Fuzzy Speculative Decoding for a Tunable Accuracy-Runtime Tradeoff

Anonymous ACL submission

Abstract

Speculative Decoding (SD) enforces strict distributional equivalence to the target model, limiting potential speed ups as distributions of near-equivalence achieve comparable outcomes in many cases. Furthermore, enforcing distributional equivalence means that users are unable to trade deviations from the target model distribution for further inference speed gains. To address these limitations, we introduce **Fuzzy Speculative Decoding (FSD)** - a decoding algorithm that generalizes SD by accepting candidate tokens purely based on the divergences between the target and draft model distributions. By allowing for controlled divergence from the target model, FSD enables users to flexibly trade generation quality for inference speed. Across several benchmarks, our method is able to achieve significant runtime improvements of over 5 tokens per second faster than SD at only an approximate 2% absolute reduction in benchmark accuracy. In many cases, FSD is even able to match SD benchmark accuracy at over 2 tokens per second faster, demonstrating that distributional equivalence is not necessary to maintain target model performance.

1 Introduction

Speculative decoding (SD), introduced by Leviathan et al. (2023) and Chen et al. (2023), is a large language model (LLM) inference acceleration algorithm that leverages a smaller, faster draft model to generate sequences of candidate tokens which are then verified and accepted in parallel by a larger target model. The speculative sampling rule that SD employs to determine which candidates to accept enforces a strict equivalence of the final sampling distribution and the original target model distribution. Thus, by cutting out the expensive sequential generation from the large target model, SD can lead to inference time reductions of around 2-3X while maintaining the same generation quality as the target model.

Despite this impressive speedup, SD suffers from two major flaws. Firstly, in order to maintain strict distributional equivalence to the target model, the SD candidate acceptance rule is overly strict, and in many cases may reject tokens that if accepted would have no impact on final generation quality (Lin et al., 2025), unnecessarily limiting the potential speed ups of SD . Secondly, the enforced distributional equivalence means that users cannot tune the SD acceptance rule to be more or less lenient in its candidate acceptance, preventing users trading deviations from the target model distribution for further inference speed gains. However, the flexibility for users to tune their LLM generation along an inference speed - generation quality tradeoff would be highly beneficial in real-world applications, as the relative importance of inference speed compared to generation quality may vary across scenarios within the same application.

To address these limitations of SD, we introduce **Fuzzy Speculative Decoding** - a generalized SD algorithm that determines token acceptance based on the divergence between the target and draft model distributions, allowing users to tune the generation quality - inference time tradeoff of their model. With FSD, users have the flexibility to tune a threshold parameter T that determines how lenient candidate acceptance should be, and thus can control how much they are willing to deviate from the target model's distribution in exchange for further runtime reductions. As it doesn't enforce strict distributional equivalence, FSD can achieve significant runtime improvements over SD by accepting a higher percentage of candidate tokens.

We conduct extensive experiments across four diverse benchmarks—spanning factoid QA, math, and coding—using three different model pairs. Our key findings are: 1. FSD matches SD's accuracy while achieving over 2 tokens per second speedup by relaxing strict distributional equivalence. 2. FSD enables greater speedups (up to 5 tokens per second) when a slight accuracy tradeoff is acceptable (approximately 2% absolute drop). 3. FSD offers a superior tunability mechanism, enabling a flexible tradeoff between the target and draft models. Compared to an alternative tunable approach that randomly assigns queries between the two models based on a predefined proportion, FSD consistently achieves higher accuracy across all speed settings.

We also perform a broad range of ablation studies, demonstrating that FSD's performance shares many similarities with SD, including dependence on draft and target model alignment for a given text and the ability to use both sample-based and greedy decoding strategies. We also show that benchmark performance under FSD is sensitive to which type of divergence used to determine token acceptance, with JS divergence performing better than KL divergence, TV distance, and top-k variants of these three divergences.

2 Previous works

Several works have sought to improve speculative decoding, primarily by increasing the acceptance rate of draft-generated tokens. Including but not limited to (1) **Verifying more tokens with tree-structured proposals:** Some methods improve efficiency by allowing the draft model to propose tokens in a tree structure, enabling the target model to verify multiple candidates in parallel using tree attention mechanisms. This expands the search space and increases the likelihood of accepting a valid token (Li et al., 2024b,a; Cai et al., 2024; Ankner et al., 2024; Miao et al., 2023; Chen et al., 2024). (2) **Aligning the draft model with the target model:** Methods include fine-tuning the draft model to mimic the target model's outputs (Zhou et al., 2023), granting the draft model access to additional representation information from the target model (AishwaryaP et al., 2024; Zhang et al., 2024b; Du et al., 2024), or even using a partial version of the target model as the draft model itself—such as using partial layers (Liu et al., 2024a; Elhoushi et al., 2024; Zhang et al., 2024a) or augmenting the target model with lightweight extensions to improve alignment (Monea et al., 2023; Fu et al., 2024; Santilli et al., 2023; Cai et al., 2024). (3) **Adaptive candidate length selection:** Instead of fixing the number of candidate tokens per step, some methods allow the draft model to determine when to stop generating (Kim et al., 2023; Huang et al., 2024), or enable the target model to verify tokens before the draft model has finished drafting (Liu et al., 2024b), leading to more flexible and efficient speculative decoding. While these methods enhance SD efficiency, they enforce strict distributional guarantees and offer limited flexibility in balancing accuracy and efficiency. In contrast, our framework demonstrates that such guarantees are unnecessary and provides tunable trade-offs. Moreover, its flexibility allows seamless integration with existing approaches, paving the way for further research and optimization.

The most similar method to ours is concurrent work Judge Decoding (JD) (Bachmann et al., 2025), an SD variant where a compact module is trained on token embeddings to 'judge' and accept candidate tokens based on correctness rather than strict alignment with the target model. This allows JD to accept more tokens than SD with minimal performance loss. However, JD has two major limitations. First, it generalizes poorly to unseen data, as token acceptance relies on a trained judgment module. Its performance drops significantly on out-of-distribution text (Bachmann et al., 2025). [1] Second, JD requires per-model training, preventing out-of-the-box use for new model pairs. In contrast, FSD is training-free, generalizes across datasets, and can be applied to any model pair out-of-the-box, effectively addressing JD's weaknesses.

3 Speculative Decoding

We start by reviewing how SD works in order to properly introduce FSD as an extension of this method.

