T2D: TARGET-TO-DRAFT MODEL PRUNING & ADAPTATION FOR EFFICIENT SPECULATIVE DECODING

Anonymous authors
Paper under double-blind review

ABSTRACT

Speculative decoding significantly accelerates autoregressive text generation in Large Language Models (LLMs), but its efficacy depends largely on two key factors: the distributional alignment between the *draft* and the *target* model, and the *draft* model's latency during inference. However, existing approaches typically utilize independently pretrained small draft models or seek to prune the target model to a manually chosen architecture, thereby failing to explicitly balance the trade-off between draft model alignment and token generation efficiency. To address this gap, we formulate draft model design as an *optimization* problem and propose T2D, a framework that constructs draft models directly from the target model using *latency-aware structured pruning* and *knowledge distillation*. In this manner, T2D obtains draft models that are jointly optimized for both measured inference latency and cross-entropy loss on the target model's responses, resulting in superlative speculative decoding performance. Through extensive experiments on multi-domain speculative decoding benchmarks and across several target model families of differing scale, we demonstrate how T2D prunes up to 90% of the target model to obtain highly performant draft models compared to competitive baselines. Moreover, draft models obtained via T2D can also be utilized by other speculative decoding token-alignment and knowledge distillation methods in a plug-and-play fashion, leading to further gains. Our findings underscore the importance of jointly navigating the draft model architecture-alignment frontier to unlock the full potential of speculative decoding for LLMs.

1 INTRODUCTION

Large Language Models (LLMs) have become the foundation of many modern AI applications, including chat assistants, code generation tools, and document summarization systems (Zhao et al., 2026; Naveed et al., 2025). As modern LLMs consist of billions of parameters, *inference efficiency* has become an important consideration alongside ensuring high-quality response generation (Zhou et al., 2024b). At the same time, improving response quality often requires larger models, making it challenging to maintain *low latency* and reasonable *serving costs* due to the ever increasing parameter scale (Lopez-Lira & Tang, 2026). This challenge has motivated a growing body of research on reducing the inference latency of autoregressive text generation in LLMs.

Speculative decoding (Leviathan et al., 2023) seeks to address this challenge without compromising output quality by incorporating a smaller model into the text generation process. Instead of generating every token with a large and high-performing model, a low latency model, known as the *draft* model, first generates a short sequence of 4-5 tokens (Leviathan et al., 2023; Chen et al., 2023). The larger original model, also known as the *target* or *verifier* model, then verifies the drafted sequence in a single forward pass and accepts only the subsequently matching tokens. Since most tokens are generated by the low latency *draft* model, and the larger model is used only for verification, speculative decoding can significantly reduce the latency of autoregressive decoding while preserving the output quality guaranteed by the *verifier*.

However, the efficiency of speculative decoding is inherently dependent on *how fast and accurately* the draft model can generate the draft sequence. If the draft model has low latency but frequently proposes incorrect tokens, most of its predictions will be rejected by the verifier, and both the draft and verifier models will need to be invoked several times over to complete a sequence, resulting in

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

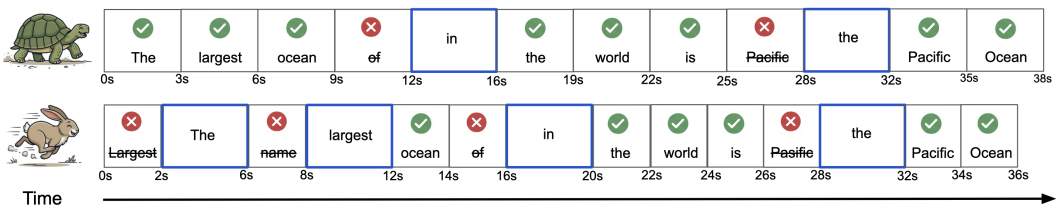

Figure 1: Example speculative-decoding traces for the same output sequence using draft models with different latencies and acceptance rates. The top row illustrates that a *more accurate* but *slower* draft model can result in *longer overall decoding* time, highlighting the *trade-off* between draft-model latency and acceptance.

slower text generation. Conversely, a highly accurate but computationally expensive draft model may offset the efficiency gains from integrating the smaller draft model into the decoding process (we visualize this trade-off in Figure 1). Thus, a *low latency architecture and target-draft agreement* are both important considerations for effective draft-model construction.

Existing approaches to draft-model construction often treat architecture design and target-model alignment as separate stages. Some methods utilize an existing, independently trained small language model and adapt it to the target model through additional training or distillation (Leviathan et al., 2023; Hu et al., 2026). However, aligning a separately trained small model with a large target model is challenging because independently trained models have distinct output distributions, making post-hoc alignment inherently difficult (Li et al., 2025). To address the challenge of aligning the draft model's output distribution with the distribution of the target model, recent work has explored the idea of constructing sparse draft models by combining weight-level *unstructured* pruning (Sun et al., 2024a; Frantar & Alistarh, 2023) of the target model with self-distillation, while largely preserving the original architecture (Lasby et al., 2025). Here, less important weights are masked to produce a sparse model while preserving the original architecture. This approach can lead to better target-draft alignment because the draft model is derived directly from the *target* model itself. However, unstructured pruning cannot provide latency benefits as irregular weight sparsity is difficult to exploit efficiently with conventional dense matrix-multiplication kernels in current Graphics Processing Unit (GPU) hardware (Rath & Maliakkal, 2026). Meaningful latency improvements instead require *structured pruning*, seeking to remove entire architectural components, such as layers or contiguous weight blocks (Men et al., 2025).

More recent work therefore proposes constructing draft models by structurally pruning the target model toward a manually chosen architecture to obtain a sufficiently small draft model (Wei et al., 2026; Yan et al., 2025). Despite their differences, these methods share a common limitation: they *manually specify* the draft architecture before considering the trade-off between decoding efficiency and alignment with the target model. As a result, the resulting draft model is not explicitly optimized for speculative decoding performance. In practice, a design that achieves strong alignment may be unnecessarily expensive in terms of inference latency, while a highly efficient design may suffer from low alignment with the target model. Consequently, the resulting draft model may be far from the Pareto-optimal balance between latency and target-model agreement for a given target model and deployment environment. Designing effective draft models therefore requires *jointly optimizing draft-model efficiency and target-draft agreement*, underpinning the basis of our work in this paper.

Contributions. To address this gap, we propose `T2D` (*Target-to-Draft*), a self-distillation framework that jointly optimizes draft-model architecture and target-model alignment. As opposed to selecting a draft architecture beforehand and then adapting it to the target model, `T2D` learns a compact draft model through target-model pruning while distilling the target model's knowledge simultaneously. During training, both decoding latency and alignment with the target model directly influence the pruning decisions, allowing the architecture to emerge from the optimization process itself naturally. This enables `T2D` to automatically discover draft models that attain better speculative decoding performance, thereby balancing efficiency and acceptance rate. As we will show in subsequent sections, `T2D` can naturally converge to the best architecture among multiple candidate pruned architectures, compared to existing pruning-based draft-model construction baselines, including ShortGPT-style pruning (Men et al., 2025) and a fixed-architecture joint pruning-and-distillation baseline inspired by Sheared-LLaMA (Xia et al., 2024; Yan et al., 2025). Across target LLMs from model families of varying sizes `T2D` can prune up to 90% of the target model to produce highly efficient draft models. Finally, we show that `T2D` can also be utilized in a plug-and-play fashion with newer distillation

methods such as AdaSpec (Hu et al., 2026), further underscoring its benefits. We summarize these contributions below:

- We propose `T2D`, a self-distillation framework for speculative decoding that jointly optimizes draft-model architecture and target-draft alignment, allowing the draft architecture to emerge from the optimization process rather than being manually specified as in prior work.
- To enable this joint optimization, we introduce a latency-aware structured pruning objective that directly incorporates the inference cost of candidate draft architectures, helping `T2D` effectively navigate decoding efficiency and alignment between the draft and target models.
- We evaluate `T2D` across multiple LLM families and sizes (Llama-2 and Vicuna-v1.5), and find that it significantly outperforms heuristic architecture choices and pruning-based baselines, achieving up to **2.46**$\times$ speculative decoding speedup across 11 diverse domains, such as coding, mathematics, reasoning, summarization, and multilingual understanding.

