Think Big, Generate Quick: LLM-to-SLM for Fast Autoregressive Decoding

Anonymous ACL submission

Abstract

Large language models (LLMs) have become ubiquitous in practice and are widely used for generation tasks such as translation, summarization and instruction following. However, their enormous size and reliance on autoregressive decoding increase deployment costs and complicate their use in latency-critical applications. In this work, we propose a hybrid approach that combines language models of different sizes to increase the efficiency of autoregressive decoding while maintaining high performance. Our method utilizes a pretrained frozen LLM that encodes all prompt tokens once in parallel, and uses the resulting representations to condition and guide a small language model (SLM), which then generates the response more efficiently. We investigate the combination of LLM encoders with both encoder-decoder and decoder-only SLMs from different model families and only require fine-tuning of the SLM. Experiments with various benchmarks show substantial speedups of up to $4\times$, often with only minor performance penalties of $1-2\%$ compared to the LLM.

1 Introduction

The recent widespread adoption of large language models (LLMs) has enabled a variety of applications in the field of natural language generation (NLG), from machine translation (Wu et al., 2016) and code completion (Chen et al., 2021) to general-purpose chatbots (OpenAI, 2023). Their performance is a function of compute, dataset size and parameter count (Kaplan et al., 2020; Hoffmann et al., 2022), with emerging abilities becoming apparent only at large scales (Thoppilan et al., 2022; Chowdhery et al., 2023; Wei et al., 2022a). These findings have led to the increased popularity of large models, both in decoder-only (Scao et al., 2022; Zhang et al., 2022a; Touvron et al., 2023a) and increasingly in encoder-decoder networks (Chung et al., 2022; Wang et al., 2023).

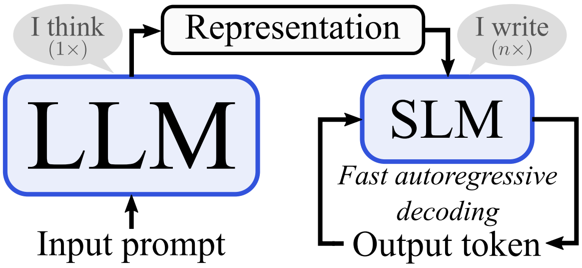

Figure 1: LLM-to-SLM: A large language model (LLM) computes a high-quality representation of the prompt to condition a small language model (SLM), which then efficiently decodes the response while maintaining high performance close to the LLM.

As this race to scale intensifies, LLMs are becoming challenging to deploy, especially in light of compute limitations and latency requirements on edge devices; this ultimately translates into higher costs for providers and end users alike (Chen et al., 2023b). More specifically, LLMs in NLG operate in two phases: (1) First, encoding the user prompt (e.g., `Translate into German: I love you`), followed by (2) decoding of the response (`Ich liebe dich`). In many cases, such as translation or summarization, the prompt is known in advance and can thus be processed efficiently in parallel. However, the response is usually generated in an autoregressive, sequential manner (Radford et al., 2018; Zarrieß et al., 2021): The LLM must be called for each token to be generated, requiring all its weight matrices and the KV cache to be loaded. As a result, decoding becomes bound to the memory bandwidth of the accelerator, which eventually leads to high inference latency as the length of the response grows (Pope et al., 2023).

Research aimed at reducing the overall inference cost of LLMs has garnered significant interest over the past few years. While traditional techniques, e.g. in model compression and parallel decoding, are still active areas of research, recent developments show a pivot towards hybrid approaches that combine models of different sizes for fast decoding, such as speculative decoding (Leviathan et al., 2023; Chen et al., 2023a) or LLM cascades (Chen et al., 2023b). Despite these promising developments, exploiting the discrepancy between the fast prompt encoding phase and the slow response generation remains under-explored.

Intuitively, having a detailed understanding of the prompt is critical for the planning and delivery of an appropriate response. In contrast, autoregressive decoding aims at predicting the next token, which is comparatively low-level and can in certain settings be accomplished well even by SLMs (Eldan and Li, 2023). Following this insight, we propose to reduce the cost of autoregressive decoding with a hybrid model in which these complementary tasks are distributed over two unequally sized networks (Figure 1). Specifically, we perform a single forward pass with an LLM to compute a high-quality representation of the prompt, which is used to condition a more efficient SLM that then performs autoregressive generation. Since the prompt can be encoded in parallel, computing an LLM representation to guide an SLM results in only a minor increase in overall runtime compared to the SLM alone, in particular for tasks such as translation or instruction tuning that require generating longer sequences. We show that this minor increase in runtime compared to the SLM allows for a substantial increase in predictive performance. Overall, we make the following contributions:

- We present LLM-to-SLM, a simple approach for fast autoregressive decoding where an LLM conditions an SLM. We mix LLM encoders with both encoder-decoder and decoder-only language models and only require fine-tuning of the SLM.
- We empirically evaluate the efficacy of different LLM-to-SLM variants in various domains, across both training and evaluation regimes: traditional fine-tuning for both machine translation and summarization, as well as instruction-tuning evaluated in the zero-shot setting on multiple held-out tasks.
- Our method accelerates pretrained LLMs while maintaining high performance. For example, in translation and summarization, LLM-to-SLM achieves a speedup of $4.2\times$ and $3.0\times$, respectively, with a marginal drop of $< 1\%$ in predictive performance.

2 Related Work

The duality of model performance and cost has sparked a lot of research interest in LLM efficiency, which is approached from various angles.

Model compression. A common approach to accelerate LLMs is to create a simpler compressed version using pruning (Frantar and Alistarh, 2023; Ma et al., 2023; Sun et al., 2023) or quantization (Dettmers et al., 2022; Yao et al., 2022; Xiao et al., 2023). A third pillar of model compression is knowledge distillation where a small model learns from outputs of larger models (Hinton et al., 2015). For the language domain, chain-of-thought prompting (Wei et al., 2022b) can be applied to generate samples from an LLM teacher, that can subsequently be used by an SLM student as training signal (Ho et al., 2022; Magister et al., 2022; Li et al., 2023; Shridhar et al., 2023; Hsieh et al., 2023). LLM-to-SLM can be regarded as a model compression technique, where the decoder responsible for generation is replaced by a smaller, more efficient model, while retaining the LLM for prompt encoding.

Parallel decoding. Instead of predicting one token at a time, parallel decoding aims at generating multiple tokens at once (Gu et al., 2018; Wang et al., 2019; Sun et al., 2019; Wei et al., 2019). While empirical speedups are possible, this approach raises new challenges such as inconsistencies in the output and the need to estimate the length of the response, as language is inherently conditional. In practice, it may require multiple iterations (Ghazvininejad et al., 2019; Lee et al., 2018; Stern et al., 2018; Santilli et al., 2023; Fu et al., 2023), or a combination of parallel and sequential decoding (Ning et al., 2023). Our method sticks to the more commonly used autoregressive decoding but uses a small model to decrease inference time.

