Lightweight Decoders for Efficient Language Models

TJ Bai
Johns Hopkins University
tbai4@jhu.edu

Arijit Nukala
Johns Hopkins University
anukala1@jhu.edu

Abstract

We investigate "deep encoder, shallow decoder" architectures for efficient autoregressive language modeling. While previous work showed promising speed-quality tradeoffs, we perform extensive ablations across more layer configurations. We benchmark against recent state-of-the-art models like DiffusER and SUNDAE. Additionally, we explore lightweight decoders using recurrent neural networks instead of transformer decoders, taking the "shallow decoder" approach to the extreme. Our aim is to identify optimal layer allocations and decoder architectures for highly efficient autoregressive language models without sacrificing quality.

1 Introduction

Much effort has been invested in optimizing transformer-based language models for efficient inference. Approaches include reducing computation through sparse or sliding window attention (Child et al. (2019), Beltagy et al. (2020)), knowledge distillation, novel model architectures, and others. A significant class of optimizations falls under the family of *non-autoregressive* (NAR) language models, which, in contrast to sequential *autoregressive* (AR) models, allow inference to be parallelized across numerous output token positions simultaneously.

NAR models are often viewed as a speed-quality trade-off. While the sequential dependency in AR models is crucial for coherent generation and high accuracy, it introduces an inherent performance ceiling. Techniques such as iterative refinement and sequence-level knowledge distillation enables NAR models to acheive a more competitive speed-quality trade-off, but their performance generally still lags significantly behind their AR counterparts.

Kasai et al. (2021) investigates "deep encoder, shallow decoder" architectures as a straightforward approach to optimize AR models for efficiency while still retaining high-quality generation for Seq2Seq translation. They argue that using an equal number of layers for the encoder and decoder is a suboptimal allocation strategy. By increasing the number of parallelizable encoder layers and reducing the serial decoder layers, often to just a single layer, we can obtain attractive speed-quality tradeoffs. Further, applying sequence-level knowledge distillation (Kim and Rush (2016)), as is often done with NAR models, allows these AR models to maintain superior performance across the board.

In this work, we aim to reproduce and extend Kasai et al. (2021). In particular, though this work importantly demonstrates the efficacy of "deep encoder, shallow decoder" architectures, the primary comparison is between 6-6 and 12-1 encoder-decoder layer allocations. We will conduct extensive ablations across a larger variety of allocations to determine more precisely the relationship between layer allocations and performance. Additionally, we will evaluate more recently competitive NAR models such as DiffusER (Reid et al. (2023)) and SUNDAE (Savinov et al. (2022)) in order to create a more comprehensive and competitive benchmark. Finally, we seek to explore a more "aggressive" decoder reduction through the use of recurrent neural networks (RNN). If the deep encoder is able to build sufficiently complex representations of the input sequence, we hypothesize that even a weak model class may be able to leverage these representations effectively and efficiently.

Preprint. Under review.

2 Project Plan

2.1 Hypothesis

Our hypothesis revolves around the efficacy of "deep encoder, shallow decoder" architectures for enhancing the speed of AR language models without compromising quality. Based on the results of Kasai et al. (2021), we propose that by allocating more layers to the encoder while reducing the depth of the decoder, we can progressively improve translation speed across various architectures while preserving generation quality. Furthermore, we hypothesize a sufficiently deep encoder with an RNN decoder could yield even higher translation speeds, albeit with a slight trade-off in translation quality.

2.2 Halfway Check-in

By the halfway point, we intend to have completed evaluation of all the AR ablations and to have begun evaluation with RNN decoders. In the second half, we seek to implement and evaluate the baseline NAR models.

2.3 Experiments

2.3.1 Methods

For each AR model, we will use the standard Transformer architecture (Vaswani et al. (2017)). As the baseline, we will evaluate 6 encoder and 6 decoder layers (6-6). Following, we will systematically evaluate a variety of layer allocations, as Kasai et al. (2021) only evaluates 12-1 as an alternative. Currently, this includes 3-3, 6-1, 6-3, and 12-6 ablations. Additionally, we will evaluate shallow RNN decoders with 1, 2, and 4 layers.

We will additionally evaluate each architecture with and without sequence-level knowledge distillation (Kim and Rush (2016)). This involves training distilled models on the beam-search outputs of a larger AR model.

2.3.2 Evaluation

Following the methods from Kasai et al. (2021), we will measure wall-to-wall translation speed from when the weights are loaded to when the last sentence is transitioned. We plan to evaluate single-sentence and batch translation speeds with batches as large as our GPUs allow. We will additionally evaluate the output translations using BLEU, as is standard (Papineni et al. (2002)).

We will benchmark our results against competitive NAR models such as DiffusER (Reid et al. (2023) and SUNDAE (Savinov et al. (2022)), in addition to the results from Kasai et al. (2021).

2.3.3 Datasets

We intend to assess translation performance on both WMT '14 (Bojar et al. (2014)) EN $\rightarrow$ FR and EN $\rightarrow$ HI to examine outcomes on datasets representing high and low resource languages. Utilizing the EN $\rightarrow$ FR dataset will enable us to directly benchmark our models against the findings from Kasai et al. (2021), Reid et al. (2023), and Savinov et al. (2022). The WMT '14 datasets also contain standard test and development sets, providing a comprehensive evaluation framework.

References

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. *arXiv:2004.05150*.

Ondřej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 12–58, Baltimore, Maryland, USA. Association for Computational Linguistics.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers.

Jungo Kasai, Nikolaos Pappas, Hao Peng, James Cross, and Noah Smith. 2021. Deep encoder, shallow decoder: Reevaluating non-autoregressive machine translation. In *International Conference on Learning Representations*.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327, Austin, Texas. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Machel Reid, Vincent Josua Hellendoorn, and Graham Neubig. 2023. DiffusER: Diffusion via edit-based reconstruction. In *The Eleventh International Conference on Learning Representations*.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. 2022. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.