{t+i} is rejected before the end of the block, the subsequent tokens are discarded and the rejected token is resampled from the adjusted distribution $p'(y_{t+i}) \propto \max(0, p(y_{t+i}) - q(y_{t+i}))$; otherwise, the tokens are all accepted

and an extra token is sampled from $p(y_{t+\gamma})$ and appended to the output sequence. This process guarantees that the sequence of accepted and resampled tokens follow the same output distribution as $p(y_{t+i})$ (Leviathan et al., 2023). The procedure is repeated until an end-of-sequence token is accepted, or the maximum sequence length T has been reached.

Efficiency measure: Acceptance rates. Each SD step takes a constant amount of time, so the wall-clock time scales linearly with the number of steps. This number is equal to the total number of steps that the target model rejects a token, plus the number of blocks accepted as a whole, where the latter term is small for large γ. This motivates us to use the *acceptance rate* as a surrogate efficiency measure for the wall-clock time. For an ideal SD process with $\gamma = \infty$, we define the **sequence-level acceptance rate** $\alpha(x)$ for a given input x as follows:

$$\alpha(x) := \frac{\mathbb{E}\left[\text{number of accepted tokens in generating } y\right]}{\mathbb{E}\left[\text{number of tokens in } y\right]} = \frac{\mathbb{E}_{y \sim p_{\leq T}(y \mid x)}\left[\sum_{t=1}^{|y|} \beta(x, y_{<t})\right]}{L_p(x)}, \quad (1)$$

where $\beta(x, y_{<t}) := \mathbb{E}_{y_t \sim q(y_t)}\left[\min\left(1, p(y_t)/q(y_t)\right)\right]$ is the token-level acceptance rate, and expectations are taken over the randomness in SD. Since speculative sampling preserves the output distribution of the target model, the denominator is simply equal to the expected length of the target output $L_p(x) := \mathbb{E}_{y \sim p_{\leq T}(y \mid x)}\left[|y|\right]$, which does not change with the draft model. Therefore, the acceptance rate is directly related to the expected total number of rejected tokens $(1 - \alpha(x)) \cdot L_p(x)$, which lower bounds the expected number of SD steps.

Efficiency measure: Block efficiency. In practice, SD is usually employed with a fixed finite block size γ; thus, a simple efficiency measure is the **block efficiency** τ. Given an input x, we compute the block efficiency $\tau(x)$ as the expected number of generated tokens per block. Note that, for a given block of size γ, the maximum value of $\tau(x)$ is $\gamma + 1$, which accounts for the case where all draft tokens are accepted and the target model appends an additional token. If we assume token-level rates $\beta(x, y_{<t})$ are i.i.d., then the acceptance rate $\alpha = \mathbb{E}\left[\beta\right]$ and $\tau(x) = (1 - \alpha^{\gamma+1})/(1 - \alpha)$ (Leviathan et al., 2023).

Walltime improvement. For given block efficiency $\tau(x)$, the expected speedup of SD is given by $\tau(x)/(c\gamma+1)$, where the relative latency c is the ratio between the times for making a single forward pass through the draft model $\mathcal{M}_q$ and target model $\mathcal{M}_p$.

4 DistillSpec: Knowledge Distillation for Speculative Decoding

As described in § 3, speculative decoding (SD) can reduce the latency of the larger (target) model with the help of a smaller (draft) model without any performance drop. However, the realized latency reduction via SD critically depends on how "well-aligned" the draft model is to the target model. Setting performance improvement of SD as a primary objective, our proposed DistillSpec method enables a closer alignment between the draft and target models by leveraging KD. We first present KD-based training of the draft model. We highlight how our objective of enhancing SD via KD influences our selection of training data generation method and divergence function – two key ingredients of DistillSpec. We then discuss how DistillSpec can be extended to lossy SD.

Let the draft model $\mathcal{M}_q^\theta$ be parameterized by θ. DistillSpec utilizes predictions from the target model $\mathcal{M}_p$ as a source of supervision (teacher) while training the draft model $\mathcal{M}_q^\theta$ (student). We assume white-box access to both models, i.e., we can obtain their token-level distributions $p(y_t)$ and $q(y_t)$, and therefore we are able to generate samples both from the target and draft models. Given a divergence function D that measures the misalignment between two distributions, KD-based training of the draft model seeks to minimize the divergence between the teacher (target) and student (draft) distributions over a training set $\mathcal{G}$:

$$\theta^* := \arg\min \mathbb{E}_{(x,y)\sim\mathcal{G}}\left[D(\mathcal{M}_p \| \mathcal{M}_q^\theta)(y|x)\right], \quad (2)$$

where $D(\mathcal{M}_p \| \mathcal{M}_q^\theta)(y|x) = \frac{1}{|y|}\sum_{t=1}^{|y|} D\big(p(\cdot \mid x, y_{<t}) \| q^\theta(\cdot \mid x, y_{<t})\big)$.We note that different KD algorithms differ in how they construct $\mathcal{G}$ and their choice of divergence D. For instance, $\mathcal{G}$ may consist of task-specific input-output pairs (X, Y) or sequences generated from $\mathcal{M}_p$ or $\mathcal{M}_q^\theta$. While *forward* KL (D_{FKL}) is the commonly used divergence for KD (Hinton et al., 2015), recent works (Agarwal et al., 2023; Wei et al., 2022) show promising results with alternative divergences, including *reverse*

Table 1: Summary of various KD algorithms in terms of training data $\mathcal{G}$ and divergence D (cf. Eq. 2). Wen et al. (2023); Agarwal et al. (2023) also consider other D; we list the most representative one.

Name	Divergence	Training Data ($\mathcal{G}$)
SeqKD (Kim & Rush, 2016) (**Black-box KD**)	FKL	Data generated by Teacher $\mathcal{M}_p$
Supervised KD (Hinton et al., 2015)	FKL	Fixed dataset of input-output pairs
ImitKD (Lin et al., 2020)	FKL	Fixed dataset + Data generated by $\mathcal{M}_q^\theta$
f-Distill (Wen et al., 2023)	TVD	Data generated by $\mathcal{M}_q^\theta$ and $\mathcal{M}_p$
On-policy GKD (Agarwal et al., 2023)	FKL / JSD	On-policy data from Student $\mathcal{M}_q^\theta$

KL (D_{RKL}), Jensen–Shannon divergence ($D_{\mathrm{JSD}[\beta]}$), and total variation distance (D_{TVD}). Further details on each divergence can be found in Appendix B.1. Table 1 summarizes various distillation algorithms, each being a specialized instance of Algorithm A.2 (Appendix).

Our choices for $\mathcal{G}$ and D are guided by how the resulting distilled model, once employed as draft model, improves the speed of SD. Towards this, we first highlight the role that the total variation distance between $p(y_t)$ and $q(y_t)$ plays in dictating the acceptance rate (§ 3)—a key efficiency measure for SD.

Acceptance rate as total variation distance. Leviathan et al. (2023, Corollary 3.6) shows that the token-level acceptance rate $\beta(x, y_{<t})$ satisfies $\beta(x, y_{<t}) = 1 - D_{\mathrm{TVD}}(p(y_t), q(y_t))$. Hence, Eq. 1 implies that maximizing the sequence-level acceptance rate $\alpha(x)$ is equivalent to minimizing the expected D_{TVD} between $p(y_t)$ and $q(y_t)$ over the output sequence distribution of $\mathcal{M}_p$, i.e.,

$$\alpha(x) = 1 - \mathbb{E}_{y \sim p_{\leq T}(y \mid x)} \Big[\sum_{t=1}^{|y|} D_{\mathrm{TVD}}(p(y_t), q(y_t)) \Big] / L_p(x). \tag{3}$$

Choice of divergence. Based on Eq. 3, it appears that directly minimizing D_{TVD} may be a principled objective for draft model distillation. While optimizing $D_{\mathrm{TVD}}(p, q)$ is theoretically inspired, our empirical study shows that such an objective may not consistently yield optimal results. We find that the choice of the most suitable divergence is highly task-dependent (§ 5.2).

Choice of training data: Sampling from the student. As for $\mathcal{G}$, Eq. 3 suggests optimizing the expected D_{TVD} over outputs generated from the teacher. However, decoding from a large teacher is generally prohibitively expensive, especially at the scale of dataset required for KD. Alternatively, one could resort to an existing ground-truth dataset, however the teacher's output distribution may deviate from the ground-truth distribution despite the teacher having been fine-tuned on it. Moreover, ground-truth datasets are often limited in size, so training *only* on such data could result in overfitting. To resolve these issues, we explore using *on-policy data* during distillation, i.e., output sequences sampled from the student itself. Besides being more computationally efficient compared to teacher generations, this approach is inspired by Gu et al. (2023); Agarwal et al. (2023), where distilling on on-policy data is shown to improve student task performance. However, different from these prior works, our primary focus is on improving the student-teacher alignment. Thus, it is not immediately clear whether minimizing the expected D_{TVD} over on-policy (student-generated) data ensures an improved acceptance rate, which is computed as an expectation over the *teacher's* output distribution (cf. Eq. 3). For this, our following result shows that this is indeed the case.

Theorem 4.1. *For SD, if the draft model $\mathcal{M}_q^\theta$ achieves on-policy KD loss $\epsilon = \mathbb{E}_{x \sim X, y \sim q_{\leq T}(y \mid x)} \left[D_{\mathrm{TVD}}(\mathcal{M}_p \| \mathcal{M}_q^\theta)(y \mid x) \right]$, then the sequence-level acceptance rate is at least*

$$\mathbb{E}_{x \sim X} \left[\alpha(x) \right] \geq 1 - T \cdot \mathbb{E}_{x \sim X} \left[\frac{T}{L_p(x)} \right] \epsilon. \tag{4}$$

When the target output length is always T, the bound simplifies to $\mathbb{E}_{x \sim X} \left[\alpha(x) \right] \geq 1 - T\epsilon$.

We defer the proof to Appendix B.2. Intuitively, it builds upon the following insights. If the on-policy KD loss is small, then, for any $1 \leq t \leq T$, the same loss evaluated only at the t-th token should also be small. Since the first token generation is independent of any other tokens, a small value of online-policy KD loss ensures that the first token distributions of the draft and target models are close. Then, an inductive argument shows that once the draft and target are similar on the first t tokens, the distributions of the $(t+1)$-th token should also be close. Our proof makes this argument rigorous by utilizing variational representations of D_{TVD}, leading to a linear error bound in T.

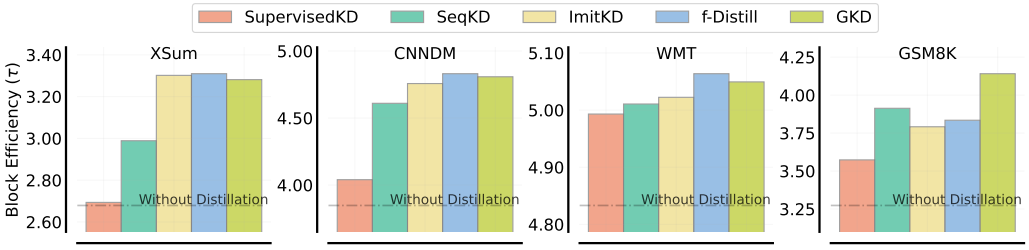


Figure 2: Distillation enhances block efficiency (τ) across diverse datasets, highlighting the superiority of model-generated data over fixed ground truth data (SupervisedKD) and emphasizing the importance of white-box distillation in addressing SeqKD's subpar performance.

DistillSpec with lossy SD. DistillSpec enhances SD efficiency without any quality loss compared to the target model $\mathcal{M}_p$. In practical applications, a reduction in model quality could be justified in order to support even faster inference with a larger target model. For such scenarios, we extend DistillSpec to lossy SD (Leviathan et al., 2023), which uses a *lenience function* $f(p, \epsilon)$ that modifies the acceptance probability from $\min(1, p(y_t)/q(y_t))$ to $\min(1, f(p(y_t), \epsilon)/q(y_t))$ (cf. § 3). Here $f : [0, 1]^2 \to \mathbb{R}^+$ is increasing and decreasing in its first and second arguments, respectively, and $\epsilon \in [0, 1]$ is a parameter we choose (cf. Algorithm A.1). In this work, we evaluate multiple choices for the lenience functions: $f_{\text{lin}}(p, \epsilon) = p/\epsilon$, $f_{\text{sq}}(p, \epsilon) = p/\epsilon^2$, and $f_{\text{exp}}(p, \epsilon) = p^\epsilon$. For example, when $\epsilon = 0.1$ and the lenience function is $f_{\text{sq}}(p, \epsilon)$, token x from $q(y_t)$ becomes hundred times more likely to be accepted by the target, thus enabling faster inference at the expense of a potential drop in generation quality. Lenience was discussed by Leviathan et al. (2023) in the context of f_{lin} and their treatment focuses solely on latency improvements, whereas we explore the use of different lenience functions as a precise control mechanism for the desired quality-latency trade-off.