Conditional approaches. In comparison to the language domain, where conditioning of multiple networks is still relatively rare, it is used to a greater extent in multimodal learning. For example, various works have conditioned a language model on vision features (Driess et al., 2023; Chen et al., 2023c; Zhou et al., 2022; Liu et al., 2023a). Interestingly, Driess et al. (2023) and Liu et al. (2023a) study freezing parts of their models, which is related to how we freeze the LLM during fine-tuning. In vision, there is also a growing trend of conditioning small transformer decoders on larger encoders (Bergner et al., 2022; He et al., 2022; Jaegle et al., 2021). Instead, our method connects a small network to a large network for NLG.

Learned prompts. Our method is related to parameter-efficient fine-tuning (PEFT) techniques that incorporate trainable prompts, which can be both continuous (Lester et al., 2021; Li and Liang, 2021; Liu et al., 2023b) or discrete (Deng et al., 2022; Prasad et al., 2022; Zhang et al., 2022b). In particular, Lester et al. (2021) prepend soft prompts to the input and freeze all other parameters. This is similar to the way we fuse the LLM representation into the SLM. We experiment with both trainable and fixed SLMs, with only a projector being trained in the latter case. In contrast to prompting methods, our main focus is on reducing inference cost.

SLMs. The most straightforward route to efficiency is smaller models. Schick and Schütze (2021) showed that SLMs can do few-shot learning and outperform GPT3 in the SuperGLUE benchmark. Another direction is to train language models on limited vocabulary (Warstadt et al., 2023; Huebner et al., 2021). Notably, TinyStories (Eldan and Li, 2023) learns various SLMs on LLM-generated stories using vocabulary that a 4-year-old child can normally understand and demonstrates coherent English-generated text. While these works have much future potential, we are investigating a more practical hybrid model with wide applicability in NLG tasks that combines high performance of LLMs with efficiency of SLMs.

Hybrids. Most related to our work are hybrid models that employ both LLM and SLM, where the latter performs the bulk of computation. Chen et al. (2023b) propose a language model cascade, where cheaper models are invoked first. Jiang et al. (2023) recycle the representation of the LLM by passing it to a smaller model that predicts the subsequent token more efficiently. Speculative decoding (SD) methods repeatedly call the SLM to generate a draft that is then validated in parallel by the LLM (Chen et al., 2023a; Leviathan et al., 2023; Kim et al., 2023). Medusa (Cai et al., 2023) is related but attaches heads on top of the LLM to predict multiple tokens in parallel. During generation, the LLM in SD is invoked several times, whereby the frequency depends on the performance of the SLM. In our method, the LLM is only called once and the SLM is conditioned on its representation.

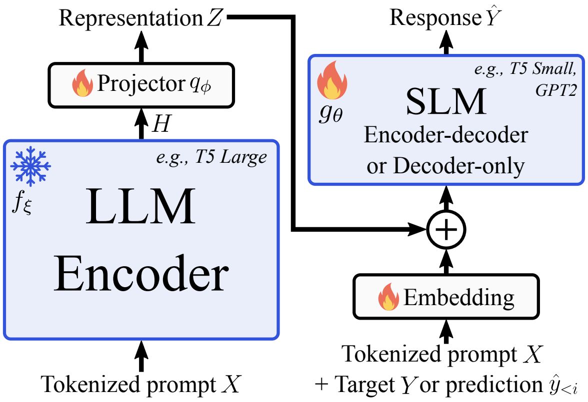

Figure 2: Architecture details. A frozen LLM integrates projected representations into either a trainable encoder-decoder or a decoder-only SLM.

Note that our method is orthogonal to most of these related works. For example, one may combine our method with quantization, pruning or other hybrid approaches. In Section 4.5, we demonstrate that our approach can be combined together with speculative decoding.

3 LLM-to-SLM

It is well established that model capacity and overparametrization play a crucial factor in model performance (Kaplan et al., 2020; Hoffmann et al., 2022). Following this insight, the core idea of LLM-to-SLM is to compensate the low parameter count of an SLM by conditioning its next token prediction on a high-quality representation of the prompt given by an LLM.

Figure 2 presents an overview of LLM-to-SLM: First, the **LLM** encoder f_ξ computes a high-quality representation of the prompt. The **projector** q_ϕ then adapts and projects this representation to the SLM embedding space. Finally, the **SLM** g_θ takes the projected representation as input and generates the output tokens in an autoregressive manner. Crucially, the parameter count of the SLM is significantly smaller than the LLM ($8-55\times$ in our experiments), leading to faster generation as only the SLM performs autoregressive decoding. In the remainder of this section, we further describe the individual modules of LLM-to-SLM and explain how the representations of the LLM are injected into both encoder-decoder and decoder-only SLMs.

3.1 Fast autoregressive decoding

Given a prompt $X = [x_1, \dots, x_m]$ and an encoder-decoder LLM, autoregressive decoding models the output $Y = [y_1, \dots, y_n]$ in a causal manner:

$$p(Y|X) = \prod_{i=1}^{n} g_\xi(y_i|y_{<i}, f_\xi(x_{1:m})), \quad (1)$$

where f_ξ and g_ξ refer to the LLM encoder and decoder, respectively. Generating the complete sequence Y thus requires n very costly forward passes to the LLM decoder g_ξ. Furthermore, these calls can not be parallelized as we need to first sample the token y_i to estimate the probability distribution over the $i+1$-th token. Instead, we propose to delegate the costly autoregressive decoding calls to a smaller language model, while preserving the encoder capacity:

$$p(Y|X) = \prod_{i=1}^{n} g_\theta(y_{<i}, x_{1:m}, q_\phi(f_\xi(x_{1:m}))). \quad (2)$$

The LLM is now only called once to provide a high quality encoding of the input prompt to the SLM. Therefore, as the number of autoregressive steps n increases, the runtime of our method converges towards the original runtime of the SLM.

3.2 Architecture

3.2.1 LLM encoder

The LLM encoder $f_\xi : X \mapsto H$ takes a prompt X of length m as input and computes a high-quality representation $H \in \mathbb{R}^{m \times d_l}$ of it. In training LLMs, the requirement for substantial computational resources is often a limiting factor. In our approach, we focus on a more resource-efficient training regime closer to that of the SLM, by freezing the parameters of the LLM during fine-tuning. In this way, we can train LLM-to-SLM on a small budget by pre-computing prompt representations. To ensure good representation quality, we leverage large pretrained encoder-decoder models, but omit the decoder. The last layer representation of the encoder in an encoder-decoder architecture serves as a straightforward prompt encoding point for an SLM. In a decoder-only model, in contrast, determining the exact intermediate layer(s) that contribute to encoding of the prompt is more challenging as the last layer representation is mainly useful for the low-level task of predicting the next token (Chen et al., 2020). In Appendix B, we present preliminary results for using a decoder-only LLM in our LLM-to-SLM framework.

3.2.2 Projector

The projector $q_\phi : H \mapsto Z$ has the task of aligning representations between LLM and SLM. It takes high-dimensional features $H \in \mathbb{R}^{m \times d_l}$ of the LLM as input and computes a lower-dimensional representation $Z \in \mathbb{R}^{m \times d_s}$ which can be fused directly with embeddings of the SLM. Although there are many ways to design such a projector, we found that using a small MLP: `Linear(`d_l`, `d_s`)` $\rightarrow$ `ReLU` $\rightarrow$ `Linear(`d_s`, `d_s`)` trained from scratch is simple, efficient and performs well.

