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

Benjamin Bergner [1 2 † *] **Andrii Skliar** [2 *] **Amelie Royer** [2 ‡] **Tijmen Blankevoort** [2 ‡] **Yuki Asano** [2 3]
Babak Ehteshami Bejnordi [2]

Abstract

Large language models (LLMs) are widely used for text generation, but their size and reliance on autoregressive decoding increase deployment costs and latency. We propose a hybrid approach that combines different-sized language models to improve efficiency while maintaining performance. Our method uses a pretrained LLM to encode prompt tokens in parallel, guiding a small language model (SLM) to generate responses more efficiently. By combining encoder-decoder LLMs with encoder-decoder and decoder-only SLMs, we achieve up to $4\times$ speedup with minor performance penalties of $1-2\%$ for translation and summarization tasks compared to the LLM.

1. Introduction

The recent widespread adoption of 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), with emerging abilities becoming apparent only at larger scales (Chowdhery et al., 2023; Wei et al., 2022). These findings have led to the increased popularity of large models, both in decoder-only (Touvron et al., 2023a) and encoder-decoder networks (Chung et al., 2022).As this race to scale intensifies, LLMs are becoming more challenging to deploy, especially in light of compute and latency requirements of edge devices, which translate into higher costs for providers and end users alike (Chen et al., 2023b).

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 be processed efficiently in parallel. However, the response is usually generated in an autoregressive manner (Radford et al., 2018; Zarrieß et al., 2021): The LLM must be called for each token to be generated, which requires loading all its weight matrices and the KV cache. 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).

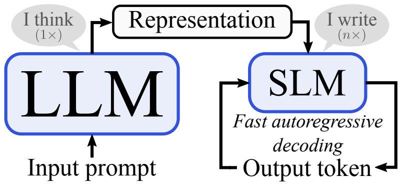

Figure 1. LLM-to-SLM: A large language model (LLM) computes a high-quality representation of the prompt to condition a small language model (SLM), which then efficiently decodes the response while maintaining high performance close to the LLM.

Following this observation, we propose a hybrid approach that targets to reduce the cost of autoregressive decoding by distributing prompt encoding and response generation over two unequally sized networks (Figure 1). Our method, called LLM-to-SLM, involves a single forward pass with an LLM to compute a high-quality representation of the prompt. Following a learnable projection, this representation is used to guide a more efficient, small language model (SLM) to perform autoregressive generation. To evaluate the efficacy of our proposed LLM-to-SLM method, we combine networks from different families and present results on multiple benchmarks. We observe speedups of up to $4.2\times$ compared to the LLM, outperforming the SLM by a large margin, while approaching the performance of the LLM in machine translation and summarization.

2. Related Work

Reducing the overall inference cost of LLMs has gained significant interest in recent years. While traditional techniques, e.g. in model compression (Frantar & Alistarh, 2023; Dettmers et al., 2022) and parallel decoding (Fu et al., 2023; Ning et al., 2023), are still active areas of research, recent developments show a pivot towards hybrid approaches that combine models of different sizes for fast decoding.

[*]Equal contribution [†]Work done as an intern at Qualcomm AI Research. [‡]Work was completed while employed at Qualcomm AI Research. [1]Hasso Plattner Institute for Digital Engineering, University of Potsdam [2]Qualcomm AI Research, Amsterdam; Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc. [3]QUVA Lab, University of Amsterdam. Correspondence to: Andrii Skliar <askliar@qti.qualcomm.com>.

Work presented at the ES-FoMo II Workshop at ICML 2024, Vienna, Austria. Copyright 2024 by the author(s).

Specifically, Chen et al. (2023b) propose a language model cascade, where cheaper models are invoked first and thus perform the bulk of computation. In contrast, 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 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) attaches heads on top of the LLM to predict multiple tokens in parallel without employing an explicit draft model. Concurrent to this work, S et al. (2024) propose generating blocks of tokens with an SLM before feeding them into an LLM, which then generates features used for improved generation of the next block.