2 Related Works

Speculative Decoding. Speculative decoding reduces the inference cost of autoregressive decoding by predicting multiple future tokens in advance. Stern et al. (2018) first explored this idea using auxiliary output heads that predict future tokens, which are verified by the same model under greedy decoding. Leviathan et al. (2023) and Chen et al. (2023) generalized the approach to sampling with a separate draft model and a rejection-sampling acceptance rule that preserves the target model's output distribution. Subsequent work has explored different algorithms for drafting and verification. For example, Medusa (Cai et al., 2024) uses multiple decoding heads and tree-structured candidates, EAGLE (Li et al., 2024) uses target-model hidden features with a lightweight trainable module, and SpecInfer (Miao et al., 2023) verifies tree-structured candidates from multiple drafts in a single pass. These methods primarily change the drafting or verification algorithm. However, to successfully apply these algorithms, a highly efficient draft model is very important. Thus, in this work, we focus on constructing high-performance draft models.

Draft Model Construction. A common approach to constructing draft models is to utilize separately pretrained language models, which can then be further trained to better match a target model (Leviathan et al., 2023; Hu et al., 2026). More recent approaches derive the draft from the target itself. SD^2 (Lasby et al., 2025), for example, combines target-model unstructured pruning with self-distillation and shows that this can improve draft-target alignment. However, unstructured pruning cannot reduce model size. Thus, structural approaches have also received attention because they produce smaller and computationally efficient dense models. Wei et al. (2026) remove Transformer layers based on their importance, while Yan et al. (2025) investigate alternative depth-width configurations and show that model shape substantially affects speculative-decoding throughput. These approaches establish that different structural choices can lead to different decoding performance, motivating our work, which searches over such choices rather than fixing them beforehand.

Structured Pruning. Structured pruning algorithms reduce model size by removing layers, attention heads, or feed-forward channels by measuring the importance of Transformer blocks (Ling et al., 2024; Men et al., 2025; Ashkboos et al., 2024; Ma et al., 2023; Xia et al., 2024; Guo et al., 2025). It provides a natural way to construct compact dense models with low latency for speculative decoding. Sheared-LLaMA (Xia et al., 2024) uses learnable pruning gates to obtain smaller models while satisfying predefined architectural constraints. However, these structured pruning approaches mostly focus on standard response-quality benchmarks such as MMLU (Hendrycks et al., 2020), GSM8K (Cobbe et al., 2021), SciQ (Welbl et al., 2017), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2021), ARC-E (Clark et al., 2018), and HellaSwag (Zellers et al., 2019). To the best of our knowledge, no prior work has explored speculative-decoding-aware pruning, where model pruning is treated as a learnable objective for improving speculative decoding performance.

3 Preliminaries and Background

LLMs and Autoregressive Decoding. An LLM M_θ generates an output sequence y autoregressively, producing one token at a time. At decoding step t, the model predicts the next token, y_t based on the input prompt x and the previously generated prefix $y_{<t} = (y_1, \ldots, y_{t-1})$. Formally, it gives a conditional probability distribution for next-token prediction: $p_\theta(y_t \mid x, y_{<t})$. LLMs are

typically built from multiple stacked Transformer (Vaswani et al., 2017) layers L, which defines the depth of the model. In addition to depth, their computational cost is determined by several key architectural dimensions, including the hidden dimension d, the number of attention heads H, and the feed-forward intermediate dimension I. These dimensions determine both the representational capacity of the model and the cost of each decoding step.

In general, larger models require more computation. For example, if $\mathcal{F}_{\text{lat}}(L, H, d, I)$ denotes the wall-clock latency of a single decoding step for a model with architecture (L, H, d, I), then $\mathcal{F}_{\text{lat}}(L, H, d, I) \geq \mathcal{F}_{\text{lat}}(L', H, d, I)$, for $L \geq L'$, where $\mathcal{F}_{\text{lat}}(.)$ is a latency estimator. A similar trend holds for other architectural dimensions, but less prominently (Yan et al., 2025). Note that with a smaller architecture, LLMs can be computationally much more efficient, motivating the research domain of speculative decoding, which we discuss next.

Speculative Decoding. Speculative decoding reduces autoregressive decoding latency by pairing a large target model M_{θ_T} with a smaller draft model M_{θ_D}. Given an input x and an already-generated and accepted prefix $\tilde{y}_{<t}$, the draft model autoregressively proposes γ candidate tokens $\hat{y}_{t:t+\gamma-1} = (\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+\gamma-1})$. The target model then verifies all γ candidates in a single forward pass by matching the next-token prediction of the target model and the proposed token of the draft model at each position. A draft token is accepted if it matches the target model's own prediction at that position. However, only the initial consecutive accepted tokens are kept. If the first mismatch occurs after a accepted tokens, the remaining draft tokens are discarded, and the target model's prediction at the mismatch position is used in their place. Thus, each verification round extends the generated sequence by $a + 1$ tokens.

The resulting speedup of speculative decoding depends primarily on the average number of accepted draft tokens, the draft-model latency, the target-model verification latency, and the draft length γ. Let $\mathcal{T}_D$ and $\mathcal{T}_T$ denote the per-token decoding latencies of the draft and target models, and let the latency ratio, $c = \mathcal{T}_D/\mathcal{T}_T$. Under the standard approximation that a target verification pass latency is close to the target decoding step latency, after simplification, the speculative-decoding speedup or ratio of speculative-decoding and autoregressive decoding throughput can be written as: $\mathcal{S} = \frac{\mathbb{E}[a]+1}{(\gamma c+1)}$, where $\mathbb{E}[a]$ is the average accepted tokens (Leviathan et al., 2023) and $\mathbb{E}[a] + 1$ is the average Accepted Length (AL) including the correction token provided by the target model. For a fixed draft length, this expression shows that speculative decoding benefits from a draft model that is both *computationally efficient* and *well aligned* with the target: lowering c shrinks the denominator, while better draft-target alignment increases $\mathbb{E}[a]$.

Notably, these two objectives are not independent, as improving one often comes at the expense of the other during draft model design. Knowledge distillation (KD) (Xu et al., 2024b) is an effective approach for improving draft-target alignment, while structured pruning of the target model (Lasby et al., 2025; Xia et al., 2024; Yan et al., 2025) is a promising direction for reducing draft-model latency while preserving alignment. In the following sections, we discuss these approaches in detail.