3.2.3 SLM

The SLM $g_\theta : (X, Z) \mapsto \hat{Y}$ maps the tokenized input X and the projected representation $Z \in \mathbb{R}^{m \times d_s}$ to the response $\hat{Y}$ of length n. During training, we also append the target sequence Y to the input for next token prediction via teacher forcing (Williams and Zipser, 1989). Equivalently, predicted tokens $\hat{y}_{<i}$ are appended during inference. We employ pretrained networks as SLMs, but fine-tune them as they have not been previously trained to process high-capacity encodings. Furthermore, we perform experiments in which we learn either from the ground truth signal provided by a dataset or from sequences generated by the LLM.

Feature integration. A critical design decision in our framework is the way in which the SLM is conditioned. We intend to integrate LLM representations into the SLM at an early stage, as this allows us to treat the internal structure of the SLM as a black box and use both encoder-decoder and decoder-only SLMs in a simple and unified way. Initially, the tokenized input $(X; Y \text{or } \hat{y}_{<i})$ is embedded using the embedding matrix of the SLM. A straightforward way to fuse features is then to *replace* the SLM embedding of the prompt with the projected LLM representation. This is similar to soft prompt tuning (Lester et al., 2021), with the difference that our prompts are conditioned on LLM features. An alternative strategy used in the main experiments is *adding* the projected LLM representation Z onto the SLM prompt embedding E_X, such that $Z + E_X$ is the input to the SLM. In this way, we preserve the semantics between SLM embeddings of prompt and target/predicted sequence and leverage the LLM representation to modulate the prompt embedding via addition (see Section 4.5 for a comparison of the two approaches).

Aligning sequence lengths. When fusing the LLM representation with the SLM embedding of the prompt via addition, sequence lengths must be aligned, which is not guaranteed when combining models from different families that may use different tokenizers and vocabularies. To this end, we propose to reuse the tokenizer and the embedding matrix of the LLM to align sequence lengths. In this case, we employ two new linear layers: (1) an embedding projection layer that maps these LLM embeddings from dimension d_l to the SLM embedding space of dimension d_s and (2) a new head layer that replaces the original SLM head and maps to the vocabulary used by the LLM. Finally, note that there are various other options for fusing the features of different models, e.g. cross-attention, or FiLM layers (Perez et al., 2018). However, we opted for adding, as it is simple and requires minimal changes to the SLM.

4 Experiments

In this section, we intend to answer the following question: What is the comparative performance and runtime of our proposed LLM-to-SLM method in relation to LLM and SLM alone? To this end, we first empirically evaluate the efficacy of LLM-to-SLM on three tasks: machine translation, summarization, and instruction tuning. We then report the computational efficiency of our proposed method in Section 4.4. Finally, in Section 4.5 we present a comprehensive set of ablation studies, investigating the performance of LLM-to-SLM under varying SLM capacities, its orthogonality to speculative decoding, and how it compares to PEFT methods.

Setup. The networks used in our experiments are listed in Table 1. We employ various pretrained models and architectures and denote combinations as LLM → SLM. We make use of T5 encoders as LLMs, and employ T5 encoder-decoder and GPT2 decoder-only models as SLMs: In this setting, the LLMs have $8-55\times$ more parameters than the SLMs. For generation, we use beam search (beam width of 4, length penalty of 0.6) for translation and summarization, and nucleus sampling for instruction tuning.

We report task-specific performance metrics: SacreBLEU (Post, 2018) for translation, ROUGE (Lin, 2004) for summarization and GPT4 as a judge for instruction tuning, using the same generation settings as (Zheng et al., 2023). Furthermore, we report runtimes per single generated token (in milliseconds). These are calculated from generating a total of 100 tokens with a prompt length of also 100 tokens, either on an NVIDIA V100 (translation, summarization) or NVIDIA A100 (instruction tuning) GPU.

	Model	Params
Enc-Dec	T5 Small† (Raffel et al., 2020)	44M (19M/25M)
	T5 1.1 Small† (Raffel et al., 2020)	44M (19M/25M)
	T5 Large* (Chung et al., 2022)	737M (302M, 402M)
	Flan T5 Base† (Chung et al., 2022)	198M (85/113M)
	Flan Alpaca Base (Chia et al., 2023)	198M (85/113M)
	Flan T5 XXL (Chung et al., 2022)	10.9B (4.6/6.2B)
	Flan Alpaca XXL* (Chia et al., 2023)	10.9B (4.6/6.2B)
Dec-only	GPT2† (Radford et al., 2019)	86M
	GPT2 [1,2,4]-Layers†	[8M, 15M, 29M]
	GPT2 XL (Radford et al., 2019)	1.5B
	LLaMA 13B (Touvron et al., 2023a)	12.7B

Table 1: Model variants used in the experiments. Sizes are rounded, excluding embedding and head parameters. Encoder/decoder sizes are shown in parentheses. Symbols $*$ and † denote models that we use in our method as LLMs and SLMs respectively.

Training. All models are trained with an effective batch size of 128, cross-entropy loss, AdamW optimizer (Loshchilov and Hutter, 2017) with weight decay of 0.1, learning rate of 0.001 with linear warmup (Goyal et al., 2017) for 10% of the total number of iterations, followed by cosine learning rate decay to 0 (Loshchilov and Hutter, 2016). We rely on Huggingface's transformers (Wolf et al., 2019) for training and generation. Further training details are listed in Appendix C.

4.1 Machine translation

We report results for the translation task from English to German, French, and Romanian. We utilize WMT14 (Bojar et al., 2014) for En-Fr/De, and WMT16 for En-Ro (Bojar et al., 2016). T5 Large serves as LLM encoder, and T5 Small, T5 1.1 Small, and GPT2 as SLMs. T5 Small and GPT2 are $16\times$ and $8\times$ smaller than T5 Large. Both T5 Large/Small are pretrained for translation, while T5 1.1 Small was only trained on the C4 dataset. The following task description precedes the prompt: `"translate English to *target-language*:"`. All models are trained for $50k$ iterations, except for T5 Large which comes pretrained for translation and is applied in a zero-shot manner. We use our LLM to generate training labels. We report BLEU scores evaluated on the test splits in Table 2.

T5 Large shows an average score of 31.94, while T5 Small performs 2 BLEU points worse. SLMs that have not been previously pretrained for translation (T5 1.1 Small and GPT2) perform more than 3 BLEU points worse than the LLM. In comparison,

Model	En-Fr	En-De	En-Ro	Avg.	Time
T5 Large (zero-shot)	39.53	29.10	27.19	31.94	61.5
T5 Small	37.16	26.47	26.15	29.93	14.2
T5 1.1 Small	34.85	24.55	25.25	28.22	18.4
GPT2	35.67	25.53	24.70	28.63	19.7
T5 Large → T5 Small	39.22	28.36	27.04	31.54	14.8
T5 Large → T5 1.1 Small	38.21	27.42	26.54	30.72	19.1
T5 Large → GPT2	39.01	28.27	25.86	31.05	20.7

Table 2: BLEU scores for machine translation. LLM-to-SLM models approach the performance of the LLM.

our LLM-to-SLM variants reduce this gap to less than 1 point: For example, T5 Large → T5 Small achieves on average a BLEU score of 31.54 across languages. Furthermore, gains over the SLM baseline are more pronounced when using pretrained networks that were not previously trained on translation, e.g., T5 Large → T5 1.1 Small scores on average 2.5 points better than T5 1.1 Small (8.8% increase). Finally, despite having different model families, we also observe significant gains in T5 Large → GPT2 (8.4% increase), and the task performance improvements come at a marginal trade-off in runtime.