Despite these promising developments, exploiting the discrepancy between fast prompt encoding and slow response generation remains underexplored. In contrast to related works, the LLM in our method is only called once and the SLM is conditioned on its representation. This makes LLM-to-SLM attractive for hybrid inference settings, where the prompt can be processed by a high-performance server while decoding is performed on an edge device.

3. LLM-to-SLM

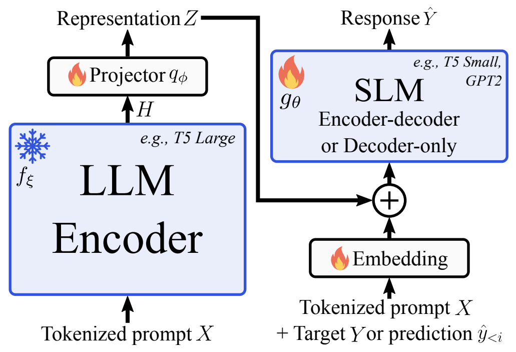

Figure 2. Architecture details. A frozen LLM encoder integrates projected representations into either a trainable encoder-decoder or a decoder-only SLM.

It is well known 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 for 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 our method: 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 and the prompt as input and generates the output tokens in an autoregressive manner. Crucially, the parameter count of the SLM is significantly smaller than the LLM ($8 - 16\times$ in our experiments), leading to faster generation as only the SLM performs autoregressive decoding. In the remainder of this section, we formalize decoding in LLM-to-SLM, describe its individual components, and further 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:

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

where f_ξ and g_ξ refer to the LLM encoder and decoder. Generating the complete sequence Y thus requires n 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:

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

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

3.2. Architecture

LLM Encoder. The LLM encoder $f_\xi : X \mapsto H$ processes a prompt X of length m into a high-quality representation $H \in \mathbb{R}^{m \times d_l}$. Training LLMs requires substantial resources, so we freeze the LLM during fine-tuning and use pretrained encoder-decoder models, omitting the decoder. This enables efficient training by precomputing prompt representations. Using last layer features from the encoder is straightforward, unlike decoder-only models where it is more challenging to identify a good prompt encoding point (see Appendix D).

Projector. The projector $q_\phi : H \mapsto Z$ aligns LLM and SLM representations. It converts high-dimensional features $H \in \mathbb{R}^{m \times d_l}$ into a lower-dimensional representation $Z \in \mathbb{R}^{m \times d_s}$, which can be directly fused with SLM embeddings. We found a small MLP: `Linear(`d_l`, `d_s`)` $\rightarrow$ `ReLU` $\rightarrow$ `Linear(`d_s`, `d_s`)` trained from scratch to be effective for this alignment.

SLM. The SLM $g_\theta : (X, Z) \mapsto \hat{Y}$ maps the tokenized input X and the projected representation Z to the response

$\hat{Y}$ of length n. We use pretrained networks as SLMs and fine-tune them, as they have not yet been trained to process high-capacity encodings. We experiment with learning from both ground truth data and LLM-generated sequences.

3.3. Conditioning the SLM

We incorporate LLM representations early into the SLM, which allows us to leverage both encoder-decoder and decoder-only SLMs. Specifically, we add the projected LLM representation Z to the SLM prompt embedding E_X, resulting in the modulated representation $Z + E_X$ (see Appendix C.2 for a comparison with other integration methods). Importantly, note that the LLM and SLM may use different tokenizers and vocabularies, leading to different sequence lengths. To align the sequences, the SLM reuses the tokenizer and a copy of the embedding matrix from the LLM. Furthermore, two new linear layers are inserted: (1) an embedding projection layer mapping from the LLM feature dimension d_l to the SLM space d_s and (2) a new head layer mapping to the LLM vocabulary.

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? We evaluate LLM-to-SLM on machine translation and summarization, report its computational efficiency and compare our method to speculative decoding and PEFT methods. We provide further details about the experimental setup and additional results in Appendix C.