Knowledge Distillation for Draft-Model Alignment. Knowledge distillation improves draft-target alignment by training the draft model to reproduce the target model's output distribution. One way to perform distillation is to directly minimize the KL divergence between the two next-token distributions (Zhou et al., 2024a), $D_{\text{KL}}\big(p_{\theta_T}(\cdot \mid x, y_{<t}) \,\|\, p_{\theta_D}(\cdot \mid x, y_{<t})\big)$. Alternatively, training sequences, $\tilde{y} = (\tilde{y}_1, \ldots, \tilde{y}_N)$, can be generated directly from the target model and can then be used to train the draft model with a standard causal cross-entropy (CE) loss:

$$\mathcal{L}_{\text{CE}}(\theta_D, \tilde{y}) = -\frac{1}{N} \sum_{t=1}^{N} \log p_{\theta_D}(\tilde{y}_t \mid x, \tilde{y}_{<t}). \tag{1}$$

Structured Pruning. Structured pruning removes entire architectural components of an LLM, such as Transformer layers, attention heads, or feed-forward channels, rather than individual weights. Given an LLM with dimensions (L, H, d, I), structured pruning yields a smaller *dense* model with dimensions (L', H', d', I'), where each retained dimension is smaller than the original. A structured-pruning pipeline generally requires a fixed architecture to be specified beforehand (Xia et al., 2024; Men et al., 2025; Yan et al., 2025). However, for speculative decoding, we do not want to determine the architecture heuristically. Rather, we want to prune the target model as much as possible while maintaining high alignment between the pruned model and the target model. This forms the basis for our technical contributions in this paper. To achieve this aim, we introduce `T2D`, a joint *pruning-and-alignment* framework that learns the trade-off between model efficiency and

Algorithm 1 T2D: Target-to-Draft Model Construction

Input: Target model M_{θ_T}, target model latency $\mathcal{T}_T$, alignment data $\mathcal{D}$, frozen latency predictor $\mathrm{MLP}_{\mathrm{lat}}$, reference CE $\mathcal{L}'_{\mathrm{CE}}$, number of epochs Ep.
Output: Dense draft model $M_{\theta^*_D}$

1: Initialize gates $\{\alpha_j\}$, $\lambda_1 \leftarrow 0$, $\lambda_2 \leftarrow 0$, $\widetilde{\mathcal{L}}_{\mathrm{CE}} \leftarrow 0$
2: **for** $e = 1, \ldots, Ep$ **do**
3: **for** $s = 1, \ldots, S$ **do**
4: Sample Hard-Concrete gates and compute expected-open probabilities $\{\bar{z}^L_\ell, \bar{z}^H_h, \bar{z}^I_{\ell,i}\}$
5: Compute estimated continuous architecture $(\bar{L}_D, \bar{H}_D, \bar{I}_D, \bar{d}_D)$ via Eq. 3
6: $\bar{\mathcal{T}} \leftarrow \exp\big(\mathrm{MLP}_{\mathrm{lat}}(\bar{L}_D, \bar{H}_D, \bar{I}_D)\big)$
7: $\mathcal{L}_{\mathrm{lat}} \leftarrow \mathcal{T}_{\mathrm{est}}/\mathcal{T}_T$
8: Apply sampled gates to M_{θ_T} and compute $\mathcal{L}_{\mathrm{CE}}$ using target-generated tokens $\tilde{y}$ from $\mathcal{D}$ ▷ Eq. 1
9: $\widehat{\mathcal{L}}_{\mathrm{CE}} \leftarrow (1-m)\widehat{\mathcal{L}}_{\mathrm{CE}} + m\mathcal{L}_{\mathrm{CE}}$
10: $g \leftarrow \max\left(\dfrac{\widehat{\mathcal{L}}_{\mathrm{CE}} - \mathcal{L}'_{\mathrm{CE}}}{\widehat{\mathcal{L}}_{\mathrm{CE}}},\, 0\right)$
11: Calculate loss, $\mathcal{L}$ via Eq. 4.
12: Update model and gate parameters by gradient descent on $\mathcal{L}$
13: $\lambda_1 \leftarrow \mathrm{clip}(\lambda_1 + \eta_1 g,\, 0,\, \lambda_1^{\max})$, $\lambda_2 \leftarrow \mathrm{clip}(\lambda_2 + \eta_2 g,\, 0,\, \lambda_2^{\max})$
14: **end for**
15: **end for**
16: Get final continuous values $\bar{L}^*_D, \bar{H}^*_D, \bar{I}^*_D$
17: Select retained layers L^*_D, attention heads H^*_D, and FFN channels I^*_D from the optimized gates
18: Instantiate a dense architecture with $d^*_D = H^*_D d_{\mathrm{head}}$
19: Copy retained parameters from M_{θ_T}
20: Continually fine-tune the obtained model on $\mathcal{D}$ for one epoch to obtain a dense and aligned $M_{\theta^*_D}$
21: **return** $M_{\theta^*_D}$

target-model alignment to construct an effective draft model for speculative decoding. We describe our proposed T2D framework in the next section.

4 Proposed Method: T2D

In Section 3, we highlighted a key challenge in draft-model construction (Lasby et al., 2025): a smaller draft model reduces decoding latency but may also lead to lower token acceptance during speculative decoding. Thus, given a target model M_{θ_T}, our goal is to construct a compact draft model by selecting an architecture that balances decoding latency and draft-target alignment. Let (L_D, H_D, d_D, I_D) denote the architectural dimensions of the draft model. We formulate draft-model construction as:

$$(L^*_D, H^*_D, d^*_D, I^*_D) = \arg\min_{(L_D,H_D,d_D,I_D)\in\mathcal{A}} \Big(\mathcal{F}_{\mathrm{lat}}(L_D, H_D, d_D, I_D), \mathcal{L}_{\mathrm{CE}}(L_D, H_D, d_D, I_D; \mathcal{D})\Big), \quad (2)$$

where $\mathcal{A}$ denotes the set of architectures reachable through structured pruning, and $\mathcal{D} = \{(x_1, \tilde{y}_1), (x_2, \tilde{y}_2), \ldots, (x_S, \tilde{y}_S)\}$ denotes the set of query-response pairs generated by M_{θ_T} and used for optimization. Directly optimizing Eq. 2 is difficult because the architectural variables are discrete and the search space is combinatorial. We therefore introduce *differentiable gates* that convert discrete pruning decisions into continuous variables, enabling gradient-based optimization inspired by Xia et al. (2024). Similarly, model latency cannot be incorporated directly into the optimization loss since $\mathcal{F}_{\mathrm{lat}}(\cdot)$ is not differentiable by definition, necessitating a differentiable proxy. In the following sections, we describe how the gates represent a differentiable architecture, how we obtain a differentiable latency estimate, and how we formulate the loss function for joint pruning and KD-based alignment training. Additionally, we present the overall T2D framework as Algorithm 1.

Learnable Structured Pruning. To enable gradient-based optimization, we introduce learnable gates over removable Transformer structures, including layer gates z^L_ℓ, attention-head gates z^H_h, and feed-forward channel gates $z^I_{l,i}$. For each gate parameter α_j, we adopt the Hard-Concrete relaxation (Louizos et al., 2017) and compute an expected-open probability $\bar{z}_j \in [0, 1]$, which serves as a differentiable approximation of a binary keep-or-remove decision (Line 1-3 in Algorithm 1). Using these probabilities, we estimate the architecture dimensions as:

$$\bar{L}_D = \sum_{\ell} \bar{z}_{\ell}^{L}, \qquad \bar{H}_D = Q \sum_{h} \bar{z}_{h}^{H}, \qquad \bar{I}_D = \frac{\sum_{\ell} \bar{z}_{\ell}^{L} \sum_{i} \bar{z}_{\ell,i}^{I}}{\sum_{\ell} \bar{z}_{\ell}^{L}}, \tag{3}$$

where Q denotes the number of query heads associated with each key-value group (Line 4-5). We estimated the hidden dimension as $\bar{d}_D = \bar{H}_D d_{\text{head}}$, where d_{head} is the head dimension. Now that we have formalized the learnable gates, we will relate latency to the learnable gates to obtain differentiable latency.

Differentiable Latency Modeling Measuring hardware latency directly for every candidate architecture during optimization is both expensive and, more importantly, non-differentiable. To estimate *differentiable latency* directly from $(\bar{I}_D, \bar{L}_D, \bar{H}_D)$, we train a lightweight latency predictor *offline*, and keep it frozen while optimizing the pruning gates. To train the latency predictor, for each candidate dense architecture, (L_D, H_D, I_D), we measure its single-token decoding latency, $\mathcal{T}_{dec}$, and generate a latency-measurement dataset. Then we train a small regression model MLP_{lat} with the dataset. The latency predictor MLP_{lat} takes the architectural dimensions (L_D, H_D, I_D) as input and predicts the logarithm of the corresponding latency. Further description of the training of the latency predictor is provided in Appendix H.