Consider a larger target model M_T and a smaller draft model M_D. The biggest bottleneck when generating from M_T individually is that tokens are sequentially dependent, and therefore each token will require a full M_T forward pass to be generate conditional on the previously generated tokens. SD mitigates this bottleneck by first generating a sequence of candidate tokens sequentially from the faster M_D, and then uses a single M_T forward pass only to *verify* which which of these tokens to accept. Provided that M_D is a good enough approximation of M_T such that a significant portion of these candidate are accepted, the runtime saved by avoiding sequential generation from M_T outweighs the additional runtime of running M_D,

[1] E.g., the accuracy on `HumanEval` drops from 86.6 to 80.4% when excluded from training (Bachmann et al., 2025), which would be unacceptable for most applications.

resulting in an overall speedup. In order to maintain M_T's full generation quality, SD accepts candidate tokens based on an acceptance rule that guarantees the final sequence of sampled tokens will still be distributed the same as they would under M_T.

At each SD step, M_D first generates a sequence of L candidate tokens, $k = [x_0, x_1...x_L]$, which are then passed through M_T to calculate the likelihood of each candidate token x_i under M_T. Using this likelihood, each candidate x_i is accepted with the probability:

$$P_{accept}(x_i) = \min\left(1, \frac{P_{M_T}(x_i|x_{<i})}{P_{M_D}(x_i|x_{<i})}\right)$$

making the final candidate token SD acceptance rule:

$$F_{accept}(x_i) = \begin{cases} 1 & \text{if } P_{accept}(x_i) > y \sim \mathcal{U}(0,1) \\ 0, & \text{else} \end{cases}$$

Once SD reaches the first rejection of the candidate sequence, it resamples a token at the rejected candidate position from the adjusted distribution:

$$M_{resample} = P_{M_T}(x_i|x_{<i}) - P_{M_D}(x_i|x_{<i})$$

(Note that P_{M_T} and P_{M_D} will already have been calculated to determine the acceptance probability.)

By accepting tokens that are *more* likely under M_D than under M_T with a probability of $\frac{P_{M_T}(x_i|x_{<i})}{P_{M_D}(x_i|x_{<i})}$ and resampling rejected tokens from an adjusted distribution, SD corrects for the bias introduced by M_D, ensuring that the final distribution remains the same as that of M_T.

3.1 Determining SD speed-ups

The inference speed up of SD heavily depends on the percentage of candidate tokens accepted. Given a fixed candidate length L, the more similar the distributions of M_D and M_T tend to be over a given generation, the more frequently candidate tokens will be accepted, and thus the greater the inference acceleration. This makes the speed-ups achieved by SD highly dependent on the distribution of text the model is generating, which we can see in Table 1. This variation in acceptance percentages based on text distributions means that each text will have an optimal candidate length L for which the SD inference speed is maximized. However, once the optimal L has been found for the given text distribution, the percentage of tokens accepted is effectively fixed, capping the inference speed of SD to a level beyond which it cannot be increased further. This is the limitation of SD that FSD attempts to address.

Dataset		**Candidate length**		
		5	**10**	**15**
CSQA	Tk. / sec	9.3	9.3	7.9
	% M_D Tk.	75.7	82.8	84.6
GSM8K	Tk. / sec	11.3	13.2	13.0
	% M_D Tk.	81.5	89.2	91.4
MMLU	Tk. / sec	7.2	7.2	6.3
	% M_D Tk.	78.7	85.6	87.5
HumanEval	Tk. / sec	13.7	16.0	16.3
	% M_D Tk.	81.5	88.7	91.4

Table 1: Inference speeds and percent of tokens originating from M_D under SD on Llama3.1 8B + 70B. Tk. / s denotes tokens per second; % M_D Tk. denotes the percentage of total generated tokens originating from M_D.

4 Fuzzy Speculative Decoding

The defining difference of FSD is that it employs a different token acceptance rule that can be tuned to be more or less lenient in its acceptance decisions based on a threshold parameter T, which can be arbitrarily set by the user. This effectively allows users to determine how much they are willing to diverge from the target distribution M_T in exchange for a higher percentage of candidates accepted, resulting in speed-ups beyond SD.

While SD determines acceptance based on the likelihood of candidate x_i under P_{M_T} and P_{M_D}, FSD calculates the distribution-level divergence between these two distributions at each candidate position. Then, based on the tunable divergence threshold T, FSD will accept a candidate token if the models' divergence at the corresponding position is less than T. This makes the FSD acceptance rule:

$$F_{accept}(x_i) = \begin{cases} 1 & \text{if } \mathrm{Div}(P_{M_T}[i], P_{M_D}[i]) < T \\ 0, & \text{else} \end{cases}$$

where $P_{M_T}[i]$ and $P_{M_T}[i]$ are the M_T and M_D next token distributions at candidate position i respectively.

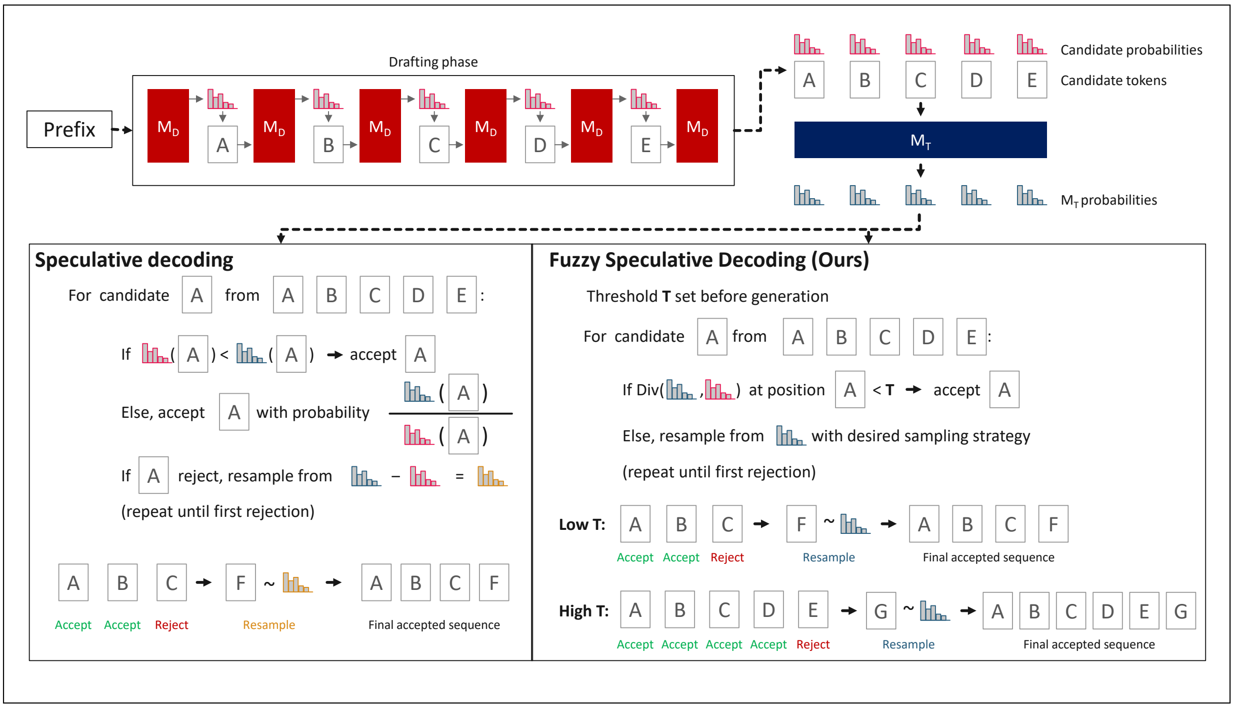


Figure 1: Visualization comparison between FSD and SD. SD accepts candidate token with a probability that depends on the relative likelihood of the candidate token under M_D and M_T. FSD determines candidate acceptance deterministically based on whether the divergence between the M_D and M_T distributions at the candidate's position exceeds a given threshold T. This lets users determine how many candidate tokens to accept by setting the threshold T accordingly.