4.1. Machine Translation

We report results for English to German, French, and Romanian translations using WMT14 (Bojar et al., 2014) for En-Fr/De and WMT16 for En-Ro (Bojar et al., 2016). T5 Large serves as the LLM encoder, and T5 Small, T5 1.1 Small, and GPT2 as the SLMs. T5 Small and GPT2 are 16× and 8× smaller than T5 Large. T5 Large and T5 Small are pretrained for translation, while T5 1.1 Small is trained on the C4 dataset. The prompt format is "`translate English to *target-language*:`". All models are trained for $50k$ iterations, except for T5 Large, which is applied in a zero-shot manner. We use our LLM to generate training labels and report BLEU scores in Table 1.

T5 Large averages a BLEU score of 31.94. T5 Small performs 2 BLEU points worse, and non-pretrained SLMs (T5 1.1 Small and GPT2) perform over 3 BLEU points worse. Our LLM-to-SLM variants close this gap, with T5 Large → T5 Small scoring 31.54 on average. Gains are more pronounced with non-pretrained networks, e.g., T5 Large → T5 1.1 Small scores 2.5 points higher than T5 1.1 Small. Notably, T5 Large → GPT2 also shows significant improvements with minimal runtime trade-offs.

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 1. BLEU scores for machine translation. LLM-to-SLM models approach the performance of the LLM.

4.2. Summarization

We assess the performance of LLM-to-SLM for summarization on CNN/Daily Mail (Hermann et al., 2015), using T5 Large as the LLM and GPT2 as the SLM. T5 Large is pretrained for summarization. Following Raffel et al. (2020), the input prompt is prefixed with "`summarize:`". All models are fine-tuned for $25k$ iterations on the training set and evaluated on the test split. Unlike translation, we train directly from ground-truth labels, as this method performs better than using LLM-generated labels. ROUGE scores and runtimes are reported in Table 2.

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

GPT2 scores 1.24 ROUGE points lower than T5 Large. However, T5 Large → GPT2 slightly exceeds T5 Large's score and achieves a 3× speedup. Notably, T5 Large → GPT2 performs comparably to a fully fine-tuned GPT2 XL model, with a decoder that is 17× smaller.

4.3. Tiny SLMs

We explore the limits of LLM-to-SLM by scaling down the SLM to a minimum. Specifically, we use T5 Large → GPT2 in the machine translation scenario and truncate the upper layers of GPT2, leading to SLMs with $d \in 1, 2, 4$ layers, having 8M, 15M, and 29M parameters, respectively. As shown in Figure 3 (right), our LLM-to-SLM models significantly outperform the corresponding SLM baselines while maintaining nearly the same runtime. Notably, T5 Large → GPT2 with $d = 4$ outperforms GPT2 and is more than twice as fast. As the size of the SLM decreases, the performance gap between the LLM and the SLM, and thus our method, increases. However, we observe even larger performance gains of LLM-to-SLM over the SLM alone in this setting. For example, T5 Large → GPT2 with $d = 1$