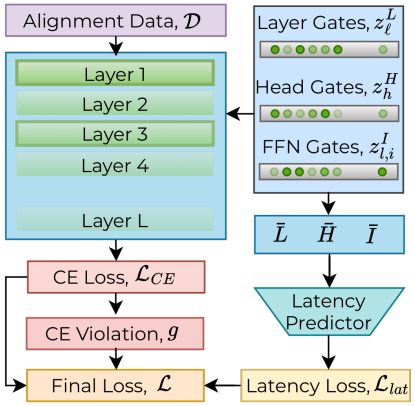

Figure 2: Overview of the T2D training pipeline. Learnable gates mask layers, attention heads, and FFN channels to jointly optimize the pruned architecture and model weights using a frozen latency predictor, target-generated cross-entropy loss, and a reference-guided penalty.

During gate optimization, we pass $(\bar{L}_D, \bar{H}_D, \bar{I}_D)$ through the frozen latency predictor. We obtain the predicted latency as, $\bar{\mathcal{T}} = \exp\left(\text{MLP}_{\text{lat}}(\bar{L}_D, \bar{H}_D, \bar{I}_D)\right)$ (Line 5). Because this predictor is differentiable with respect to its architectural inputs, gradients can flow from the latency objective all the way back to the pruning gates during backpropagation, giving us a latency-aware training signal. With learnable latency via $\bar{\mathcal{T}}$ and alignment via Eq. 1, we now combine them into the single joint objective that drives our pruning procedure.

Latency-Aware Pruning and Knowledge Distillation We optimize the gates using three complementary loss terms: (1) a latency objective, (2) the CE loss, and (3) a reference-guided penalty that limits how much the CE is allowed to degrade. Our entire framework is visualized in Figure 2. The latency objective normalizes the predicted draft latency by the measured latency of the unpruned target model: $\mathcal{L}_{\text{lat}} = \bar{\mathcal{T}}/\mathcal{T}_T$ (Line 6). The raw CE loss $\mathcal{L}_{\text{CE}}$ is computed on the currently masked model using the target-generated response $\tilde{y}$ from the alignment data by using Eq. 1. (Line 7). We weight the latency and CE terms by μ_{lat} and μ_{CE}, respectively.

Since optimization starts from the *unpruned* target model, the latency term needs to be dominant during the early stages of training to initiate pruning. However, if pruning is too aggressive, the CE loss can spike rapidly as useful structures are masked too quickly. To guard against this, we impose a reference-based one-sided augmented-Lagrangian penalty. The penalty remains inactive while the smoothed CE, $\widehat{\mathcal{L}}_{\text{CE}}$, remains below the reference $\mathcal{L}'_{\text{CE}}$, and increases only when the constraint is violated. We calculate the smoothed CE loss, $\widehat{\mathcal{L}}_{\text{CE}}$, using a smoothing factor m, which controls how much the current loss influences the penalty (Line 8). Thus, our final optimization loss becomes:

$$\mathcal{L} = \mu_{\text{lat}} \mathcal{L}_{\text{lat}} + \mu_{\text{CE}} \mathcal{L}_{\text{CE}} + \lambda_1 g + \lambda_2 g^2, \tag{4}$$

where g determines the amount of constraint violation, calculated similarly to Line 10 of Algorithm 1, and λ_1 and λ_2 are the learnable penalty weights, whose learning is controlled by the hyperparameters η_1, η_2, $\lambda_1^{\max}$, and $\lambda_2^{\max}$ (Line 13). Finally, we update the model weights, gates, and penalty weights based on the calculated loss (Lines 10-12). This joint objective pushes the estimated decoding latency down while actively preventing alignment from drifting significantly from the acceptable tolerance. The two goals motivated in Equation 2 of Section 3 are therefore optimized *simultaneously*, over the *same* gate parameters, rather than in separate stages. After optimization, we convert the learned continuous gate scores into a concrete draft architecture, as described next.

Draft Model Construction and Continual Finetuning. We convert the continuous gate scores into discrete decisions after the joint training (Lines 17-18). We then instantiate a new, dense Transformer using the selected layers, attention structures, and feed-forward channels, and copy their corresponding parameters from the trained target model (Line 19). Finally, we fine-tune the resulting draft model on $\mathcal{D}$ for one epoch to let the weights readjust and mitigate the effects of converting continuous gate values into discrete decisions. Thus, we obtain the final draft model $M_{\theta^*_D}$, derived from M_{θ_T} by truly reducing the number of parameters.

5 Experiments and Results

Having described T2D in Section 4, we now explain our experimental setup, including the target and initial draft models, the training and validation datasets, benchmark dataset, and evaluation metrics. We then discuss our results subsequently.

Models. We evaluate T2D on Llama-2-7B-Chat-HF (Touvron et al., 2023), which has been widely used as a base model for both pruning and speculative decoding research (Yan et al., 2025; Sun et al., 2024b), enabling direct comparison with prior work and their design decisions. To assess the scalability of T2D across model sizes, we further evaluate it on Vicuna-13B-v1.5 (Zheng et al., 2023), a comparatively larger model. Both these models provide a representative evaluation across different model families and scales. For each family, we construct the draft model from the respective target model, similarly as described in Section 4, and use it in speculative decoding.

Training Dataset Construction. For each target model, we construct a target-generated alignment dataset inspired by Wu et al. (2024) from a mixture of publicly available instruction tuning corpora, including Dolly-15K (Conover et al., 2023), Alpaca (Taori et al., 2023), WizardLM (Xu et al., 2024a), UltraChat (Ding et al., 2023), OpenOrca (Lian et al., 2023), WildChat (Zhao et al., 2024), MetaMathQA (Yu et al., 2024), CodeFeedback (Zheng et al., 2024), Bactrian-X (Li et al., 2023), and xP3 (Muennighoff et al., 2023). We construct 250,000 question-answer pairs for the training set $\mathcal{D}$, which is used during pruning and alignment optimization, and randomly sample 5,000 pairs for the validation set $\mathcal{D}_{\text{val}}$. The validation set is used only for hyperparameter optimization. We provide further details of the training dataset in Appendix C.

Benchmark Dataset. We evaluate the performance of the draft models on the SPEED Bench (Abramovich et al., 2026) speculative decoding benchmark, which contains 798 queries across 11 domains. We provide details of the SPEED benchmark in Appendix B.

Baselines. Given that target-model-pruning-based draft-model construction is an emerging research direction with limited prior work, we compare T2D against state-of-the-art draft-construction approaches that isolate the effect of joint architecture optimization: *Wide* Draft Models (Yan et al., 2025) and ShortGPT (Men et al., 2025). Additional details of these baselines are provided in Appendix D due to space limits. Similar to prior work (Leviathan et al., 2023), we report the speculative decoding speedup, $\mathcal{S}$, relative to autoregressive decoding as our main metric in results.

Now, we discuss and compare T2D with the baselines. We first evaluate T2D 's performance under the simple distillation process described in Section 4. We then analyze the trade-off curve that T2D seeks to optimize and examine how it balances latency and CE loss during joint optimization. Next, we show that T2D-derived draft models can be utilized in a plug-and-play fashion with speculative decoding-specific alignment techniques. Finally, we compare T2D with independently developed, extensively trained draft models and show that such models are not well suited for speculative decoding because they are optimized only for better text generation, not explicitly for this objective.