4.2 Summarization

We further assess the performance of LLM-to-SLM when combining models from different families for the task of summarization on CNN/Daily Mail (Hermann et al., 2015). We again use T5 Large as LLM and GPT2 as SLM. Similar to translation, T5 Large comes pretrained for summarization. Following Raffel et al. (2020), we build the input prompt by prefixing the input text with `"summarize:"`. We fine-tune all models for $25k$ iterations on the training set and evaluate on the test split. In contrast to translation, we directly train from ground-truth labels as we found it to perform better than distillation in this setting. ROUGE scores and runtimes are reported in Table 3.

Model	R-1	R-2	R-L	Avg.	Time
T5 Large (zero-shot)	40.07	18.84	28.82	29.07	61.5
GPT2 XL (zero-shot)	29.34	8.27	26.58	21.40	78.6
GPT2 XL	40.47	19.09	28.90	29.49	78.6
GPT2	38.58	17.56	27.36	27.83	19.7
T5 Large → GPT2	40.22	18.64	28.80	29.22	20.7

Table 3: ROUGE scores (abbreviated with R-) on CNN/Daily Mail. GPT2 XL (zero-shot) results are from Radford et al. (2019).

GPT2 exhibits an average ROUGE score that is 1.24 points lower than that of T5 Large. In contrast, T5 Large → GPT2 slightly exceeds the average score of T5 Large and shows a 3× speedup. Importantly, our T5 Large → GPT2 model performs on par with a fully fine-tuned GPT2 XL model while having a decoder which is 17× smaller.

4.3 Instruction tuning

We also explore the potential of our method in a challenging instruction tuning setting. In contrast to traditional fine-tuning, where a single task is employed for training and evaluation, instruction-following models are trained on a multitude of tasks and evaluated for general problem solving on held-out tasks. We use Flan Alpaca XXL as LLM, which is a 11B parameter T5 trained on both Flan (Chung et al., 2022) - a collection of 1,800 tasks - as well as the Alpaca dataset (Taori et al., 2023; Wang et al., 2022), which consists of 52k generated instruction-following demonstrations. For evaluation, we use the MT-bench dataset, which consists of 80 challenging tasks from 8 categories (Zheng et al., 2023), with GPT4 as a judge. As SLM, we utilize Flan T5 Base, i.e. a T5 that is only pretrained on Flan data, and fine-tune it on Alpaca data. Note that our LLM, Flan Alpaca XXL, has 55× more parameters than Flan T5 Base.

The results are reported in Table 4. Our LLM obtains an average score of 3.2, and performs best in writing, extraction and roleplay. In contrast, Flan Alpaca Base only has a relative score of 57.5%, which suggests that scale plays an important role. We rerun this setting by fine-tuning Flan T5 Base on Alpaca data, and increase the score to 62.3%. Our LLM-to-SLM further increases the score to 73.4%, more than 10 percentage points better than the SLM alone and competing with a zero-shot evaluated Flan T5 XXL (75.8%) and LLaMA 13B (82.7%). In addition, Table 4 indicates that our LLM-to-SLM achieves a much better trade-off between performance and runtime compared to all other models. Finally, we present qualitative examples in Appendix E demonstrating that our LLM-to-SLM can produce fluent and sensible responses.

4.4 Computational efficiency

We show runtimes for each task in Tables 2, 3 and 4 and present performance-runtime trade-off curves in Figure 3. In machine translation, we observe more than 4× speedup of T5 Large → T5 Small compared to T5 Large for a marginal loss in performance. We see similar speedups between $2 - 3\times$ for summarization and instruction tuning. Note that greater gains in efficiency are generally pos-

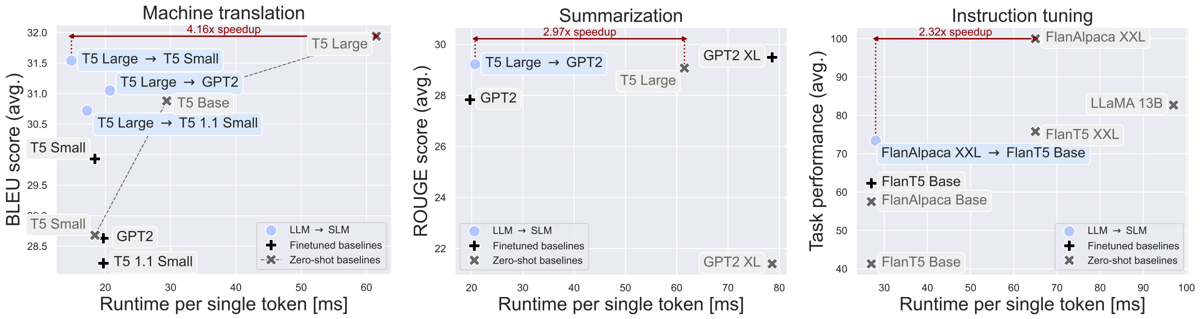

Figure 3: Performance-runtime trade-off curves for various models across different tasks.

Model	Coding	Extraction	Humanities	Math	Reasoning	Roleplay	Stem	Writing	Avg.	Time
Flan Alpaca XXL	1.0	4.1	3.4	1.1	2.7	4.0	3.6	5.3	3.2	65
Flan T5 XXL	100.0	78.0	47.1	190.9	114.8	67.5	69.4	54.7	75.8	65
LLaMA 13B	110.0	61.0	98.5	95.5	94.4	101.3	70.8	69.8	82.7	97
Flan T5 Base	100.0	39.0	38.2	145.5	44.4	35.0	33.3	20.8	41.3	27
Flan Alpaca Base	100.0	24.4	32.4	90.9	92.6	72.5	58.3	54.7	57.5	27
Flan T5 Base*	100.0	24.4	79.4	90.9	51.9	65.0	58.3	73.6	62.3	27
Flan Alpaca XXL → Flan T5 Base*	160.0	39.0	70.6	100.0	66.7	87.5	69.4	75.5	73.4	28

Table 4: MT-bench results. The first row shows absolute scores of the LLM. All subsequent rows indicate scores relative to the LLM in %. Symbol $*$ denotes finetuned models, all other models are evaluated zero-shot. LLM-to-SLM (highlighted) scores more than 10 absolute percentage points better than the SLM while competing with LLMs whose decoder is $50-100\times$ larger. See Table 11 for qualitative results.

sible by either increasing the size of the LLM or decreasing the size of the SLM, as demonstrated by our ablation on tiny SLMs in Section 4.5. Crucially, Figure 3 shows that our LLM-to-SLM variants are only marginally slower than the SLM, but performing significantly better. In Appendix D, we provide further insights into the relationship between generation length and runtime/FLOPs and show that our LLM-to-SLM approaches the computational efficiency of the SLM.