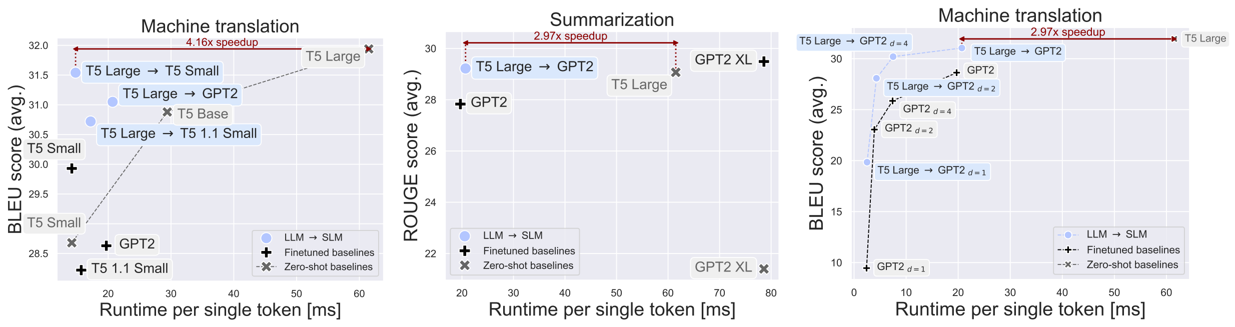

Figure 3. Performance-runtime trade-off curves for various models for machine translation (left) and summarization (middle) tasks. Results of LLM-to-SLM when truncating SLMs to 1, 2, and 4 layers (right).

performs about 10 BLEU points better than the SLM.

4.4. Computational Efficiency

We show runtimes for each task in Tables 1 and 2 and performance-runtime trade-off curves in Figure 3. In machine translation, T5 Large → T5 Small is over $4\times$ faster than T5 Large with minimal performance loss. Summarization shows a similar gain with around a $3\times$ speedup. Figure 3 illustrates that LLM-to-SLM variants are only slightly slower than the SLM but perform significantly better. Further insights into generation length, runtime, and FLOPs are in Appendix E, showing that LLM-to-SLM approaches the computational efficiency of the SLM.

4.5. Speculative decoding

While our empirical evaluations on translation and summarization indicate that LLM-to-SLM can achieve comparable performance to the LLM, such behaviour is generally not guaranteed. Speculative decoding (SD), on the other hand, guarantees to match the distribution of the LLM, albeit at the cost of invoking the LLM multiple times (Chen et al., 2023a). In Table 3, we report results comparing our method to SD in translation (English to French, greedy decoding). Our method (T5 Large → T5 Small) shows similar performance to SD and the LLM, but a much larger speedup ($4.15\times$ compared to $1.53\times$ in SD) since our method requires only a single call to the LLM.

Method	Score	Time
T5 Large (zero-shot)	38.9	61.5
T5 Small	35.1	14.2
SD (T5 Large, T5 Small)	38.9	40.3
T5 Large → T5 Small	**38.5**	**14.8**

Table 3. Comparison of BLEU scores and runtime in ms/token between LLM-to-SLM and speculative decoding for English to French translation. Notation: SD (target model, draft model).

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

Table 4. PEFT results using fixed T5 Small for translation and GPT2 for summarization. Average BLEU scores are reported for WMT and average ROUGE scores for CNN/DM. In our method, T5 Small and GPT2 are conditioned on T5 Large representations.

4.6. Parameter-efficient fine-tuning

Our method shares similarities with PEFT methods. Soft prompt tuning prepends learnable tokens to the prompt embedding, while we add LLM representations element-wise to the SLM's prompt embedding, a form of conditional prompting. In Table 4, we compare our method with PEFT by freezing the SLM and training only the projector. PEFT hyperparameters are adjusted to match the total parameter count of our projector (see Appendix C.1 for details). The results show that our method outperforms all PEFT methods in both summarization and translation, indicating that conditioning on the LLM improves performance over non-conditional approaches. Additional PEFT results for a more limited training regime are in Appendix C.3.

5. Conclusion

In this work, we introduced LLM-to-SLM, a novel hybrid framework designed to accelerate autoregressive decoding by strategically combining an LLM with an SLM. This approach takes advantage of parallel prompt encoding by the LLM and efficient response generation by the SLM. Specifically, the LLM first encodes a prompt into a high-quality representation, which is then used to condition the SLM for efficient yet accurate decoding. Our evaluations on various benchmarks have shown that LLM-to-SLM can achieve significant speed improvements, providing $3-4\times$ faster generation with only minimal losses in predictive performance — about $1-2\%$ for translation and summarization compared to using an LLM alone.

References

Bojar, O., Buck, C., Federmann, C., Haddow, B., Koehn, P., et al. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the ninth workshop on statistical machine translation*, pp. 12–58, 2014.

Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Logacheva, V., Monz, C., et al. Findings of the 2016 conference on machine translation (wmt16). In *First conference on machine translation*, pp. 131–198. Association for Computational Linguistics, 2016.

Brown, T., Mann, B., Ryder, N., Subbiah, M., and Kaplan, J. D. e. a. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf.

Cai, T., Li, Y., Geng, Z., Peng, H., and Dao, T. Medusa: Simple framework for accelerating llm generation with multiple decoding heads, 2023.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*, 2023a.

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

Chen, M., Tworek, J., Jun, H., Yuan, Q., Pinto, H. P. d. O., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.

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

Frantar, E. and Alistarh, D. Sparsegpt: Massive language models can be accurately pruned in one-shot. 2023.

Fu, Y., Bailis, P., Stoica, I., and Zhang, H. Breaking the sequential dependency of llm inference using lookahead decoding, November 2023. URL https://lmsys.org/blog/2023-11-21-lookahead-decoding/.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, large minibatch sgd: Training imagenet in 1 hour. *arXiv preprint arXiv:1706.02677*, 2017.

Hermann, K. M., Kocisky, T., Grefenstette, E., Espeholt, L., Kay, W., Suleyman, M., and Blunsom, P. Teaching machines to read and comprehend. *Advances in neural information processing systems*, 28, 2015.

Hoffmann, J., Borgeaud, S., Mensch, A., Buchatskaya, E., Cai, T., Rutherford, E., Casas, D. d. L., Hendricks, L. A., Welbl, J., Clark, A., et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Jiang, Y., He, Q., Zhuang, X., Wu, Z., Wang, K., Zhao, W., and Yang, G. Recyclegpt: An autoregressive language model with recyclable module. *arXiv preprint arXiv:2308.03421*, 2023.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Kim, S., Mangalam, K., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Big little transformer decoder. *arXiv preprint arXiv:2302.07863*, 2023.

Lester, B., Al-Rfou, R., and Constant, N. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

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

Li, X. L. and Liang, P. Prefix-tuning: Optimizing continuous prompts for generation. *arXiv preprint arXiv:2101.00190*, 2021.

Lin, C.-Y. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pp. 74–81, 2004.

Loshchilov, I. and Hutter, F. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Ning, X., Lin, Z., Zhou, Z., Yang, H., and Wang, Y. Skeleton-of-thought: Large language models can do parallel decoding. *arXiv preprint arXiv:2307.15337*, 2023.

OpenAI. Gpt-4 technical report, 2023.

Pope, R., Douglas, S., Chowdhery, A., Devlin, J., Bradbury, J., Heek, J., Xiao, K., Agrawal, S., and Dean, J. Efficiently scaling transformer inference. *Proceedings of Machine Learning and Systems*, 5, 2023.

Post, M. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pp. 186–191, Belgium, Brussels, October 2018. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W18-6319.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. Improving language understanding by generative pre-training. 2018.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

S, A. P., Nair, P. A., Samaga, Y., Boyd, T., Kumar, S., Jain, P., and Netrapalli, P. Tandem transformers for inference efficient llms, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

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

Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.

Appendix

A. 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.

B. Limitations

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. Another limitation of this work is our focus on encoder-decoder LLMs. We discuss this design decision in Section 3.2 and provide preliminary results for decoder-only models as LLMs within our framework in Appendix D, finding that the performance strongly depends on the layer from which the LLM features are extracted. Further investigation is required to see how decoder-only features can be used in the LLM-to-SLM setting. Finally, we only used 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., 2022). 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.