In the case of candidate token rejection, FSD will sample from $P_{M_T}[i]$, that is the original target model distribution at the rejected position, with whatever sampling method the user sets for the generation. The full FSD algorithm is depicted if Figure 1 as a side-by-side comparison with SD.

4.1 Intuitive motivation

Just like SD, FSD aims to accept candidate tokens at positions for which M_T and M_D are similar. Instead of relying on strict equivalence in final distribution, FSD relies on the fact that across an entire generation, M_T and M_D will produce similar tokens when the divergence between their distributions is low. This in turn means that at positions with low divergence, we can likely use tokens sampled from M_D in place of those sampled from M_T with minimal impact on the final generation.

By tuning T, users can directly dictate how lenient candidate acceptance rule should be, thereby implicitly determining how much they are willing to allow the final sampling distribution to diverge from M_T in exchange for further runtime reductions. In addition, as the FSD acceptance rule becomes more relaxed, users can also increase the candidate length L past the value that was optimal for SD to realize even further reductions in inference time.

As a general framework, FSD can use any divergence type that relies solely on P_{M_T} and P_{M_D}. In this work, we focused on KL divergence, JS divergence, and total variation distance. We defines these divergences in Appendix A. We also perform an empirical evaluation of FSD performance under at these different divergence types in our Appendix C, which indicates that JS divergence is the best performing divergence type.

4.2 Final divergence from M_T under FSD

Unlike SD, FSD does not enforce distributional equivalence to M_T. Tokens generated via FSD are sampled from a distribution that has diverged from M_T by an amount dependent on the threshold T. Specifically, when generating a sequence of N tokens, the divergence between FSD sequence-level distribution and the M_T sequence level distribution is upper bounded by:

$$\mathrm{Div}(P_{M_T}(x_{1:N}), P_{\mathrm{FSD}}(x_{1:N})) \leq N \cdot \%_{M_D} \cdot T$$

where P_{FSD} is the distribution of a sequence sampled from M_D and M_T using FSD, N is the sequence length, $\%_{M_D}$ is the percentage of final tokens originating from M_D, and T is the divergence threshold set by the user. We show the derivation of this bound in Appendix B.

While this bound establishes a theoretical limit on divergence, it doesn't directly indicate how FSD impacts downstream performance. The relationship between sequence-level divergence and generation quality is non-trivial, as performance degradation depends not only on the magnitude of sequence-level divergence, but also on *which* tokens the models diverge on. Thus, an empirical evaluation is necessary to quantify how different choices of T impact model performance.

5 Main experiments

5.1 Experiment design

We tested FSD at various thresholds in comparison to SD on a range of benchmarks, reporting benchmark accuracy, inference speed (tokens/second), and average length of accepted candidates sequences for three of these threshold levels (denoted FSD (Low), (Med.), and (High)). We evaluated on CommonsenseQA (Talmor et al., 2019) for factual knowledge, GSM8K (Cobbe et al., 2021) for math, MMLU (Hendrycks et al., 2021) [2] for general knowledge and reasoning, and HumanEval (Chen et al., 2021) for code generation. We performed experiments on 3 M_D - M_T model pairs of varying size: Llama3.1 8B + 70B (Grattafiori et al., 2024), Gemma2 2B + 27B (Team et al., 2024), and Qwen2.5 7B + 32B (Qwen et al., 2025). All Gemma2 and Qwen2.5 tests were performed on 2 A6000s, while the Llama3.1 tests were performed on 2 A100s. We use a batch size of 1 for all experiments. JS divergence was chosen as the divergence type following a preliminary experiments that indicated it performed the best. An in depth explanation of the experiment design can be found in Appendix D, and the results of our preliminary divergence type comparison in Appendix C.

5.2 Implementation

To perform our experiments, we modified huggingface's transformers library (Wolf et al., 2020) to implemented FSD within the library's assisted generation functionality [3]. This allows us to easily test FSD using the transformers library and allows for a fair comparison to SD, which is implemented in the library by default.

5.3 FSD performance

We present our experimental results in Table 2 and in Figure 2.

FSD generally matches SD accuracy at noticeably faster inference speeds. When setting T to lower values, FSD's accuracy converges to the level of SD, often reaching this level while accepting more candidate tokens and thereby realizing greater runtime improvements. This clearly demonstrates that in many cases, the distributional equivalence enforced by SD is not necessary maintain the full M_T performance level. Particularly notable are the Llama3.1 and Qwen2.5 GSM8K results, in which FSD is able to outperform SD at around 3 and 4 tokens per second faster, respectively.

As mentioned in section 2, many other SD extensions have been able to achieve SD performance at faster generation speeds, so this finding isn't necessarily unique to FSD. However, these prior methods all still enforce strict distributional equivalence to M_T, making our findings notable as they demonstrate this equivalence is often not necessary. Furthermore, given this fundamental difference, our method could easily be applied to these existing SD extensions in order to further extend their respective speedups, as well as introduce the accuracy - runtime tunability we describe below to these otherwise inflexible methods. We leave this exploration to future works.