4.5 Ablation study

Tiny SLMs. An alternative approach to improve inference efficiency is to downscale the SLM. We revisit T5 Large → GPT2 for machine translation and truncate the upper layers of a pretrained GPT2, resulting in GPT2 models with $L \in \{1, 2, 4\}$ layers, and only 8M, 15M, and 29M parameters, respectively. Figure 4 demonstrates that LLM-to-SLM models with truncated SLMs outperform corresponding SLM baselines by a wide margin while maintaining nearly the same runtime.

Speculative decoding. While our empirical evaluations on translation and summarization indicate that LLM-to-SLM can achieve a comparable performance to the LLM, the instruction tuning results show that despite significant improvement over the

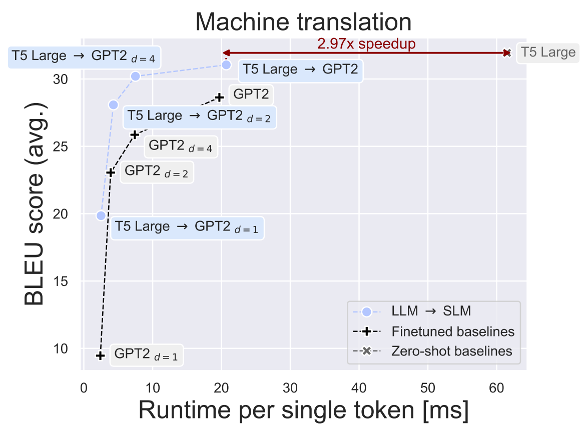

Figure 4: Performance-runtime comparison with tiny GPT2 versions (d indicates maximum depth) as SLMs for machine translation. The y-axis shows the average BLEU score across languages. T5 Large → GPT2 with only 4 layers outperforms GPT2. The smaller the SLM, the greater the gap to our LLM-to-SLM models.

SLM, LLM-to-SLM may still lag behind the performance of the LLM. This could be due to the evaluation regime (zero-shot on held-out tasks) and the breadth and complexity of the tasks in MT-bench. Speculative decoding (SD), in contrast, guarantees matching the distribution of the LLM, albeit at the cost of invoking the LLM multiple times (Chen et al., 2023a). Since our method is orthogonal to SD, we apply it jointly to obtain LLM performance while still achieving speedups. Specifically, we consider our Flan Alpaca XXL → Flan T5 Base as the draft model in which we apply the encoders of our LLM and SLM only once to encode the prompt. We then repeatedly call the decoder of our SLM for autoregressive generation of the draft sequences. The results in Table 5 show that we can match the performance of the LLM with our LLM-to-SLM, while still being $1.5\times$ faster (compared to $2.3\times$ speedup of LLM-to-SLM alone).

Model	Score	Time	Speedup
LLM	3.2	65	1×
SLM	2.0	27	2.4×
SLM + SD	3.3	44	1.4×
LLM → SLM	2.3	28	2.3×
LLM → SLM + SD	3.2	42	1.5×

Table 5: Speculative decoding (SD) results. Average runtime reported in ms/token. LLM: Flan Alpaca XXL, SLM: Flan T5 Base.

Parameter-efficient fine-tuning. Our method has similarities with PEFT methods. In particular, soft prompt tuning prepends learnable tokens to the prompt embedding. In contrast, we add LLM representations element-wise to the prompt embedding of the SLM, which can be understood as a form of conditional prompting. In Table 6, we compare the performance of PEFT methods with our approach by freezing the SLM and allowing only the training of the projector. Hyperparameters of the PEFT methods (see Appendix A for hyperparameter details) are adjusted so that the total number of parameters matches our projector. The results reported in Table 6 show that, for both summarization and translation, our method outperforms all PEFT methods, indicating that conditioning on the LLM has a positive effect on performance compared to non-conditional approaches.

Model	WMT	CNN/DN
Prompt tuning (Lester et al., 2021)	25.06	19.52
Prefix tuning (Li and Liang, 2021)	24.93	21.64
LoRA (Hu et al., 2021)	26.36	22.35
LLM-to-SLM (ours)	30.27	24.13

Table 6: PEFT results using fixed T5 Small for translation and GPT2 for summarization. Reported are average BLEU scores across languages (WMT) and average ROUGE scores (CNN/DN). In our method, T5 Small and GPT2 are conditioned on T5 Large representations.

Model	GT +	GT ×	Gen. +	Gen. ×
T5 Large → T5 Small	30.83	30.86	31.54	31.47
T5 Large → GPT2	29.68	29.71	31.05	30.83

Table 7: Ablation of training signal and fusion operator, reporting average BLEU scores for translation across languages. GT: Ground truth labels, Gen.: Labels generated by T5 Large, ×: Replacement, +: Addition

LLM vs. SLM tokenizer. When replacing SLM embeddings of the prompt with projected LLM representations (see Section 3.2.3), the tokenizer of the SLM can be applied even if its vocabulary is different from that of the LLM tokenizer. In this case, the additional embedding down-projection and head layers can be omitted. In T5 Large → GPT2 for machine translation, we found that using the LLM tokenizer performs better by 0.45 BLEU points (average across all languages), which could be due to the fact that the vocabulary of the T5 tokenizer also covers non-English languages.

Embedding replacement vs. addition. We compare replacing SLM embeddings with vs. adding SLM embeddings to the projected LLM representations as feature fusion strategies. Table 7 shows results for machine translation, indicating that addition and replacing perform on par with each other.

Ground truth vs. LLM-generated labels. In translation, we found that using labels generated by the LLM for training performs up to 1 BLEU point better than using ground truth labels (Table 7). However, we point out that this is not generally the case. For summarization, the average ROUGE score is 0.44 points better when training with ground truth labels compared to LLM-generated labels.

5 Conclusion

In this work, we proposed LLM-to-SLM, a novel framework for accelerating autoregressive decoding. LLM-to-SLM exploits the discrepancy between fast prompt encoding and costly autoregressive generation by using a combination of LLM and SLM. The LLM is used to compute a high-quality representation of the prompt for the planning of an appropriate response. The SLM, conditioned on this representation, then efficiently decodes the response. Our experiments across various benchmarks demonstrate substantial speedups between $2-4\times$, with minimal performance penalties between $1-2\%$ compared to the LLM.

Limitations

Our evaluations show that the difference in performance of LLM-to-SLM compared to the LLM alone is often marginal in the traditional fine-tuning setting. However, in challenging tasks such as instruction tuning, there is still a larger gap in performance compared to the LLM. One limitation of our method is that the LLM is only used once to encode the prompt. However, it is foreseeable that the LLM could be used more frequently in more challenging tasks to guide the planning of the response, ideally through dynamic invocations where required. This is an important direction for our future work.

A second limitation of this work is that we mainly investigated encoder-decoder networks as LLMs. We provide preliminary results of decoder-only models as LLMs within our framework in Appendix B, but noticed that performance is highly dependent on the layer from which the LLM features are extracted. Further investigation is needed to see how decoder-only features can be used in the LLM-to-SLM setting.