5 Experiments

5.1 Enhancing speculative decoding through distillation

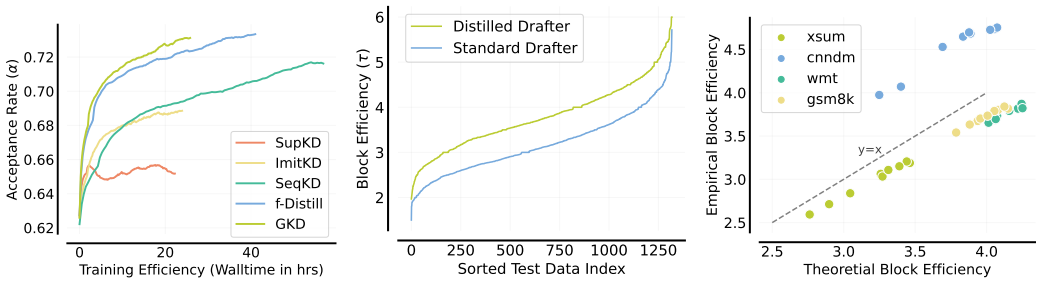
We evaluate the effectiveness of KD in improving the speed of speculative decoding (SD). We specifically investigate its impact on the enhancement of acceptance rate, block efficiency, and latency.

Tasks and models. Following Leviathan et al. (2023), we evaluate two model types: 1) GPT-like decoder-only Transformer models trained using the standard autoregressive objective on LM1B task (Chelba et al., 2013), where the target and draft models have 234M and 33M parameters, respectively; and 2) Standard encoder-decoder T5 v1.1 models (Raffel et al., 2020) supervised fine-tuned on four different tasks, with T5-XL (3B) and T5-Small (77M) serving as the target and draft models, respectively. As for the four datasets, we utilize two datasets form the T5 paper, namely WMT EnDe (Bojar et al., 2014) and CNN/DM (Hermann et al., 2015) which deal with translation and text summarization, respectively. The remaining two tasks used to test T5 models are XSum (Narayan et al., 2018) and GSM8K (Cobbe et al., 2021), which deal with abstractive summarization and arithmetic reasoning, respectively. See Appendix C for more details.

Training data for KD. We study five KD algorithms outlined in Table 1. However, for SeqKD (Kim & Rush, 2016) and f-Distill (Wen et al., 2023), we opt for an online data generation approach from the teacher instead of a conventional fixed offline teacher-generated dataset. This approach, while computationally expensive, yields a more diverse dataset. For GKD, we solely rely on the data generated by the online student model, excluding the static ground truth data. All data generated by either the teacher or the student is based on temperature sampling with a temperature of 1.0 (see Appendix D.1.3 for an ablation study on sampling temperature).

Evaluation. For each pair of target model and draft model, we measure the empirical acceptance rate α and block efficiency τ with three different block sizes $\gamma \in \{3, 5, 7\}$, both with and without distillation. As per Leviathan et al. (2023), we measure the relative wall time improvements with a batch size of 1 for both greedy sampling ($T = 0$) and standard temperature sampling ($T = 1$).

Results. Figure 1 shows that the impact of distillation on SD speed is evident, consistently yielding a 10−46% improvement across various datasets. This effect is most pronounced when employing greedy decoding. The summary of results for different block sizes and decoding strategies across



(a) Acceptance rate v.s. Walltime (b) Instance-level Block Eff. (τ) (c) Agreement of τ

Figure 3: (a) White-box KD using teacher logits and model-generated data is crucial. Draft model's on-policy data can be as effective as target model data. GKD achieves the best wall-time performance improvement. (b) Distillation improves the block efficiency for all examples (c) Empirical block efficiency aligns well with its D_{TVD}-based theoretical counterpart.

five datasets is presented in Table D.1 (Appendix), highlighting the superior performance of the KD algorithm in terms of latency. The findings demonstrate that KD significantly enhances the acceptance rate and block efficiency for both decoder-only and encoder-decoder models across all datasets. Distillation algorithms utilizing model-generated data consistently outperform other approaches, resulting in approximately a $\sim$20% additional speedup compared to standard SD on LM1B, XSum, CNN/DM, and GSM8K. However, the gains on the WMT dataset are marginal, as the preliminary model already achieves a high acceptance rate and block efficiency without KD.

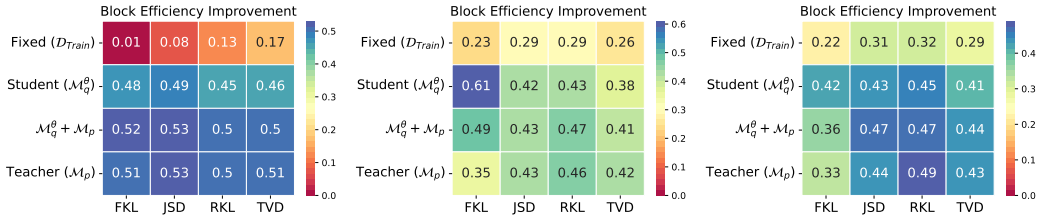
Figure 2 presents a comparison of block efficiency across different algorithms, employing temperature sampling ($T = 1$) with a block size $\gamma = 7$. The figure showcases the utility of model generated data as using a fixed ground truth data (i.e., Supervised KD) ranked the lowest across all settings except WMT. In contrast, f-Distill and GKD which use the purely model generated data significantly outperform the other counterparts. Besides, the subpar performance of SeqKD, despite purely trained on the data generated by the target model, implies that white-box distillation (information from the target model's logits) is vital for SD. This is corroborated by Figure 3a, which illustrates the evolution of the acceptance rate throughout the training. Supervised KD ranks lowest, and its performance plateaus as training progresses due to a static dataset. In contrast, all other algorithms that employ model-generated data continue to improve. Despite f-Distill being much more computationally costly than GKD due to using teacher generated data, both exhibit comparable performance. Notably, GKD achieves the best wall-time performance improvement. (see Appendix D.1.2 for more visualizations on performance improvement during training).

We investigate whether KD improves block efficiency universally or mainly impacts a subset of examples. We depict the change in block efficiency per example in Figure 3b. The results reveal a consistent enhancement in block efficiency across most examples, as also seen in various datasets (see Figure D.12). Figure 3c illustrates a strong agreement between theoretical and empirical block efficiency values for several distilled models (each model as a filled circle). Despite theoretical values occasionally overestimating or underestimating empirical values due to potential deviations from the i.i.d. token-level assumption (cf.§3), the ranking of distilled models remains highly consistent. In summary, these findings affirm that KD effectively optimizes block efficiency.

5.2 DistillSpec recipe

We now focus on identifying the optimal KD approach for SD. Following the training and evaluation protocols in § 5.1, we explore four training data construction methods and four divergence functions on XSum and GSM8K. In particular, we explore the following variants of training data: 1) Fixed ground-truth dataset $\mathcal{D}_{\text{Train}}$, 2) Data generated only from the draft $\mathcal{M}_q^\theta$, 3) Data generated only from teacher $\mathcal{M}_p$, 4) Data generated from both $\mathcal{M}_q^\theta$ and $\mathcal{M}_p$ in equal proportion. We further examine the following divergences: 1) Forward KL (FKL), 2) Jenson-Shannon divergence (JSD), 3) Reverse KL (RKL), and 4) Total variation distance (TVD).

Importance of training data and divergence in DistillSpec. Figure 4 illustrates the block efficiency improvement on XSum and GSM8K, in line with observations from § 5.1. Note that using model-generated data (last three rows) yields superior performance than using a fixed dataset (first row). Specifically, on XSum with greedy decoding, using data generated from both $\mathcal{M}_q^\theta$ and $\mathcal{M}_p$



(a) XSum: τ (Greedy) (b) GSM8K: τ (Greedy) (c) GSM8K: τ (Non-greedy)

Figure 4: **DistillSpec Recipe**. We report improvement in empirical block efficiency after distillation on (a) XSum with greedy sampling, and GSM8K with (b) greedy and (c) non-greedy sampling. The optimal divergence function and data construction to improve the block efficiency highly depends on the task. We should treat them as a hyperparameter on our task of interest.

gives the best performance, with JSD slightly outperforming the other divergences. However, on GSM8K with greedy sampling, FKL with only draft $\mathcal{M}_q^\theta$ generated data emerges as the best KD setup. In contrast, with temperature sampling (at $T = 1$), a different trend is observed as RKL combined with data generated by $\mathcal{M}_p$ is the most effective. See Appendix D.2.1 for results on different datasets and decoding strategies. Nonetheless, using only draft generated data is competitive.

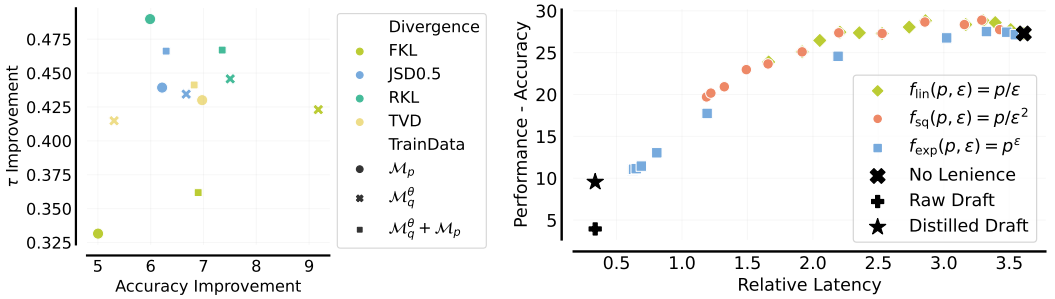
Impact of distillation on draft quality vs. compatibility. We also study how different distillation approaches affect draft model's task performance and if the same design choices are optimal for improving both draft task performance and its utility for SD (cf. Figure D.18, D.19). Similar to our earlier observations, the use of generated data is paramount for improving draft performance. More notably, utilizing data generated from $\mathcal{M}_q^\theta$ yields comparable or superior results compared to using data generated from $\mathcal{M}_p$. However, the choice of the optimal KD algorithm largely depends on the task and the underlying decoding strategy. More interestingly, Figure 5a highlight a dichotomy between block efficiency improvements and task performance gains via KD as a drafter with high task performance does not necessarily indicate a powerful drafter for SD. See Appendix D.2.2 for more results on different datasets and decoding strategies.

Recommendation. Interestingly, although TVD is the objective we aim to optimize for SD (cf. Eq. 3), its direct optimization does not yield the best performance in most of the settings explored. We generally find that the choice of the divergence in KD is a hyperparameter that needs to be tuned based on the task and decoding strategy used. For training data construction, we propose using draft $\mathcal{M}_q^\theta$ for data generation as it can achieve similar or superior performance compared to teacher $\mathcal{M}_p$, but at a much lower cost.

5.3 Quality versus latency trade-off

Lossy speculative decoding. We analyze the quality-latency trade-off using lossy SD variants, as detailed in Algorithm A.1. Figure 5b illustrates that employing either KD ($\star$) or SD ($\times$) alone does not fully bridge the performance or latency gaps, respectively. In such cases, a leniency parameter can help interpolate between these two approaches, as demonstrated in Figure 5b. Interpolating on GSM8K presents challenges, as f_{lin} still results in high performance and latency when using a lenience of 10^{-5}, while f_{sq} pushes it further along the trade-off curve. Although f_{exp} can still interpolate, it yields a worse trade-off. Interestingly, it is possible to significantly reduce latency while almost preserving the quality in GSM8K, possibly because many tokens have little impact on the final performance, and accepting them is a model preference with minimal effect on generation quality. See Appendix D.3.1 for comparison between raw draft and distilled draft models, where we show that distilled draft achieves a much better trade-off.

DistillSpec meets model garden. In practical scenarios, we often have access to multiple models of different sizes – a model garden – to design the inference pipeline. We consider this setting by focusing on the T5-models with five sizes: T5-Small (77M), T5-Base (250M), T5-Large (800M), T5-XL (3B), and T5-XXL (11B). We study four different quality-latency trade-off curves using KD and SD: 1) **Raw:** Deploying supervised fine-tuned (SFT) T5 models; 2) **Distilled:** Applying KD to T5 models for optimizing downstream task performance 3) **Speculative:** Applying SD to T5 models; and 4) **DistillSpec:** Applying KD to T5 models and using distilled models as target and draft.



(a) Performance vs. SD Compatibility (GSM8K)

(b) DistillSpec with Lossy SD (GSM8K)

Figure 5: (a) The improvement on speculative decoding and downstream task performance are different. A high performance model do not imply it will be good for speculative decoding. (b) We employ leniency as a precise control to achieve the desired quality-latency trade-off.

