The Green KNIGHT: Green Machine Translation with Knowledge-Distilled, Narrow, Inexpensive, Greedy, Hybrid Transformers

Anonymous ACL submission

Abstract

State-of-the-art neural machine translation (NMT) models deliver high-quality translations at the expense of large inference latency and energy consumption, requiring vast GPU fleets and contributing significantly to carbon emissions. To democratize and "green" NMT, we introduce the Green KNIGHT, a hardware-agnostic collection of recipes to optimize model performance in terms of speed and energy consumption, with only a minor trade-off in quality. On two high-resource benchmarks we show up to 91× CPU speedup and 94% energy savings for En→De, and 65× speedup and 10% energy usage for En→Ko; while incurring only minor losses of 9% relative BLEU. Our results prove that efficient and environmentally conscious NMT can be realized through optimizations build on well-understood, off-the-shelf techniques with no custom low-level code required, making our approach immediately deployable in real-world translation pipelines.

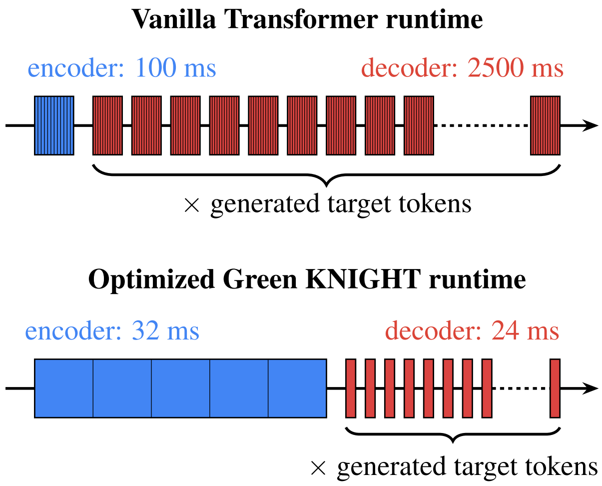


Figure 1: Runtime breakdown for the vanilla Transformer (100 ms encoder + 2500 ms decoder per batch, 95% spent in the decoder) versus our optimized Green KNIGHT model. By shifting the workload toward the encoder (32 ms) and drastically reducing decoder cost (24 ms), our design ultimately yields up to **91× speedup** and **94% energy savings** for En→De.

1 Introduction

Neural Machine Translation (NMT) has rapidly become the standard for automated language transfer, achieving human-competitive fluency and adequacy across dozens of language pairs with Transformer architectures (Vaswani et al., 2017). Most research aims at improving model performance, which usually goes hand in hand with larger and in particular deeper models whose inference time and energy consumption worsen due to the quadratic dependence on target length of the autoregressive decoder as well as the high cost of beam search.

In particular, as shown in Figure 1, the conventional Transformer 'big' model can spend over 95% of its per-batch runtime in the decoder. This imbalance not only throttles throughput but also drives up energy usage. Recent years have seen large language models (LLMs) surpass traditional Transformer models in translation quality (Kocmi et al., 2024). However, LLMs tend to be larger by magnitudes and more expensive to operate than Transformers, which is not reflected by the difference in quality. This heavy resource burden renders LLM-based translation even less sustainable for most enterprises and hinders broader deployment. In contrast, we aim at NMT with a better quality-speed/energy trade-off than recent LLMs and therefore focus on improving encoder-decoder models.

Parallel to our goal of making machine translation more efficient, we want to reduce the ecological footprint of machine translation systems in order to reduce significant carbon emissions—an issue which has been advocated before in the context of general AI (Strubell et al., 2019; Schwartz et al., 2020) and NMT systems in particular (Shterionov and Vanmassenhove, 2023), but received only minor attention by the broad NMT community. Hence, we ask the question: How "green" can machine translation become while still being practicable?

To tackle these issues, we aim at gaining as much translation speed and energy savings as possible while not losing more than 10 % relative translation quality measured in BLEU as well as COMET or BLEURT. As a solution we introduce the Green KNIGHT, a recipe to build hardware-agnostic Transformer models for green machine translation. They combine inference-time optimizations such as greedy decoding and dynamic 8-bit quantization with architectural changes to decrease the decoder work load. In particular, we investigate fast hybrid models with RNN (Cho et al., 2014; Bahdanau et al., 2015) and SSRU decoders (Kim et al., 2019), augmented with knowledge distillation (Kim and Rush, 2016).

The main contributions of our work are:

1. We comprehensively analyze the stacked effect of various inference and architecture optimizations, considering quality, speed and energy measurements for every step and all contributions mentioned next.