FSD achieves even greater runtime improvements over SD when slight accuracy loss is acceptable. As T increases, FSD is able to achieve runtime speedups far greater than SD while only sacrificing small reductions in benchmark accuracy. The higher the divergence from M_T we are willing to tolerate when accepting tokens, the greater the runtime improvement over SD. While benchmark accuracy does eventually degrade as T increases, we note how minimal this deterioration is. For instance, FSD with Llama3.1 8B + 70B on CSQA achieves a 6 token per second increase over the inference speed of SD in exchange for only a 2% absolute reudction in accuracy. We expect that in many applications of LLMs, such a runtime improvement would likely justify these small

[2] Due to runtime constraints, we used a subset of the full MMLU dataset. This subset was sampled such that the relative prevalence of each question subject was preserved

[3] We will release our codebase upon publication

	GSM8K			CSQA			MMLU			HumanEval		
	Llama3.1 8B + 70B											
	Acc	Spd	ALen	Acc	Spd	ALen	Acc	Spd	ALen	Acc	Spd	ALen
M_D	84.6	31.8	-	73.8	32.5	-	72.2	32.8	-	63.2	33.0	-
M_T	94.9	8.5	-	83.6	8.9	-	86.2	9.3	-	79.1	9.3	-
SD	95.1	16.8	9.7	84.1	13.5	1.96	84.8	15.8	3.37	77.4	20.5	7.6
FSD (Low)	95.2	19.5	11.8	84.0	14.4	3.32	84.0	17.0	3.9	78.9	22.3	8.5
FSD (Med.)	94.3	21.2	12.4	83.7	17.5	4.3	83.0	18.1	4.1	77.6	23.2	8.6
FSD (High)	93.1	22.0	13.5	82.1	19.5	8.14	82.6	18.8	4.2	77.4	23.6	8.9
	Gemma2 2B + 27B											
	Acc	Spd	ALen	Acc	Spd	ALen	Acc	Spd	ALen	Acc	Spd	ALen
M_D	57.5	28.5	-	64.6	31.3	-	55.2	24.3	-	40.9	17.9	-
M_T	90.7	8.8	-	83.0	9.1	-	75.3	9.4	-	75.6	9.6	-
SD	90.8	16.2	5.7	83.1	11.5	2.07	76.8	12.2	2.7	76.2	12.4	3.7
FSD (Low)	89.6	18.4	6.8	82.3	13.9	2.5	75.6	13.3	2.9	78.7	13.6	4.02
FSD (Med.)	88.5	19.4	7.1	81.6	15.7	3.2	75.4	15.5	3.5	77.8	14.1	4.2
FSD (High)	86.1	21.5	11.1	79.5	17.5	3.9	74.2	16.1	3.7	75.8	14.3	4.3
	Qwen2.5 7B + 32B											
	Acc	Spd	ALen	Acc	Spd	ALen	Acc	Spd	ALen	Acc	Spd	ALen
M_D	89.9	34.8	-	80.2	36.6	-	71.9	35.6	-	68.1	26.9	-
M_T	94.9	8.8	-	86.9	9.1	-	82.7	9.6	-	80.9	9.6	-
SD	95.1	17.4	6.6	86.8	14.0	2.7	82.2	16.0	3.2	82.1	15.2	3.7
FSD (Low)	94.7	21.4	8.2	86.6	16.1	3.3	82.0	18.0	3.7	81.9	17.1	4.3
FSD (Med.)	94.2	22.4	9.2	86.1	19.5	6.6	81.6	19.5	4.0	79.0	17.2	4.4
FSD (High)	94.0	22.0	9.3	85.9	20.9	6.9	81.7	20.7	4.46	78.3	17.7	4.6

Table 2: Benchmark performance of FSD at varying threshold levels compared to M_D, M_T, and SD. "Acc" refers to the QA accuracy. "Spd" refers to Inference Speed (tokens/sec.). "ALen" refers to the average accepted sequence length.

reductions in generation quality.

FSD allows for a previously unattainable accuracy - runtime tunability. The accuracy - runtime tunability of FSD is demonstrated in Figure 2. A model with good tunability should satisfy two key requirements: (1) it should allow flexibility in adjusting the speed-accuracy trade-off across the speed axis, and (2) it should achieve the highest possible accuracy compared to other methods at the same speed. Unlike SD, which has a fixed efficiency, FSD enables flexible adjustments along the speed axis while maintaining minimal accuracy degradation, thereby meeting the first requirement. To evaluate the second requirement, we introduce a *random allocation* baseline, where queries are randomly assigned between the target and draft models, allowing tunability by adjusting the proportion of queries sent to the target model. We represent this baseline with a greyline interpolating between the target and draft models. As shown in Figure 2, FSD consistently outperforms the random allocation method across all speeds, demonstrating not only its flexibility but also its superior tunability.

6 Ablation studies

6.1 FSD and SD variation across datasets

As expected, the acceptance percentages and thereby the runtime improvements of both FSD and SD are highly dependent on the benchmark. We observe that FSD follows the same trends in acceptance percentages across datasets that SD does. That is, the benchmarks on which SD accept more candidate tokens (of course at the M_T accuracy level) are also the benchmarks on which FSD can accepts more candidates when set to match this M_T accuracy level.

This trend points to an underlying difference in draft and target model alignment across datasets which is affecting both methods ability to accept tokens. We illustrate this difference in M_D and M_T model alignment across datasets in Figure 5, which shows the distribution of JS divergences between the Llama models on a subset of question from each dataset. As we can see, the divergences are much more heavily skewed to be much lower on datasets for which both SD and FSD accept more tokens,

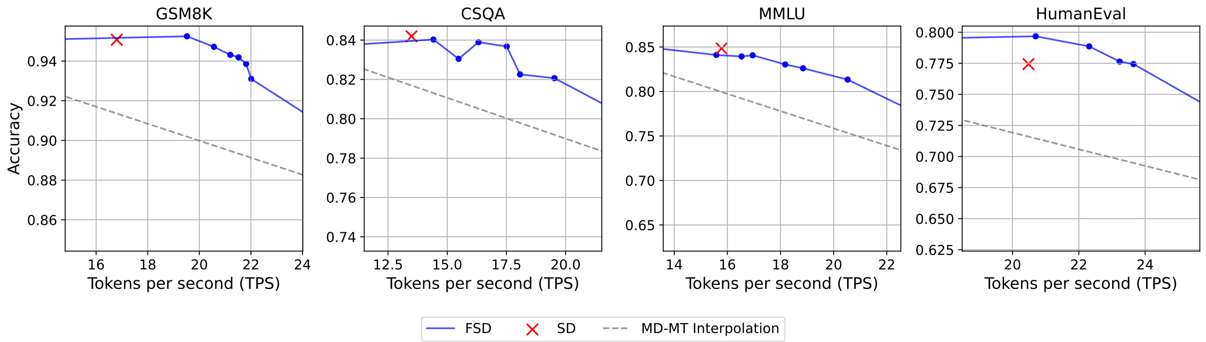


Figure 2: FSD Benchmark accuracy - inference speed trade off compared to SD. Results were collected with Llama3.1 8B + 70B as model pair

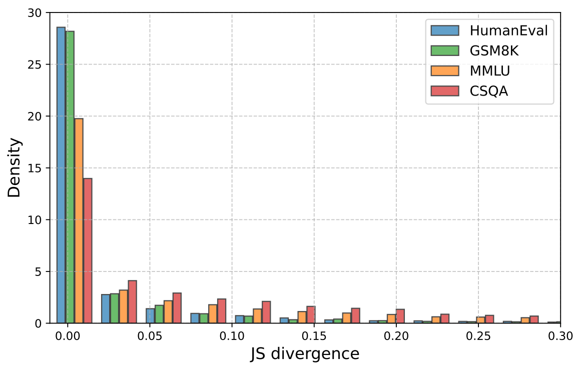


Figure 3: Distributions of JS divergences between M_D and M_T across tested datasets. Long tail of distributions (JS div. ≥ 0.3) truncated for better visibility.