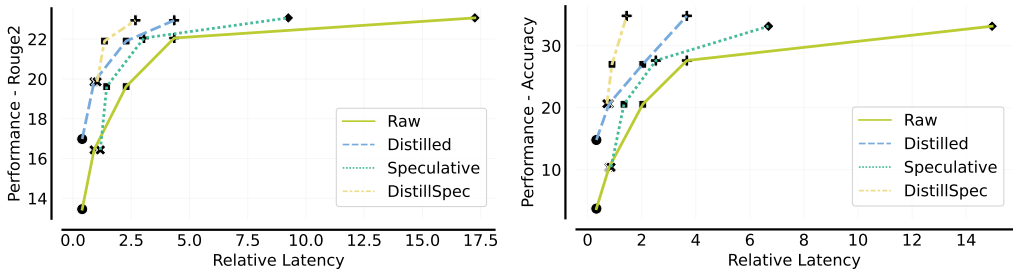


Figure 6: DistillSpec excels in quality and latency, offering a remarkable 6.4x and 10.7x latency reduction for XSum (left) and GSM8K (right), while maintaining nearly identical performance.

Figure 6 illustrates that SD effectively shifts the trade-off curve leftward, especially with larger model sizes. However, its efficacy diminishes with smaller model sizes when the relative computation time between the draft and target models is closely matched. In contrast, distillation, which optimizes the model for downstream task performance, appears to offer a superior trade-off between quality and latency, particularly for smaller models. Conversely, a reverse trend is observed for larger model sizes when evaluating the model with temperature sampling. Figure D.25a indicates a substantial gap between the distilled model and the larger teacher model, while the SD-based method significantly reduces latency. This suggests that when stringent performance and decoding strategy constraints are in place, SD remains a valuable approach. Notably, our method, DistillSpec, which combines the benefits of distillation and SD, consistently achieves the best trade-off between quality and latency, resulting in an impressive reduction in latency while maintaining nearly identical performance. Specifically, DistillSpec reduces relative latency from 17.3 to 2.7 and from 15.0 to 1.4 on XSum and GSM8K, respectively, representing speedup improvements of 6.4× and 10.7×. In contrast, the Rouge2 Score only experiences a marginal decrease, shifting from 23.1 to 23.0 on XSum, while the model accuracy on GSM8K actually improves, rising from 33.1 to 34.8.

6 Conclusion

In this paper, we evaluate the efficacy of white-box knowledge distillation (KD) in enhancing speculative decoding (SD) through improved alignment between target and draft models. A thorough analysis is conducted to understand the impact of training data construction and divergence functions on KD performance. We underscore the significance of utilizing model-generated data and argue that employing the draft model's on-policy data during KD is a cost-efficient method for realizing the improved alignment. Additionally, we assess the trade-off between quality and latency within the scope of lenience and availability of multiple models of varying quality and size, concluding that KD procures a superior trade-off compared to standard SD. The optimal strategy involves initially applying KD for downstream task performance, followed by SD, resulting in a six to ten-fold decrease in latency with negligible performance loss. Our study contributes novel insights into the white-box KD algorithms for LLMs and provides guidance on striking an effective balance between quality and latency using KD and SD.

AUTHOR CONTRIBUTIONS

YZ led the project and conducted all distillation experiments and evaluations for T5 models. YZ also wrote the initial draft of the paper and made all the plots and tables. KL provided a theoretical justification of the proposed method via Theorem 4.1 and revised the paper. ASR and AKM gave high-level guidance on the project and revised the paper. AR and SK gave high-level feedback on the project. JK supervised the project and served as a host to YZ at Google Research, and conducted evaluation of the decoder-only models. RA served as an advisor, provided detailed technical feedback, and revised the paper.

ACKNOWLEDGMENTS

We would like to extend a special thank you to Neha Gupta, Wittawat Jitkrittum, Nino Veillard, Yaniv Leviathan, Matan Kalman, Danny Vainstein, Natan Potikha, Ananda Theertha Suresh, Laz Karydas, Aishwarya PS, Pranav Nair, Praneeth Netrapalli, Nikunj Saunshi, Ziteng Sun, Keiran Paster, Olivier Bachem, Aleksandra Faust for insightful discussion and valuable feedback.

REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos, Matthieu Geist, and Olivier Bachem. Generalized knowledge distillation for auto-regressive language models. *arXiv preprint arXiv:2306.13649*, 2023.

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.

Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ale s Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W/W14/W14-3302.

Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 535–541, 2006.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, and Tri Dao. Medusa: Simple framework for accelerating llm generation with multiple decoding heads. https://github.com/FasterDecoding/Medusa, 2023a.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. *arXiv preprint arXiv:2305.17126*, 2023b.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One billion word benchmark for measuring progress in statistical language modeling. *arXiv preprint arXiv:1312.3005*, 2013.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.

Yao Fu, Hao Peng, Litu Ou, Ashish Sabharwal, and Tushar Khot. Specializing smaller language models towards multi-step reasoning. *arXiv preprint arXiv:2301.12726*, 2023.

Tao Ge, Heming Xia, Xin Sun, Si-Qing Chen, and Furu Wei. Lossless acceleration for seq2seq generation with aggressive decoding. *arXiv preprint arXiv:2205.10350*, 2022.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in neural information processing systems*, pp. 1693–1701, 2015.

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

Ferenc Huszár. How (not) to train your generative model: Scheduled sampling, likelihood, adversary? *arXiv preprint arXiv:1511.05101*, 2015.

Yoon Kim and Alexander M Rush. Sequence-level knowledge distillation. *arXiv preprint arXiv:1606.07947*, 2016.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, pp. 19274–19286. PMLR, 2023.

Alexander Lin, Jeremy Wohlwend, Howard Chen, and Tao Lei. Autoregressive knowledge distillation through imitation learning. *arXiv preprint arXiv:2009.07253*, 2020.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Rae Ying Yee Wong, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. Specinfer: Accelerating generative llm serving with speculative inference and token tree verification. *arXiv preprint arXiv:2305.09781*, 2023.

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. *ArXiv*, abs/1808.08745, 2018.

OpenAI. Gpt-4 technical report. *ArXiv*, abs/2303.08774, 2023. URL `https://api.semanticscholar.org/CorpusID:257532815`.

Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

Ankit Singh Rawat, Manzil Zaheer, Aditya Krishna Menon, Amr Ahmed, and Sanjiv Kumar. When in doubt, summon the titans: Efficient inference with large models. *arXiv preprint arXiv:2110.10305*, 2021.

H. A. Rowley, S. Baluja, and T. Kanade. Neural network-based face detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(1):23–38, 1998. doi: 10.1109/34.655647.

Noam Shazeer and Mitchell Stern. Adafactor: Adaptive learning rates with sublinear memory cost. In *International Conference on Machine Learning*, pp. 4596–4604. PMLR, 2018.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. Instantaneous grammatical error correction with shallow aggressive decoding. *arXiv preprint arXiv:2106.04970*, 2021.

Ziteng Sun, Ananda Theertha Suresh, Jae Hun Ro, Ahmad Beirami, Himanshu Jain, Felix Yu, Michael Riley, and Sanjiv Kumar. Spectr: Fast speculative decoding via optimal transport. In *Workshop on Efficient Systems for Foundation Models @ ICML2023*, 2023. URL `https://openreview.net/forum?id=d0mGsaheuT`.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. Stanford alpaca: An instruction-following llama model. `https://github.com/tatsu-lab/stanford_alpaca`, 2023.

Romal Thoppilan, Daniel De Freitas, Jamie Hall, Noam Shazeer, Apoorv Kulshreshtha, Heng-Tze Cheng, Alicia Jin, Taylor Bos, Leslie Baker, Yu Du, et al. Lamda: Language models for dialog applications. *arXiv preprint arXiv:2201.08239*, 2022.

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

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

Yuqiao Wen, Zichao Li, Wenyu Du, and Lili Mou. f-divergence minimization for sequence-level knowledge distillation. *arXiv preprint arXiv:2307.15190*, 2023.

Zhixiang (Eddie) Xu, Matt J. Kusner, Kilian Q. Weinberger, Minmin Chen, and Olivier Chapelle. Classifier cascades and trees for minimizing feature evaluation cost. *Journal of Machine Learning Research*, 15(62):2113–2144, 2014.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Appendix

Table of Contents

A DISTILLSPEC ALGORITHMS

Algorithm A.1 Speculative decoding step

Require: Target Model ($\mathcal{M}_p$), Draft Model ($\mathcal{M}_q$), Context ($\rho = \{x, y_{<t}\}$)
Hyperparameters: Block Size (γ), Lenience Function $f(p, \epsilon)$ ($\epsilon = 1$ for lossless and $\epsilon < 1$ for lossy decoding)

for all $i = 0$ to $\gamma - 1$ **do**
  $q_{t+i}(y) \leftarrow \mathcal{M}_q\left(x, y_{<t+i}\right), \quad y_{t+i} \sim q_{t+i}(y)$ ▷ Sample γ tokens from $\mathcal{M}_q$ autoregressively.
end for
$(p_t(y), \ldots, p_{t+\gamma}(y)) \leftarrow (\mathcal{M}_p(x, y_{<t}), \ldots, \mathcal{M}_p(x, y_{<t+\gamma}))$ ▷ Run $\mathcal{M}_p$ in parallel.
$\forall i \mid t \leq i < t + \gamma, \quad r_i = \frac{f(p_i(y), \epsilon)}{q_i(y)}$ ▷ Compute the rejection thresholds.
$u_t \sim \text{Uniform}(0, 1), \ldots, u_{t+\gamma-1} \sim \text{Uniform}(0, 1)$ ▷ Generate γ random values.
$n \leftarrow \min\left(\{i \mid 0 \leq i < \gamma, u_{t+i} > r_{t+i}\} \cup \{\gamma\}\right)$ ▷ Determine the number of accepted tokens n.
if $n < \gamma$ **then**
  $y_{t+n} \sim \text{norm}\left(\max\left(0, p_{t+n}(y) - q_{t+n}(y)\right)\right)$ ▷ Sample from the adjusted distribution.
else
  $y_{t+n} \sim p_{t+n}(y)$ ▷ Sample from $\mathcal{M}_p$.
end if
Return $\{x, y_{<t+n+1}\}$ ▷ Append n tokens from $\mathcal{M}_q$ and one token from $\mathcal{M}_p$.

Algorithm A.2 Knowledge distillation

Require: Target Model ($\mathcal{M}_p$), Draft Model ($\mathcal{M}_q^\theta$), Dataset (X, Y) contain input x and possibly output y.
Hyperparameters: Fixed data fraction $\lambda_1 \in [0, 1]$, Student data fraction $\lambda_2 \in [0, 1]$, Divergence Function D, Learning rate η.

$u_1 \sim \text{Uniform}(0, 1), u_2 \sim \text{Uniform}(0, 1)$ ▷ Generate two random values.
if $u_1 \leq \lambda_1$ **then**
  $B = \{(x_b, y_b)\}_{b=1}^B$, where $(x_i, y_i) \sim (X, Y)$ ▷ Sample inputs and outputs from (X, Y).
else
  $B' = \{(x_b)\}_{b=1}^B$, where $x_i \sim X$ ▷ Sample a batch of inputs from X.
  if $u_2 \leq \lambda_2$ **then**
    $B = \{(x_b, y_b)\}_{b=1}^B$, where $x_i \sim B'$, $y_i \sim \mathcal{M}_q^\theta(\cdot|x_i)$ ▷ Sample data from $\mathcal{M}_q$.
  else
    $B = \{(x_b, y_b)\}_{b=1}^B$, where $x_i \sim B'$, $y_i \sim \mathcal{M}_p(\cdot|x_i)$ ▷ Sample data from $\mathcal{M}_p$.
  end if
end if
Return $\theta \leftarrow \theta - \eta \frac{1}{B} \sum_{(x,y) \in B} \nabla_\theta D(\mathcal{M}_p \| \mathcal{M}_q^\theta)(y|x)$ ▷ Update θ to minimize $D(\mathcal{M}_p \| \mathcal{M}_q^\theta)$.

B METHOD

B.1 DESCRIPTION OF DIVERGENCE FUNCTIONS

Below are some common divergence functions used in distillation, given two discrete probability distribution $P(\mathcal{C})$ and $Q(\mathcal{C})$.

Kullback-Leibler (KL) divergence.

$$D_{\mathrm{KL}}(P\|Q) = \sum_{c\in\mathcal{C}} P(c)\log\frac{P(c)}{Q(c)} \tag{B.1}$$

Note that the KL divergence is not symmetric, that is, $D_{\mathrm{KL}}(P\|Q) \neq D_{\mathrm{KL}}(Q\|P)$. As such, we refer to $D_{\mathrm{KL}}(P\|Q)$ as the **forward KL** (FKL) while $D_{\mathrm{RKL}}(P||Q) := D_{\mathrm{KL}}(Q\|P)$ as the **reverse KL** (RKL) between P and Q. Note that the FKL under an empirical data distribution corresponds to maximum likelihood estimation (**MLE**), which we typically optimize in supervised learning given a fixed dataset.