2. A comparison of various hybrid translation models under the applied inference and architecture optimizations.

3. The evaluations on two real-life tasks that include multiple domains. We show that our final model yields 91× faster translations and saves 94% energy on the English→German task, while being 65× faster and using only 10% energy for English→Korean translations.

The final model runs equally fast on CPU and GPU, making it a viable deployment option even in resource-constrained environments. Our work proves that easy-to-implement methods can make MT substantially more time- and energy efficient, encouraging more research teams to consider Green AI (Schwartz et al., 2020) when developing and deploying NMT systems.

2 Related Work

Kim et al. (2019) introduce a suite of modeling and engineering improvements for fast NMT. They enhance teacher-student training with multi-agent dual learning and noisy back-translation, and replace self-attention in the decoder with a lightweight recurrent unit (SSRU), tying weights between decoder layers to reduce parameters and improve CPU cache efficiency. Their inference optimizations include 8-bit quantization, 16-bit GPU inference, and concurrent GPU streams, all implemented in a custom C++ Marian framework. Their models achieve up to 24× CPU and 14× GPU speedups over their 2018 baselines without BLEU loss. In contrast, our work uses standard PyTorch, explores a broader range of decoder sizes, and employs a simpler knowledge distillation approach. We also explore more RNN decoder variants and do not rely on C++-specific optimizations or weight tying.

Hsu et al. (2020) empirically combine several known techniques, including multi-agent dual learning for distillation, SSRU and AAN decoders (Zhang et al., 2018), removal of the decoder feed-forward network (FFN), and a deep encoder–shallow decoder structure (12/1 layers). They further reduce parameters by pruning attention heads, achieving 109% CPU and 84% GPU speedups with 25% fewer parameters, while matching the Transformer "base" quality. Unlike their work, we do not focus on parameter reduction, but rather on maximizing speed and energy efficiency with minimal quality loss. We systematically evaluate various decoder sizes, tune the beam size, and find that removing the decoder's FFN is not beneficial in our setting.

Lin et al. (2021) present a combination of simple, hardware-agnostic techniques for efficient Transformer inference, such as tuning the vocabulary size, using a shallow decoder, pruning attention heads, dropping the decoder FFN, and factorizing the output projection. They also employ weight distillation and "weak" distillation (training without dropout or label smoothing in the shallow decoder). Their approach yields a $\sim 3.5\times$ speedup without quality degradation. In contrast, we achieve much higher speedups (up to 91×) by allowing small quality degradation and incorporating RNN-based decoders.

Lin et al. (2023) present an NMT system optimized for mobile deployment through three key architecture improvements: reducing vocabulary size instead of using embedding factorization, reducing model width rather than using parameter sharing, and employing a deep encoder with a shallow decoder. Combined with knowledge distillation, dropout removal, and optimization of integer operations, their 10MB model achieves a $\sim 47\times$ speedup while maintaining 88.4% of

Transformer-big performance, with 99.5% memory reduction. Their 20MB variant achieves 95.5% baseline performance with a $\sim$ 27.7$\times$ speedup. Unlike their approach, our work does not require custom hardware-specific implementations, and we thoroughly investigate beam size tuning and RNN decoder alternatives.

3 Transformer Models

Since the introduction of the Transformer (Vaswani et al., 2017), this network architecture is the de-facto standard for machine translation. The model can be described as being composed of two main parts; an encoder which compresses the input source sentence, and a decoder which autoregressively—i.e. word-by-word—generates the hypothesis in the target language. Both, encoder and decoder consist of a stack of layers which transform dense vector-representations of a fixed model dimension d. The encoder as well as the decoder layers consist of a self-attention component, which scales quadratically in the input sequence length, as well as a feedforward sub-layer. In addition, the decoder also has a cross-attention component of similar complexity as the self-attention.

Despite encoder and decoder having a comparable total number of floating point operations (roughly weighted 2:3 due to the cross-attention in the decoder), the computations of the encoder can be parallelized while the decoder in this architecture is inherently autoregressive. This effect is further amplified by beam search, and in our case leads to 95% of the total computation time being spent in the decoder (as presented in Figure 1).

4 Inference Optimizations

We commence with optimizing inference as it is independent of any system, i.e. it introduces no retraining burden and can be adopted with minimal engineering effort.

4.1 Greedy Search

To begin with, we reduce the number of candidate hypotheses per time step by shrinking the beam size down to 1. Moreover, by utilizing greedy decoding, we eliminate all overhead due to beam management.