5.1 Main Results

Table 1 presents the decoding speedup for all methods. T2D outperforms all baselines in all domains. In Llama-2-7B-Chat, the T2D-developed draft model achieves up to 1.74× speedup, which is up to 20% higher speedup than heuristic architecture choices. In the comparatively larger model Vicuna-13B-v1.5, T2D achieves an overall speedup of 1.89 and up to 9% higher speedup than heuristic baselines, outperforming the baselines in all domains. Similar trends appear in other models, showing that T2D can effectively converge to a better architectural and alignment tradeoff and achieve the best overall standard speculative decoding performance. We provide additional details on T2D-

Table 1: Speculative decoding speedup across different evaluation domains. For each target model, we compare draft models constructed by ShortGPT, *Wide*, and T2D. The highest speedup (×) for each target model and domain is shown in bold.

Target Model	Draft	Size (B)	Code	Hum.	Math	Multilin.	QA	RAG	Reason.	Role.	Stem	Summ.	Writing	Overall
Llama-2-7B-Chat	ShortGPT	1.27	1.72	1.30	1.57	1.46	1.06	1.57	1.66	1.20	1.43	1.41	1.41	1.45
	Wide	1.27	1.82	1.46	1.74	1.83	1.34	1.63	1.73	1.25	1.58	1.43	1.50	1.59
	T2D	0.86	**1.98**	**1.59**	**1.92**	**2.03**	**1.46**	**1.79**	**1.90**	**1.35**	**1.72**	**1.60**	**1.62**	**1.74**
Vicuna-13B-v1.5	ShortGPT	1.91	1.95	1.66	1.82	2.22	1.40	1.74	1.86	1.30	1.70	1.73	1.62	1.74
	Wide	1.91	2.06	1.67	1.92	2.22	1.44	1.79	1.91	1.33	1.72	1.74	1.66	1.78
	T2D	1.21	**2.13**	**1.79**	**1.99**	**2.46**	**1.47**	**1.97**	**2.04**	**1.39**	**1.83**	**1.88**	**1.73**	**1.89**

derived architectures in Appendix E. Our qualitative analysis shows that the T2D Llama-2 draft model generates mostly coherent responses even after 90% pruning, demonstrating the effectiveness of joint optimization. Further analysis is provided in Appendix I owing to space considerations.

5.2 Why is T2D better than random manual architecture selection?

To investigate the reason behind the effectiveness of T2D, we performed a pruning ratio sweeping experiment in which we pruned the Llama-2-7B-chat-hf model by 65%, 71%, 77%, 83%, 89%, and 95% with ShortGPT. After pruning, we fine-tuned the resulting draft models for one epoch on target-model-generated responses. Finally, after fine-tuning, we evaluated the models on the SPEED Benchmark and plotted the change in speedup, the latency ratio relative to the target model, and the average acceptance-length ratio relative to the maximum acceptance length. Figure 3 shows the trade-off between acceptance length and latency for different pruning ratios. With aggressive pruning, both the latency and the acceptance length of the draft model decrease. However, speedup peaks at 89% pruning and then declines, showing that the best trade-off occurs around 89% pruning, where the rate of latency decrease outweighs the rate of acceptance-rate decrease. As a result, speedup is highest at that point. T2D finds this maximum speedup point by pruning Llama-2-7B-Chat naturally up to 90% and reducing latency while keeping the acceptance-length decrease rate as low as possible. As a result, the T2D-developed draft model achieves the best overall speedup in an *automated* fashion, whereas the other pruning baselines require a fixed prior pruning budget, limiting their efficacy.

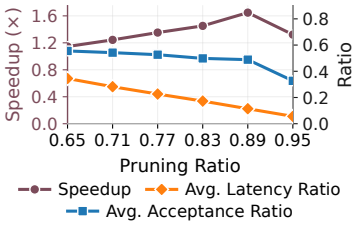

Figure 3: Trade-off curve of acceptance ratio and latency across pruning ratio and speedup of the draft model across pruning ratio of Llama-2-7B-Chat model when pruned with ShortGPT.

5.3 How does T2D prune the model while balancing CE loss and latency?

In Figure 4, we show the ratios of different LLM components along with the pruning ratio for Llama-2-7B-Chat. Additionally, we plot the latency and the CE loss over the training period. Initially, the model has very high CE loss because the gates are randomly initialized, and very high latency because pruning has not yet started. After some steps, the CE loss goes down as the model weights adjust. Then, near 10% of training, pruning starts after a warm-up period, and the CE loss begins to increase. After the CE loss rises to a certain value, the Lagrange penalty increases, and the CE loss weights λ_1 and λ_2 are increased significantly. As a result, the CE loss starts to decrease after that point, and the pruning rate is balanced against the CE loss. After 90% pruning, the pruning ratio flattens out. Over the training period, layer pruning contributes more, while heads and FFN are pruned less, because most of the latency benefit comes from layer pruning, suggesting that wider models are better for speculative decoding. We present the pruning ratio, latency, and CE loss curves for other models in Appendix F. The overall trends are similar to Llama-2-7B-Chat. However, for larger

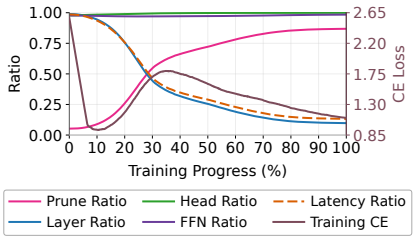

Figure 4: Pruning ratio of the Llama-2-7B-Chat model over the training period along with the kept percentage (ratio) of the Layers, Heads, and FFN. We also plot latency and CE loss.

models, T2D prunes the heads and FFN as well, since these components have higher dimensions in larger models and also contribute a non-negligible portion of the latency. T2D achieves a better latency-alignment trade-off under simple supervised fine-tuning-based distillation. However, more advanced alignment techniques have been proposed specifically for speculative decoding, such as AdaSpec (Hu et al., 2026). In the next section, we investigate whether T2D continues to outperform alternative methods when combined with speculative decoding-specific alignment techniques.

5.4 Can T2D Models Adapt to Other Alignment Methods?

In our experiments, we utilized target model-generated responses for distillation and fine-tuned the draft model accordingly. However, for even better alignment and performance, T2D-derived draft models can also be adapted to more advanced alignment methods such as AdaSpec. To evaluate whether T2D models can continue improving under domain-specific specialized alignment, we fine-tune all Llama-2-7B-Chat-based draft models using AdaSpec on the GSM8K dataset (Cobbe et al., 2021) and evaluate their speculative decoding performance on the GSM8K test set, following the same evaluation protocol as AdaSpec. In Table 2, we present the performance of all draft models. Under AdaSpec distillation on the GSM8K dataset, T2D outperforms the other baselines, achieving up to 20% higher speedup. This shows that T2D is a superior draft-model construction algorithm that can be adapted to other distillation methods in a plug-and-play fashion and achieve even greater speedups than heuristic architecture-selection methods.

Table 2: Speculative decoding performance comparison of draft models under the AdaSpec distillation method (best in **bold**)

Draft Model	Speedup
ShortGPT	1.43
Wide	1.59
T2D	**1.79**

5.5 Our T2D draft models better than an independently trained draft model?

We compare the T2D-produced Llama-2-7B-Chat draft model with the Sheared-Llama-1.3B model, which is an independently pretrained model on 50B tokens, which is about 340× more data than T2D takes during optimization. We fine-tune the Sheared-Llama-1.3B model with the Llama-2-7B-chat-hf generated responses similarly to T2D for one epoch to readjust the weights. We call the finetuned model Sheared-Llama-1.3B-ft and report the overall speedup of both models in Table 3. The Sheared-Llama-1.3B-ft model achieves only a $1.18\times$ overall speedup, whereas the T2D draft model achieves a significantly higher $1.74\times$ speedup. This is because Sheared-Llama is designed for general text generation quality rather than optimizing the latency-alignment trade-off for speculative decoding. As a result, it delivers lower speculative decoding performance than T2D. These results further demonstrate that T2D is a more effective draft-model construction method for identifying speculative-decoding-aware draft models that can significantly accelerate autoregressive decoding. We provide per-domain speedup comparison in Appendix G.