Jensen–Shannon (JS) divergence.

$$D_{\mathrm{JS}}(P\|Q) = \frac{1}{2}(D_{\mathrm{KL}}(P\|M) + D_{\mathrm{KL}}(Q\|M), \quad \text{where } M = \frac{1}{2}(P+Q) \tag{B.2}$$

Generalized Jensen-Shannon divergence ($D_{\mathrm{JSD}[\beta]}$).

$$D_{\mathrm{JSD}[\beta]}(P\|Q) = \beta D_{\mathrm{KL}}\Big(P\Big\|\beta P + (1-\beta)Q\Big) + (1-\beta)D_{\mathrm{KL}}\Big(Q\Big\|\beta P + (1-\beta)Q\Big) \tag{B.3}$$

Interestingly, it can be proved that $\lim_{\beta\to 0}\frac{D_{JSD(\beta)}(P\|Q)}{\beta} = D_{\mathrm{KL}}(P\|Q)$ (Huszár, 2015). As such, $D_{\mathrm{JSD}[\beta]}$ behaves similarly to forward KL for small values of β. Similarly, $D_{\mathrm{JSD}[\beta]}$ has similar behavior to reverse KL for β close to 1, since $D_{\mathrm{JSD}[\beta]}(P\|Q) = D_{\mathrm{JSD}[1-\beta]}(Q\|P)$.

Total variation distance (TVD).

$$D_{\mathrm{TVD}}(P\|Q) = \sum_{c\in\mathcal{C}} |\frac{P(c)-Q(c)}{2}| \tag{B.4}$$

B.2 JUSTIFICATION OF USING ON-POLICY DATA

In this section, we prove Theorem 4.1 to justify our use of on-policy data. We follow the notations in Sections 3 and 4. In addition, we write $\epsilon(x) := \mathbb{E}_{y\sim q_{\le T}(y\,|\,x)}\left[D_{\mathrm{TVD}}(\mathcal{M}_p\|\mathcal{M}_q^\theta)(y\,|\,x)\right]$ for the distillation loss of a single input x.

First, we decompose $\alpha(x)$ and $\epsilon(x)$ into sums of contributions from each token to ease the analysis:

Lemma B.1. *For all x, $\alpha(x) = 1 - \frac{1}{L_p(x)}\sum_{t=1}^{T} A_t$ and $\epsilon(x) \ge \frac{1}{T}\sum_{t=1}^{T} E_t$, where*

$$A_t(x) := \mathbb{E}_{y\sim p_{\le T}(y\,|\,x)}\left[\mathbb{1}_{\{t\le|y|\}} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right] \tag{B.5}$$

$$E_t(x) := \mathbb{E}_{y\sim q_{\le T}(y\,|\,x)}\left[\mathbb{1}_{\{t\le|y|\}} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right] \tag{B.6}$$

Proof. Recall that by Equation 3 we have

$$\alpha(x) = 1 - \frac{\mathbb{E}_{y\sim p_{\le T}(y\,|\,x)}\left[\sum_{t=1}^{|y|} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right]}{L_p(x)}.$$

We can rewrite $\sum_{t=1}^{|y|} D_{\mathrm{TVD}}(p(y_t), q(y_t)) = \sum_{t=1}^{T} \mathbb{1}_{\{t\le|y|\}} D_{\mathrm{TVD}}(p(y_t), q(y_t))$ and then swap the order between the summation and the expectation to obtain

$$\alpha(x) = 1 - \frac{\sum_{t=1}^{T} \mathbb{E}_{y\sim p_{\le T}(y\,|\,x)}\left[\mathbb{1}_{\{t\le|y|\}} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right]}{L_p(x)},$$

which proves Equation B.5.

Similarly, by definition of $\epsilon(x)$ and D_{TVD} we have

$$\begin{aligned}\epsilon(x) &= \mathbb{E}_{y\sim q_{\le T}(y\,|\,x)}\left[\frac{1}{|y|}\sum_{t=1}^{|y|} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right] \\ &\ge \mathbb{E}_{y\sim q_{\le T}(y\,|\,x)}\left[\frac{1}{T}\sum_{t=1}^{|y|} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right].\end{aligned}$$

We can rewrite $\sum_{t=1}^{|y|} D_{\mathrm{TVD}}(p(y_t), q(y_t)) = \sum_{t=1}^{T} \mathbb{1}_{\{t\le|y|\}} D_{\mathrm{TVD}}(p(y_t), q(y_t))$ and then swap the order between the summation and the expectation to obtain

$$\epsilon(x) \ge \frac{1}{T}\sum_{t=1}^{T} \mathbb{E}_{y\sim q_{\le T}(y\,|\,x)}\left[\mathbb{1}_{\{t\le|y|\}} D_{\mathrm{TVD}}(p(y_t\,|\,x,y_{<t}), q(y_t\,|\,x,y_{<t}))\right],$$

which proves Equation B.6. □

Lemma B.1 motivates us to study $A_t(x)$ and $E_t(x)$ instead. Below we rewrite them as variational forms that will be used later. For this, we introduce some defintiions.

Definition B.1. *For any sequence $z \in \{\mathtt{P},\mathtt{Q}\}^\tau$ that consists only of letters $\mathtt{P}$ and $\mathtt{Q}$, we define $\mathcal{M}(x,y,z)$ as the distribution of sequences sampled as follows:*

1. *Initialize a sequence of tokens as y;*
2. *If there are $t-1$ tokens, sample the t-th token from $\mathcal{M}_p$ if $z_t = \mathtt{P}$, and from $\mathcal{M}_q$ otherwise;*
3. *Repeat until an end-of-sequence token is sampled, or the sequence length has reached τ.*

We use the shorthand $\mathtt{P}^k$ and $\mathtt{Q}^k$ to denote the sequence of k consecutive letters of $\mathtt{P}$ and $\mathtt{Q}$ respectively. We use Ω^k to denote the set of all possible strings of length k, and $\delta : \Omega^t \to [-1/2, 1/2]$ to denote a function that maps a sequence of t tokens to a real number in $[-1,1]$. We abuse the notation and assign $\delta(y) = 0$ for all $y \notin \Omega^t$.

Lemma B.2. *For all x and all $1 \le t \le T$,*

$$A_t(x) = \sup_{\delta:\Omega^t \to [-1/2,1/2]} \left\{ \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^t)} [\delta(y)] - \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1}\mathtt{Q})} [\delta(y)] \right\}, \tag{B.7}$$

$$E_t(x) = \sup_{\delta:\Omega^t \to [-1/2,1/2]} \left\{ \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{t-1}\mathtt{P})} [\delta(y)] - \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^t)} [\delta(y)] \right\}. \tag{B.8}$$

Proof. For a fixed pair of x and y, we rewrite the total variance distance between $p(y_t \mid x, y_{<t})$ and $q(y_t \mid x, y_{<t})$ as the following variational form:

$$\begin{aligned} &D_{\mathrm{TVD}}(p(y_t \mid x, y_{<t}), q(y_t \mid x, y_{<t})) \\ &\quad = \sup_{\tilde{\delta}:\Omega \to [-1/2,1/2]} \left\{ \underbrace{\mathbb{E}_{y_t \sim p(y_t \mid x, y_{<t})} \left[\tilde{\delta}(y_t)\right] - \mathbb{E}_{y_t \sim q(y_t \mid x, y_{<t})} \left[\tilde{\delta}(y_t)\right]}_{=:\Delta(x, y_{<t}, \tilde{\delta})} \right\}. \end{aligned}$$

Then after taking the expectations we have

$$\begin{aligned} A_t(x) &= \mathbb{E}_{y \sim p_{\le T}(y \mid x)} [D_{\mathrm{TVD}}(p(y_t \mid x, y_{<t}), q(y_t \mid x, y_{<t}))] \\ &= \mathbb{E}_{y \sim p_{\le T}(y \mid x)} \left[\sup_{\tilde{\delta}:\Omega \to [-1/2,1/2]} \left\{ \Delta(x, y_{<t}, \tilde{\delta}) \right\} \right] \\ &= \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1})} \left[\sup_{\tilde{\delta}:\Omega \to [-1/2,1/2]} \left\{ \mathbb{1}_{\{|y|=t-1\}} \Delta(x, y, \tilde{\delta}) \right\} \right] \\ &= \sup_{\delta:\Omega^t \to [-1/2,1/2]} \left\{ \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1})} \left[\mathbb{1}_{\{|y|=t-1\}} \Delta(x, y, \delta(y, \cdot)) \right] \right\}, \end{aligned}$$

where the third equality uses the observation that for any function f, $\mathbb{E}_{y \sim p_{\le T}(y \mid x)} [f(x, y_{<t})]$ can be replaced with $\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1})} \left[\mathbb{1}_{\{|y|=t-1\}} f(x, y)\right]$, and the last equality swaps the order between the expectation and the supremum. Finally, we move the expectations in Δ to merge with the expectation outside and obtain:

$$\begin{aligned} A_t(x) &= \sup_{\delta:\Omega^t \to [-1/2,1/2]} \left\{ \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^t)} \left[\mathbb{1}_{\{|y|=t\}} \delta(y)\right] - \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1}\mathtt{Q})} \left[\mathbb{1}_{\{|y|=t\}} \delta(y)\right] \right\} \\ &= \sup_{\delta:\Omega^t \to [-1/2,1/2]} \left\{ \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^t)} [\delta(y)] - \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1}\mathtt{Q})} [\delta(y)] \right\}, \end{aligned}$$

where the last step is due to our abuse of notation that $\delta(y) = 0$ for all $y \notin \Omega^t$. This proves Equation B.7, and Equation B.8 can be proved similarly. □

With our variational forms of $A_t(x)$ and $E_t(x)$, we obtain the following lemma for bounding $A_t(x)$ in terms of $E_t(x)$.

Lemma B.3. *For all x and $1 \le t \le T$,*

$$A_t(x) \le 2 \sum_{k=1}^{t-1} E_k(x) + E_t(x). \tag{B.9}$$

Proof. It suffices to give an upper bound on $\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^t)} [\delta(y)]$ and a lower bound on $\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1}\mathtt{Q})} [\delta(y)]$ for all $\delta : \Omega^t \to [-1/2, 1/2]$.

For all $1 \le k \le t$, we can replace the first $\mathtt{P}$ in $\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P}^{t-k+1})} [\delta(y)]$ with $\mathtt{Q}$ by only introducing an error of $E_{k+1}(x)$:

$$\begin{aligned} \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P}^{t-k+1})} [\delta(y)] &= \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P})} \left[\mathbb{1}_{\{|y|=k\}} \mathbb{E}_{y' \sim \mathcal{M}(x,y,\mathtt{P}^{t-k})} [\delta(y')]\right] \\ &\le \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^k)} \left[\mathbb{1}_{\{|y|=k\}} \mathbb{E}_{y' \sim \mathcal{M}(x,y,\mathtt{P}^{t-k})} [\delta(y')]\right] + E_k(x) \\ &= \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^k\mathtt{P}^{t-k})} [\delta(y)] + E_k(x), \end{aligned}$$

where the second inequality holds because we can define a function $\delta' : \Omega^k \to [-1/2, 1/2], \delta(y) = \mathbb{1}_{\{|y|=k\}} \mathbb{E}_{y' \sim \mathcal{M}(x,y,\mathtt{P}^{t-k})} [\delta(y')]$ and then apply Equation B.8.

Now taking a telescoping sum over $1 \le k \le t$, we obtain

$$\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^t)} [\delta(y)] \le \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^t)} [\delta(y)] + \sum_{k=1}^{t} E_k(x). \tag{B.10}$$

Similarly, for all $1 \le k \le t-1$, we can replace the first $\mathtt{P}$ in $\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P}^{t-k}\mathtt{Q})} [\delta(y)]$ with $\mathtt{Q}$ by only introducing an error of $E_k(x)$:

$$\begin{aligned}
\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P}^{t-k}\mathtt{Q})} [\delta(y)] &= \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P})} \left[\mathbb{1}_{\{|y|=k\}} \mathbb{E}_{y' \sim \mathcal{M}(x,y,\mathtt{P}^{t-k-1}\mathtt{Q})} [\delta(y')]\right] \\
&= -\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k-1}\mathtt{P})} \left[-\mathbb{1}_{\{|y|=k\}} \mathbb{E}_{y' \sim \mathcal{M}(x,y,\mathtt{P}^{t-k-1}\mathtt{Q})} [\delta(y')]\right] \\
&\ge -\left(\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k})} \left[-\mathbb{1}_{\{|y|=k\}} \mathbb{E}_{y' \sim \mathcal{M}(x,y,\mathtt{P}^{t-k-1}\mathtt{Q})} [\delta(y')]\right] + E_k(x)\right) \\
&= \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^{k}\mathtt{P}^{t-k-1}\mathtt{Q})} [\delta(y)] - E_k(x).
\end{aligned}$$

Taking a telescoping sum over $1 \le k \le t-1$ gives

$$\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1}\mathtt{Q})} [\delta(y)] \ge \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{Q}^t)} [\delta(y)] - \sum_{k=1}^{t-1} E_k(x). \tag{B.11}$$

Subtracting Equation (B.11) from Equation (B.10), we have the following holds for all functions $\delta : \mathcal{S} \to [-1/2, 1/2]$:

$$\mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^t)} [\delta(y)] - \mathbb{E}_{y \sim \mathcal{M}(x,\varnothing,\mathtt{P}^{t-1}\mathtt{Q})} [\delta(y)] \le \sum_{k=1}^{t} E_k(x) + \sum_{k=1}^{t-1} E_k(x) = 2\sum_{k=1}^{t-1} E_k(x) + E_t(x),$$

which proves the claim. □

Proof for Theorem 4.1. Taking a sum of Equation B.9 over $1 \le t \le T$, we have

$$\sum_{t=1}^{T} A_t(x) \le \sum_{t=1}^{T} (1 + 2(T-t)) E_t(x).$$

Combining this with Lemma B.1 gives

$$\begin{aligned}
\alpha(x) = 1 - \frac{1}{L_p(x)} \sum_{t=1}^{T} A_t &\ge 1 - \frac{1}{L_p(x)} \sum_{t=1}^{T} (1 + 2(T-t)) E_t(x) \\
&\ge 1 - \frac{2T}{L_p(x)} \sum_{t=1}^{T} E_t(x) \\
&\ge 1 - \frac{2T^2}{L_p(x)} \epsilon(x),
\end{aligned}$$

which proves the theorem statement after taking the expectation over $x \sim X$. □

C Implementation Details

C.1 Datasets

In this section, we present a comprehensive overview of the datasets employed in this study.