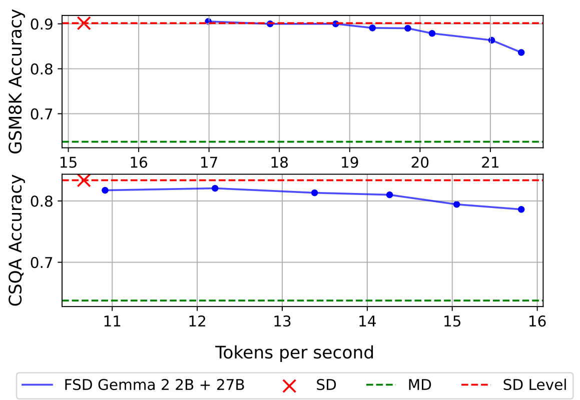


Figure 4: FSD performance on GSM8K and CSQA with greedy decoding from M_T distribution in case of rejection. SD baselines also used greedy decoding. Model pair used was Gemma2 2B + 7B.

such as GSM8K and HumanEval. Intuitively, this makes sense: the more similar distributions tend to be across a given text generation, the lower their divergences, and thus the more candidates FSD will accept at a given threshold. Likewise, the more similar the distributions, the more likely it is that SD accepts a candidate, since the acceptance probabilities will tend to be higher. Thus, it makes sense that both FSD and SD's runtimes follow the same trend across benchmarks.

6.2 Threshold-accuracy relationship

As previously discussed, the relationship between FSD threshold, the percentage of M_D tokens accepted, and downstream benchmark performance is highly dependent on the dataset and the candidate length L. Thus, before conducting any calibration tests, users will not know what acceptance percentage and downstream performance correspond to each threshold T and candidate length L.

However, we find that the performance level corresponding to a given threshold loosely generalizes across datasets, giving users a good starting point when setting T on an unknown distribution. As an example of this, Table 3 shows the performance of FSD for all three model pairs at a single selected threshold specific to each pair. We can see that for all three pairs, FSD with this constant threshold consistently achieves approximately SD accuracy at around 1-3 tokens per second faster than SD across all datasets. Thus, similar to how certain candidate lengths are known to be good starting points for SD and can later be tuned based on the specific text distribution, we show that the similar out-of-the-box values thresholds values exist for FSD.

6.3 Greedy decoding vs. sample-based decoding

As described in the experiment setup, we used greedy decoding to generate candidate sequences from M_D, and used sample-based decoding to sample from the M_T in the case of candidate rejection. While greedy decoding from M_D is standard prac-

Dataset	Qwen2.5 7B + 32B			Llama3.1 8B + 70B			Gemma2 2B + 27B		
	SD Acc.	FSD Acc. @ T = 0.4	Speedup over SD (tokens / second)	SD Acc.	FSD Acc. @ T = 0.3	Speedup over SD (tokens / second)	SD Acc.	FSD Acc. @ T = 0.7	Speedup over SD (tokens / second)
GSM8K	95.1	94.7	3.4	95.1	94.7	3.7	90.8	89.6	2.1
CSQA	86.8	86.4	3.6	84.2	83.8	2.8	83.1	82.3	2.4
MMLU	82.3	82.1	2.8	84.8	84.1	1.2	76.8	74.8	1.9
HumanEval	82.1	81.9	2.9	77.4	77.6	2.7	76.2	77.8	1.7

Table 3: FSD performance comparison across datasets at set thresholds

tice when using SD, both greedy and sample-based decoding are regularly used in SD to sample from the adjusted distribution in case of rejection. Thus, the question arises whether FSD is also able to accommodate for greedy decoding, in addition to sample-based, in the case of candidate rejection.

To test this, we evaluated FSD performance on GSM8K and CSQA with greedy decoding and compared this performance to that of SD under greedy decoding, to see whether the performance trend is similar to what we observe in Table 2. As we can see in Figure 4, FSD seems to follow the same performance trend observed in the main results under greedy decoding. We can again see FSD converge to SD performance at lower thresholds, and achieve significant runtime improvements at the cost of accuracy at higher thresholds. Again we can also see that the higher model alignment on GSM8K we discussed above allows FSD to achieve more impressive results over SD on this dataset, while the performance on CSQA is slightly weaker. This all is consistent with our main results in Table 2.

7 Discussion

7.1 Potential further developments

Unlike the probabilistic acceptance rule of SD, the FSD acceptance criteria is deterministic given the M_D and M_T logits. This means that FSD allows for the generation of a token-level dataset of acceptance / rejected labels, since the FSD acceptance decision relies solely on the M_D and M_T distributions at each tokens position. This unlocks the possibility of training a classifier to predict which tokens will be accepted and which will be rejected, based purely on the tokens up to the position being generated. Such a classifier can be used to dynamically set the candidate length generated by the draft model, reducing the number of rejected tokens at each SD step and thereby further increasing the inference time speed ups.

The second area that we feel has potential for future development is the testing and development of a novel divergence types to identify which candidate tokens should be accepted with limited impact on generation quality. Given that FSD was already able to achieve very impressive results with simple divergence types like KL divergence and JS divergence, we expect that the divergences tailored specifically to this methods are likely to further mitigate the deterioration of generation quality as the acceptance threshold T increases and allow FSD to maintain quality at even higher generation speeds. Judge Decoding (Bachmann et al., 2025) attempts a similar approach to this by using learned token correctness to determine acceptance, however as discussed this method doesn't generalize, leaving this direction open for further research.

8 Conclusion

We have introduced FSD - a modified SD algorithm that can accept divergence allows users to tune how much divergence from M_T they are willing to accept in exchange for runtime improvement beyond SD. This flexibility to achieve significantly higher runtimes, in addition an ability to match SD generation quality at faster inference in certain scenarios, makes FSD novel alternative to SD that we expect can be valuable in many LLM applications. We have shown that FSD is able to achieve very strong empirical results on-par with SD, and is able to achieve considerably higher generation speeds the cost of only minor deteriorations in generation quality.