C. Further results and ablations

C.1. Experimental setup

Setup. The networks used in our experiments are listed in Table 5. We employ various pretrained models and architectures and denote combinations as LLM $\rightarrow$ SLM. We make use of a T5 encoder as LLM, and employ T5 encoder-decoder and GPT2 decoder-only models as SLMs: In this setting, the LLM has $8-16\times$ more parameters than the SLMs. Since the LLM remains frozen in our method, we rely on a well-performing LLM pre-trained for the respective tasks. For generation, we use beam search (beam width of 4, length penalty of 0.6). We report task-specific performance metrics: SacreBLEU (Post, 2018) for translation and ROUGE (Lin, 2004) for summarization. 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 on an NVIDIA V100 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)
Dec-only	GPT2† (Radford et al., 2019)	86M
	GPT2 [1,2,4]-Layers†	[8M, 15M, 29M]
	GPT2 XL (Radford et al., 2019)	1.5B

Table 5. Model variants used in the experiments. Sizes are rounded, excluding embedding and head parameters. Encoder/decoder sizes are shown in parentheses. Symbols $*$ and $\dagger$ 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 & 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 & Hutter, 2016) and a linear learning rate warmup (Goyal et al., 2017) for 10% of the total number of iterations is used. We rely on Huggingface's transformers (Wolf et al., 2019) for training and generation. 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. Details on computational resources used for training and evaluation are specified in Table 6.

Task	Model type	GPU	Num GPUs	Batch size per GPU	Num iterations
Translation	SLM	V100	1	16	50k
	LLM→SLM				
Summarization	SLM	V100	1	16	25k
	LLM→SLM				
	LLM		8	8	5k

Table 6. 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.

PEFT hyperparameters. Table 7 shows the hyperparameters for the prompt tuning, prefix tuning, and LoRA methods considered in the parameter-efficient fine-tuning experiments 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 7. Hyperparameters for prompt tuning, prefix tuning and LoRA.

C.2. Ablation study

Embedding addition vs. replacement. Instead of adding projected LLM representations to SLM embeddings, we can alternatively replace the SLM embeddings of the prompt with the projected LLM encoding. Table 8 shows results comparing these feature fusion strategies for machine translation, indicating that addition and replacing perform on par with each other.

LLM vs. SLM tokenizer. When replacing the SLM embedding of the prompt with the projected LLM encoding, 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 also 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.

Model	GT $+$	GT $\times$	Gen. $+$	Gen. $\times$
T5 Large → T5 Small	30.83	30.86	**31.54**	31.47
T5 Large → GPT2	29.68	29.71	**31.05**	30.83

Table 8. 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, $\times$: Replacement, $+$: Addition

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 8). 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.

C.3. Limited-data PEFT

In our ablation study on PEFT techniques (see Section 4.6), we utilize a relatively large number of training examples ($>$ 20k). On the other hand, often smaller datasets are used for (parameter-efficient) fine-tuning. In Table 9, we show extended results covering a limited training regime, exemplary for translation (English to French). Specifically, we limit the training set to 20k examples and fine-tune for 20 epochs, corresponding to approximately 3k iterations. We compare this setting to full training as used in the main paper, i.e., 50k iterations on 6.4M examples.

Method	Full training	Limited training
Prompt tuning	36.02	30.59
Prefix tuning	35.43	30.07
LoRA	36.70	32.38
LLM $\rightarrow$ SLM (ours)	**39.35**	**36.89**

Table 9. Comparison of full and limited training for translation (English to French)

Our LLM-to-SLM outperforms all other PEFT methods also in the limited training regime. However, to achieve a performance close to that of the LLM, it is beneficial to use a training setting similar to that of the LLM.

D. Decoder-only models as LLM

Compared to encoder-decoder based LLMs, where the last layer representation of the encoder serves as a straightforward prompt encoding point, in a decoder-only model there is no specific layer that can be explicitly identified as the encoded representation of the prompt. We thus experiment with extracting representations from different layers of the LLM before passing it to the projector and subsequently to the SLM. In Table 10, we show results for the summarization task, using a fine-tuned GPT2 XL as the LLM and a smaller GPT2 as the SLM (see Table 5 for parameter counts). Surprisingly, we observe that the performance deteriorates with the depth of the LLM. Although LLM-to-SLM performs slightly better than the SLM alone by using very early layer representations of GPT2 XL, our model based on an encoder-decoder LLM, T5 Large $\rightarrow$ GPT2, performs considerably better (avg. ROUGE score of 29.22, see also Table 2).