Table 3: Speculative decoding speedup (×) comparison between finetuned Sheared-Llama and T2D with Llama-2-7B-Chat as the target model. The highest speedup is shown in **bold**.

Draft Model	Size (B)	Speedup
Sheared-Llama-1.3B-ft	1.27	1.18
T2D	0.86	**1.74**

6 Conclusion

In this work, we address a key limitation of existing draft-model construction methods for speculative decoding: their reliance on fixed draft architectures chosen before considering alignment with the target model. We propose T2D, a self-distillation framework that jointly optimizes draft-model architecture and target-model alignment, allowing the draft architecture to emerge directly from the optimization process. To enable this, we introduce a latency-aware structured pruning objective that combines a differentiable estimate of inference latency with a target-distilled cross-entropy objective, guided by a reference-based penalty that prevents pruning from outpacing alignment. Experiments across the Llama-2 and Vicuna-v1.5 families show that T2D consistently outperforms heuristic architecture choices and pruning-based baselines, demonstrating that effective draft models can be discovered directly from the target model through joint optimization of latency and alignment.

7 AI USE STATEMENT

We used generative AI tools solely to assist with proofreading, such as correcting spelling and grammar in the manuscript.

8 REPRODUCIBILITY STATEMENT

We provide full implementation details necessary to reproduce our results, including the T2D training procedure (Algorithm 1), hyperparameters, and loss formulation (Equation 4) in Section 4 and Appendix D. Details of the training dataset construction, including source corpora and sampling procedure, are given in Section 5 and Appendix C, and the SPEED Bench evaluation protocol is described in Appendix B. We provide our code in an open-source repository: `https://anonymous.4open.science/r/T2D-C310`.

9 ETHICS STATEMENT

This work aims to improve the efficiency of LLM inference through speculative decoding without introducing new capabilities, data, or risks beyond those already associated with the target models from which we prune and distill. All datasets used for alignment training are publicly available instruction-tuning corpora, and all target models are open-weight models used in accordance with their respective licenses. We do not anticipate direct negative societal impacts from this work. Instead, by reducing inference latency and computational cost, our method may help reduce the environmental and resource footprint associated with deploying large language models.

REFERENCES

Talor Abramovich, Maor Ashkenazi, Izzy Putterman, Benjamin Chislett, Tiyasa Mitra, Bita Darvish Rouhani, Ran Zilberstein, and Yonatan Geifman. Speed-bench: A unified and diverse benchmark for speculative decoding. *arXiv preprint arXiv:2604.09557*, 2026.

Saleh Ashkboos, Maximilian Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. Slicegpt: Compress large language models by deleting rows and columns. In *International Conference on Learning Representations*, volume 2024, pp. 11682–11701, 2024.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

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

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

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Mike Conover, Matt Hayes, Ankit Mathur, Jianwei Xie, Jun Wan, Sam Shah, Ali Ghodsi, Patrick Wendell, Matei Zaharia, and Reynold Xin. Free dolly: Introducing the world's first truly open instructiontuned llm. 2023.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. Enhancing chat language models by scaling high-quality instructional conversations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 3029–3051, 2023.

Elias Frantar and Dan Alistarh. SparseGPT: Massive language models can be accurately pruned in one-shot. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 10323–10337. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/frantar23a.html.

Kaan Gokcesu and Hakan Gokcesu. Generalized huber loss for robust learning and its efficient minimization for a robust statistics. *arXiv preprint arXiv:2108.12627*, 2021.

Jialong Guo, Xinghao Chen, Yehui Tang, and Yunhe Wang. Slimllm: Accurate structured pruning for large language models. *arXiv preprint arXiv:2505.22689*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

Yuezhou Hu, Jiaxin Guo, Xinyu Feng, and Tuo Zhao. Adaspec: Selective knowledge distillation for efficient speculative decoders. *Advances in Neural Information Processing Systems*, 38:88736–88758, 2026.

Mike Lasby, Nish Sinnadurai, Valavan Manohararajah, Sean Lie, Yani Ioannou, and Vithursan Thangarasa. Sd $ˆ 2$: Self-distilled sparse drafters. *arXiv preprint arXiv:2504.08838*, 2025.

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

Haonan Li, Fajri Koto, Minghao Wu, Alham Fikri Aji, and Timothy Baldwin. Bactrian-x: Multilingual replicable instruction-following models with low-rank adaptation. *arXiv preprint arXiv:2305.15011*, 2023.

Yuetai Li, Xiang Yue, Zhangchen Xu, Fengqing Jiang, Luyao Niu, Bill Yuchen Lin, Bhaskar Ramasubramanian, and Radha Poovendran. Small models struggle to learn from strong reasoners. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 25366–25394, 2025.

Yuhui Li, Fangyun Wei, Chao Zhang, and Hongyang Zhang. Eagle: Speculative sampling requires rethinking feature uncertainty. *arXiv preprint arXiv:2401.15077*, 2024.

Wing Lian, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, and "Teknium". Openorca: An open dataset of gpt augmented flan reasoning traces. *Hugging Face dataset repository*, 2023.

Gui Ling, Ziyang Wang, Yuliang Yan, and Qingwen Liu. Slimgpt: Layer-wise structured pruning for large language models. *Advances in Neural Information Processing Systems*, 37:107112–107137, 2024.

Alejandro Lopez-Lira and Yuehua Tang. Can chatgpt forecast stock price movements? return predictability and large language models. *Journal of Financial Economics*, 184:104335, 2026.

Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through $ l_0 $ regularization. *arXiv preprint arXiv:1712.01312*, 2017.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

Xin Men, Mingyu Xu, Qingyu Zhang, Qianhao Yuan, Bingning Wang, Hongyu Lin, Yaojie Lu, Xianpei Han, and Weipeng Chen. Shortgpt: Layers in large language models are more redundant than you expect. In *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 20192–20204, 2025.

Xupeng Miao, Gabriele Oliaro, Zhihao Zhang, Xinhao Cheng, Zeyu Wang, Zhengxin Zhang, Rae Ying Yee Wong, Alan Zhu, Lijie Yang, Xiaoxiang Shi, et al. Specinfer: Accelerating generative large language model serving with tree-based speculative inference and verification. *arXiv preprint arXiv:2305.09781*, 2023.

Niklas Muennighoff, Thomas Wang, Lintang Sutawika, Adam Roberts, Stella Biderman, Teven Le Scao, M Saiful Bari, Sheng Shen, Zheng Xin Yong, Hailey Schoelkopf, et al. Crosslingual generalization through multitask finetuning. In *Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 1: Long papers)*, pp. 15991–16111, 2023.

Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian. A comprehensive overview of large language models. *ACM Transactions on Intelligent Systems and Technology*, 16(5):1–72, 2025.

Plawan Kumar Rath and Rahul Maliakkal. Weight pruning amplifies bias: A multi-method study of compressed llms for edge ai. In *2026 IEEE World AI IoT Congress (AIIoT)*, pp. 0292–0299. IEEE, 2026.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

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

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. A simple and effective pruning approach for large language models. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=PxoFut3dWW.

Mingjie Sun, Zhuang Liu, Anna Bair, and Zico Kolter. A simple and effective pruning approach for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 4942–4964, 2024b.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Stanford alpaca: An instruction-following llama model, 2023.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Hanyu Wei, Zunhai Su, Peng Lu, Chao Li, Spandan Tiwari, Ashish Sirasao, and Yuhan Dong. Sdfp: Speculative decoding with fit-pruned models for training-free and plug-and-play llm acceleration. In *International Conference on Intelligent Computing*, pp. 518–528. Springer, 2026.

Johannes Welbl, Nelson F Liu, and Matt Gardner. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pp. 94–106, 2017.

Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 944–964, 2024.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. Sheared llama: Accelerating language model pre-training via structured pruning. In *International Conference on Learning Representations*, volume 2024, pp. 5385–5409, 2024.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=CfXh93NDgH.

Xiaohan Xu, Ming Li, Chongyang Tao, Tao Shen, Reynold Cheng, Jinyang Li, Can Xu, Dacheng Tao, and Tianyi Zhou. A survey on knowledge distillation of large language models. *arXiv preprint arXiv:2402.13116*, 2024b.

Minghao Yan, Saurabh Agarwal, and Shivaram Venkataraman. Decoding speculative decoding. In *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)*, pp. 6460–6473, 2025.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pp. 45040–45061, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th annual meeting of the association for computational linguistics*, pp. 4791–4800, 2019.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Zican Dong, Yupeng Hou, Beichen Zhang, Yingqian Min, Junjie Zhang, Peiyu Liu, et al. A survey of large language models. *Frontiers of Computer Science*, 20(12):2012627, 2026.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=Bl8u7ZRlbM`.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023.

Tianyu Zheng, Ge Zhang, Tianhao Shen, Xueling Liu, Bill Yuchen Lin, Jie Fu, Wenhu Chen, and Xiang Yue. Opencodeinterpreter: Integrating code generation with execution and refinement. *arXiv preprint arXiv:2402.14658*, 2024.

Yongchao Zhou, Kaifeng Lyu, Ankit Singh Rawat, Aditya Krishna Menon, Afshin Rostamizadeh, Sanjiv Kumar, Jean-François Kagy, and Rishabh Agarwal. Distillspec: Improving speculative decoding via knowledge distillation. In *International Conference on Learning Representations*, volume 2024, pp. 32011–32050, 2024a.

Zixuan Zhou, Xuefei Ning, Ke Hong, Tianyu Fu, Jiaming Xu, Shiyao Li, Yuming Lou, Luning Wang, Zhihang Yuan, Xiuhong Li, et al. A survey on efficient inference for large language models. *arXiv preprint arXiv:2404.14294*, 2024b.

APPENDIX

A LIMITATION

Our findings are based on the models, datasets, and experimental settings used in this work. Results may vary with different experimental setups, such as hardware configurations or sampled datasets. We recommend interpreting our results in the context of the experimental setup described in the paper and appendix.

B BENCHMARK DATASET DETAILS

We use 798 samples from the SPEED Bench speculative decoding benchmark to evaluate all methods across 11 diverse domains: Coding, Humanities, Mathematics, Multilingual, Question Answering (QA), Retrieval-Augmented Generation (RAG), Reasoning, Roleplay, STEM, Summarization, and Writing. The benchmark covers a broad range of generation tasks, enabling a comprehensive assessment of speculative decoding performance under different workloads. Table 4 summarizes the number of evaluation samples in each domain.

Table 4: Number of evaluation samples from SPEED Bench used in our experiments.

Code	Hum.	Math	Multilin.	QA	RAG	Reason.	Role.	STEM	Summ.	Writing	Total
80	80	80	80	80	71	51	48	80	78	70	798

C TRAINING DATASET

As discussed in Section 5, we use queries from open-source instruction-tuning datasets and generate responses using the target models. We randomly sample 250K queries, which cover approximately 95% of the Llama tokenizer vocabulary. In total, these queries correspond to approximately 0.022B Llama tokens.

Although the same set of queries is used for all target models, the number and length of the generated responses vary across models. To characterize the scale of the resulting training data, Table 5 reports the total number of generated response tokens and the average response length for each target model.

Table 5: Statistics of the generated training data. Both models are prompted with the same set of 250K instruction-tuning queries.

Target Model	Total Response Tokens (B)	Avg. Response Length
Llama-2-7B-Chat	0.147	588
Vicuna-13B-v1.5	0.144	577

D IMPLEMENTATION DETAILS

We explain our implementation details briefly below.

Baselines. We use the *Wide* model proposed by Yan et al. (2025) as our primary baseline. The *Wide* architecture is obtained by pruning the target model, and we refer to this approach as *Wide* throughout the paper. Specifically, we use the best-performing *Wide* architecture reported by Yan et al. (2025), which was optimized on Llama-2-7B-Chat. We then prune each target model according to this architecture and train it on the same dataset used for `T2D`. Following pruning, the resulting *Wide* models are further fine-tuned using the same supervised fine-tuning dataset. For consistency, we also adopt the same 5-layer *Wide* architecture proposed by Yan et al. (2025) across all target models. Similar to `T2D`, the *Wide* models are jointly optimized for two epochs and then continually fine-tuned for one additional epoch after pruning. We also compare our method with a simpler ShortGPT-based draft-model construction method with the same architecture as the Wide model.

Hyperparameters. We summarze the hyperparameters used in T2D model development.

Hardware Configurations. The experiments were conducted on a Linux server with 14x NVIDIA DGX B200 GPUs with 192 GB VRAM/GPU.

Table 6: Hyperparameters used for pruning and fine-tuning in T2D framework.

Hyperparameter	Value
Target modules	Layers, attention heads, FFN channels
Reference CE loss ($\mathcal{L}'_{CE}$)	0.5
Cross-entropy loss weight (μ_{CE})	1.0
Latency loss weight (μ_{lat})	$200 - 400$
Joint optimization epochs	2
Continual fine-tuning epochs	1
Global training batch size	32
Sequence length	4096
Fine-tuning learning rate	1×10^{-4}
Layer gate learning rate	1×10^{-3}
Head gate learning rate	1×10^{-3}
FFN gate learning rate	3×10^{-3}
Penalty learning rate (η_1)	1×10^{-3}
Penalty learning rate (η_2)	1×10^{-3}

E T2D-DERIVED DRAFT MODEL ARCHITECTURES

Table 7 presents the final draft-model architectures derived by T2D for Llama-2-7B-Chat and Vicuna-13B-v1.5.

Table 7: Architectural specifications of the T2D-derived draft models for Llama-2-7B-Chat and Vicuna-13B-v1.5.

T2D Draft Model	Params (B)	Layers	Hidden Size	Num Heads	FFN Size
Llama-2-7B-Chat	0.86	3	4096	32	10752
Vicuna-13B-v1.5	1.21	4	4480	35	11264

F RATIO OF WEIGHTS, CE LOSS, AND LATENCY OVER THE TRAINING PERIOD FOR VICUNA-13B-V1.5

We provide the ratios of different components of Vicuna-13B-v1.5 during the T2D joint optimization in Figure 5. The figure shows that, over the training period, the estimated latency of the model decreases, along with the ratio of the pruned model size with respect to the target model. However, when looking at the individual components, we observe that the number of attention heads in Vicuna-13B-v1.5 initially decreases due to pruning and later increases slightly as some heads are retained. For the FFN, the ratio initially decreases and eventually flattens. These trends show that T2D can automatically adjust the pruning ratios of different components during optimization.

G PER DOMAIN SPEEDUP COMPARISON WITH SHEARED LLAMA

Table 8 shows the speedup of Sheared-Llama-1.3B and the T2D draft model across domains. T2D achieves higher speedup in every domain because of the latency-aware optimization in T2D.

Table 8: Speculative decoding speedup ($\times$) of Sheared-Llama-1.3B and T2D. The highest speedup for each domain is shown in **bold**.

Draft Model	Size (B)	Code	Hum.	Math	Multilin.	QA	RAG	Reason.	Role.	Stem	Summ.	Writing	Overall
Sheared-Llama-1.3B-ft	1.27	1.32	1.11	1.29	1.16	0.90	1.36	1.40	1.12	1.20	0.73	1.27	1.18
T2D	0.86	1.98	1.59	1.92	2.03	1.46	1.79	1.90	1.35	1.72	1.60	1.62	1.74

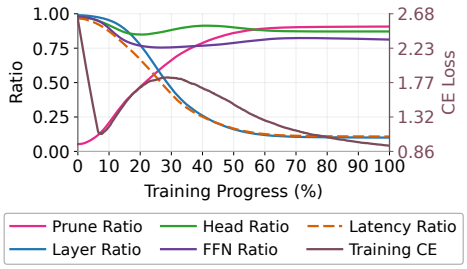

Figure 5: Pruning ratio of the target model (Vicuna-13B-v1.5) over the training period, along with the kept percentage (ratio) of the Layers, Heads, and FFN. We also plot latency and CE loss.