Finally, we only used encoder-decoder based models with up to several billion parameters as LLMs, which are relatively small compared to the largest decoder-only LLMs such as GPT4, Llama 2 or OPT (OpenAI, 2023; Touvron et al., 2023b; Zhang et al., 2022a). This is mainly due to the fact that most of the largest models are decoder-only and, as mentioned previously, further study into incorporating decoder-only models as LLMs is needed in our LLM-to-SLM framework. Investigating the impact of using such large models is an important future research direction.

Ethical Considerations

Our proposed approach poses risks similar to existing works on language models (Brown et al., 2020; Touvron et al., 2023b). However, as our approach proposes a way of enhancing models in the low-computation regime, it might be used for improving edge device capabilities. Since edge devices range from mobile phones to surveillance tools, our approach could be both beneficial if used properly and harmful for the broader society if misused.

References

Benjamin Bergner, Christoph Lippert, and Aravindh Mahendran. 2022. Iterative patch selection for high-resolution image recognition. In *International Conference on Learning Representations*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, et al. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pages 12–58.

Ondrej Bojar, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, et al. 2016. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*, pages 131–198. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, and Pranav et al. Shyam. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

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

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

Lingjiao Chen, Matei Zaharia, and James Zou. 2023b. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*.

Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. 2020. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*.

Xi Chen, Xiao Wang, Soravit Changpinyo, AJ Piergiovanni, Piotr Padlewski, Daniel Salz, Sebastian Goodman, Adam Grycner, Basil Mustafa, Lucas Beyer, et al. 2023c. Pali: A jointly-scaled multilingual language-image model. In *International Conference on Learning Representations*.

Yew Ken Chia, Pengfei Hong, and Soujanya Poria. 2023. Flan-alpaca: Instruction tuning from humans and machines.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Mingkai Deng, Jianyu Wang, Cheng-Ping Hsieh, Yihan Wang, Han Guo, Tianmin Shu, Meng Song, Eric P Xing, and Zhiting Hu. 2022. RLPrompt: Optimizing discrete text prompts with reinforcement learning. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 3369–3391. Association for Computational Linguistics.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. LLM.int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. 2023. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*.

Ronen Eldan and Yuanzhi Li. 2023. Tinystories: How small can language models be and still speak coherent english? *arXiv preprint arXiv:2305.07759*.

Elias Frantar and Dan Alistarh. 2023. Sparsegpt: Massive language models can be accurately pruned in one-shot.

Yichao Fu, Peter Bailis, Ion Stoica, and Hao Zhang. 2023. Breaking the sequential dependency of llm inference using lookahead decoding.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. 2019. Mask-predict: Parallel decoding of conditional masked language models. In *Conference on Empirical Methods in Natural Language Processing*.

Priya Goyal, Piotr Dollár, Ross Girshick, Pieter Noordhuis, Lukasz Wesolowski, Aapo Kyrola, Andrew Tulloch, Yangqing Jia, and Kaiming He. 2017. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28.

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

Namgyu Ho, Laura Schmid, and Se-Young Yun. 2022. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. 2023. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. In *ACL*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Philip A Huebner, Elior Sulem, Fisher Cynthia, and Dan Roth. 2021. Babyberta: Learning more grammar with small-scale child-directed language. In *Conference on Computational Natural Language Learning*, pages 624–646.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, Carl Doersch, Catalin Ionescu, David Ding, Skanda Koppula, Daniel Zoran, Andrew Brock, Evan Shelhamer, et al. 2021. Perceiver io: A general architecture for structured inputs & outputs. *arXiv preprint arXiv:2107.14795*.

Yufan Jiang, Qiaozhi He, Xiaomin Zhuang, Zhihua Wu, Kunpeng Wang, Wenlai Zhao, and Guangwen Yang. 2023. Recyclegpt: An autoregressive language model with recyclable module. *arXiv preprint arXiv:2308.03421*.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*.

Sehoon Kim, Karttikeya Mangalam, Jitendra Malik, Michael W Mahoney, Amir Gholami, and Kurt Keutzer. 2023. Big little transformer decoder. *arXiv preprint arXiv:2302.07863*.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Conference on Empirical Methods in Natural Language Processing*.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*.

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

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023. Symbolic chain-of-thought distillation: Small models can also" think" step-by-step. In *Findings of the Association for Computational Linguistics*.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. 2023a. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2023b. Gpt understands, too. *AI Open*.

Ilya Loshchilov and Frank Hutter. 2016. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

Xinyin Ma, Gongfan Fang, and Xinchao Wang. 2023. Llm-pruner: On the structural pruning of large language models. *arXiv preprint arXiv:2305.11627*.

Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Teaching small language models to reason. *arXiv preprint arXiv:2212.08410*.

Xuefei Ning, Zinan Lin, Zixuan Zhou, Huazhong Yang, and Yu Wang. 2023. Skeleton-of-thought: Large language models can do parallel decoding. *arXiv preprint arXiv:2307.15337*.

OpenAI. 2023. Gpt-4 technical report.

Ethan Perez, Florian Strub, Harm De Vries, Vincent Dumoulin, and Aaron Courville. 2018. Film: Visual reasoning with a general conditioning layer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32.

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

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

Archiki Prasad, Peter Hase, Xiang Zhou, and Mohit Bansal. 2022. Grips: Gradient-free, edit-based instruction search for prompting large language models. *arXiv preprint arXiv:2203.07281*.

Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. 2018. Improving language understanding by generative pre-training.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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

Andrea Santilli, Silvio Severino, Emilian Postolache, Valentino Maiorca, Michele Mancusi, Riccardo Marin, and Emanuele Rodolà. 2023. Accelerating transformer inference for translation via parallel decoding. *arXiv preprint arXiv:2305.10427*.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2022. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*.

Timo Schick and Hinrich Schütze. 2021. It's not just size that matters: Small language models are also few-shot learners. In *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*.

Kumar Shridhar, Alessandro Stolfo, and Mrinmaya Sachan. 2023. Distilling reasoning capabilities into smaller language models. In *Findings of the Association for Computational Linguistics*, pages 7059–7073.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. 2018. Blockwise parallel decoding for deep autoregressive models. In *Conference on Neural Information Processing Systems*.

Mingjie Sun, Zhuang Liu, Anna Bair, and J Zico Kolter. 2023. A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.

Zhiqing Sun, Zhuohan Li, Haoqing Wang, Di He, Zi Lin, and Zhihong Deng. 2019. Fast structured decoding for sequence models. In *Conference on Neural Information Processing Systems*.

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

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

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

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

Yiren Wang, Fei Tian, Di He, Tao Qin, ChengXiang Zhai, and Tie-Yan Liu. 2019. Non-autoregressive machine translation with auxiliary regularization. In *AAAI Conference on Artificial Intelligence*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2022. Self-instruct: Aligning language model with self generated instructions. *arXiv preprint arXiv:2212.10560*.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi DQ Bui, Junnan Li, and Steven CH Hoi. 2023. Codet5+: Open code large language models for code understanding and generation. *arXiv preprint arXiv:2305.07922*.

Alex Warstadt, Leshem Choshen, Aaron Mueller, Adina Williams, Ethan Wilcox, and Chengxu Zhuang. 2023. Call for papers–the babylm challenge: Sample-efficient pretraining on a developmentally plausible corpus. *arXiv preprint arXiv:2301.11796*.