XSum (Narayan et al., 2018). The Extreme Summarization (XSum) dataset serves as an evaluation benchmark for abstractive single-document summarization systems. This dataset comprises 226,711 news articles, sourced from BBC articles spanning the years 2010 to 2017. These articles encompass a wide range of domains, including News, Politics, Sports, Weather, Business, Technology, Science, Health, Family, Education, Entertainment, and Arts. Summarization performance is evaluated using ROUGE scores on the validation dataset split of XSum, primarily emphasizing ROUGE-2, while observing similar trends in ROUGE-LSum and ROUGE-1. A maximum input length of 1024 and a maximum output length of 64 are employed for distillation and evaluation.

CNN/DM (Hermann et al., 2015). The CNN/Daily Mail (CNN/DM) dataset is tailored for text summarization. It comprises abstractive summary bullets generated by humans from news stories on CNN and Daily Mail websites, presented in the form of questions with entities hidden. These questions are answered using corresponding passages from the source text. Similar to XSum, ROUGE scores on the validation dataset are reported, primarily emphasizing ROUGE-2, with observations in ROUGE-LSum and ROUGE-1. A maximum input length of 2048 and a maximum output length of 128 are used for distillation and evaluation.

WMT EnDe (Bojar et al., 2014). The WMT14 EnDe dataset stands as a standard benchmark for machine translation. It entails the task of translating English text into German while preserving content, semantics, and style. Evaluation relies on the BLEU score, measuring the similarity of machine-translated text to high-quality reference translations. A maximum input length of 80 and a maximum output length of 80 are employed for distillation and evaluation, with performance assessed on the original test split.

GSM8K (Cobbe et al., 2021). The GSM8K dataset comprises 8.5K high-quality, linguistically diverse grade school math word problems crafted by human problem writers. The dataset is divided into 7.5K training problems and 1K test problems, with solutions typically requiring 2 to 8 steps involving elementary calculations using basic arithmetic operations. To enhance reasoning abilities, we explored distillation alongside the zero-shot chain-of-thought (CoT) method, as described in Agarwal et al. (2023). A maximum input length of 256 and a maximum output length of 320 are used for distillation and evaluation.

LM1B Chelba et al. (2013). The One Billion Word dataset (LM1B) is a widely recognized benchmark for language modeling. The training and held-out data are derived from the WMT 2011 News Crawl dataset, created using Bash shell and Perl scripts. A maximum input length of 128 and a maximum output length of 128 are used for distillation and evaluation.

C.2 Models

In accordance with Leviathan et al. (2023), we evaluate two model types: 1) GPT-like decoder-only Transformer models trained using the standard autoregressive objective on LM1B task (Chelba et al., 2013), where the target and draft models have 234M and 33M parameters, respectively; and 2) Standard encoder-decoder T5 v1.1 models (Raffel et al., 2020) supervised fine-tuned on four different tasks, with T5-XL (3B) and T5-Small (77M) serving as the target and draft models, respectively.

The $\mathcal{M}_p$ for decoder-only model experiment has: dimension 1024, feed-forward dimension 4096, 12 layers, and 16 attention heads. The corresponding $\mathcal{M}_q^\theta$ has 33M parameters: dimension 512, feed-forward dimension 1024, 4 layers, and 4 attention heads. All models utilize the T5 tokenizer with 32k tokens. As for the T5 base checkpoints, we start from LM-adapted T5v1.1 models. These LM-adapted models are initialized from T5v1.1 and trained for an additional 100K steps on the LM objective discussed in the T5 paper (Raffel et al., 2020). These checkpoints are open-sourced at `https://console.cloud.google.com/storage/browser/t5-data/pretrained_models`.

In our experiments, both the student and teacher models for the distillation process are initialized from the supervised fine-tuning on the original training dataset. This process is detailed as follows:

- **XSum**. For small, base, large, XL and XXL models, we use LM-Adapted T5v1.1 models supervised fine-tuned for 100K, 50K, 30k, 20K and 8K steps respectively.
- **CNN/DM**. For small, base, large, XL and XXL models, we use LM-Adapted T5v1.1 models supervised fine-tuned for 200K, 80K, 20k, 20k and 4K steps respectively.
- **WMT**. For small, base, large, XL and XXL models, we use LM-Adapted T5v1.1 models supervised fine-tuned for 250K, 250K, 110k , 50K and 10K steps respectively.
- **GSM8K**. All models were supervised fine-tuned starting from FLAN-T5 models on the Palm-540 generated CoT dataset for 10K steps.

C.3 DISTILLATION

We employ the Adafactor optimizer (Shazeer & Stern, 2018) to train our draft student model, denoted as $\mathcal{M}_q^\theta$, across all our experiments, following the guidelines outlined in Algorithm A.2. In the context of our knowledge distillation (KD) loss function, as defined in Eq. 2, we maintain the temperatures for both the target model, denoted as T_p, and the draft model, denoted as T_q, at a constant value of 1.0. It is imperative to emphasize the significance of preserving this uniform temperature setting, as it plays a pivotal role in speculative decoding, ensuring a consistent and coherent semantic interpretation for both $\mathcal{M}_p$ and $\mathcal{M}_q^\theta$. A summary of the hyperparameters used in our knowledge distillation process can be found in Table C.1.

Table C.1: Hyperparameter Summary for Distillation Experiments.

Hyperparameter	**Value**
Training Steps	300,000
Batch size	32
Dropout	0.0
Learning Rate (LR)	0.0003
LR Warmup Steps	5,000
LR Cooldown (Begin, End)	(150,000, 300,000)
Warmup Schedule	Linear (from 0 to LR)
Cooldown Schedule	Cosine Decay (from LR to 0.1LR)

C.4 EVALUATION

To obtain scores for each task (specifically, Rouge2 for XSum and CNN/DM, BLEU for WMT, and Accuracy for GSM8K), we employ the evaluation methodology as outlined by Agarwal et al. (2023) to assess all examples within the test or validation sets and subsequently report the average performance. For the assessment of empirical speculative decoding metrics, encompassing empirical acceptance rate and empirical block efficiency, we conduct evaluations on all instances within the test or validation sets and then report the average performance. For measuring the actual latency, we adhere to the procedure detailed in Leviathan et al. (2023), where both our target model and draft model are executed on the same device, specifically a TPUv4, without utilizing model parallelism. To gauge the real latency, we randomly sample 500 examples from either the test or validation set, and measure the decoding time with a batch size of 1. This measurement procedure is repeated three times, and the mean performance is reported. It is worth noting that we have observed minimal variance across different random seeds in our results.

D Additional Results

D.1 Enhancing speculative decoding through knowledge distillation

Table D.1: DistillSpec improves the sepeculative decoding across various datasets, block sizes for both greedy decoding and temperature sampling.

Datasets	Models	Sampling		w/o Distillation		with Distillation			
		TEMP	γ	τ	SPEED	τ	SPEED	Δ	Method
XSum	Target M_p T5-XL (3B)	T=0	3	2.31	1.44×	2.62	1.58×	+0.14×	f-Distill
			5	2.57	1.43×	3.08	1.62×	+0.19×	f-Distill
			7	2.68	1.36×	3.31	1.57×	+0.21×	f-Distill
	Draft M_q T5-SMALL (77M)	T=1	3	2.19	1.40×	2.58	1.57×	+0.17×	f-Distill
			5	2.39	1.37×	3.01	1.61×	+0.25×	f-Distill
			7	2.47	1.28×	3.21	1.55×	+0.27×	f-Distill
CNNDM	Target M_p T5-XL (3B)	T=0	3	2.83	1.89×	3.19	2.11×	+0.22×	f-Distill
			5	3.46	2.07×	4.13	2.42×	+0.35×	GKD
			7	3.85	2.07×	4.83	2.53×	+0.46×	f-Distill
	Draft M_q T5-SMALL (77M)	T=1	3	2.49	1.71×	2.87	1.93×	+0.23×	f-Distill
			5	2.89	1.77×	3.52	2.12×	+0.35×	f-Distill
			7	3.08	1.71×	3.92	2.12×	+0.41×	f-Distill
WMT	Target M_p T5-XL (3B)	T=0	3	3.22	2.08×	3.30	2.13×	+0.05×	f-Distill
			5	4.17	2.33×	4.32	2.41×	+0.08×	f-Distill
			7	4.83	2.36×	5.06	2.46×	+0.10×	f-Distill
	Draft M_q T5-SMALL (77M)	T=1	3	2.73	1.77×	2.88	1.83×	+0.07×	GKD
			5	3.29	1.83×	3.48	1.93×	+0.10×	GKD
			7	3.54	1.72×	3.85	1.84×	+0.12×	f-Distill
GSM8K	Target M_p T5-XL (3B)	T=0	3	2.60	1.51×	2.96	1.69×	+0.18×	GKD
			5	3.06	1.42×	3.68	1.65×	+0.22×	GKD
			7	3.27	1.36×	4.14	1.60×	+0.24×	GKD
	Draft M_q T5-SMALL (77M)	T=1	3	2.58	1.48×	2.84	1.64×	+0.16×	f-Distill
			5	3.03	1.39×	3.45	1.58×	+0.19×	f-Distill
			7	3.23	1.33×	3.84	1.53×	+0.20×	f-Distill
LM1B	Target M_p GPT-LIKE (234M)	T=0	3	2.96	3.66×	3.13	3.97×	+0.31×	f-Distill
			5	3.69	3.35×	3.92	3.51×	+0.16×	f-Distill
			7	4.15	2.52×	4.55	2.72×	+0.20×	f-Distill
	Draft M_q GPT-LIKE (33M)	T=1	3	2.51	2.34×	2.69	2.45×	+0.11×	f-Distill
			5	2.90	2.79×	3.20	3.02×	+0.23×	f-Distill
			7	3.10	1.98×	3.51	2.18×	+0.20×	f-Distill

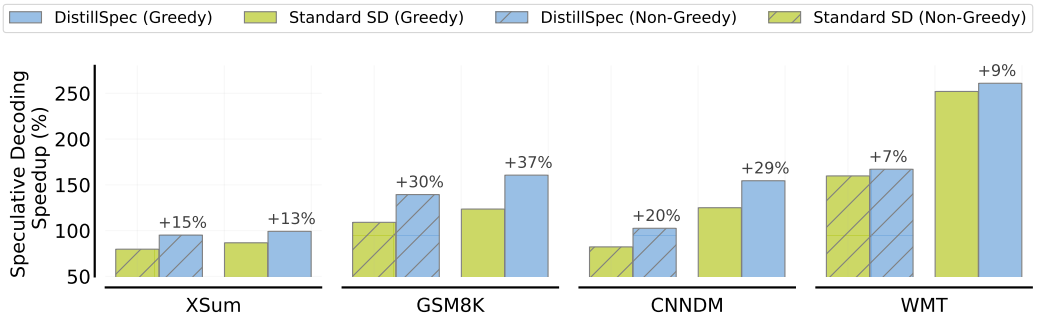


Figure D.1: The distilled draft model, specifically T5-small, derived from the teacher model, T5-XL, is capable of generalizing to a larger target model, T5-XXL, resulting in a substantial acceleration in various scenarios.