9 Limitations

The biggest limitation of our method is that it is not preemptively known what threshold T will result in what downstream generation performance, as this relationship is highly dependent on the distribution of the text being generated and the candidate length L. Thus, a practical application of FSD will have to either perform calibration tests on a text distribution similar to the eventual generation distribution, or will have to use a potentially suboptimal out-of-the-box value similar to those discussed in section 6.2. However, we note that SD suffers from a similar reliance on hyperparameter tuning, as its inference speed is highly dependent on using the correct L. An incorrect selection of L can result in SD having no impact on or even decreasing the generation speed compared to M_T. Thus, FSD's sensitivity to T is simply an additional reliance on hyperparameters. Another major limitation of our method is that FSD, is unable to theoretically guarantee that the generation quality of M_T will be maintained, making SD a safer choice for applications in which generation quality is significantly more important than inference speed. However, our results indicate that FSD is *empirically* able to maintain M_T performance, often even at noticeably higher inference speeds, so in practice we feel this lack of theoretical guarantee is not a major issue.

References

S AishwaryaP, Pranav Ajit Nair, Yashas Samaga, Toby Boyd, Sanjiv Kumar, Prateek Jain, and Praneeth Netrapalli. 2024. Tandem transformers for inference efficient llms. *ArXiv*, abs/2402.08644.

Zack Ankner, Rishab Parthasarathy, Aniruddha Nrusimha, Christopher Rinard, Jonathan Ragan-Kelley, and William Brandon. 2024. Hydra: Sequentially-dependent draft heads for medusa decoding. *ArXiv*, abs/2402.05109.

Gregor Bachmann, Sotiris Anagnostidis, Albert Pumarola, Markos Georgopoulos, Artsiom Sanakoyeu, Yuming Du, Edgar Schönfeld, Ali Thabet, and Jonas Kohler. 2025. Judge decoding: Faster speculative sampling requires going beyond model alignment. *Preprint*, arXiv:2501.19309.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. 2024. Medusa: Simple llm inference acceleration framework with multiple decoding heads. *Preprint*, arXiv:2401.10774.

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

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021. Evaluating large language models trained on code. *Preprint*, arXiv:2107.03374.

Zhuoming Chen, Avner May, Ruslan Svirschevski, Yuhsun Huang, Max Ryabinin, Zhihao Jia, and Beidi Chen. 2024. Sequoia: Scalable, robust, and hardware-aware speculative decoding. *ArXiv*, abs/2402.12374.

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

Cunxiao Du, Jing Jiang, Yuanchen Xu, Jiawei Wu, Sicheng Yu, Yongqi Li, Shenggui Li, Kai Xu, Liqiang Nie, Zhaopeng Tu, and Yang You. 2024. Glide with a cape: A low-hassle method to accelerate speculative decoding. *ArXiv*, abs/2402.02082.

Mostafa Elhoushi, Akshat Shrivastava, Diana Liskovich, Basil Hosmer, Bram Wasti, Liangzhen Lai, Anas Mahmoud, Bilge Acun, Saurabh Agarwal, Ahmed Roman, Ahmed Aly, Beidi Chen, and Carole-Jean Wu. 2024. LayerSkip: Enabling early exit inference and self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12622–12642, Bangkok, Thailand. Association for Computational Linguistics.

Yichao Fu, Peter D. Bailis, Ion Stoica, and Hao Zhang. 2024. Break the sequential dependency of llm inference using lookahead decoding. *ArXiv*, abs/2402.02057.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. 2024. The llama 3 herd of models. *Preprint*, arXiv:2407.21783.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Preprint*, arXiv:2009.03300.

Kaixuan Huang, Xudong Guo, and Mengdi Wang. 2024. Specdec++: Boosting speculative decoding via adaptive candidate lengths. *ArXiv*, abs/2405.19715.

Sehoon Kim, Karttikeya Mangalam, Suhong Moon, Jitendra Malik, Michael W. Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Speculative decoding with big little decoder. In *Neural Information Processing Systems*.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding. *Preprint*, arXiv:2211.17192.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024a. EAGLE-2: Faster inference of language models with dynamic draft trees. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 7421–7432, Miami, Florida, USA. Association for Computational Linguistics.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. 2024b. Eagle: Speculative sampling requires rethinking feature uncertainty. *ArXiv*, abs/2401.15077.

Zicheng Lin, Tian Liang, Jiahao Xu, Qiuzhi Lin, Xing Wang, Ruilin Luo, Chufan Shi, Siheng Li, Yujiu Yang, and Zhaopeng Tu. 2025. Critical tokens matter: Token-level contrastive estimation enhances llm's reasoning capability. *Preprint*, arXiv:2411.19943.

Jiahao Liu, Qifan Wang, Jingang Wang, and Xunliang Cai. 2024a. Speculative decoding via early-exiting for faster LLM inference with Thompson sampling control mechanism. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3027–3043, Bangkok, Thailand. Association for Computational Linguistics.

Tianyu Liu, Yun Li, Qitan Lv, Kai Liu, Jianchen Zhu, and Winston Hu. 2024b. Parallel speculative decoding with adaptive draft length. *ArXiv*, abs/2408.11850.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, Chunan Shi, Zhuoming Chen, Daiyaan Arfeen, Reyna Abhyankar, and Zhihao Jia. 2023. Specinfer: Accelerating large language model serving with tree-based speculative inference and verification. *Proceedings of the 29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 3*.

Giovanni Monea, Armand Joulin, and Edouard Grave. 2023. Pass: Parallel speculative sampling. *ArXiv*, abs/2311.13581.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. 2025. Qwen2.5 technical report. *Preprint*, arXiv:2412.15115.