H Development of the MLP Latency Predictor

To obtain a differentiable estimate of decoding latency, we train a lightweight MLP latency predictor offline before the joint pruning-and-distillation optimization. The predictor takes the estimated architectural dimensions (L_D, H_D, I_D), corresponding to the retained layers, attention heads, and FFN size, and predicts the log latency of a single cached decoding step.

Training Data. We construct a dataset of candidate architectures by varying the number of layers, attention heads, and FFN dimensions. For each architecture, we instantiate an uninitialized model of the corresponding shape and directly measure single-token decoding latency on the target hardware. The median latency across repeated measurements is used as the regression target.

Model and Training. The latency predictor is a three-layer MLP with two hidden layers of size 256 and SiLU activations. Inputs are standardized using training-set statistics, and the model is trained to predict log latency using the Huber loss (Gokcesu & Gokcesu, 2021). To encourage smooth interpolation between measured architectures, we apply small random perturbations to the input dimensions during training. After training, the predictor is frozen and used throughout the joint optimization process to provide differentiable latency estimates.

I Qualitative Analysis of Speculative Decoding Performance

Text generation quality of `T2D`. As we have already discussed, `T2D` prunes the target model by up to 90%. Following this extensive pruning and alignment, we investigate whether the resulting draft model retains coherent text-generation capability. Table 9 provides representative responses generated by the `T2D` Llama-2 draft model for a range of random general-purpose queries. Most of the time, the model is successful in staying relevant to the query and topic, correctly following instructions and producing on-topic, coherent answers.

Comparison with baselines on Standard Speculative Decoding. As discussed in the previous section, `T2D` generates coherent and meaningful text, which is important for a draft model because its predictions are used to propose tokens during speculative decoding. We now compare the token sequences generated by `T2D`, ShortGPT, and Wide, with all three evaluated against the same target model. Figure 6 shows decoding traces from three domains: RAG-style extraction, reasoning, and open-ended writing. Accepted tokens are shown in `green`, rejected tokens in ~~`red`~~, and the target model's corrections in **`blue`**.

Across the examples, the three draft models often make similar mistakes early in the generation. For example, they may propose `The` instead of `Based` at the beginning of a response, or `text` instead of `data/question`. These cases suggest that some errors are shared across the pruned draft models rather than being specific to a particular construction method. At the same time, `T2D` gets several finer details right where the baselines do not. In Example 1, for instance, `T2D` correctly predicts the capitalized `D` in `D party`, while both ShortGPT and Wide predict the lowercase `d`. In Example 3, `T2D` behaves similarly to ShortGPT and maintains a long accepted span, while Wide introduces several additional rejections even though it eventually produces similar text. Overall,

Table 9: Example response generated by the T2D Llama-2 draft model, with observations on response quality.

Prompt	T2D Draft Model Response (excerpt)	Observation
How are you?	I'm just an AI, I don't have personal experiences or emotions, so I don't have a physical body or a specific situation. However, I can provide information on a wide range of topics, including but not limited to: 1. Travel and exploration: I can provide information on different cultures, languages, and landscapes, and I can also provide information on different cultures and landscapes. 2. Culture and traditions: I can provide information on different cultures and traditions [...]	Relevant and appropriate framing (correctly disclaims lacking emotions) repetition.
What is the capital of France?	The capital of France is Paris.	Fully correct and relevant; concise factual answer.
Can you help me in my study?	Of course! I'd be happy to help you in your study. Here are some steps you can take to help me improve my study skills: 1. Identify the type of study: Determine which type of study you want to focus on, such as studies that you want to study for a specific purpose or subject. 2. Research the topic: Identify the topic or subject you want to focus on, such as the impact of social media on mental health [...]	Relevant and largely coherent, giving actionable study advice.
What is a computer?	A computer is a type of computer that is designed to perform specific tasks, such as image processing, data analysis, and machine learning. It is typically used in a variety of applications, including: 1. Image processing: A computer can process images, such as images of objects, scenes, or time-series data, and can be used to perform tasks such as image compression, feature extraction, and image processing [...]	Topically relevant (stays within computing/AI applications) but the core definition is circular.
What is Artificial Intelligence?	Artificial Intelligence (AI) is a rapidly growing field that has the potential to revolutionize various industries, including healthcare, finance, transportation, and education. Here are some of the key aspects of AI: 1. Machine Learning: AI systems can be used to analyze large datasets, identify patterns, and make predictions, and optimize them for better decision-making. 2. Natural Language Processing (NLP): AI systems can be used to analyze large amounts of text data, such as customer reviews, marketing campaigns, and customer feedback, to understand their context and make more informed decisions [...]	Correct and directly relevant; accurately defines AI and enumerates coherent, describes the subfields (ML, NLP), demonstrating strong topical grounding.

these traces show that T2D can maintain fluent generation and capture details such as capitalization and entity-level tokens, while still exhibiting some of the common errors shared by other pruned draft models.

Example 1: Context Tabular Extraction (RAG)
Target: "Based on the provided data, the names of the representatives from the D party are:"

T2D (Ours)
~~The~~ → **Based** on the ~~information~~ → **provided** ~~text~~ → **data,** the names of the representatives from the D party are:

ShortGPT
~~The~~ → **Based** on the ~~given~~ → **provided** ~~text~~ → **data,** the ~~represent~~ → **names** of the representatives from the ~~d~~ → **D** party are:

Wide
~~The~~ → **Based** on the provided ~~text~~ → **data,** the ~~represent~~ → **names** of the representatives from the ~~d~~ → **D** party are:

Example 2: Multi-Step Reasoning
Target: "Based on the provided question, the answer is (E) No, the jury's award is discretionary..."

T2D (Ours)
~~The~~ → **Based** on the ~~information~~ → **provided** ~~text~~ → **question,** the answer is (~~X~~ → **E**) No, the ~~passage~~ → **jury's** award is ~~not~~ → **discretionary**

ShortGPT
~~The~~ → **Based** on the ~~information~~ → **provided** ~~text~~ → **question,** the answer is (~~B~~ → **E**) No, ~~because~~ → **the** ~~app~~ → **jury's** award is ~~not~~ → **discretionary**

Wide
~~The~~ → **Based** on the ~~given~~ → **provided** ~~text~~ → **question,** the answer ~~to~~ → **is** (~~X~~ → **E**) ~~Yes~~ → **No,** the ~~evidence~~ → **jury's** award is ~~not~~ → **discretionary**

Example 3: Pedagogical Writing (Open-Ended)
Target: "Sure, I'd be happy to help you write a lesson plan for a kindergarten middle class activity themed "I'm a Little Scientist"..."

T2D (Ours)
~~This~~ → **Sure** , ~~here~~ → **I** 'd be happy to help you write a lesson plan for a kindergarten middle class activity themed "I'm a Little Scientist"

ShortGPT
~~This~~ → **Sure** , ~~here~~ → **I** 'd be happy to help you write a lesson plan for a kindergarten middle class activity themed "I'm a Little Scientist"

Wide
Sure, ~~here~~ → **I** ~~can~~ → **'** d be happy to help you ~~with~~ → **write** a lesson plan for a kindergarten middle class activity ~~!~~ → **themed** ~~by~~ → **"** I'm a Little Scientist"

Figure 6: Examples of speculative decoding outputs across different domains.