D.1.1 Empirical Block Efficiency Improvement

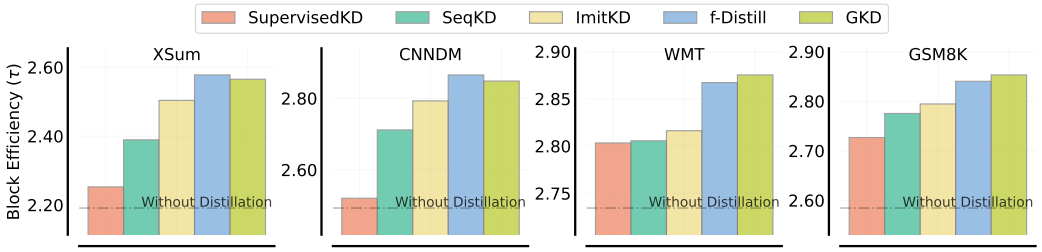


Figure D.2: Empirical block efficiency improvement of DistillSpec for non-greedy sampling ($T = 1$) and block size $\gamma = 3$. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. The dashed line indicates the block efficiency of speculative decoding using a non-distilled draft model. DistillSpec outperforms standard speculative decoding across all of the distillation methods being considered, with f-Distill and GKD yielding the highest gains.

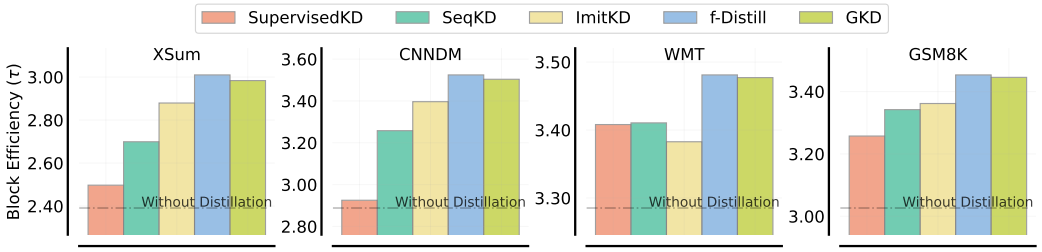


Figure D.3: Empirical block efficiency improvement of DistillSpec for non-greedy sampling ($T = 1$) and block size $\gamma = 5$. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. The dashed line indicates the block efficiency of speculative decoding using a non-distilled draft model. DistillSpec outperforms standard speculative decoding across all of the distillation methods being considered, with f-Distill and GKD yielding the highest gains.

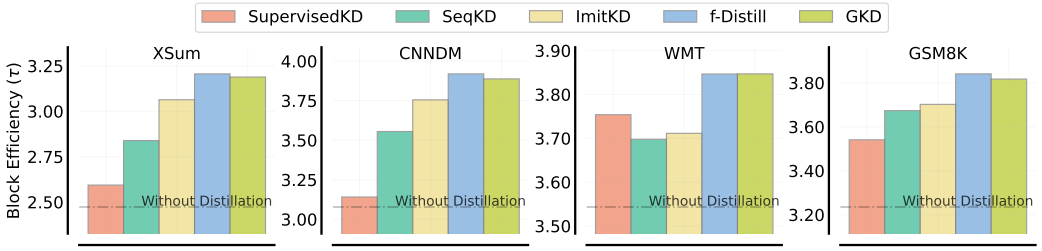


Figure D.4: Empirical block efficiency improvement of DistillSpec for non-greedy sampling ($T = 1$) and block size $\gamma = 7$. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. The dashed line indicates the block efficiency of speculative decoding using a non-distilled draft model. DistillSpec outperforms standard speculative decoding across all of the distillation methods being considered, with f-Distill and GKD yielding the highest gains.

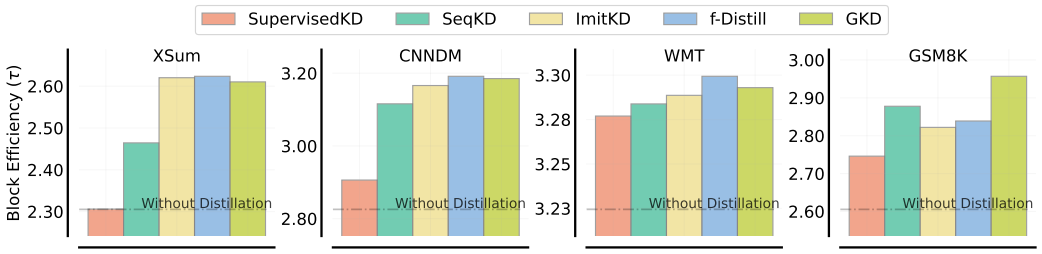


Figure D.5: Empirical block efficiency improvement of DistillSpec for greedy sampling ($T = 0$) and block size $\gamma = 3$. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. The dashed line indicates the block efficiency of speculative decoding using a non-distilled draft model. DistillSpec outperforms standard speculative decoding across all of the distillation methods being considered, with GKD weakly outperforming the other methods on average.

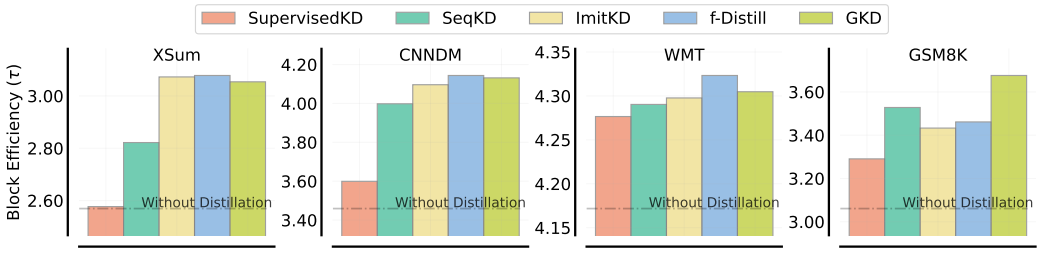


Figure D.6: Empirical block efficiency improvement of DistillSpec for greedy sampling ($T = 0$) and block size $\gamma = 5$. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. The dashed line indicates the block efficiency of speculative decoding using a non-distilled draft model, with GKD weakly outperforming the other methods on average.

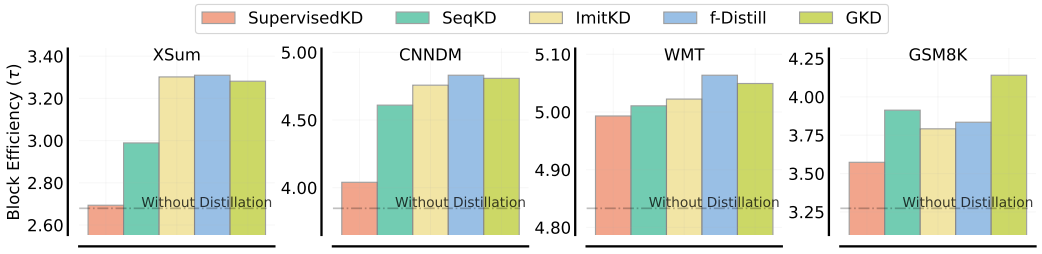
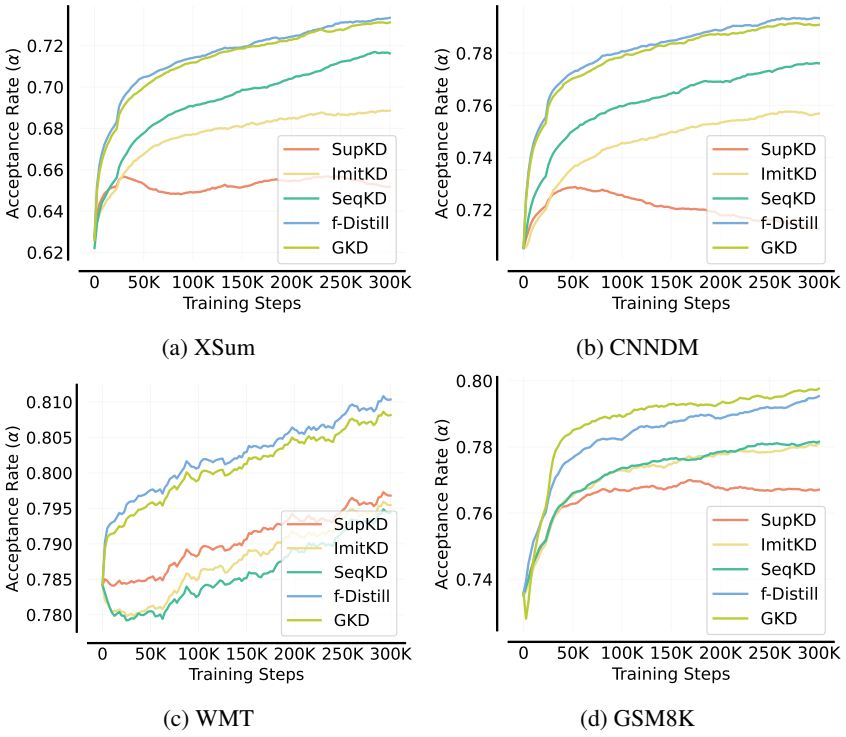


Figure D.7: Empirical block efficiency improvement of DistillSpec for greedy sampling ($T = 0$) and block size $\gamma = 7$. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. The dashed line indicates the block efficiency of speculative decoding using a non-distilled draft model, with GKD weakly outperforming the other methods on average.

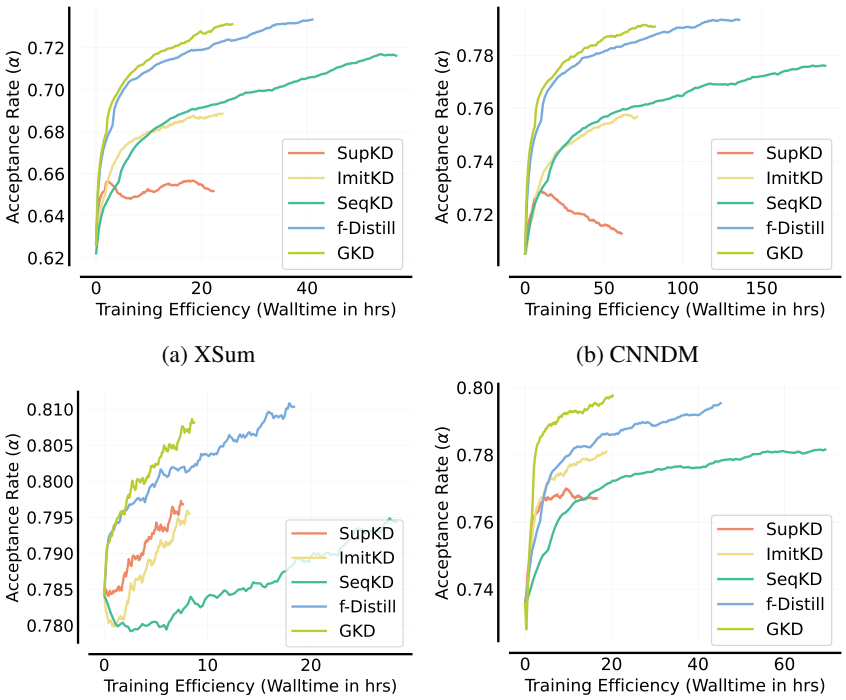
D.1.2 PERFORMANCE IMPROVEMENT OVER TIME



(a) XSum (b) CNNDM

(c) WMT (d) GSM8K

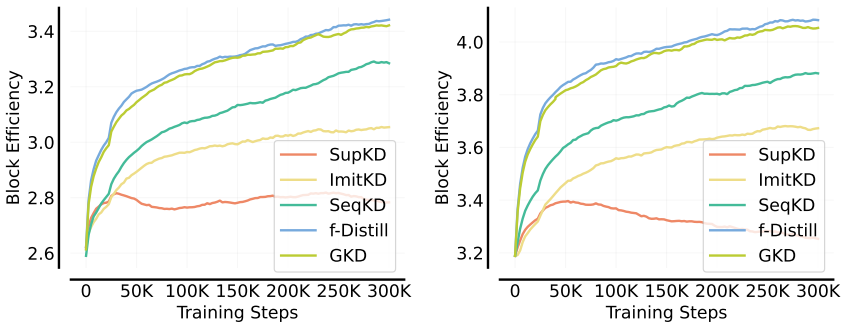
Figure D.8: Progression of the acceptance rate α of DistillSpec over the training of the draft model, measured by the number of training steps. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. GKD and f-Distill yields the most consistent improvement in α over training steps, while SupKD yields the least improvement and exhibits declining acceptance rates after 40k training steps on XSum and CNNDM.



(a) XSum (b) CNNDM

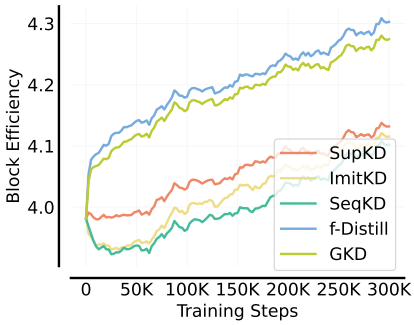
(c) WMT (d) GSM8K

Figure D.9: Progression of the acceptance rate of DistillSpec over the training of the draft model, measured by the training wall time. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. GKD and f-Distill yield the most consistent improvement in α over wall-time training, while SupKD yields the least improvement and even exhibits an inflection in the acceptance rate early during training.