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodola. 2023. Accelerating transformer inference for translation via parallel decoding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 12336–12355, Toronto, Canada. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. Commonsenseqa: A question answering challenge targeting commonsense knowledge. *Preprint*, arXiv:1811.00937.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, Johan Ferret, Peter Liu, Pouya Tafti, Abe Friesen, Michelle Casbon, Sabela Ramos, Ravin Kumar, Charline Le Lan, Sammy Jerome, Anton Tsitsulin, Nino Vieillard, Piotr Stanczyk, Sertan Girgin, Nikola Momchev, Matt Hoffman, Shantanu Thakoor, Jean-Bastien Grill, Behnam Neyshabur, Olivier Bachem, Alanna Walton, Aliaksei Severyn, Alicia Parrish, Aliya Ahmad, Allen Hutchison, Alvin Abdagic, Amanda Carl, Amy Shen, Andy Brock, Andy Coenen, Anthony Laforge, Antonia Paterson, Ben Bastian, Bilal Piot, Bo Wu, Brandon Royal, Charlie Chen, Chintu Kumar, Chris Perry, Chris Welty, Christopher A. Choquette-Choo, Danila Sinopalnikov, David Weinberger, Dimple Vijaykumar, Dominika Rogozińska, Dustin Herbison, Elisa Bandy, Emma Wang, Eric Noland, Erica Moreira, Evan Senter, Evgenii Eltyshev, Francesco Visin, Gabriel Rasskin, Gary Wei, Glenn Cameron, Gus Martins, Hadi Hashemi, Hanna Klimczak-Plucińska, Harleen Batra, Harsh Dhand, Ivan Nardini, Jacinda Mein, Jack Zhou, James Svensson, Jeff Stanway, Jetha Chan, Jin Peng Zhou, Joana Carrasqueira, Joana Iljazi, Jocelyn Becker, Joe Fernandez, Joost van Amersfoort, Josh Gordon, Josh Lipschultz, Josh Newlan, Ju yeong Ji, Kareem Mohamed, Kartikeya Badola, Kat Black, Katie Millican, Keelin McDonell, Kelvin Nguyen, Kiranbir Sodhia, Kish Greene, Lars Lowe Sjoesund, Lauren Usui, Laurent Sifre, Lena Heuermann, Leticia Lago, Lilly McNealus, Livio Baldini Soares, Logan Kilpatrick, Lucas Dixon, Luciano Martins, Machel Reid, Manvinder Singh, Mark Iverson, Martin Görner, Mat Velloso, Mateo Wirth, Matt Davidow, Matt Miller, Matthew Rahtz, Matthew Watson, Meg Risdal, Mehran Kazemi, Michael Moynihan, Ming Zhang, Minsuk Kahng, Minwoo Park, Mofi Rahman, Mohit Khatwani, Natalie Dao, Nenshad Bardoliwalla, Nesh Devanathan, Neta Dumai, Nilay Chauhan, Oscar Wahltinez, Pankil Botarda, Parker Barnes, Paul Barham, Paul Michel, Pengchong Jin, Petko Georgiev, Phil Culliton, Pradeep Kuppala, Ramona Comanescu, Ramona Merhej, Reena Jana, Reza Ardeshir Rokni, Rishabh Agarwal, Ryan Mullins, Samaneh Saadat, Sara Mc Carthy, Sarah Cogan, Sarah Perrin, Sébastien M. R. Arnold, Sebastian Krause, Shengyang Dai, Shruti Garg, Shruti Sheth, Sue Ronstrom, Susan Chan, Timothy Jordan, Ting Yu, Tom Eccles, Tom Hennigan, Tomas Kocisky, Tulsee Doshi, Vihan Jain, Vikas Yadav, Vilobh Meshram, Vishal Dharmadhikari, Warren Barkley, Wei Wei, Wenming Ye, Woohyun Han, Woosuk Kwon, Xiang Xu, Zhe Shen, Zhitao Gong, Zichuan Wei, Victor Cotruta, Phoebe Kirk, Anand Rao, Minh Giang, Ludovic Peran, Tris Warkentin, Eli Collins, Joelle Barral, Zoubin Ghahramani, Raia Hadsell, D. Sculley, Jeanine Banks, Anca Dragan, Slav Petrov, Oriol Vinyals, Jeff Dean, Demis Hassabis, Koray Kavukcuoglu, Clement Farabet, Elena Buchatskaya, Sebastian Borgeaud, Noah Fiedel, Armand Joulin, Kathleen Kenealy, Robert Dadashi, and Alek Andreev. 2024. Gemma 2: Improving open language models at a practical size. *Preprint*, arXiv:2408.00118.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing. *Preprint*, arXiv:1910.03771.

Jun Zhang, Jue Wang, Huan Li, Lidan Shou, Ke Chen, Gang Chen, and Sharad Mehrotra. 2024a. Draft& verify: Lossless large language model acceleration via self-speculative decoding. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11263–11282, Bangkok, Thailand. Association for Computational Linguistics.

Lefan Zhang, Xiaodan Wang, Yanhua Huang, and Ruiwen Xu. 2024b. Learning harmonized representations for speculative sampling.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. 2023. Distillspec: Improving speculative decoding via knowledge distillation. *ArXiv*, abs/2310.08461.

A Divergence definitions

Kullback–Leibler (KL) Divergence:

$$D_{\mathrm{KL}}(P_{M_T} \| P_{M_D}) = \sum_{t \in \mathcal{V}} P_{M_T}(t \mid x) \log \left(\frac{P_{M_T}(t \mid x)}{P_{M_D}(t \mid x)} \right)$$

where $\mathcal{V}$ is the vocabulary, $P_{M_T}(t \mid x)$ is the probability assigned by model M_T to token t given context x, $P_{M_D}(t \mid x)$ is the probability assigned by model M_D to token t given context x.

Jensen–Shannon (JS) Divergence:

$$D_{\mathrm{JS}}(P_{M_T} \| P_{M_D}) = \frac{1}{2} D_{\mathrm{KL}}(P_{M_T} \| M) + \frac{1}{2} D_{\mathrm{KL}}(P_{M_D} \| M)$$

where $M(t \mid x)$ is the mixture distribution (average of P_{M_T} and P_{M_D}):

$$M(t \mid x) = \frac{P_{M_T}(t \mid x) + P_{M_D}(t \mid x)}{2}$$

and D_{KL} is the Kullback–Leibler divergence as defined above.

Total Variation (TV) Distance:

$$D_{\mathrm{TV}}(P_{M_T}, P_{M_D}) = \frac{1}{2} \sum_{t \in \mathcal{V}} |P_{M_T}(t \mid x) - P_{M_D}(t \mid x)|$$

where: $P_{M_T}(t \mid x)$ and $P_{M_D}(t \mid x)$ are the probabilities from models M_T and M_D respectively, as defined above.

B Derivation of FSD sequence-level divergence bound

B.1 KL divergence bound

Starting with the sequence-level KL divergence decomposed autoregressively:

$$D_{\text{KL}}(P_{M_T} \| P_{M_{FSD}})$$

$$= \sum_{t=1}^{T} E_{P_{M_T}(x_{1:t-1})} \left[D_{\text{KL}}(P_{M_T}(t \mid x) \| P_{M_{FSD}}(t \mid x)) \right]$$

By assumption, at each step when the M_D - M_T divergence exceeds τ, P_{M_T} is used instead of P_{M_D}, making the divergence 0. Let p_{use} be the probability that P_{M_D} is used:

$$D_{\text{KL}}(P_{M_T}(t \mid x) \| P_{M_{FSD}}(t \mid x)) \leq p_{\text{use}} \tau$$

Summing over T steps:

$$D_{\text{KL}}(P_{M_T} \| P_{M_D}) \leq \sum_{t=1}^{T} p_{\text{use}} \tau = T p_{\text{use}} \tau$$

B.2 JS divergence bound

The JS divergence is defined as:

$$D_{\text{JS}}(P_{M_T} \| P_{M_D}) = \frac{1}{2} D_{\text{KL}}(P_{M_T} \| M) + \frac{1}{2} D_{\text{KL}}(P_{M_D} \| M)$$

Using the KL decomposition for both terms and applying the same per-step bound τ for when P_{M_D} is used:

$$D_{\text{JS}}(P_{M_T} \| P_{M_D}) \leq \frac{1}{2} \sum_{t=1}^{T} p_{\text{use}} \tau + \frac{1}{2} \sum_{t=1}^{T} p_{\text{use}} \tau = T p_{\text{use}} \tau$$

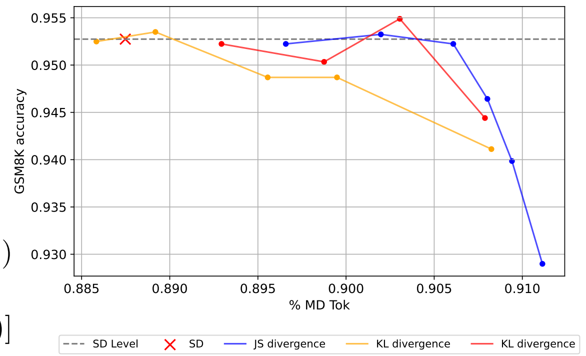
B.3 TV distance bound

The sequence-level TV distance decomposes similarly via subadditivity:

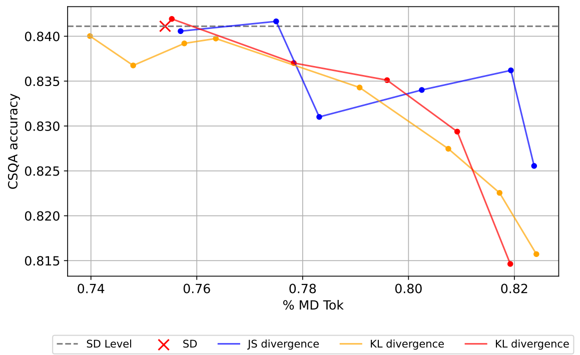
$$D_{\text{TV}}(P_{M_T}, P_{M_D}) \leq \sum_{t=1}^{T} E_{P_{M_T}(x_{1:t-1})} [D_{\text{TV}}(P_{M_T}(t \mid x), P_{M_D}(t \mid x))]$$

By assumption, if P_{M_D} is used, the per-step TV distance is bounded by τ:

$$D_{\text{TV}}(P_{M_T}(t \mid x), P_{M_D}(t \mid x)) \leq p_{\text{use}} \tau$$



(a) FSD performance of different divergence types on GSM8K.



(b) FSD performance of different divergence types on CSQA.

Figure 5: Comparison of FSD performance on GSM8K and CSQA with different divergence types.

Summing over T steps:

$$D_{\text{TV}}(P_{M_T}, P_{M_D}) \leq \sum_{t=1}^{T} p_{\text{use}} \tau = T p_{\text{use}} \tau$$

Final Result: For all three divergences, the upper bound is:

$$D(P_{M_T} \| P_{M_D}) \leq T p_{\text{use}} \tau.$$

C Divergence comparison under FSD

We referenced in the section 5.1, we performed preliminary tests on the difference divergence types to see which divergence was best able to maintain SD accuracy as T increases. The results of this preliminary experiments can be seen below.

D In-depth experiment design

Below is the experiment design we followed to collect our main results.

For each benchmark, we start by empirically determining the approximately optimal SD candidate length L by testing SD with $L = [5, 10, 15, 20]$ on a small subset of questions, and select the L

with the fastest inference speed as the candidate length to be used in our SD baseline. We denote this SD optimal candidate length as L'. We then test FSD with threshold $T = [0.1, 0.2, ...0.9, 1.0]$ at L' on the same small subset of question to determine the threshold T_{SD} that accepts approximately the same percentage of candidate tokens as SD. Starting from this 'equivalent' T_{SD}, we then evaluate FSD's benchmark performance at threshold increasing in increments of 0.1, until benchmark performance has degraded by approximately 20% of the performance difference between M_D and M_T. (e.g. if M_T scores 90%, M_D scores 80%, we test FSD at increasing T until accuracy reaches $90 - ((90 - 80) * 0.2) = 88\%$) For each threshold, we complete complete 3 trials, using greedy decoding to generate the candidate sequences from M_D and sample-based decoding to sample from M_T in the case of candidate rejection. We use the same sampling strategy for our SD baseline, as this is the default for the huggingface assisted generation implementation we used.

Importantly, as the acceptance percentage increases beyond that of SD, L' may no longer be the optimal candidate length. Thus, we increased L' to the next highest length in $[5, 10, 15, 20]$ if we observed that FSD is accepting close to all candidates.

To quantify the performance-runtime tunability of our method, we report the FSD benchmark accuracy, inference speed (tokens/second), and average length of accepted candidates sequences at three increasing threshold levels (denoted FSD (Low), (Med.), and (High)). These three levels are meant to simulate scenarios in which users are willing to accept increasing drops in generation quality in exchange for increasing generation speeds.

Random baseline In Table 2, we can clearly see that benchmark accuracy is highly sensitive to the percentage of candidate tokens accepted. For every benchmark, FSD accuracy is almost identical to SD accuracy when the threshold T is set such FSD accepts a similar percentage of candidate tokens. This begs the question: is benchmark performance simply a function of the candidate acceptance percentage, irrespective of *which* tokens are being accepted?

To test this, we performed a random FSD baseline, in which FSD was set to randomly accept a certain percentage of candidate tokens. By doing this, we are able to determine whether the divergences between distributions is an effective method of determining which tokens can be accepted with minimal impact on downstream performance, or whether this performance is mostly determined by *how many* M_D tokens are accepted. We report these results in Figure 6. As expected, we can see that FSD with random candidate acceptance does significantly worse than regular divergence-based FSD, even when significantly fewer candidates from M_D are being accepted. Thus, it does seem that M_D-M_T divergence is an effective criteria for deciding which candidates to accept, implying that the development of better divergences will likely improve FSD performance even further.

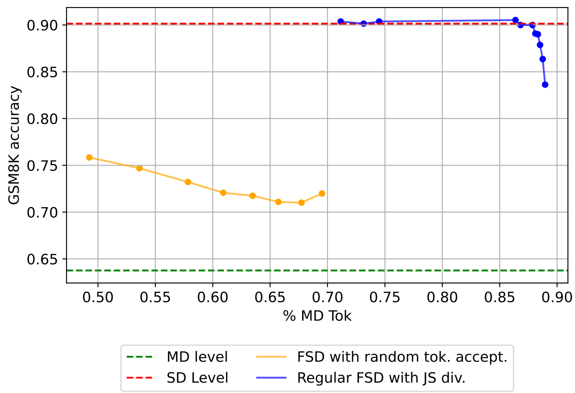


Figure 6: CSQA performance of regular FSD vs FSD with random token acceptance at varying percentages of M_D tokens. % MD Tok. denotes the percentage of final generated tokens originating from M_D. Experiment was performed on Gemma2 2B + 27B model pair