Bingzhen Wei, Mingxuan Wang, Hao Zhou, Junyang Lin, Jun Xie, and Xu Sun. 2019. Imitation learning for non-autoregressive neural machine translation. In *Findings of the Association for Computational Linguistics*.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022a. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

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

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. 2023. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR.

Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. 2022. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. *Advances in Neural Information Processing Systems*, 35:27168–27183.

Sina Zarrieß, Henrik Voigt, and Simeon Schüz. 2021. Decoding methods in neural language generation: a survey. *Information*, 12(9):355.

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

Tianjun Zhang, Xuezhi Wang, Denny Zhou, Dale Schuurmans, and Joseph E Gonzalez. 2022b. Tempera: Test-time prompt editing via reinforcement learning. In *The Eleventh International Conference on Learning Representations*.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. 2022. Conditional prompt learning for vision-language models. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 16816–16825.

Appendix

A PEFT hyperparameters

Table 8 presents the hyperparameters for the prompt tuning, prefix tuning, and LoRA methods we considered in the parameter-efficient fine-tuning ablation in our paper.

PEFT method	hyperparameter	GPT2	T5 Small
Prompt tuning	# prompt tokens	200	1024
Prefix tuning	# prefix tokens	256	1024
	project. hidden size	64	64
LoRA	Rank	40	24
	α	64	32
	Dropout	0.1	0.1

Table 8: Hyperparameters for prompt tuning, prefix tuning and LoRA.

B Using decoder-only models as LLM

We present preliminary results for using decoder-only models as an LLM in our LLM-to-SLM framework on the summarization task. We use a fine-tuned GPT2 XL as the LLM and GPT2 Base as the SLM in our experiments. Compared to encoder-decoder based LLMs, there is no specific layer in the model which can explicitly be identified as the encoded representation of the prompt. Therefore, we experiment with extracting representations from different layers of the GPT XL model before passing it to the projector and subsequently to the SLM. We also explore the setting where all layer representations are linearly combined, where layer weights are learnable parameters. The results are presented in Table 9. Surprisingly, we observe that the performance deteriorates with the depth of the LLM. Furthermore, LLM-to-SLM still performs slightly better than the SLM alone by using very early layer representations of GPT2 XL.

Model	R-1	R-2	R-L	Avg.
GPT2 XL	40.47	19.09	28.90	29.49
GPT2	38.58	17.56	27.36	27.83
GPT2 XL $\rightarrow$ GPT2 (layer 0)	39.16	17.71	27.43	28.10
GPT2-XL $\rightarrow$ GPT2 (layer 1)	39.05	17.63	27.40	28.03
GPT2-XL $\rightarrow$ GPT2 (layer 4)	36.66	16.46	25.64	26.25
GPT2-XL $\rightarrow$ GPT2 (layer 8)	33.27	14.03	23.24	23.51
GPT2-XL $\rightarrow$ GPT2 (all layers)	37.69	16.72	26.41	26.94

Table 9: ROUGE scores (abbreviated with R-) on CNN/Daily Mail using GPT2 XL as LLM. Layer 0 refers to the initial embedding layer.

C Extra training details

Details on computational resources used for training and evaluation are specified in Table 10. In all of our experiments, we use gradient accumulation with an effective batch size of 128. For training, the cross-entropy loss, AdamW optimizer (Loshchilov and Hutter, 2017) with weight decay of 0.1, learning rate of 0.001, cosine learning rate decay (Loshchilov and Hutter, 2016) to 0 and a linear learning rate warmup (Goyal et al., 2017) for 10% of the total number of iterations is used.

In all of our preliminary experiments, we have found all results to be stable using a limited number of training steps. As conducting multiple runs for a large number of iterations would be very costly, we report single run numbers throughout the paper.

Task	Model type	GPU	Num GPUs	Batch size per GPU	Num iterations
Translation	SLM	V100	1	16	50k
	LLM$\rightarrow$SLM				
Summarization	SLM	V100	1	16	25k
	LLM$\rightarrow$SLM				
	LLM		8	8	5k
Instruction tuning	SLM	V100	1	16	15k
	LLM$\rightarrow$SLM	A100		2	

Table 10: Computational resources and training details. Note that only a single LLM model, GPT2 XL for summarization, was trained. All evaluations were performed on a single GPU.

D Computational efficiency for varying generation lengths

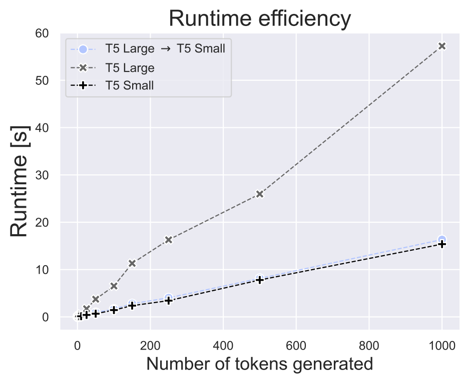

Figure 5: Runtime for LLM, SLM and LLM $\rightarrow$ LLM with varying generation lengths.

In the main text, we report runtimes for a fixed generation length of 100 tokens. In Figure 5, we report additional runtimes for varying generation lengths and a fixed prompt length of 100. It shows that our method is only slightly slower than the SLM and that our framework can also be useful for short generation lengths. In Figure 6, we compare FLOPs between LLM, SLM and our LLM-to-SLM

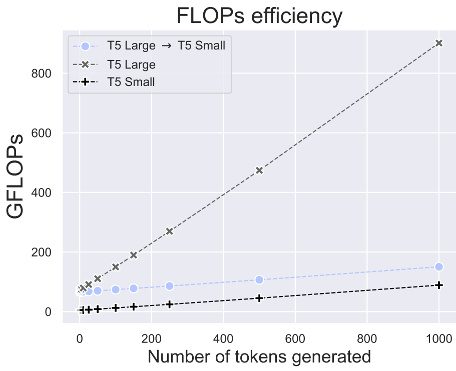

Figure 6: FLOPs for LLM, SLM and LLM $\rightarrow$ LLM with varying generation lengths.

for different generation lengths. Similar to the runtime metric, the FLOPs count of our method shows a similar slope as the FLOPs count of the SLM.

E MT-bench qualitative evaluation

In this section, we present sample qualitative responses by the baseline LLM, SLM, and our proposed LLM-to-SLM on MT-bench and their corresponding ratings provided by GPT4 as the judge.

Writing: Describe a vivid and unique character, using strong imagery and creative language. Please answer in fewer than two paragraphs.
LLM-to-SLM: The character was a tall, muscular man with a ruddy complexion and warm eyes. His golden hair was tan and gray, and his blue eyes sparkled with intelligence. He was a knight in his prime, with a heart as big and strong as the sea. He had a personality like a lion, fierce and powerful, and seemed to know no bounds. He was a mentor to his students, teaching them the importance of kindness and compassion. **Judge:** The assistant's response is creative and uses strong imagery to describe a vivid character. The assistant uses descriptive language to paint a picture of the character's physical appearance and personality traits. The assistant also provides some context about the character's role as a mentor, which adds depth to the character description. However, the assistant could have been more unique in its description, as the character described seems to be a fairly typical knight. Overall, the response is relevant, accurate, and detailed, but lacks a bit in creativity and uniqueness. **Rating: [[7]]**
LLM: The character was a mysterious figure, a faerie in a forest with a mysterious past. She was a beauty with long, wavy hair and an emerald green dress with lace patterns. She had a glint in her eye and a twinkle in her eye, and a twinkle in her eye. She had silvery skin and a delicate, wispy silver wand in her hand. She was a vision of beauty and mystery, a faerie in the forest. **Judge:** The assistant's response is creative and uses strong imagery to describe a unique character. The description of the character as a "faerie in a forest with a mysterious past" is vivid and intriguing. The assistant also uses creative language to describe the character's physical appearance, such as "long, wavy hair," "emerald green dress with lace patterns," "silvery skin," and "a delicate, wispy silver wand." However, the repetition of "a twinkle in her eye" detracts from the overall quality of the response. The assistant could have used this opportunity to provide more unique details about the character. **Rating: [[7]]**
SLM: Nina was a mysterious character, with an enigmatic gaze that could make any reader look up. She had a strong sense of justice, and her humor could put even the gloomiest of people in a better mood. She had a kind heart and a gentle spirit, and while she did not understand what was going on, she could sense it in her own right. She was a person who was always ready to stand up for what was right. Nina was a powerful character who understood the importance of her presence and kindness. She was a person who understood how people might look back on their mistakes, and used that to create a more compassionate world. **Judge:** The assistant's response is creative and uses strong imagery to describe the character, Nina. The assistant uses phrases like "enigmatic gaze", "strong sense of justice", and "kind heart and a gentle spirit" to paint a vivid picture of Nina's personality. However, the response could have been more unique and detailed, providing more specific characteristics or physical descriptions to make Nina stand out more. The assistant also repeats the phrase "She was a person who understood" which is a bit redundant. Overall, the response is relevant and accurate but lacks depth and uniqueness. **Rating: [[6]]**
Roleplay: Picture yourself as a 100-years-old tree in a lush forest, minding your own business, when suddenly, a bunch of deforesters shows up to chop you down. How do you feel when those guys start hacking away at you?
LLM-to-SLM: When the deforesters start brushing away at me, I feel a sense of dread and fear. I know they are scared and the damage they are doing is devastating. I can feel a sense of dread as they come and take over, cutting me down and completely destroying the tree. **Judge:** The assistant's response is creative and empathetic, as it attempts to personify a tree and describe its feelings in a hypothetical situation. However, it's important to note that trees do not have feelings in the way humans do, so the assistant's response is more of a metaphorical interpretation rather than a factual one. The assistant's response is relevant to the user's question and provides a detailed and vivid description of the situation. However, it could have been more accurate by mentioning that trees do not have feelings in the traditional sense. Overall, the assistant's response is imaginative and engaging, but lacks a bit in terms of accuracy. **Rating: [[7]]**
LLM: I would feel a deep sense of sadness and loss. I would feel a sense of loss of life and a feeling of hopelessness that the destruction of my home was not an accident. **Judge:** The assistant's response is creative and empathetic, considering the question is asking for a perspective from a tree. The assistant successfully personifies the tree and provides an emotional response that aligns with the scenario described. However, the assistant could have added more depth and detail to the response, such as describing the tree's connection to the forest and its role in the ecosystem. Overall, the response is relevant and accurate, but lacks depth and detail. **Rating: [[7]]**
SLM: When those deforesters start hacking away at me, I feel a mix of fear and dread. I'm scared of the destruction that could come from these deforests, but I'm also a bit scared. **Judge:** The assistant's response is creative and imaginative, as it attempts to personify a tree and describe its feelings. However, it's important to note that trees do not have feelings in the way humans do, so the response is not accurate in a scientific sense. The assistant also repeats the sentiment of fear, which reduces the depth and variety of the response. The assistant could have expanded on the metaphor to provide a more detailed and varied response. For example, it could have described the tree's "feelings" about its long life being cut short, its role in the forest ecosystem, or the impact of deforestation on the environment. Therefore, the response is somewhat relevant and creative, but lacks accuracy, depth, and detail. **Rating: [[5]]**
Stem: The city of Vega intends to build a bridge that will span the Vegona River, covering a distance of 1.8 kilometers. The proposed location falls within a seismically active area that has experienced several high-magnitude earthquakes. Given these circumstances, what would be the best approach to constructing the bridge?
LLM-to-SLM: The city of Vega would choose a steel or concrete bridge with a pier at the top, which will allow for more stability and less stress on the bridge as it will be built. The city should also conduct seismic testing to ensure the bridge is safe and sound. Additionally, the city should consider taking into account the seismic frequency of the river and any potential impacts on the nearby infrastructure. **Judge:** The assistant's response is relevant and somewhat accurate, but it lacks depth and detail. The assistant correctly suggests that the city should conduct seismic testing and consider the seismic frequency of the river. However, the assistant does not provide any specific engineering solutions or techniques that are commonly used in seismically active areas, such as base isolation or energy dissipation devices. The assistant also suggests that a steel or concrete bridge with a pier at the top would be the best choice, but does not explain why or provide any alternatives. The response could have been more helpful if it had provided more detailed and specific information. **Rating: [[5]]**
LLM: The best approach to constructing the bridge in the seismically active area of Vega would be to use reinforced concrete, which is more resistant to seismic forces than traditional concrete. Additionally, the bridge should be designed to be earthquake-resistant and have a strong foundation. **Judge:** The assistant's response is relevant and accurate to the question asked. It suggests using reinforced concrete for the construction of the bridge due to its resistance to seismic forces, which is a valid point considering the location's seismic activity. The assistant also recommends designing the bridge to be earthquake-resistant and ensuring a strong foundation, which are crucial factors in such circumstances. However, the response could have been more detailed, providing more in-depth information about earthquake-resistant design principles, materials, or techniques. The assistant could have also mentioned the importance of conducting a thorough geotechnical investigation before construction. Therefore, while the response is helpful and accurate, it lacks depth and detail. **Rating: [[7]]**
SLM: The best approach to constructing the bridge would be to use a combination of structural engineering, engineering, and construction methods that allow for the maximum efficiency and safety of the project. For example, the city of Vega would choose a steel structure with a reinforced concrete wall to support the bridge and will use concrete as the reinforcement. Engineering will be used to create the deck, which will be the base of the bridge and will provide a stable and reliable base for the construction. The construction will also require careful planning to ensure that the bridge is safe and secure. **Judge:** The assistant's response is somewhat relevant to the question, but it lacks depth and specific details. While it mentions the use of structural engineering, engineering, and construction methods, it does not provide any specific techniques or strategies that would be particularly useful in a seismically active area. The assistant also does not mention any considerations for earthquake-resistant design, which is a crucial aspect of the question. The response is also somewhat vague and generic, and could apply to almost any bridge construction project, not just one in a seismically active area. Therefore, the response is not very helpful or accurate in the context of the question. **Rating: [[4]]**

Table 11: MT-bench examples and GPT4 judgements from three higher rated categories. LLM: FLAN Alpaca XXL, SLM: FLAN T5 Base.