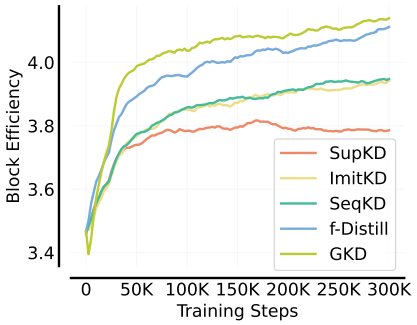


(a) XSum

(b) CNNDM

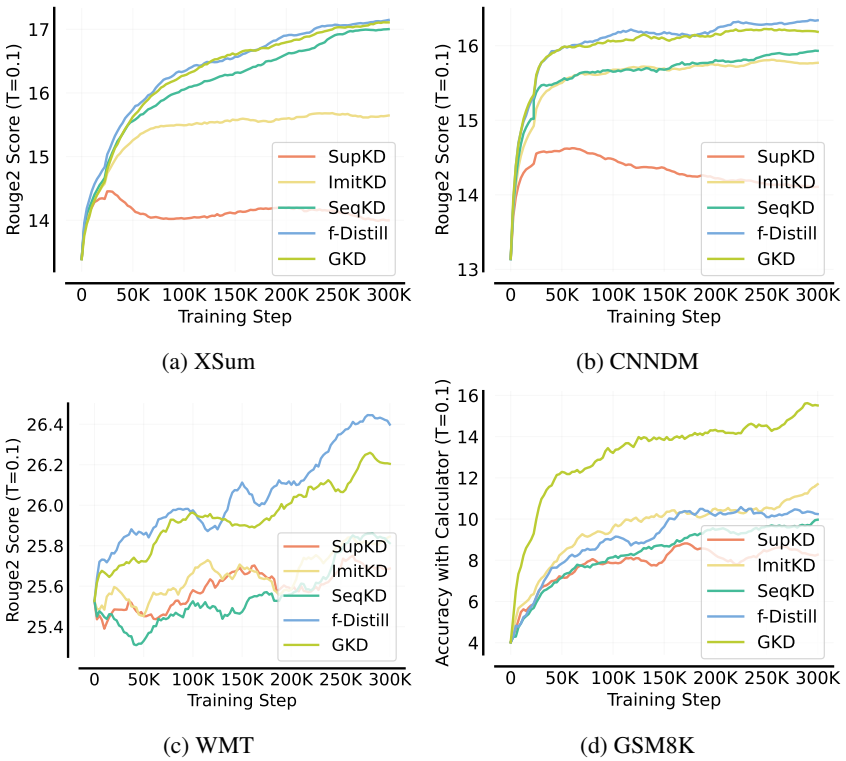


(c) WMT



(d) GSM8K

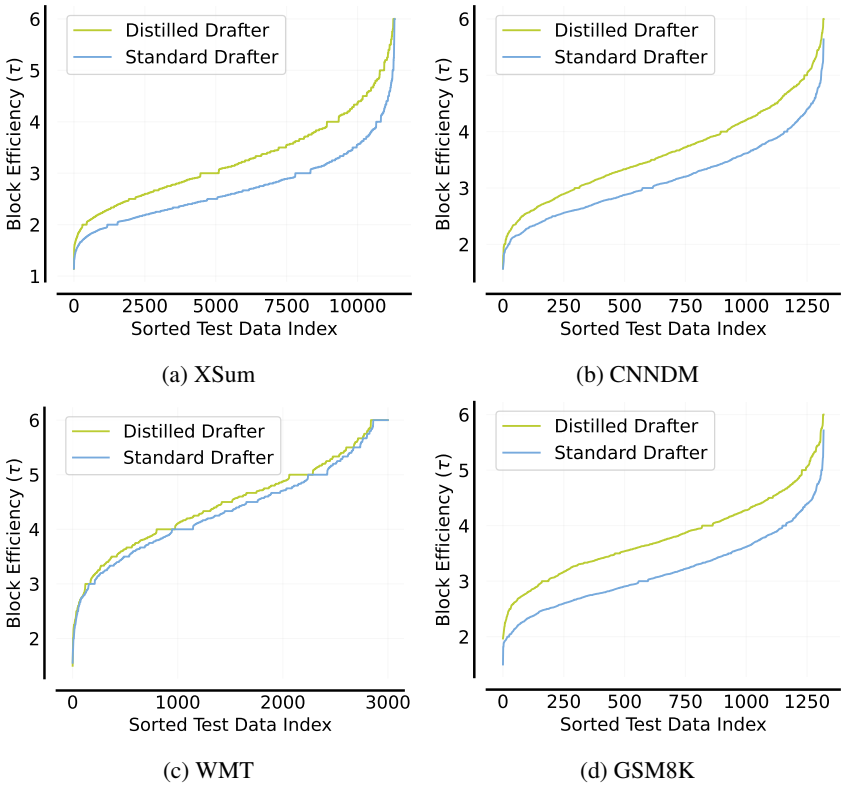
Figure D.10: Progression of the block efficiency τ of DistillSpec over the training of the draft model, measured by the number of training steps. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. GKD and f-Distill yield the most consistent improvement in τ over training, while SupKD yields the least improvement and exhibits a drop in block efficiency after 40k training steps on XSum and CNNDM.



(a) XSum (b) CNNDM

(c) WMT (d) GSM8K

Figure D.11: Progression of the task performance of draft model over the course of its training, as measured by the number of training steps. Task performance is based on sampling temperature $T = 0.1$ and is measured by the Rouge2 score on the XSum and CNNDM datasets, and the accuracy score with the help of a calculator on GSM8K. The draft model is trained using one of the distillation methods listed in Table 1 of Section 4. GKD and f-Distill yield the most consistent improvement in task performance over the course of training, while SupKD yields the least improvement and exhibits declining Rouge2 scores after 40k training steps on XSum and CNNDM.



(a) XSum

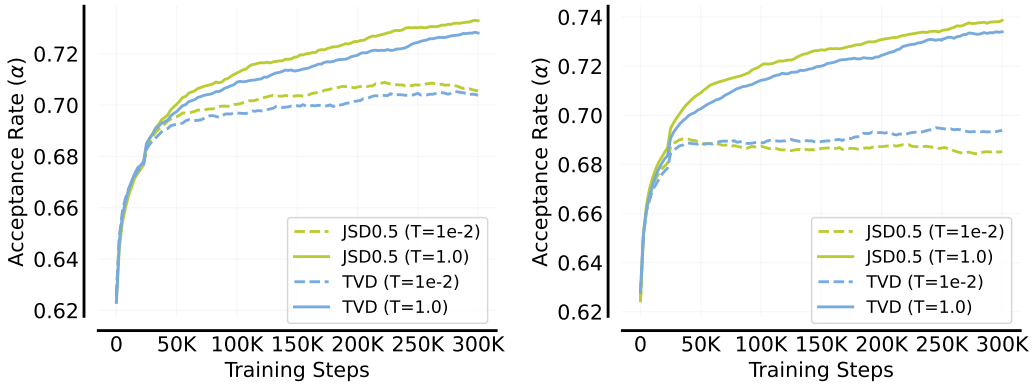
(b) CNNDM

(c) WMT

(d) GSM8K

Figure D.12: Comparison of the block efficiency of drafter models trained with distillation vs. without distillation. The draft model is trained using f-Distill on XSum and CNNDM, and on-policy GKD on GSM8K. For each dataset and drafter, the block efficiency is sorted and plotted from lowest to highest index. The higher position of the curve for the distilled drafter at each index indicates that the distribution of block efficiency of the distilled drafter *first-order stochastically dominates* the block efficiency of the non-distilled drafter.

D.1.3 SAMPLING TEMPERATURE EFFECT



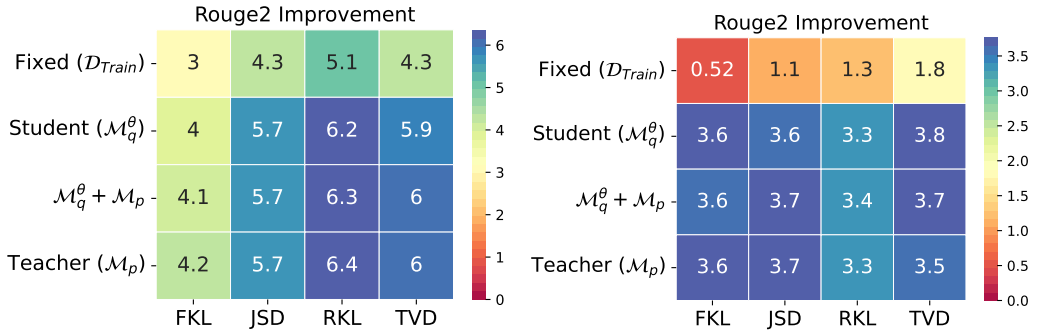
(a) Distilling on student generations (on-policy).

(b) Distilling on target model generations.

Figure D.13: Effect of the sampling temperature on XSum decoding. We assess how varying the sampling temperature influences the performance of our model in two distinct scenarios: (a) the drafter model is trained on its own generated sequences (on-policy distillation), and (b) the drafter model is trained on sequences sampled from the target model. In both instances, we find that using a high sampling temperature is paramount for attaining superior performance.

D.2 DISTILLATION RECIPE

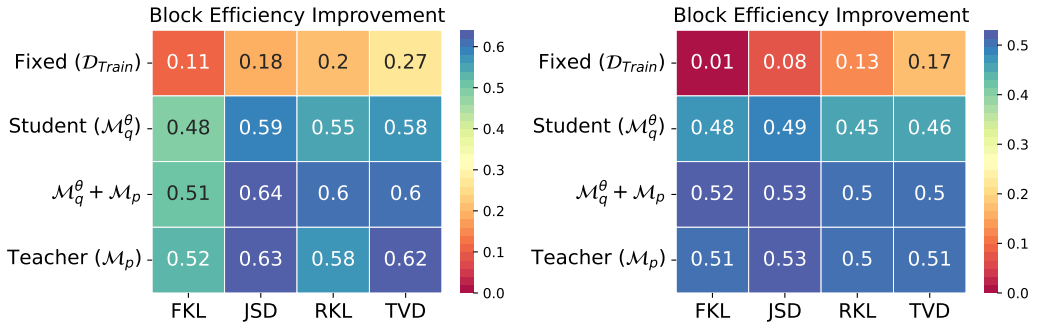
D.2.1 SCORE AND BLOCK EFFICIENCY IMPROVEMENT



(a) Non-greedy decoding ($T = 1$)

(b) Greedy decoding ($T = 0$)

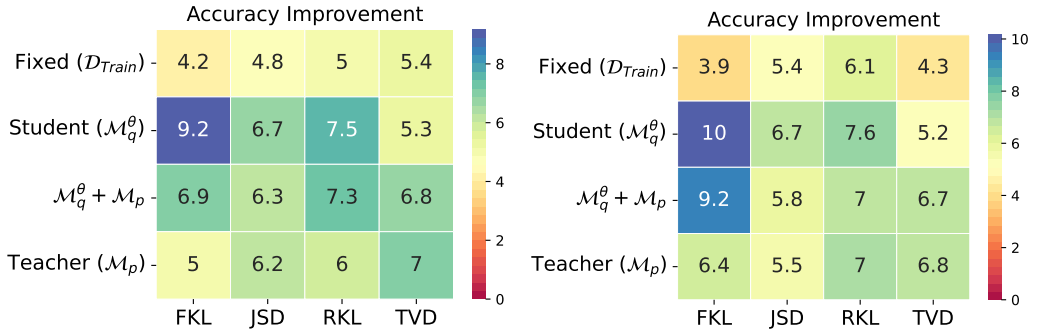
Figure D.14: Improvement in the Rouge2 score of DistillSpec over standard speculative decoding on XSum given by different combinations of divergence functions and generative distributions used in the draft model distillation, using greedy ($T = 0$) and non-greedy ($T = 1$) sampling.



(a) Non-greedy decoding ($T = 1$)

(b) Greedy decoding ($T = 0$)

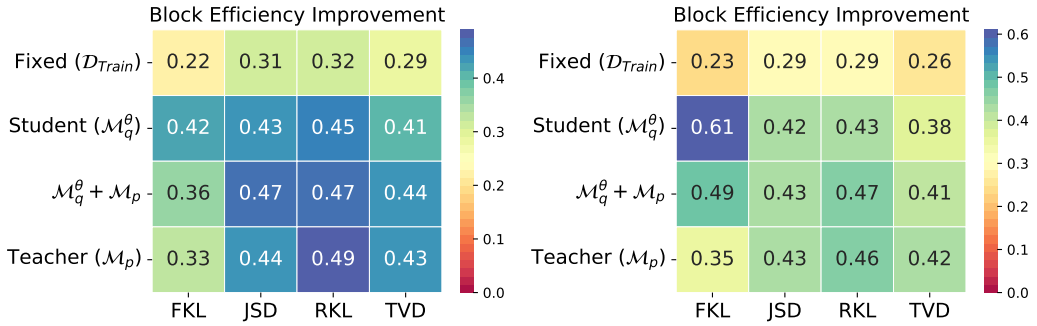
Figure D.15: Improvement in the block efficiency of DistillSpec over standard speculative decoding on XSum given by different combinations of divergence functions and generative distributions used in the draft model distillation, using greedy ($T = 0$) and non-greedy ($T = 1$) sampling.