4.2 Quantization

The most expensive operation in the Transformer are vector-matrix multiplications. In our measurements, they take around 55% of the total baseline computation time.

These operations can be sped up by computing them in 8-bit integer arithmetic, with hardware acceleration on recent CPUs (Bhandare et al., 2019). We apply a dynamic post-training quantization scheme which computes ranges of the activations on-the-fly during inference (Tang et al., 2024) and apply it to all major vector-matrix multiplications: the feedforward blocks, the attention key/value/query computation and projection, the softmax, and the RNN cells (in case of the SSRU and LSTM experiments).

5 Architectural Optimizations

As our baseline, we adopt the state-of-the-art Transformer 'big' model, in which the encoder and decoder share the same depth. As previously shown in Figure 1, the decoder dominates the encoder in inference runtime. Hence, our focus lies on applying architectural modifications that shift computation towards the encoder and thus streamline the decoder, as illustrated in Figure 2.

5.1 Layer Reallocation & Reduction

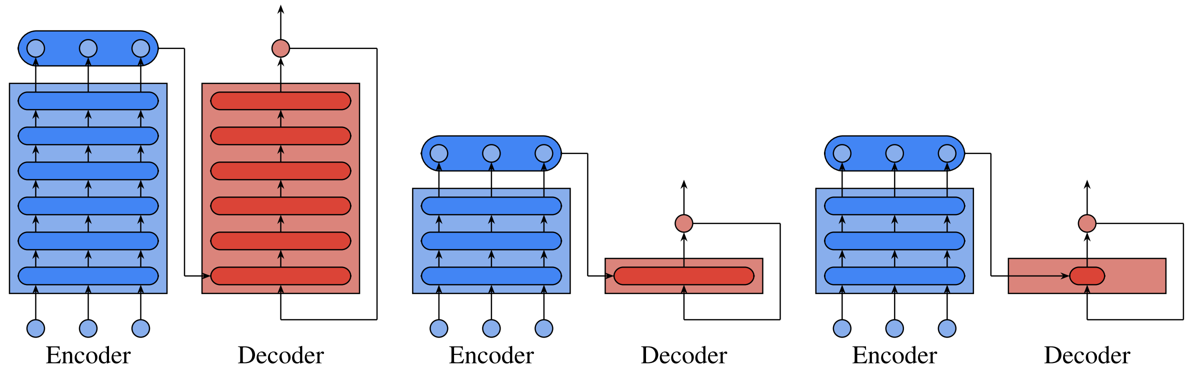
At first, we shift all decoder layers but one to the encoder, leveraging the parallelism in the encoder. The reallocation results in a model comprising $L_{\mathrm{E}} + L_{\mathrm{D}} - 1$ encoder layers and a single-layer decoder.

Moreover, we empirically investigate the trade-off between translation quality and efficiency regarding translation speed and energy consumption. Since the mapping between encoder depth and quality-efficiency ratio is not linear, our focus does not lie on reducing the encoder to as few layers as possible, but rather on finding a good optimum.

The combined effect of both steps is illustrated in Figure 2b. In comparison to the baseline model (Figure 2a), the resulting model contains only a shallow single-layer decoder and an encoder that is significantly less deep.

5.2 Decoder Width Compression

After having shrunk the decoder to just a single layer (see Figure 2b), we aim at further improving translation speed by also using reducing the decoder hidden dimension d to a smaller dimension $d' < d$, as pictured in Figure 2c. Accordingly, we also decrease the size of the forward-projections (which in the baseline was chosen as $4d$) to $4d'$. As every component in the Transformer decoder



(a) Baseline model. (b) Reallocated & reduced layers. (c) Compressed decoder width.

Figure 2: Architecture optimizations.

scales with the model dimension, this reduces the computational load of the entire decoder.

5.3 Replacing Transformer with Interleaved RNN Decoder

Finally, we replace the Transformer decoder with a custom lightweight RNN module. First, we experiment with replacing the self-attention layer of the decoder with an RNN layer (Kim et al., 2019; Hsu et al., 2020; Lin et al., 2021). We tested a standard LSTM (Hochreiter and Schmidhuber, 1997) and an optimized SSRU (Kim et al., 2019). Unlike previous work, we use only one decoder layer (see Section 5.1). In our initial experiments with LSTM, we observed a notable decrease in quality, with some smaller models practically diverging (see Table 4). We assume that this is due to the fact that the RNN cell is never exposed to hidden representations of the encoder. Therefore, we rewire the decoder by enriching the hidden state of the RNN cell h_t with a cross-attention to the hidden representation of the encoder H_{enc} (Bahdanau et al., 2015):

$$\begin{aligned} h'_t, c_t &= \mathrm{RNN}(x_t, h_{t-1}, c_{t-1}), \\ h_t &= h'_t + \mathrm{Attention}(H_{enc}, h'_t). \end{aligned} \tag{1}$$