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

E. Computational efficiency for varying generation lengths

In the main text, we report runtimes for a fixed generation length of 100 tokens. In Figure 4, 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 5, we compare FLOPs between LLM, SLM and our LLM-to-SLM 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.

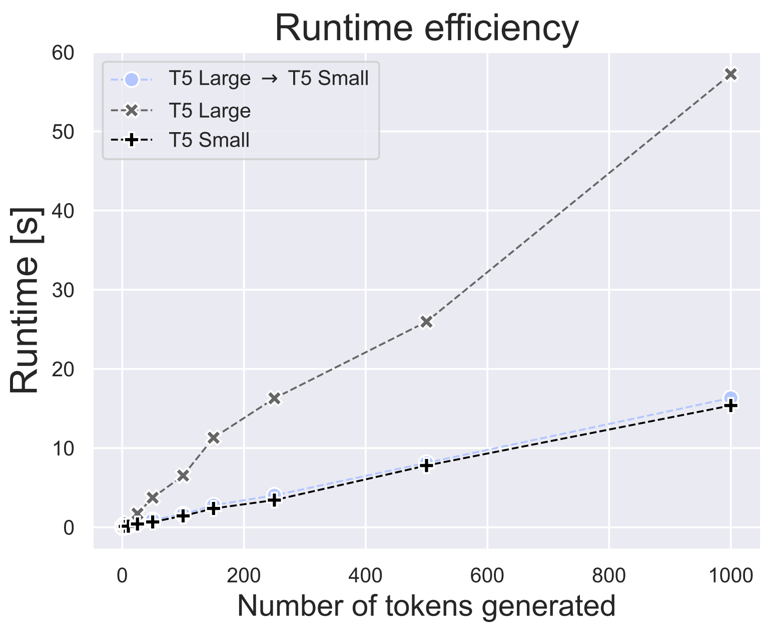

Figure 4. Runtime for LLM, SLM and LLM → SLM with varying generation lengths.

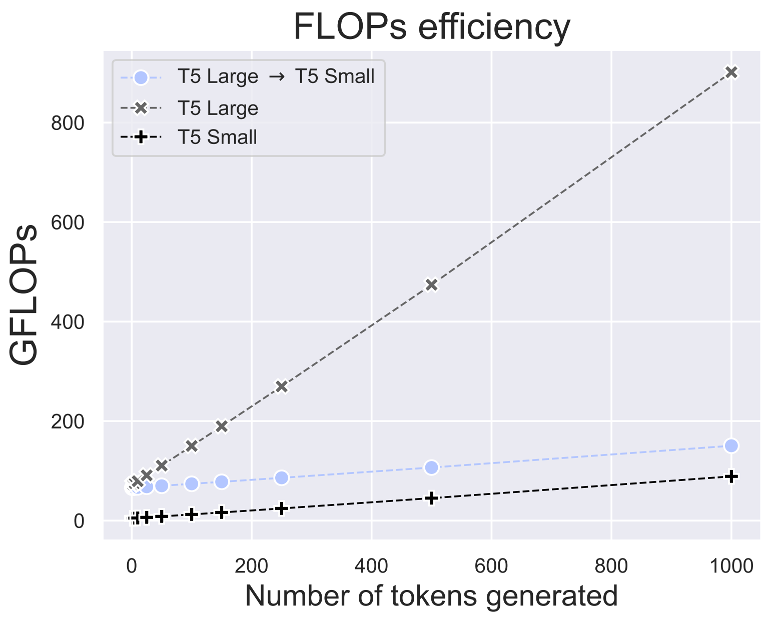

Figure 5. FLOPs for LLM, SLM and LLM → SLM with varying generation lengths.