(a) Non-greedy decoding ($T = 1$)

(b) Greedy decoding ($T = 0$)

Figure D.16: Improvement in the Rouge2 score of DistillSpec over standard speculative decoding on GSM8K given by different combinations of divergence functions and generative distributions used in the draft model distillation, using greedy ($T = 0$) and non-greedy ($T = 1$) sampling.

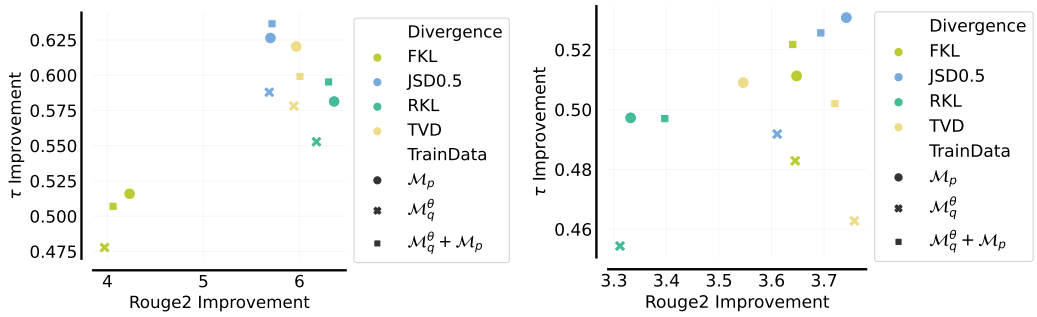


(a) Non-greedy decoding ($T = 1$)

(b) Greedy decoding ($T = 0$)

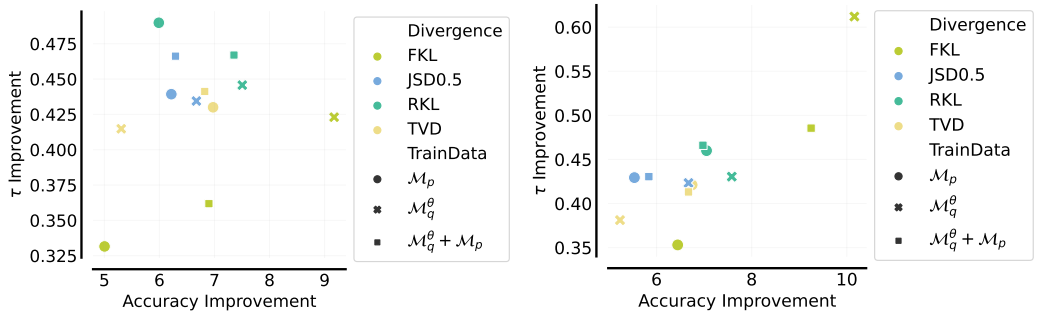
Figure D.17: Improvement in the block efficiency of DistillSpec over standard speculative decoding on GSM8K given by different combinations of divergence functions and generative distributions used in the draft model distillation, using greedy ($T = 0$) and non-greedy ($T = 1$) sampling.

D.2.2 IMPACT OF DISTILLATION ON DRAFT QUALITY VS. COMPATIBILITY



(a) Non-greedy decoding ($T = 1$) (b) Greedy decoding ($T = 0$)

Figure D.18: Correlation plot of the improvements in the block efficiency (y-axis) and Rouge2 score (x-axis) of DistillSpec over standard speculative decoding on XSum given by different combinations of divergence functions and generative distributions used in the draft model distillation (see Section 4), using greedy ($T = 0$) and non-greedy ($T = 1$) sampling.

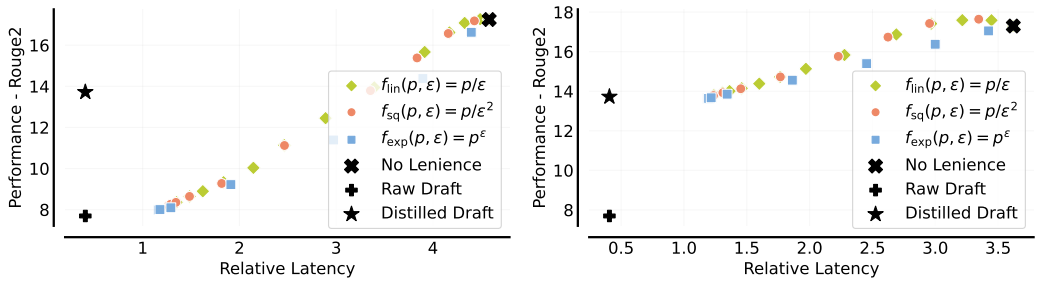


(a) Non-greedy decoding ($T = 1$) (b) Greedy decoding ($T = 0$)

Figure D.19: Correlation plot of the improvements in the block efficiency (y-axis) and Accuracy (x-axis) of DistillSpec over standard speculative decoding on GSM8K given by different combinations of divergence functions and generative distributions used in the draft model distillation (see Section 4), using greedy ($T = 0$) and non-greedy ($T = 1$) sampling.

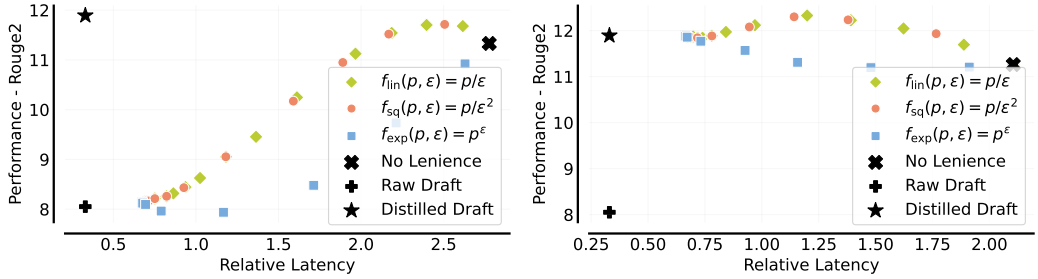
D.3 Quality versus latency trade-off

D.3.1 Lossy speculative decoding



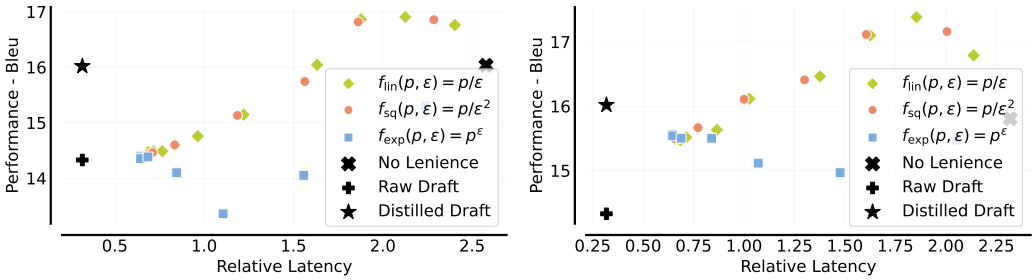
(a) Using non-distilled drafter model (standard SD) (b) Using distilled drafter modell (DistillSpec)

Figure D.20: Tradeoff between task performance and decoding latency on XSum enabled by combining lossy speculative decoding with the alternative variants of DistillSpec presented in Section 4. The higher position and flatter slope of the tradeoff curve for DistillSpec indicate that the method enables larger latency gains for smaller reduction in model quality than are feasible under standard SD.



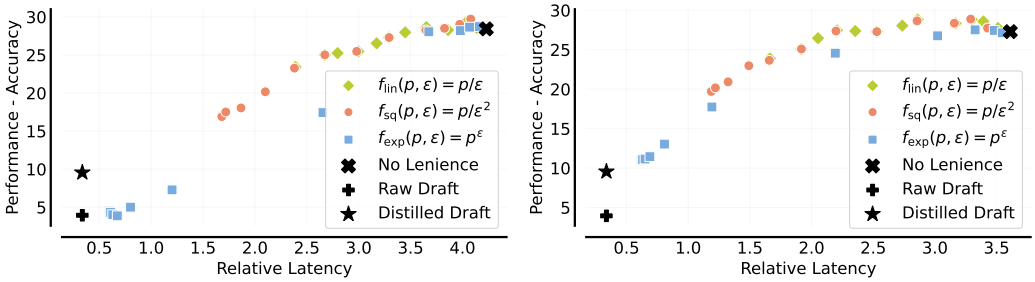
(a) Using non-distilled drafter model (standard SD) (b) Using distilled drafter modell (DistillSpec)

Figure D.21: Tradeoff between task performance and decoding latency on CNNDM enabled by combining lossy speculative decoding with the alternative variants of DistillSpec presented in Section 4. The higher position and flatter slope of the tradeoff curve for DistillSpec indicate that the method enables larger latency gains for smaller reduction in model quality than are feasible under standard SD.



(a) Using non-distilled drafter model (standard SD) (b) Using distilled drafter modell (DistillSpec)

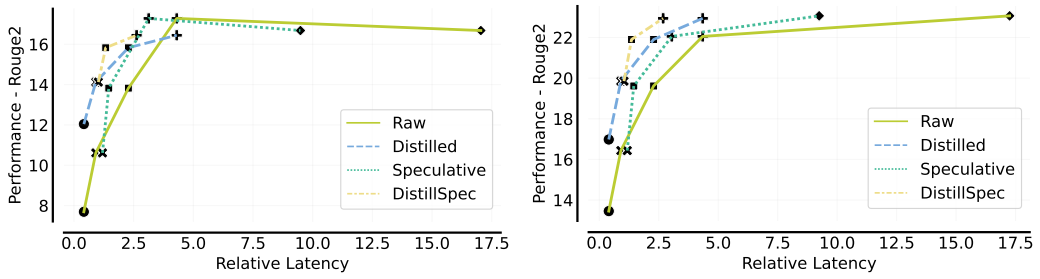
Figure D.22: Tradeoff between task performance and decoding latency on WMT enabled by combining lossy speculative decoding with the alternative variants of DistillSpec presented in Section 4. While the similar height and slope of the tradeoff curves of DistillSpec and standard SD indicate a comparable quality-latency tradeoff between the two setups, DistillSpec exhibits lower latency overall.



(a) Using non-distilled drafter model (standard SD) (b) Using distilled drafter modell (DistillSpec)

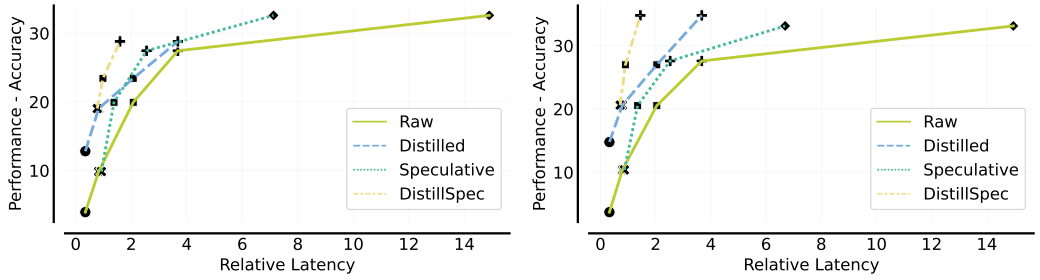
Figure D.23: Tradeoff between task performance and decoding latency on GSM8K enabled by combining lossy speculative decoding with the alternative variants of DistillSpec presented in Section 4. While the similar height and slope of the tradeoff curves of DistillSpec and standard SD indicate a comparable quality-latency tradeoff between the two setups, DistillSpec exhibits lower latency overall.

D.3.2 DISTILLSPEC MEETS MODEL GARDEN



(a) Non-greedy decoding ($T = 1$) (b) Greedy decoding ($T = 0$)

Figure D.24: Quality-latency trade-off curves on GSM8K using greedy ($T = 0$) or non-greedy ($T = 1$) sampling, for a single target LLM trained without distillation ("Raw"), trained by distilling from a larger LLM ("Distilled"), speculative decoding using a draft model trained without distillation ("Speculative"), and speculative decoding using a distilled target LLM with a distilled draft model ("DistillSpec").



(a) Non-greedy decoding ($T = 1$) (b) Greedy decoding ($T = 0$)

Figure D.25: Quality-latency trade-off curves on GSM8K using greedy ($T = 0$) or non-greedy ($T = 1$) sampling, for a single target LLM trained without distillation ("Raw"), trained by distilling from a larger LLM ("Distilled"), speculative decoding using a draft model trained without distillation ("Speculative"), and speculative decoding using a distilled target LLM with a distilled draft model ("DistillSpec").