In this way, the RNN cell has a direct information path to the source sentence. We dub this method *interleaving*, as RNN cell and cross-attention computations are interleaved between token positions. As the original SSRU design lacks a hidden state, we integrate one by concatenating the input embedding x_t with the previous output h_t, i.e., $x_t = [x_t, h_t]$.

6 Training Optimizations

After improving translation speed and energy consumption by optimizing inference and the model architecture, we address the question of how to optimize the training of our models. Here, the goal is different, as we aim at gaining better models in terms of translation quality rather than making them more time and energy efficient.

To achieve this goal, we apply sequence-level **knowledge distillation** (KD; Kim and Rush, 2016) using the baseline model as the teacher model p_{T}, which aids the training of the student model p_{S} using the aforementioned optimized architecture. However, instead of using the cross-entropy between student and teacher, we use the Kullback-Leibler divergence $\mathcal{D}_{\mathrm{KL}}(\cdot,\cdot)$ to compute the additional training loss $\mathcal{L}_{\mathrm{KD}}$:

$$\mathcal{L}_{\mathrm{KD}} = -\frac{1}{2}\Big(\mathcal{D}_{\mathrm{KL}}(p_{\mathrm{S}}, p_{\mathrm{T}}) + \mathcal{D}_{\mathrm{KL}}(p_{\mathrm{T}}, p_{\mathrm{S}})\Big). \tag{2}$$

The resulting overall training loss is computed as

$$\mathcal{L} = \alpha \cdot \mathcal{L}_{\mathrm{CE}} + \beta \cdot \mathcal{L}_{\mathrm{KD}}. \tag{3}$$

$\mathcal{L}_{\mathrm{CE}}$ is the cross-entropy loss of the student model and α, β are corresponding hyper parameters.

7 Experiments

We evaluate our methods on two real-world, large-scale machine translation tasks: En→De and En→Ko. Although we evaluate our proposed recipes on both tasks, due to limited space we move the intermediate results for En→Ko to the Appendix A.

The overall goal is to build a system that preserves 90% translation quality of the baseline, while yielding as much translation speed and energy savings as possible.

7.1 Data

We aim to evaluate systems in an realistic, industry-scale setting by training general-domain MT systems on a large dataset. To evaluate the general-domain quality of the MT systems, we chose four test sets of varying domains for both the En→De and En→Ko task. The corresponding training data was selected as a mix of in-domain data matching the test set domains, and general out-of-domain and crawled data.

For En→De, we use four test sets from the movie subtitles (OpenSubtitles 2018 test set), talks (TED tst2018), news (WMT newstest2019), and parliament speech domain (Europarl ST test) (Tiedemann, 2012; Tiedemann and Thottingal, 2020). Concatenated, this test set spans a total of 9k sentences with 154k source words. As training data, we select four corresponding corpora from OPUS (OpenSubtitles 2018, TED 2020, NewsCommentary & Global Voices, Europarl), and mix it with out-of-domain data in a ratio of 3:1. In total, the models are trained on 90M bilingual sentence pairs with 1.4B target words—detailed statistics can be found in the Appendix B.

For En→Ko, we report on four test sets comprised of subtitles (2018 OpenSubtitles test set), talks (TED tst2018), newswire texts from FBIS (2013 test set), as well as a general-domain test set from the Korean-English treebank (2013 version). In total, these test sets have 10k sentences with 123.6k source words. Generally, less data is publicly available for this language pair and we train on a total of 28M sentence pairs with 201M target words from these corpora.

7.2 Setup

We train models based on the Transformer architecture and implemented them in PyTorch 2.5 (Paszke et al., 2019). We apply byte-pair encodings (Sennrich et al., 2016; Kudo and Richardson, 2018) to the training data and obtain a source and target vocabulary of 30k and 10k units respecively. Segmentation and casing is encoded via two separate translation factors (Wilken and Matusov, 2019). Our models are trained for 250 sub-epochs of 1M sentences each using the Adam optimizer (Loshchilov and Hutter, 2019) with weight decay 0.01 and base learning rate of $3{\cdot}10^{-4}$, 0.1 label smoothing and 0.1 dropout. For more details we refer to Appendix B.

In the following, we abbreviate a model with L_E encoder and L_D decoder layers by $L_\mathrm{E}/L_\mathrm{D}$, e.g. a 12/12 model refers to a system with 12 encoder and decoder layers each.

For testing, we evaluate both the translation quality, as well as the inference speed and energy consumption of each of the trained model configurations. For this we compute BLEU (Papineni et al., 2002) and COMET (Rei et al., 2020) for each of the language pair's four test sets, and in the interest of readability report the average of each metric in the tables. BLEU is computed via Sacrebleu (Post, 2018) with paired approximate randomization (Riezler and III, 2005), COMET uses the checkpoint `Unbabel/wmt22-comet-da` and bootstrap resampling (Koehn, 2004). As the use of white spaces is inconsistent in Korean, we report character-level BLEU for this language pair, and report BLEURT (Sellam et al., 2020) instead of COMET[1].

Profiling the CPU time and energy is done on a single fixed machine which resembles a server-typical setup with a total of 128 Intel® Xeon® Gold 6438Y+ CPUs (Sapphire Rapid) and 500 GB RAM running Ubuntu 22. We reserve this machine exclusively for the translation executable which we give access to a subset of 8 of these CPUs. For each model, we concatenate the four language-pair specific test sets, and then measure the total time and energy the translation itself takes. Sequences are sorted by their length to reduce the amount of padding, translating 16 sentences at a time in each batch. Our models are converted into TorchScript and automatically optimized using PyTorch's JIT engine[2]. As it may take longer to translate when the model and data is first loaded , we run a translation of all data once as a warmup phase before profiling the actual runtime and energy usage.

The CPU energy usage is measured in a separate thread by rapidly polling the current power usage[3] and integrating over time. This also includes the passive energy usage by running the node and operation system. As the runtime and energy consumption varies between different executions, we run each inference three times and then report the median to omit outliers. For GPU inference we use one NVIDIA RTX 2080 Ti[4].

We report the translation speed measured in untokenized English words per second (WPS), and the energy consumption in kJ for running the complete inference.

[1] In our experiments, COMET turned out highly unreliable for Korean; achieving ≈ 50 COMET even for completely broken models as can also be seen in Table 1 of Lee et al. (2025)'s work.

[2] Using `torch.jit.optimize_for_inference`

[3] Via s-tui, https://github.com/amanusk/s-tui

[4] Running `nvidia-smi --query-gpu=power.draw` periodically to poll the power usage

Beam Size	BLEU	COMET	WPS	kJ
32	35.3	85.4	18	2462
16	35.3	85.3	31	1429
12	35.3	85.4	49	931
8	35.3	85.3	77	588
4	35.1	85.3	135	338
2	34.8	85.2	184	239
1	34.3	85.0	280	166
greedy	34.3	85.0	329	145
+ quantize	**34.4**	**84.8**	**776**	**66**

Table 1: Impact of beam size on quality, speed and energy for the En→De 12/12 system. We also investigate greedy search and quantization.

Enc	Dec	BLEU	COMET	WPS	kJ
12	12	34.4	84.8	776	66
18	6	34.3	85.0	1.1k	51
21	3	34.2	84.9	1.5k	45
23	1	33.7	83.4	1.9k	42
12	1	33.5	83.0	2.8k	39
6	1	33.0	82.3	3.8k	37
5	**1**	**32.8**	**82.2**	**4.1k**	**37**
4	1	32.3	81.7	4.4k	36
3	1	31.8	81.1	4.8k	35
2	1	30.9	80.3	5.1k	36
1	1	28.7	77.8	5.6k	35

Table 2: Impact of layer reallocation and reduction on quality, speed and energy for En→De. All systems here utilize greedy search and quantization.

Decoder	BLEU	COMET	WPS	kJ
‘big’	32.8	82.2	4.1k	37
‘base’	32.3	81.3	5.1k	36
‘small’	**31.7**	**79.8**	**6.1k**	**35**
‘tiny’	31.1	78.0	7.2k	35

Table 3: Impact of decoder width on quality, speed and energy for the En→De quantized 5/1 system with greedy search.

7.3 Evaluation: Inference Optimizations

At first, we investigate the effect of the beam size during inference in terms of translation quality, speed and energy. The results for En→De are presented in Table 1. Starting from the baseline beam size of 8, increasing it stepwise to 32 does not effect quality, but slows down translation from 77 to 18 WPS while also increasing the energy consumption by a factor of 4. This underlines that higher beam sizes yield no improvement (Yang et al., 2018).

Moving into the opposite direction, decreasing the beam size down to 1 yields a 3.6× speedup and a 3.5× energy efficiency boost at the cost of 1.0 BLEU and 0.3 COMET. To discard the overhead of beam search, we replace it by a greedy search implementation. This does not affect quality, but further improves speed and energy to 329 WPS and 145 kJ, respectively.

Additionally quantizing the model results in a translation speed of 776 WPS while consuming 66 kJ, which corresponds to improvement factors of 10.1 and 8.9 over the baseline. If not mentioned otherwise, we use greedy search and quantization for all evaluations to follow, all being carried out on CPUs.

7.4 Evaluation: Architectural Optimizations

We then investigate the impact of reallocating layers from the decoder to the encoder (see Table 2, lines 1–4). Starting from the 12/12 baseline model, we can shift 6 or even 9 layers without seeing a degradation in quality, increasing translation speed from 776 to 1.1k WPS. Moving all but one layer to the encoder results in the 23/1 model which is nearly 2.5× faster than the 12/12 model at the cost of 0.7 BLEU points. The energy consumption is improved by a factor of more than 1.5.

Furthermore, we decrease the number of layers in the decoder down to a point where it is still acceptable in terms of translation quality, bearing in mind that we also want to improve the decoder width. The results are presented in the bottom part of Table 2. Given our minimum 90% threshold of our baseline, i.e. 31.8 BLEU, the 3/1 model with 4.8k WPS would be still acceptable. However, as we want to have the possibility to further improve the decoder width, we settle with the 5/1 model as trade-off between quality and efficiency. Note, that the number of encoder layers has only a minor effect on the energy consumption (lines 4-11).

Having settled on a quantized 5/1 model with greedy search, we proceed with decoder-width optimization (see Table 3). Starting with a Transformer ‘big’ decoder, we stepwise halve the decoder dimension d until reaching the Transformer ‘tiny’ model with only $1/8$ dimension width in comparison to Transformer ‘big’ (where $d = 1024$). The number of attention heads and the feedforward di-

Decoder	BLEU	COMET	WPS	kJ
Transformer	**31.7**	79.8	6.1k	35
SSRU	30.5†	78.4†	**7.3k**	35
+ interleave	30.6†	79.8	**7.3k**	35
LSTM	7.6†	43.3†	6.0k	35
+ interleave	31.4	**80.5**†	7.0k	35

Table 4: Different decoder architectures, applied to the En→De quantized 5/1 model with the 'small' decoder size. † indicates a statistically significant difference w.r.t. the Transformer decoder ($p < 0.005$).

Decoder	KD	BLEU	COMET	WPS
Transformer	✗	31.7	79.8	6.1k
	✓	**32.2**	80.5	6.1k
SSRU interl.	✗	30.6	79.8	7.3k
	✓	31.7	81.0	**7.4k**
LSTM interl.	✗	31.4	80.5	7.0k
	✓	32.0	**81.3**	7.0k

Table 5: Impact of knowledge distillation (KD) on quality, speed and energy for the En→De quantized 5/1 system with a Transformer 'small' decoder and greedy search. All systems have consumption of 34–35 kJ. All three KD systems have pairwise statistically significantly different COMET scores ($p < 0.005$).

mension is down-scaled accordingly. Although the quality loss per step is 0.5–0.6 BLEU, the gain in speed becomes less while the energy consumption is basically constant. Thus, we choose the Transformer 'small' decoder (decoder model dimension $d = 256$, 4 attention heads, $d_{\text{ff}} = 1024$) for further experiments, as it offers a good trade-off between the speedup and quality loss.

7.5 Evaluation: RNN Decoder Replacement

A common approach to speed up the inference for MT is to replace the decoder with an RNN (Kim et al., 2019; Hsu et al., 2020; Lin et al., 2021). However, existing research has not studied the performance of such hybrid models when they are combined with inference optimizations such as greedy search and quantization—which for themselves already yield a speedup of factor 10 at only minor drop in quality.

In Figure 3 we show that the effect of quantization and beam size reduction varies between different metrics: we compare our hybrid models with Transformer and LSTM decoders (with interleaving). Across all metrics, the hybrid LSTM decoder performs better than the Transformer decoder when using beam search and not applying quantization. However, in BLEU this advantage vanishes when decoding with quantization and greedy search. For COMET and BLEURT, this is not the case.

We proceed to compare different RNN architectures: the standard LSTM (Hochreiter and Schmidhuber, 1997) and the optimized SSRU (Kim et al., 2019); and apply our proposed interleaving approach to both models.

As shown in Table 4, the interleaved SSRU decoder is faster but the quality is worse. Interleaving proves to be crucial for the LSTM decoder, which offers the best speed-quality trade-off: It has the best COMET score of 80.5, and despite its 31.4 BLEU being slightly lower than the 31.7 BLEU gained with the Transformer decoder, this difference is statistically not significant. At the same time it is **91× faster** than the baseline model.

7.6 Evaluation: Training Optimization

After having improved the model speed and energy consumption, we shift our focus on improving its translation quality through knowledge distillation (KD). We utilize the strongest model—the 12/12 Transformer 'big' baseline—as the teacher and weigh the CE and KD losses with $\alpha = 0.5$ and $\beta = 1.0$, respectively, since this setting performed best in prior experiments.

Table 5 presents the performance of the models trained with KD in comparison to their counterparts without. The Transformer system with KD performs best in BLEU with a small 0.2 advantage over the interleaved LSTM with KD, which in turn outperforms the Transformer by 0.8 COMET and is faster by 0.9k WPS. On the other hand, the interleaved SSRU offers the fastest inference speed with 7.4k WPS, but its performance in both BLEU and COMET is worse by 0.3 in comparison to the interleaved LSTM.

Note, that the BLEU score of 31.7 gained by the interleaved SSRU with KD corresponds to only 89.8% translation quality w.r.t. our baseline Transformer system, which is does not fulfill our goal to preserve at least 90% quality.

7.7 Evaluation: Summary

Table 6 summarizes the gains by all optimization techniques proposed in this work when applied incrementally for both the En→De and En→Ko tasks.

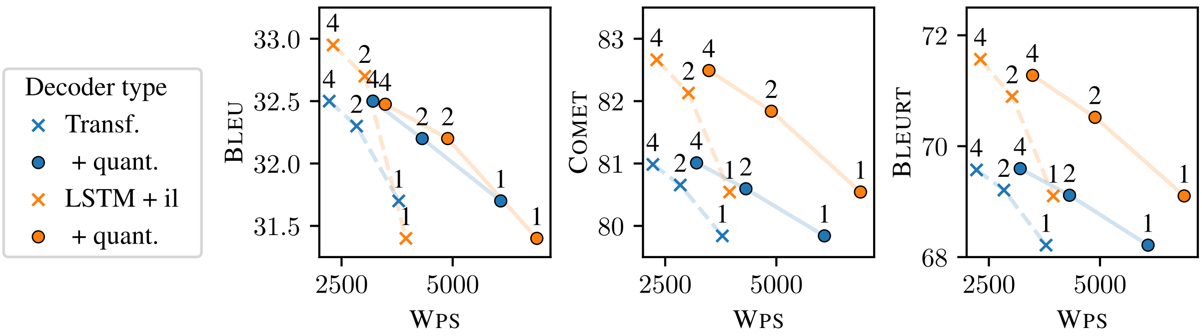


Figure 3: Comparing the decoding quality and speed, with different beam sizes: 4, 2, and 1 (using greedy search). All models use the 5/1 architecture with either a Transformer 'small' or interleaved LSTM 'small' decoder.

Technique	English → German						English → Korean			
	Quality		CPU		GPU		Quality		CPU	
	BLEU	COMET	WPS	kJ	WPS	kJ	BLEU	BLEURT	WPS	kJ
Transformer 'big'	35.3	85.3	77	588	484	76	28.1	57.6	90	413
+ greedy search	34.3	85.0	329	145	1.3k	23	27.3	56.9	316	123
+ quantize	34.4	84.8	776	66	n.a.	n.a.	27.2	56.8	785	60
+ depth opt.	32.8	82.2	4.1k	37	6.0k	5.6	25.6	55.3	3.6k	42
+ width opt.	31.7	79.8	5.8k	35	6.6k	3.4	24.9	54.2	5.4k	41
+ LSTM interl.	31.4	80.5	7.0k	35	6.9k	3.0	24.7	54.2	5.8k	40
+ KD	**32.0**	**81.3**	**7.0k**	**35**	**6.9k**	**2.9**	**25.7**	**55.4**	**5.9k**	**40**

Table 6: Incrementally applying all proposed techniques to the En→De and En→Ko task. We report inference speed (WPS) and energy consumption (kJ) on CPU and GPU.

Details on all intermediate results for En→Ko are to be found in the Appendix A.

For En→De, we preserve 90.7% relative BLEU and 95.3% relative COMET, i.e. we lose 3.3 BLEU and 4.0 COMET absolute. At the same time, we gain **91× translation speed** and **94.0% energy savings** on CPU. An extended analysis reveals that the speed gains are particularly high for long sequences (see appendix, Figure 4).

On GPU we obtain 14.3× translation speed and 96.2% energy savings. Note, that the final model is 14.5× faster on CPU than the vanilla Transformer 'big' model on GPU. Overall, our final model is as fast on CPU as it is on GPU. Note, that no GPU kernels exist in PyTorch 2.5 for quantization. Hence, all GPU results are obtained without quantization.

In the En→Ko test sets, there are 1.19 target tokens per source token on average, whereas in the En→De test sets this number is 1.36. As there are less target words to be generated per source word in the En→Ko task, there are less decoding steps, which explains why for En→Ko the baseline system achieves a translation speed of 90 WPS.

Overall, the speedup for En→Ko is 65-fold and our finals systems achieves 90% energy savings, while still achieving 91% relative BLEU and 96% relative BLEURT, which underlines the general applicability and gains of our proposed optimizations. However, as there are less decoding steps than in the En→De task on average, there is also less gain to be expected by the proposed techniques.

8 Conclusion

In this work, we introduced Green KNIGHT, an easy-to-cook recipe that substantially accelerates inference and reduces the energy consumption of NMT models while incurring only minimal loss in translation quality. In contrast to specialized low-level or hardware-specific optimizations, Green KNIGHT achieves these gains through widely used and well-understood tools and methods, making it immediately adoptable in production NMT pipelines. Our experiments on two language pairs show that this carefully crafted recipe achieves speed-ups up to 91×, reduces energy consumption up to 94%, while losing not more than 4.7% relative COMET or 9.5% relative BLEU compared to the baseline. Crucially, the final models achieve the same throughput on both the CPU and GPU, significantly contributing to democratizing NMT.

Limitations

The empirical relevance of this work might be limited by the tasks we report on and the evaluation. We report on two high-resource datasets translating from English as the source language. Although, according to our findings, the encoder (and therefore the source language) does not seem to be the bottleneck, further investigation would be needed to confirm this. We base our findings on higher-resource language pairs and do not investigate low-resource settings or other language pairs.

Due to resource constraints, we were also only able to perform a single training run per reported system. Furthermore, our evaluation is limited to automatic metrics. We validate our results using a range of automatic metrics (COMET, BLEURT, and BLEU), but we do not perform a human evaluation.

We only reported results measured on a single machine and one specific driver version for the measurement of translation speed and energy. Although the setup used is typical for a server CPU, using different hardware might impact the translation speed and energy consumption. Furthermore, time and energy measurements inherently suffer from some variance between runs, which can depend on exterior factors such as the server's temperature or system background jobs.

Potential Risks

As our primary results are based on automated metrics, they do not necessarily reflect the quality as assessed by humans. This is especially true for neural metrics such as COMET and BLEURT, which are not auditable and may lead to unpredictable results in varying domains and language pairs. E.g., we observed that COMET did not give reasonable scores for our English to Korean evaluations, despite the authors claiming it should work for this language pair[5]. Relying blindly on these metrics to make decisions can lead to potential misjudgments in translation quality.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Aishwarya Bhandare, Vamsi Sripathi, Deepthi Karkada, Vivek Menon, Sun Choi, Kushal Datta, and Vikram A. Saletore. 2019. Efficient 8-bit quantization of transformer neural machine language translation model. *CoRR*, abs/1906.00532.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Fangxiaoyu Feng, Yinfei Yang, Daniel Cer, Naveen Arivazhagan, and Wei Wang. 2022. Language-agnostic BERT sentence embedding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 878–891. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yi-Te Hsu, Sarthak Garg, Yi-Hsiu Liao, and Ilya Chatsviorkin. 2020. Efficient inference for neural machine translation. In *Proceedings of SustaiNLP: Workshop on Simple and Efficient Natural Language Processing*, pages 48–53, Online. Association for Computational Linguistics.

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.

Young Jin Kim, Marcin Junczys-Dowmunt, Hany Hassan, Alham Fikri Aji, Kenneth Heafield, Roman Grundkiewicz, and Nikolay Bogoychev. 2019. From research to production and back: Ludicrously fast neural machine translation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 280–288, Hong Kong. Association for Computational Linguistics.

Tom Kocmi, Eleftherios Avramidis, Rachel Bawden, Ondrej Bojar, Anton Dvorkovich, