

KV Caching and Speculative Decoding at GPT-2 Scale: Acceptance, Cost Ratio, and the Limits of Speedup

Stanford CS224N Default Project (inference-time optimization extension)

Amulya Parthasarathy
Department of Computer Science
Stanford University
amulyasp@stanford.edu

Abstract

Autoregressive Transformers generate text one token at a time, so decoding is often the latency bottleneck. This project studies two inference-time optimizations for GPT-2 on sonnet generation: key-value caching and speculative decoding. KV caching gives a clear positive result. It removes 70–77× of repeated prefix processing, but wall-clock speed improves by only about 1.3×, which shows that GPT-2-scale decoding is still dominated by per-step overhead. Speculative decoding is then implemented in both greedy and sampling-correct forms, with token-identical greedy checks and a four-stage correctness test suite for sampling. Across eight GPT-2-family draft-target pairs, no configuration beats the KV-cached baseline. The one clear stand out is DistilGPT2 → GPT-2 small, which reaches 74.9% greedy acceptance because DistilGPT2 was distilled from GPT-2 small. That alignment is target-specific: acceptance drops to 46.2% against GPT-2 medium and 32.3% against GPT-2 XL. Holding the target fixed at GPT-2 XL makes the tradeoff clear: cheaper drafts improve the wall-clock ratio, stronger drafts improve acceptance, and no pair gets enough of both at once. A seven-run distillation study helps explain why the other pairs do not close the gap. Training a student on only 131 sonnets improves acceptance only slightly, which points to a combination of limited data, exposure bias, and the architectural mismatch between student and teacher. The main result is a careful negative result: speculative decoding is correct at GPT-2 scale, but no existing GPT-2-family draft is both cheap enough and aligned enough to produce real wall-clock speedup. The DistilGPT2 result suggests that a purpose-built draft trained on sufficient data could change this outcome.

1 Key Information

- **External collaborators:** None.
- **Mentor:** Julie Kallini.
- **Sharing project:** Not shared with another course.
- **Late days:** None used.

2 Introduction

Autoregressive language models decode one token at a time. This makes inference slow because every new token depends on the full prefix. Two common ideas try to reduce this cost. KV caching avoids recomputing attention over the whole prefix. Speculative decoding goes further by asking a cheaper draft model to propose several tokens, then letting the target model verify them in one pass [1, 2].

This project asks a narrow question. For GPT-2 decoders [3] fine-tuned on sonnet generation, how much real speedup is possible on a single GPU from KV caching and speculative decoding, while keeping greedy generation exact and sampling distributionally correct? GPT-2 is useful here because it is small enough to fit cleanly inside the CS224N project codebase and easy to instrument carefully.

The scale difference matters because published speculative-decoding results typically use targets in the 7B–70B+ parameter range with purpose-built draft models of 1B–7B parameters, where the target-to-draft cost ratio is large enough for speculation to pay off [4]. GPT-2 lets the same idea be tested in a regime where every part of the system can be measured closely. If speculation still helps here, that would strengthen the case that it is a broadly useful decoding strategy. If it does not, the failure can reveal what conditions are actually needed for speedup.

The starting point is KV caching, which turns out to help much less than the raw token-count reduction suggests. Speculative decoding is then tested across eight GPT-2-family draft–target pairs in both greedy and sampling-correct modes, and a break-even analysis and a small distillation study explain where the expected speedup disappears. At GPT-2 scale, a draft model has to clear a much higher bar than it first appears. See Appendix A for additional plots and tables.

Deviations from proposal. The original proposal planned layer-drop draft models and a relaxed top- k acceptance rule. Layer-drop drafts produced a mismatch between truncated hidden states and the language-model head (8.4% acceptance), so the project had to pivot to architecturally self-consistent multi-size model pairs. The relaxed top- k rule was replaced by sampling-correct speculative decoding, which preserves the target distribution exactly [1, 2]. The project also adds several pieces of analysis not in the proposal, including a four-stage sampling correctness test suite, a break-even cost-ratio analysis, cross-GPU validation, and a seven-variant distillation study, all to find out what conditions the GPT-2 family would need to satisfy for speculative decoding to deliver real speedup.

3 Related Work

Transformers [5] are the standard architecture for modern language modeling, and GPT-2 [3] is a canonical decoder-only example. KV caching is a standard inference optimization that stores per-layer keys and values so the model does not recompute the full prefix at every time step [6].

Speculative decoding accelerates autoregressive generation by letting a smaller draft model propose several tokens and then letting the target model verify them in one pass. Chen et al. and Leviathan et al. give the standard accept–reject construction that preserves the target distribution exactly [1, 2]. At large scale, these methods can produce real speedups because the target model is expensive and a good draft can amortize the verification pass.

A more recent empirical study by Yan et al. shows that draft latency matters at least as much as draft quality, and that larger or more accurate drafts do not automatically give better throughput [4]. That result is directly relevant here. The GPT-2 family gives a controlled way to study the same tradeoff at smaller scale.

Knowledge distillation [7] trains a student to match a teacher. DistilGPT2 [8] is especially useful in this project because it is a professionally distilled draft from GPT-2 small. It acts as a strong positive control for alignment. If even that pair does not break even, then the bar for a naive student draft is likely much higher than it first appears.

Contributions and novelty. KV caching, speculative decoding, and knowledge distillation are established ideas. This project does not claim algorithmic novelty. The contribution is implementation and analysis. All code for KV caching, cache threading through GPT-2 layers, the speculative controller in greedy mode and sampling-correct mode, correctness checks, and the benchmark instrumentation was implemented from scratch in the CS224N GPT-2 codebase. No library-level speculative decoding routine such as HuggingFace’s `assistant_model` API was used. The greedy controller computes the target argmax at each verification position, accepts the longest prefix where draft and target agree, and replaces the first mismatch with the target token. The sampling-correct controller implements the accept–reject scheme of Chen et al. and Leviathan et al. (Eqs. (1)–(2)), with full pseudocode in Appendix A.1. A distillation framework was additionally developed with AI tool assistance (detailed in the collaboration statement) to diagnose low acceptance rates. The main novelty is therefore analytical. It is a detailed GPT-2-scale case study showing that speculative

decoding can be correct and still fail to speed up when the available draft models do not satisfy the joint acceptance and wall-clock cost-ratio conditions needed for break-even, and that limited distillation data is a primary reason the gap remains open.

4 Approach

4.1 Pipeline and baseline

The implementation was done in three phases. The starting point was baseline decoding without caching. The next phase was KV-cached decoding, followed by speculative decoding built on top of the KV-cached baseline. This matters because speculation needs to be compared against the strongest target-only baseline, not against an intentionally weak uncached loop.

The original generation code was also refactored into reusable parts for next-logit computation, logit post-processing (temperature scaling, top- p filtering), and token selection. This made it possible to share the same sampling and greedy logic across baseline decoding and speculative decoding. CUDA-synchronized timing and forward-pass counters were also added at the same time for instrumentation so that every result could be benchmarked using the same logic.

4.2 KV caching

KV caching stores attention keys and values at each transformer layer and appends only the new token at each step. For batch size B , heads H , head dimension D , and current sequence length T , each layer stores

$$K_{\text{cache}}, V_{\text{cache}} \in \mathbb{R}^{B \times H \times T \times D}.$$

At the next step, only the projections for the new token are computed and concatenated along the time axis. This required changes in the attention module, the GPT-2 block, and the model forward path. Correctness was checked with allclose logit comparisons and token-identical greedy generation.

4.3 Speculative decoding

Speculative decoding uses a draft model q and a target model p . Given a prefix, the draft proposes a block of K tokens (generated autoregressively through the draft model). The target then runs once on the prefix plus all K draft tokens, which produces the verification logits for each speculative position plus one extra bonus position.

Greedy mode. In greedy mode, a draft token is accepted if it matches the target argmax at that position. If the first mismatch happens at position j , then the first $j - 1$ draft tokens are kept and the token at position j is replaced by the target token. If all K draft tokens are accepted, the target also emits one bonus token from the next position. After every rejection, both draft and target caches are truncated to the committed prefix.

Sampling-correct mode. The sampling-correct path implements the standard accept-reject rule [1, 2]. For a draft token $x_i \sim q(\cdot \mid x_{<i})$, the token is accepted with probability

$$a_i = \min\left(1, \frac{p(x_i \mid x_{<i})}{q(x_i \mid x_{<i})}\right). \quad (1)$$

If a token is rejected, the corrected token is sampled from the residual distribution

$$r(x) \propto \max(0, p(x \mid x_{<i}) - q(x \mid x_{<i})). \quad (2)$$

This guarantees that the final output distribution matches the target exactly. A vectorized implementation gave about a 6–7% benchmark improvement over the original per-position loop, and both implementations were cross-validated against each other.

Correctness was established in four stages: a toy distribution test, self-draft validation, a real-model Monte Carlo test, and an end-to-end marginal comparison. See Appendix A.2 for details.

4.4 Distillation as a diagnostic tool

The core contributions of this project are KV caching and speculative decoding. When the initial benchmarks showed low acceptance across all independently fine-tuned GPT-2 pairs, DistilGPT2 provided a critical clue. This professionally distilled 82M model reached 74.9% acceptance against GPT-2 small, proving that distillation can produce the kind of draft–target alignment that speculation needs. That result raised a natural follow-up question: could a similar distillation be done for larger targets like GPT-2 large? A distillation framework was built to find out. Seven variants were tested, spanning soft KL alignment, hard cross-entropy, sequence-level distillation on teacher-generated rollouts, KL regularization, and multi-prompt data augmentation. The goal was not to propose a new distillation method, but to understand why the independently fine-tuned pairs showed low acceptance and whether naive distillation on a small dataset could close the gap. See Appendix A.7 for experimental details and per-variant results, and Section 6.5 for the main analytical conclusions.

4.5 Metrics and break-even analysis

The main inference metrics are acceptance rate, wall-clock speedup relative to the KV-cached baseline, and forward-pass counts for both target and draft models. The simplest break-even condition is

$$t + \frac{d}{\rho} < 1, \quad \text{equivalently} \quad \rho > \frac{d}{1-t}, \quad (3)$$

where t is target forwards per generated token, d is draft forwards per generated token, and ρ is the target-to-draft wall-clock cost ratio. This ignores controller overhead (cache management, accept/reject logic), so it is optimistic. If a pair fails this test, real speedup is not expected. Figure 2 (left) visualizes this boundary for all model pairs.

5 Experiments

5.1 Data and setup

The default project tasks are sentiment classification on SST and CFIMDB, paraphrase detection on Quora, and sonnet generation on Shakespeare. The extension in this report focuses on sonnet generation because speculation only helps when the generated sequence is long enough to amortize the draft work. The sonnet split contains 143 training examples. Twelve were held out for inference benchmarks, leaving 131 sonnets for the distillation experiments. Table 1 lists all model configurations.

Benchmarks were run on NVIDIA T4 (16 GB) and A100 (80 GB) GPUs. GPT-2 small and medium were fine-tuned on sonnets for 10 epochs with learning rate 2×10^{-5} . GPT-2 large and GPT-2 XL were trained and benchmarked on A100 because of memory limits. The main speculative-decoding sweeps used $K \in \{2, 4, 8\}$. Distillation experiments ran for up to 20 epochs with per-epoch acceptance evaluation and best-checkpoint selection. All speed numbers are relative to the KV-cached baseline.

Table 1: Model specifications.

Model	Layers	Hidden	Heads	Params	Role
DistilGPT2	6	768	12	82M	Draft
GPT-2 small	12	768	12	124M	Draft / Target
GPT-2 medium	24	1024	16	355M	Draft / Target
GPT-2 large	36	1280	20	774M	Target
GPT-2 XL	48	1600	25	1558M	Target

5.2 Evaluation methods

Three groups of metrics are used.

Correctness metrics. In greedy mode, speculative decoding must produce exactly the same tokens as KV-cached greedy decoding. Token identity was verified on all 12 held-out sonnet prompts for

every greedy benchmark. In sampling mode, correctness means that the final output distribution matches the target model. This was checked with four tests: toy distributions, self-draft validation, a real-model Monte Carlo test, and an end-to-end marginal comparison (Appendix A.2).

Efficiency metrics. Wall-clock speedup is the main efficiency metric. It is defined as KV-cached baseline time divided by speculative-decoding time for the same prompts. All timings use CUDA synchronization, which forces the CPU to wait for all GPU kernels to finish before reading the clock. Without this step, GPU kernel launches overlap with timing calls and the measured latency reflects launch overhead rather than actual computation. Supporting metrics are acceptance rate, target and draft forward passes per generated token, and latency in ms/token. These numbers feed into the break-even analysis in Eq. (3).

Downstream task metrics. The base models are also evaluated on the default CS224N project tasks. Sentiment uses accuracy, paraphrase detection uses accuracy on Quora dev and test, and sonnet generation uses ChrF. Because inference optimizations do not change model weights, these scores stay unchanged and simply confirm that the default-project baselines were met. Sonnet generation is the efficiency focus since it produces sequences long enough for speculation. Paraphrase detection produces only one or two tokens of output, so speculative decoding cannot help there.

5.3 Results

Default-project performance. Table 2 confirms that fine-tuned models meet default-project baselines.

Table 2: Downstream task performance.

Task	Metric	Result
SST sentiment	Accuracy	0.518
CFIMDB sentiment	Accuracy	0.984
Paraphrase detection	Accuracy (dev / test)	0.897 / 0.895
Sonnet generation	ChrF (dev / test)	44.302 / 44.131

KV caching. KV caching removes a large amount of repeated prefix work but gives only a modest wall-clock gain. Table 3 shows a 70–77× reduction in processed prefix tokens, but only 1.26–1.28× speedup.

Table 3: KV caching on GPT-2 small, T4. Prefix counts sum over 12 held-out prompts.

Mode	Prefix (no cache)	Prefix (cached)	Reduction	Time (no cache)	Time (cached)	Speedup
Greedy (EOS)	108,682	1,553	70×	12.66s	10.03s	1.26×
Fixed-128	148,992	1,926	77×	17.04s	13.35s	1.28×

The large gap between token reduction and wall-clock speedup shows that at GPT-2 scale, per-step overhead (kernel launches, memory transfers, framework bookkeeping) dominates over the attention computation saved by caching. This motivates speculative decoding as an attempt to reduce the *number* of serial target forward passes rather than the cost of each one.

Speculative decoding across model pairs. Table 4 reports greedy speculative decoding at $K=4$. Rows are grouped by target so the bottom three rows give a fixed-target GPT-2 XL comparison. All runs are token-identical to the KV-cached baseline. No pair reaches speedup above 1.0×. The one clear standout is DistilGPT2 → GPT-2 small. It reaches 74.9% acceptance because DistilGPT2 was distilled from GPT-2 small, so the draft and target distributions are unusually well aligned. That advantage does not carry over to other targets: DistilGPT2 acceptance drops to 46.2% against GPT-2 medium and 32.3% against GPT-2 XL. Across the independently fine-tuned GPT-2 pairs, acceptance ranges from 25.8% to 54.1%, with a non-monotonic pattern (small→XL at 37.5% exceeds small→large at 25.8%), but no pair satisfies both the acceptance and measured cost-ratio conditions needed for break-even. A full DistilGPT2 → GPT-2 small K-sweep is reported in Appendix Table 5.

Table 4: Greedy speculative decoding at $K=4$, grouped by target. Speedup is relative to the KV-cached baseline. All runs are token-identical to the KV-cached baseline. Measured ρ is the target-to-draft wall-clock cost ratio from self-draft timing runs (T4 values marked $\sim$ are approximate). Nominal ρ is the $L \times d^2$ FLOP estimate.

Draft $\rightarrow$ Target	Accept	Speedup	Measured ρ	Nominal ρ	GPU
DistilGPT2 $\rightarrow$ GPT-2 small	74.9%	0.81 $\times$	$\sim 2.0\times$	2.0 $\times$	T4
GPT-2 small $\rightarrow$ GPT-2 medium	54.1%	0.66 $\times$	$\sim 2.0\times$	3.6 $\times$	T4
DistilGPT2 $\rightarrow$ GPT-2 medium	46.2%	0.80 $\times$	$\sim 3.8\times$	7.1 $\times$	T4
GPT-2 medium $\rightarrow$ GPT-2 large	29.9%	0.40 $\times$	$\sim 1.5\times$	2.3 $\times$	T4
GPT-2 small $\rightarrow$ GPT-2 large	25.8%	0.52 $\times$	$\sim 3.0\times$	8.3 $\times$	T4
GPT-2 medium $\rightarrow$ GPT-2 XL	43.3%	0.58 $\times$	1.96 $\times$	4.9 $\times$	A100
GPT-2 small $\rightarrow$ GPT-2 XL	37.5%	0.73 $\times$	4.12 $\times$	17.4 $\times$	A100
DistilGPT2 $\rightarrow$ GPT-2 XL	32.3%	0.80 $\times$	7.36 $\times$	34.7 $\times$	A100

Fixed-target XL comparison. Fixing the target at GPT-2 XL gives the cleanest view of the acceptance–cost–ratio tradeoff. GPT-2 medium gives the highest acceptance at 43.3%, but it is itself too expensive as a draft, so the measured target-to-draft wall-clock ratio is only 1.96 $\times$ and speedup is just 0.58 $\times$. DistilGPT2 is much cheaper, so the measured ratio rises to 7.36 $\times$ and speedup gets closer to break-even at 0.80 $\times$, but acceptance falls to 32.3%. GPT-2 small sits between them at 37.5% acceptance, 4.12 $\times$ measured ratio, and 0.73 $\times$ speedup. This is the core tradeoff. Full fixed-target GPT-2 XL sweeps are in Appendix Table 7.

Distillation investigation. To understand why independently fine-tuned GPT-2 pairs show low acceptance, seven distillation variants were tested for GPT-2 small $\rightarrow$ GPT-2 large. The undistilled baseline is 18.56% acceptance. All seven variants stay in a narrow 19–22% band, with the best result at 22.0% from multi-prompt teacher rollouts with KL regularization. This small gain is the main result. Naive distillation on 131 sonnets does not come close to reproducing the alignment seen in DistilGPT2 $\rightarrow$ GPT-2 small. That gap points to three limits at once: too little data, exposure bias during free-running generation, and potential architectural mismatch between a 124M student and a 774M teacher. See Appendix Table 9 for the per-variant breakdown, and Section 6.5 for the analysis.

Figures. Figure 1 summarizes the benchmark results, and Figure 2 summarizes the break-even and distillation analysis.

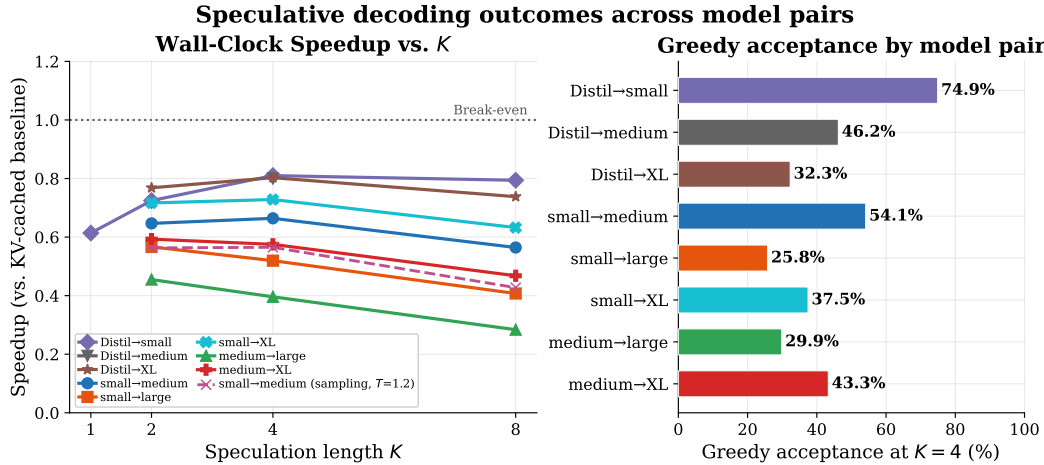


Figure 1: Main speculative-decoding results. Left: wall-clock speedup versus speculation length K for representative draft–target pairs. DistilGPT2 $\rightarrow$ GPT-2 small is the strongest positive control but remains below 1.0 $\times$, and the GPT-2 medium $\rightarrow$ GPT-2 XL boundary-case run also remains below break-even. Right: greedy acceptance at $K=4$ across model pairs. Professional distillation produces the strongest draft–target alignment, while fine-tuned GPT-2 pairs show non-monotonic acceptance.

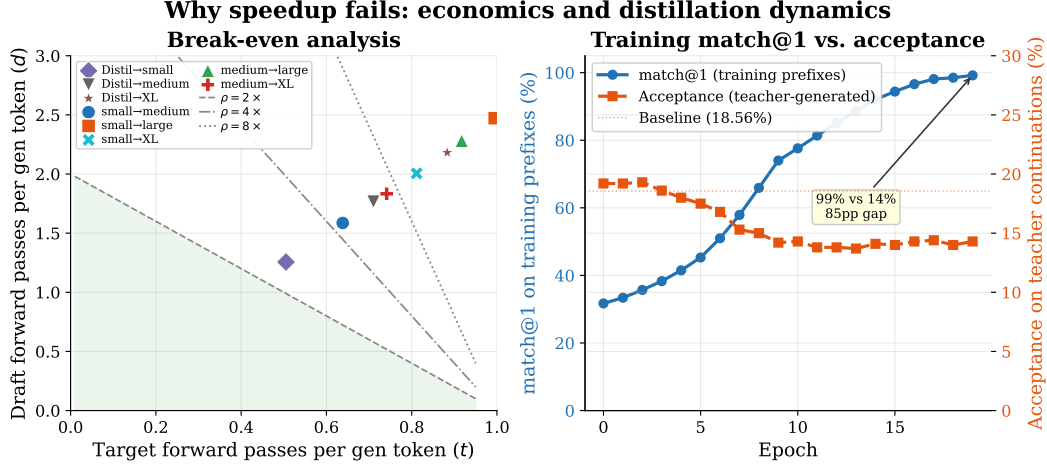


Figure 2: Main analytical results. Left: break-even analysis in the (t, d) plane, where t is target forward passes per generated token and d is draft forward passes per generated token. Lines show the cost ratio ρ required for break-even; the relevant comparison for each pair is the measured ρ from Table 4. All evaluated pairs remain in the no-speedup region. Right: the smoking-gun distillation result. Under pure hard-CE distillation, match@1 on training prefixes rises toward 99% while acceptance on teacher-generated continuations drops toward 14%, showing that teacher-forced agreement does not imply free-running speculative-decoding compatibility.

6 Analysis

6.1 Why KV caching helps less than the token count suggests

KV caching removes most of the repeated prefix computation, but GPT-2 small still speeds up by only about $1.3\times$. This means the bottleneck is not just attention FLOPs. At this scale, kernel launch cost, memory traffic, and framework overhead still matter a lot. That is why speculative decoding is attractive in principle. The goal is not only to make one target pass cheaper, but to reduce how many serial target passes are needed.

6.2 Acceptance and cost ratio must be good at the same time

For speculative decoding, the fixed-target GPT-2 XL comparison makes the main point concrete. For GPT-2 medium $\rightarrow$ GPT-2 XL at $K=4$, the measured forward rates are about $t \approx 0.74$ target passes and $d \approx 1.84$ draft passes per generated token. Eq. (3) then requires

$$\rho > \frac{1.84}{1 - 0.74} \approx 7.1.$$

The measured wall-clock ratio is only $1.96\times$, so negative speedup is expected. GPT-2 small improves the measured ratio to $4.12\times$, but acceptance drops to 37.5%. DistilGPT2 improves the measured ratio again to $7.36\times$ and reaches the best XL speedup at $0.80\times$, but that still misses break-even because acceptance falls to 32.3%. A 32.3% acceptance rate does not mean 32.3% faster decoding. It only means the draft agrees with the target often enough to save some target work. This failure is clear from the raw forward-pass counts. In the fixed-length benchmark (12 prompts $\times$ 128 new tokens each = 1536 generated tokens), the KV-cached baseline costs about 1536 GPT-2 XL forward passes. Speculative decoding reduces the target count to 1358 GPT-2 XL verification passes, saving 178 expensive target passes. But it also adds 3354 DistilGPT2 draft forward passes, which at a $7.36\times$ wall-clock cost ratio amount to $3354/7.36 \approx 456$ GPT-2-XL-equivalent passes. The saved target work (178) is much smaller than the extra draft work (456), so the speculative path is slower overall. Appendix A.10 gives the full derivation and connects to the break-even plot in Figure 2.

A counterexample from the independently fine-tuned GPT-2 family is also worth noting. Small $\rightarrow$ XL reaches higher acceptance than small $\rightarrow$ large (37.5% versus 25.8%), even though GPT-2 XL is the

larger target. This non-monotonic result shows that “bigger target equals worse acceptance” is too simple a summary for independently fine-tuned pairs. The consistent finding is that no GPT-2-family pair improves acceptance and cost ratio enough at the same time for speculation to break even.

6.3 Measured versus nominal cost ratios

The nominal $L \times d^2$ (layers $\times$ hidden-dimension²) FLOP ratios for GPT-2 XL against DistilGPT2, GPT-2 small, and GPT-2 medium are $34.7\times$, $17.4\times$, and $4.9\times$, respectively. But self-draft timing runs on A100 (Appendix Table 8), where each model is benchmarked alone in KV-cached baseline mode, give much smaller wall-clock ratios: $7.36\times$, $4.12\times$, and $1.96\times$. This compression explains why none of the XL runs breaks even, and is consistent with Yan et al., who show that draft latency scales more like layer depth than raw parameter-count ratios would suggest [4]. At GPT-2 scale, both draft and target models are dominated by memory movement and fixed per-step overhead, so the usable wall-clock cost ratio is far smaller than the nominal FLOP ratio. In larger-model settings, speculative decoding can still succeed not because these effects disappear, but because the target becomes much more expensive in wall-clock time and latency-optimized drafts create a larger practical target/draft asymmetry.

6.4 The DistilGPT2 story is real, but it is target-specific

DistilGPT2 remains the clearest positive control in the project. It was distilled from GPT-2 small, and that is exactly where it performs best. Against GPT-2 small, greedy sonnet acceptance reaches 74.9%. Against GPT-2 medium, it falls to 46.2%. Against GPT-2 XL, it falls again to 32.3%. A paraphrase-alignment probe on the Quora cloze task (Appendix A.9) shows the same pattern in a simpler one-token setting. Expected overlap is 0.972 for DistilGPT2 $\rightarrow$ GPT-2 small, 0.874 for DistilGPT2 $\rightarrow$ GPT-2 medium, and 0.756 for DistilGPT2 $\rightarrow$ GPT-2 XL. That confirms the good DistilGPT2 result is not generic but is strongest for the source target.

Sampling acceptance is consistently lower than greedy acceptance across all pairs because it penalizes probability-mass misallocation across the entire vocabulary, not just top-1 disagreement. See Appendix A.4 for a temperature sweep and detailed comparison.

6.5 Distillation diagnostics: exposure bias and the limits of naive distillation

The most revealing distillation result comes from the pure hard-CE run (Figure 2, right panel). Match@1 on teacher-forced prefixes rises steadily and eventually reaches 99.2%, but free-running acceptance on teacher-generated continuations falls to about 14%. That 85-percentage-point gap is direct evidence of exposure bias: the student learns to imitate the teacher on training prefixes, but not in the distribution of contexts that comes during actual speculative decoding. Sequence-level distillation on teacher-generated rollouts partially addresses this, and KL regularization stabilizes training dynamics (Appendix Figure 5, right), but the gains are incremental. The distillation training set of 131 sonnets (393 with augmentation) is orders of magnitude smaller than the full WebText corpus used to produce DistilGPT2, which is the most likely reason why seven different training strategies all land in the same narrow 19–22% band. DistilGPT2 is an even smaller model (82M) yet reaches 74.9% acceptance against GPT-2 small, because it was trained on a massive corpus with an architecture purpose-built to match the target.

7 Conclusion

This project implemented KV caching and speculative decoding from scratch in the CS224N GPT-2 codebase and tested them on sonnet generation. KV caching clearly helps, but the wall-clock gain is much smaller than the reduction in repeated token work. Speculative decoding is correct in greedy and sampling modes, but no evaluated GPT-2-scale pair beats the KV-cached baseline. The clearest positive result is DistilGPT2 $\rightarrow$ GPT-2 small. That pair has the highest acceptance because DistilGPT2 was distilled from GPT-2 small, so their token distributions are unusually well aligned. That alignment does not transfer to larger targets such as GPT-2 medium or GPT-2 XL. Holding the target fixed at GPT-2 XL sharpens this: GPT-2 medium offers the highest acceptance, DistilGPT2 offers the strongest measured wall-clock ratio, GPT-2 small lies between them, and none reaches $1.0\times$ speedup. Across the rest of the GPT-2 family, the available drafts never give both strong enough

acceptance and a large enough measured cost ratio at the same time. The distillation experiments explain why. Training a student on only 131 sonnets improves acceptance only slightly, even with sequence-level data and KL regularization. The evidence points to a combination of too little data, exposure bias, and the architectural mismatch between student and teacher. The DistilGPT2 result suggests that the missing ingredient is not speculative decoding itself, but a purpose-built draft trained on much more data to match a specific target. The main lesson is that speculative decoding is governed by the joint economics of *acceptance* and *measured wall-clock cost ratio*, not by either quantity alone. At GPT-2 scale, the available drafts fall short on one or both of these requirements, so correct speculation still fails to deliver real speedup. This limitation is specific to the regime studied here: all experiments use GPT-2-scale models (82M–1.56B parameters) and a 143-sonnet dataset, whereas published large-scale studies often report measured wall-clock cost ratios in the $10\text{--}20\times$ range, where speculative decoding becomes viable [4, 2]. The negative result is therefore still informative: it identifies the exact conditions needed for speedup and shows clearly why the GPT-2 family falls short of them.

References

- [1] Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. Accelerating large language model decoding with speculative sampling, 2023.
- [2] Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 19274–19286. PMLR, 23–29 Jul 2023.
- [3] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Technical Report, 2019.
- [4] Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 6460–6473, Albuquerque, New Mexico, April 2025. Association for Computational Linguistics.
- [5] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.
- [6] Reiner Pope, Sholto Douglas, Aakanksha Chowdhery, Jacob Devlin, James Bradbury, Jonathan Heek, Kefan Xiao, Shivani Agrawal, and Jeff Dean. Efficiently scaling transformer inference. In D. Song, M. Carbin, and T. Chen, editors, *Proceedings of Machine Learning and Systems*, volume 5, pages 606–624. Curan, 2023.
- [7] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015.
- [8] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. In *NeurIPS EMC² Workshop*, 2019.

Collaboration and AI tool use. ChatGPT and Claude were used to (i) explore background material on KV caching and speculative decoding, (ii) edit prose for clarity and grammar, and (iii) assist with developing the distillation training pipeline, which was not in the original project scope and was added to diagnose low acceptance rates. GitHub Copilot was used for coding assistance (e.g., autocomplete and refactoring suggestions). All three tools were also used at times during debugging to help interpret error messages and reason about potential fixes. All AI-suggested code was manually reviewed, edited, and tested before inclusion. All project ideas, technical decisions, experimental design, and analysis are the author’s own.

A Additional Results and Supporting Material

This appendix contains extra details, supporting plots, and validation data referenced from the main text. The main paper is written to stand on its own.

A.1 Speculative decoding algorithm

Algorithm 1 gives the complete pseudocode for one speculative round, covering both greedy and sampling-correct modes. This is the algorithm of Chen et al. [1] and Leviathan et al. [2]. The greedy path (lines 7–12) accepts draft tokens that match the target argmax and replaces the first mismatch. The sampling-correct path (lines 13–19) uses the probability-ratio acceptance test (Eq. (1)) and draws the corrected token from the residual distribution (Eq. (2)) on rejection. Both paths truncate draft and target KV caches to the committed prefix length after each round (line 21) and collect a bonus token when all K drafts are accepted (line 22).

Algorithm 1 One speculative decoding round (greedy or sampling-correct)

Require: prefix tokens $x_{1:t}$, draft model q , target model p , speculation length K , mode $\in \{\text{greedy, sample}\}$

Ensure: accepted tokens appended to prefix, both KV caches updated

- 1: Draft K tokens autoregressively: $d_k \sim q(\cdot \mid x_{1:t}, d_{1:k-1})$ for $k = 1 \dots K$
- 2: Run target once on $[x_{1:t}, d_{1:K}]$ to get logits at all $K+1$ positions
- 3: Apply warping (temperature, top- p) to both draft and target logits
- 4: $n_{\text{accepted}} \leftarrow 0$
- 5: **for** $k = 1$ to K **do**
- 6: **if** mode = greedy **then**
- 7: **if** $d_k = \arg \max p(\cdot \mid x_{1:t}, d_{1:k-1})$ **then**
- 8: $n_{\text{accepted}} \leftarrow n_{\text{accepted}} + 1$ {draft matches target argmax}
- 9: **else**
- 10: Replace d_k with $\arg \max p(\cdot \mid x_{1:t}, d_{1:k-1})$; **break**
- 11: **end if**
- 12: **else**
- 13: {sampling-correct mode}
- 14: Draw $u \sim \text{Uniform}(0, 1)$
- 15: **if** $u < \min(1, p(d_k)/q(d_k))$ **then**
- 16: $n_{\text{accepted}} \leftarrow n_{\text{accepted}} + 1$ {accept per Eq. (1)}
- 17: **else**
- 18: Sample replacement from $r(x) \propto \max(0, p(x) - q(x))$; **break** {Eq. (2)}
- 19: **end if**
- 20: **end if**
- 21: **end for**
- 22: Truncate draft and target KV caches to $t + n_{\text{accepted}} + 1$
- 23: **if** $n_{\text{accepted}} = K$ **then**
- 24: Emit bonus token from target logits at position $K+1$
- 25: **end if**

A.2 Sampling correctness validation

Greedy correctness is verified by direct token comparison: every greedy speculative run must produce tokens identical to the KV-cached baseline, as confirmed across all 12 held-out sonnets for every configuration. Sampling correctness requires statistical validation because the output is stochastic, so four complementary tests were used.

1. **Toy distribution test.** Five hand-built (p, q) pairs with vocabulary sizes from 3 to 50 were sampled 200K–500K times each. Maximum marginal error was below 0.002.
2. **Self-draft validation.** Using GPT-2 small as both draft and target gave 100% acceptance. Greedy generation was token-identical to the KV-cached baseline and sampling reduced to ordinary target sampling.

3. **Real-model Monte Carlo test.** Using real draft and target logits at a single position, the accept–reject algorithm was simulated 50K times on the extracted distributions (no additional model forward passes) and the empirical token-frequency histogram was compared against the target distribution. The metrics were maximum absolute difference and total variation distance. Both the loop and vectorized implementations passed ($\max |\Delta p| < 0.002$ on sonnet prompts, < 0.010 on general prompts), confirming that accepted tokens follow the target distribution rather than the draft distribution.
4. **End-to-end marginal comparison.** Over 2K full generations, the maximum absolute difference between speculative and KV-cached per-position marginals stayed below 0.03.

A.3 DistilGPT2 $\rightarrow$ GPT-2 small K-sweep

Table 5: DistilGPT2 $\rightarrow$ GPT-2 small, greedy K-sweep on A100 and T4. All runs token-identical to the KV-cached baseline.

K	Accept	Speedup (A100)	Speedup (T4)	Target fwd	Draft fwd
1	90.2%	0.62 $\times$	0.61 $\times$	1620	1620
2	84.6%	0.80 $\times$	0.72 $\times$	1148	1719
4	74.9%	0.83 $\times$	0.81 $\times$	776	1930
8	58.4%	0.81 $\times$	0.79 $\times$	554	2452

$K=4$ gives the best speedup even though acceptance is lower than at $K=1$ or $K=2$, because the reduction in target forward passes (from 1620 to 776) more than compensates for the increase in draft forward passes (from 1620 to 1930). At $K=8$, acceptance drops far enough that draft overhead outweighs the further target reduction.

A.4 Greedy versus sampling acceptance

Greedy acceptance measures top-1 agreement between draft and target ($\arg\max q = \arg\max p?$), while sampling acceptance measures distributional overlap via Eq. (1). Sampling acceptance is consistently lower. For small $\rightarrow$ medium at $K=4$, sampling acceptance is 38.4% versus 54.1% greedy, because sampling penalizes probability-mass misallocation across the entire vocabulary. A near-greedy temperature test ($T=0.001$) on DistilGPT2 $\rightarrow$ GPT-2 small confirms convergence: sampling acceptance reaches 74.1%, within 0.8 percentage points of greedy (74.9%), validating that the sampling path correctly degenerates toward greedy as temperature approaches zero (Table 6).

Table 6 shows a counterintuitive result: higher temperature yields higher sampling acceptance in the DistilGPT2 pair, because flattening both distributions toward uniform makes probability ratios $p(x)/q(x)$ closer to 1. This reinforces that sampling acceptance and greedy acceptance measure fundamentally different properties, distributional similarity versus mode agreement, which is an important distinction for distillation where KL-based training optimizes distributional similarity while greedy decoding depends on mode agreement.

Table 6: DistilGPT2 $\rightarrow$ GPT-2 small (sampling, $K=4$, top- $p=0.9$). Greedy shown for reference.

Strategy	Accept (A100)	Accept (T4)	Speedup (A100)	Speedup (T4)
Greedy	74.9%	74.9%	0.83 $\times$	0.81 $\times$
Sample $T=0.001$	74.1%	74.1%	0.76 $\times$	0.87 $\times$
Sample $T=0.8$	42.2%	41.4%	0.55 $\times$	0.54 $\times$
Sample $T=1.0$	41.6%	44.5%	0.55 $\times$	0.57 $\times$
Sample $T=1.2$	46.6%	48.7%	0.59 $\times$	0.60 $\times$

A.5 Fixed-target GPT-2 XL sweeps

Table 7: Greedy sweeps with GPT-2 XL as the fixed target on A100.

Draft	K	Accept	Speedup	Target fwd	Draft fwd	Measured ρ
GPT-2 medium	2	58.9%	$0.59\times$	1424	2129	$1.96\times$
GPT-2 medium	4	43.3%	$0.58\times$	1140	2819	$1.96\times$
GPT-2 medium	8	27.3%	$0.47\times$	990	4342	$1.96\times$
GPT-2 small	2	51.9%	$0.72\times$	1520	2272	$4.12\times$
GPT-2 small	4	37.5%	$0.73\times$	1246	3079	$4.12\times$
GPT-2 small	8	23.1%	$0.63\times$	1106	4852	$4.12\times$
DistilGPT2	2	47.7%	$0.77\times$	1582	2366	$7.36\times$
DistilGPT2	4	32.3%	$0.80\times$	1358	3354	$7.36\times$
DistilGPT2	8	19.4%	$0.74\times$	1230	5410	$7.36\times$

This sweep is the clearest summary of the results. Moving from medium to small to DistilGPT2 steadily improves the measured target-to-draft cost ratio, but acceptance falls in the opposite direction. Within each draft group, $K=4$ consistently gives the best or near-best speedup, matching the pattern from the DistilGPT2 K -sweep (Table 5). At $K=2$, acceptance is high but each round amortizes only two draft tokens. At $K=8$, acceptance drops severely (27.3%, 23.1%, 19.4% for medium, small, and DistilGPT2 respectively), and the excess draft computation negates the target savings. The fact that the optimal K is stable across drafts with very different acceptance levels suggests that the tradeoff between target amortization and draft overhead is governed more by the speculation mechanism itself than by the specific draft–target pair.

A.6 Self-draft sanity runs and effective wall-clock ratio

Table 8: Self-draft greedy runs at $K=1$ on A100. These confirm controller correctness (100% acceptance, approximately $0.5\times$ speedup as expected when doubling work) and provide direct estimates of per-token wall-clock costs.

Pair	Accept	Speedup	Baseline ms/tok	Spec. ms/tok
DistilGPT2 $\rightarrow$ DistilGPT2	100%	$0.51\times$	4.2	8.3
GPT-2 small $\rightarrow$ GPT-2 small	100%	$0.49\times$	7.5	15.1
GPT-2 medium $\rightarrow$ GPT-2 medium	100%	$0.50\times$	15.8	31.5
GPT-2 XL $\rightarrow$ GPT-2 XL	100%	$0.49\times$	30.9	62.5

These runs imply measured wall-clock ratios of $30.9/4.2 \approx 7.36\times$ for XL/DistilGPT2, $30.9/7.5 \approx 4.12\times$ for XL/small, and $30.9/15.8 \approx 1.96\times$ for XL/medium. The approximately $0.5\times$ speedup across all four model sizes also serves as a sanity check on controller overhead. Self-draft at $K=1$ doubles the model computation (one draft forward plus one target forward instead of one baseline forward), so the expected speedup is exactly $0.5\times$ if the controller adds no overhead. The measured values ($0.49\text{--}0.51\times$) confirm that speculative-decoding bookkeeping (cache management, accept/reject logic, tensor slicing) adds negligible wall-clock cost relative to the model forward passes themselves.

A.7 Distillation experiment details

Seven distillation variants were tested for GPT-2 small (124M) as draft against GPT-2 large (774M) as teacher, using greedy $K=4$ acceptance as the evaluation metric. Four variants used ground-truth sonnet training data, and three used teacher-generated rollouts. Table 9 gives the per-variant breakdown. All ground-truth variants peaked at epoch 1–2 and degraded with further training. Sequence-level distillation on teacher-generated rollouts partially addressed this by training on the distribution of contexts that arise during free-running generation. Adding KL regularization (70% weight) stabilized training dynamics and prevented the peak-then-degrade pattern, allowing

acceptance to climb steadily through all 10 epochs. Multi-prompt augmentation (393 rollouts from 1-, 2-, and 3-line prompts) provided $3\times$ the original data and achieved the best result at 22.0%.

Table 9: Distillation results for GPT-2 small $\rightarrow$ GPT-2 large, greedy $K=4$. Acceptance measured on a teacher-generated evaluation protocol.

Variant	Config	Best accept	Epoch	Δ
Baseline, no distill	—	18.56%	—	—
<i>Ground-truth training data</i>				
Soft KL	$T=2, \alpha=0.7$	19.8%	1	+1.2
Hard CE only	$T=1, \beta=1$	19.3%	2	+0.7
Soft KL + hard CE	$T=2, \alpha=0.7, \beta=1$	20.3%	1	+1.8
Sharp KL only	$T=1, \alpha=1$	19.9%	2	+1.3
<i>Teacher-generated rollouts</i>				
Seq-level CE	131 rollouts	20.7%	2	+2.1
Seq-level + KL reg	131 rollouts	21.1%	9	+2.5
Multi-prompt + KL reg	393 rollouts	22.0%	7	+3.4

The Epoch column reveals two distinct training regimes. All four ground-truth variants peak at epoch 1–2 and then degrade, which is the memorization signature confirmed by the KL divergence growth in Figure 5 (right panel). In contrast, the KL-regularized variants peak at epoch 7–9, showing that regularization fundamentally changes training dynamics by preventing the student from overfitting to the teacher’s choices on a small number of training sonnets. Despite this improvement in training stability, the best regularized result (22.0%) is still only 3.4 percentage points above the undistilled baseline, confirming that training methodology alone cannot close the gap given the limited data and architectural mismatch.

A.8 Supporting figures

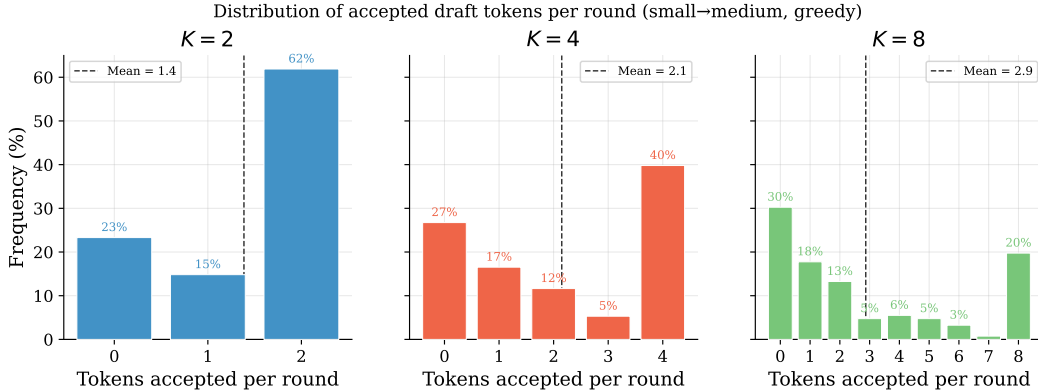


Figure 3: Accepted-length distributions for $K \in \{2, 4, 8\}$ (GPT-2 small $\rightarrow$ medium, greedy). The tall bar at accepted length 0 (full rejection at position 1) grows with K .

The accepted-length distributions illustrate why larger K does not automatically help. At $K=2$, 62% of rounds accept all draft tokens (the rightmost bar), so most rounds are productive. At $K=4$, full acceptance drops to 40% and the zero-acceptance bar (draft rejected at position 1, wasting the entire round) grows to 27%. At $K=8$, only 20% of rounds accept all eight tokens and 30% reject immediately. The mean accepted length grows from 1.4 to 2.1 to 2.9, which shows diminishing returns per additional draft token. Each extra unit of K buys less than one extra accepted token on average, while the draft computation cost grows linearly in K .

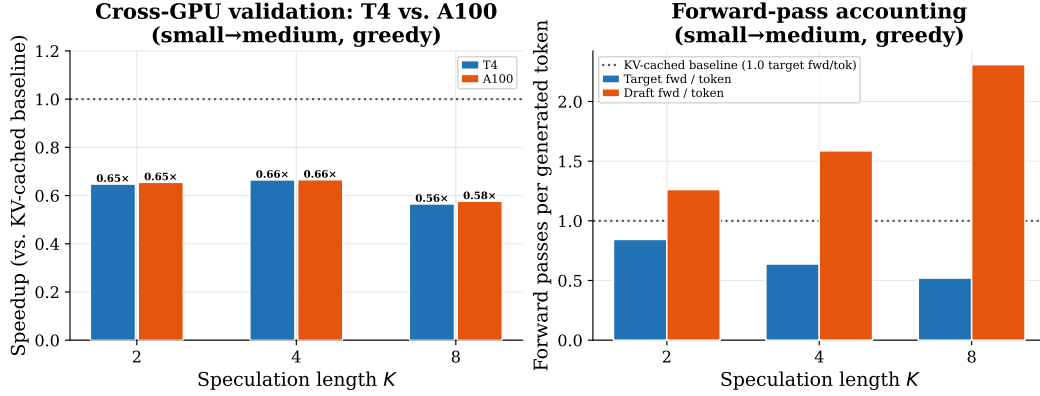


Figure 4: Left: cross-GPU comparison (T4 vs. A100) showing near-identical speedup ratios, which supports an algorithmic bottleneck rather than a hardware one. Right: forward-pass accounting showing target and draft forwards per token versus K , which helps interpret the break-even condition in Eq. (3).

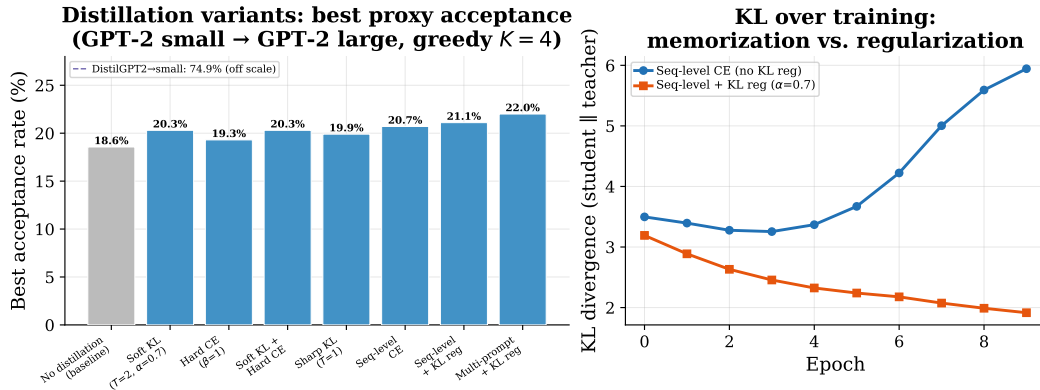


Figure 5: Left: acceptance across all seven distillation variants plus baseline (18.56%). All variants fall in a narrow 19–22% band. Right: KL divergence across training epochs. Pure sequence-level CE (blue) shows KL growth from 3.50 to 5.94, indicating memorization. Adding KL regularization (orange) stabilizes training and prevents the peak-then-degrade pattern.

A.9 Paraphrase alignment probe

Sonnet acceptance is measured across hundreds of autoregressive positions, making it hard to isolate where draft–target agreement breaks down. To address this, a paraphrase alignment probe was developed as a simpler diagnostic. Each of the 40,429 examples in the Quora paraphrase dev split is formatted as a cloze-style prefix ending at the yes/no decision point. Both the draft and target models (pretrained, not fine-tuned) produce a probability distribution over the next token at that position, isolating distributional agreement at a single position without the complexities of multi-step generation.

Three metrics are computed from these distributions. $\mathbb{E}[\text{accept}]$ is the expected single-token acceptance rate under the sampling-correct scheme (Eq. (1)), which directly measures how often a speculative draft token would survive verification. KL divergence measures the overall distance between the two distributions. Greedy match is simply whether draft and target agree on the argmax token. In the scatter plots (Figure 6), each dot is one example, with the draft probability of “yes” on one axis and the target probability on the other. Points near the diagonal indicate strong agreement.

The Pearson r value measures whether the two models at least rank examples similarly, even if they disagree on absolute probabilities.

Table 10 and Figures 6–7 show the results for all eight pairs. The ordering broadly supports the sonnet findings, but there is an important limitation. The probe does not perfectly predict sonnet acceptance (compare the $\mathbb{E}[\text{accept}]$ and Sonnet accept columns in Table 10). For example, small→XL has the worst paraphrase overlap (0.729) but better sonnet acceptance (37.5%) than small→large (0.787 overlap, 25.8% sonnet acceptance). This divergence arises because the probe measures agreement at a single yes/no position on pretrained models for paraphrase prediction, while sonnet acceptance measures agreement across hundreds of positions on fine-tuned models generating in a specific domain. The probe is therefore best understood as a supporting diagnostic of draft–target compatibility, not as a quantitative predictor of generation-time acceptance.

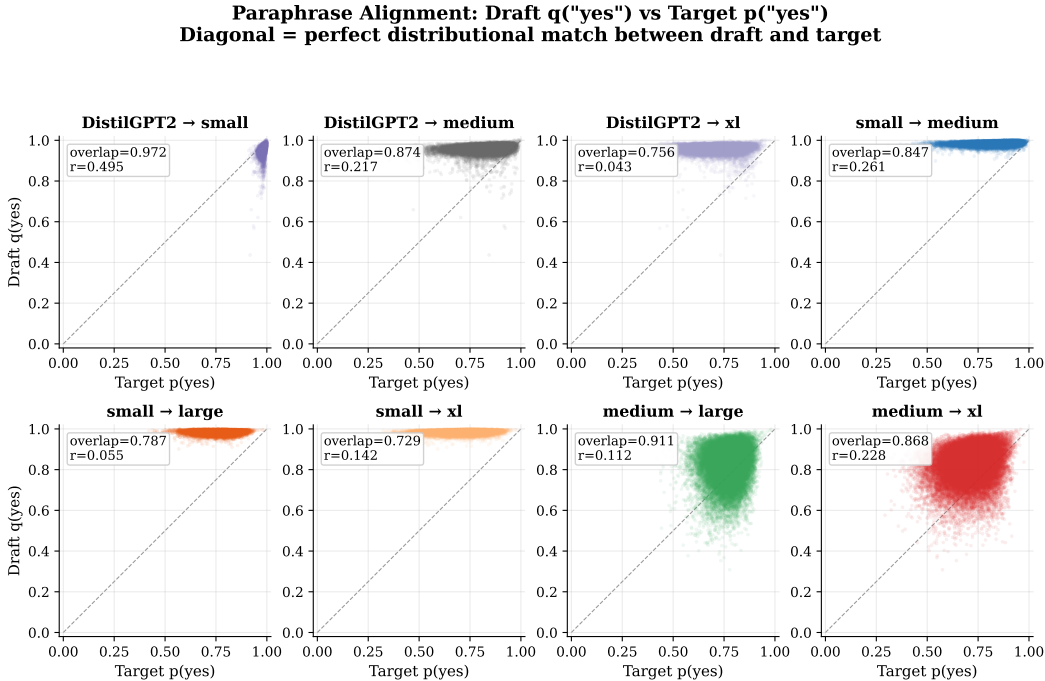


Figure 6: Paraphrase alignment probe: per-example draft probability $q(\text{"yes"})$ versus target probability $p(\text{"yes"})$ on the Quora cloze dev split (40,429 examples, pretrained models). The diagonal indicates perfect distributional agreement. *Overlap* is the expected $K=1$ sampling acceptance, and r is the Pearson correlation across examples. DistilGPT2 panels show that professional distillation yields unusually tight alignment to the source target GPT-2 small, with weaker alignment to GPT-2 medium and GPT-2 XL.

The scatter plots reveal three distinct alignment regimes. DistilGPT2 → GPT-2 small (overlap 0.972, $r=0.495$) shows tight clustering along the diagonal, confirming that professional distillation produces unusually strong per-example agreement with the source target. At the other extreme, DistilGPT2 → GPT-2 XL (overlap 0.756, $r=0.043$) and small → large (overlap 0.787, $r=0.055$) show wide scatter with near-zero correlation, meaning the two models barely agree on which examples are likely paraphrases. The remaining pairs fall between these extremes, with moderate overlap and low-to-moderate correlation. The correlation values are particularly informative: even when two models have reasonable overlap (because both assign high probability to “yes” on average), low r means they disagree about *which* examples are confident paraphrases, which would cause frequent position-by-position mismatches during multi-step speculative decoding.

Table 10: Paraphrase alignment on the Quora cloze task (pretrained models, dev split, 40,429 examples). $\mathbb{E}[\text{accept}]$ is the expected $K=1$ sampling acceptance under the sampling-correct scheme, greedy match is top-1 agreement on the yes/no token, and sonnet accept is greedy $K=4$ acceptance from the sonnet benchmarks. The probe measures draft–target distributional agreement, not end-to-end speed.

Pair	$\mathbb{E}[\text{accept}]$	KL	Greedy match	Sonnet accept
DistilGPT2 $\rightarrow$ GPT-2 small	0.972	0.013	1.000	74.9%
GPT-2 medium $\rightarrow$ GPT-2 large	0.911	0.045	0.998	29.9%
DistilGPT2 $\rightarrow$ GPT-2 medium	0.874	0.136	0.998	46.2%
GPT-2 medium $\rightarrow$ GPT-2 XL	0.868	0.083	0.985	43.3%
GPT-2 small $\rightarrow$ GPT-2 medium	0.847	0.290	0.998	54.1%
GPT-2 small $\rightarrow$ GPT-2 large	0.787	0.457	0.999	25.8%
DistilGPT2 $\rightarrow$ GPT-2 XL	0.756	0.364	0.987	32.3%
GPT-2 small $\rightarrow$ GPT-2 XL	0.729	0.647	0.987	37.5%

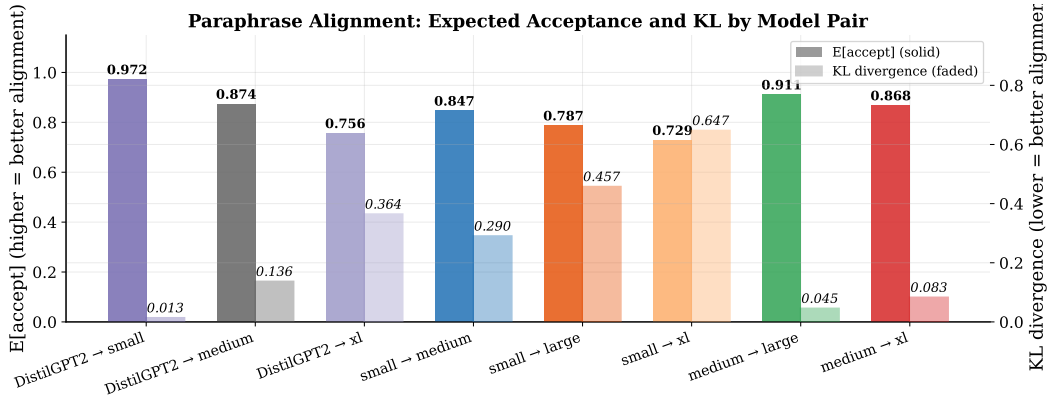


Figure 7: Paraphrase alignment probe on the Quora cloze task (pretrained models, dev split, 40,429 examples). Solid bars show expected acceptance $\mathbb{E}[\min(1, p(x)/q(x))]$ for the yes/no next-token distribution, and faded bars show KL divergence. DistilGPT2 $\rightarrow$ GPT-2 small shows the strongest alignment, and the three DistilGPT2 rows illustrate the target-specific nature of professional distillation: overlap drops from 0.972 against GPT-2 small to 0.874 against GPT-2 medium and 0.756 against GPT-2 XL.

A.10 Worked example: why DistilGPT2 $\rightarrow$ GPT-2 XL still fails

This section walks through the cost accounting for DistilGPT2 $\rightarrow$ GPT-2 XL at greedy $K=4$, first in raw forward-pass counts and then expressed as averages per generated token, which is the form used in the break-even plot (Figure 2, left).

Benchmark setup. The speed comparison uses a *fixed-length* benchmark. Each of the 12 held-out sonnet prompts is forced to generate exactly 128 new tokens (EOS is suppressed as a stopping condition), so both the KV-cached baseline and speculative decoding produce the same output workload:

$$N_{\text{tok}} = 12 \times 128 = 1536 \text{ generated tokens.}$$

This isolates decoding efficiency from EOS-length variation and makes the forward-pass and speedup comparisons interpretable.

Recorded metrics. The benchmark records four metrics for this model pair (Table 7, $K=4$ row). The first three were recorded during the speculative decoding run itself, while the fourth (ρ) was measured separately from standalone baseline runs.

- **Acceptance** = 32.3%: the fraction of draft-proposed tokens that matched the GPT-2 XL argmax and were kept.
- **Target forward passes** = 1358: the total number of GPT-2 XL verification calls across all 12 prompts.
- **Draft forward passes** = 3354: the total number of DistilGPT2 next-token calls across all 12 prompts. Each call produced one proposed token, so the draft proposed 3354 tokens in total. Only 32.3% of those proposals survived the target’s verification.
- **Measured $\rho = 7.36\times$** : the ratio of per-token wall-clock costs between GPT-2 XL and DistilGPT2, computed from the self-draft timing runs on A100 (Table 8). Each model was run alone in KV-cached baseline mode on the same 12-prompt benchmark, with no speculative decoding involved, giving average per-token latencies of 30.9 ms/tok for GPT-2 XL and 4.2 ms/tok for DistilGPT2, so $\rho = 30.9/4.2 \approx 7.36$. This means that, in the KV-cached baseline benchmark, generating one token with GPT-2 XL takes about 7.36 times as long as generating one token with DistilGPT2. This ratio is then used as a proxy for the relative cost of target and draft forward passes in the break-even analysis.

These counts represent the number of times each model was called (one call per forward pass), which differs from the number of output tokens (1536). The target count (1358) is less than 1536 because one target verification pass can sometimes accept several draft proposals at once and produce multiple output tokens in a single speculative round (and, when all K are accepted, add a bonus token). The draft count (3354) is greater than 1536 because the draft proposes up to K tokens per round, and many proposals do not survive verification.

Raw-cost view: saved versus paid. In the KV-cached baseline, generating 1536 tokens costs about 1536 GPT-2 XL forward passes (approximately one per output token). Speculative decoding saves some of those expensive passes:

$$1536 - 1358 = 178 \text{ GPT-2 XL passes avoided.}$$

However, it also adds 3354 DistilGPT2 passes. Converting draft work into GPT-2-XL-equivalent cost using $\rho = 7.36$ gives

$$\frac{3354}{7.36} \approx 456 \text{ target-equivalent passes.}$$

So the total speculative cost is

$$1358 + 456 = 1814 \text{ target-equivalent passes,}$$

compared with 1536 for the baseline. The speculative path costs about $1814/1536 \approx 1.18$ times as much as the baseline, predicting a speedup of about $1536/1814 \approx 0.85\times$. The measured speedup is slightly worse at $0.80\times$ because this simplified model ignores some runtime overhead and treats per-pass costs as constant.

Per-token view: connection to the break-even plot. The same arithmetic can be expressed as averages per generated token by dividing the forward-pass counts by 1536:

$$t = \frac{1358}{1536} = 0.884, \quad d = \frac{3354}{1536} = 2.184.$$

So on average the target needed 0.884 forward passes per output token (barely less than the baseline’s 1.0), and the draft needed 2.184 forward passes per output token. Substituting into the break-even condition from Eq. (3):

$$t + \frac{d}{\rho} = 0.884 + \frac{2.184}{7.36} \approx 1.18 > 1.$$

This is the same 18% overshoot as in the raw-cost view. Equivalently, this operating point would need

$$\rho > \frac{2.184}{1 - 0.884} \approx 18.8\times$$

to break even, far above the measured $7.36\times$.

Why low acceptance hurts in two ways at once. With only 32.3% acceptance, the target still has to verify frequently ($t = 0.884$, close to 1.0), so the expensive target savings are small. At the same time, the draft has to propose many tokens that do not survive ($d = 2.184$, well above 1.0), so the cheap draft work accumulates into a significant cost. The result is that the saved GPT-2 XL work (178 passes) is smaller than the extra draft work (456 target-equivalent passes), and the pair remains below break-even despite the favorable $7.36\times$ wall-clock ratio.

How to read the break-even plot. Figure 2 (left) shows each model pair as a point in the (t, d) plane. The three dashed lines mark the break-even boundary

$$d = \rho(1 - t)$$

for guide values $\rho = 2\times$, $4\times$, and $8\times$. Conceptually, a pair can achieve speedup only if its point falls below the boundary corresponding to its measured ρ . In the current plot, only the $\rho = 2\times$ region is shaded; the $\rho = 4\times$ and $\rho = 8\times$ lines are shown as visual guides. The actual break-even boundary for DistilGPT2 $\rightarrow$ GPT-2 XL lies slightly below the $\rho = 8\times$ guide line, so being well above that line confirms the pair is far from the speedup region.

A.11 Project progression

The project progressed in four stages.

1. Baseline decoding and KV caching were implemented first, which established the KV-cached baseline that speculation had to beat.
2. Speculative decoding was then added in greedy and sampling-correct forms, along with the four-stage correctness checks.
3. Distillation experiments were added after the first benchmarks showed that independently fine-tuned pairs had low acceptance. This stage produced the match@1 versus acceptance smoking-gun result.
4. GPT-2 XL experiments were added last. Those runs completed the family sweep and made the fixed-target XL tradeoff much clearer.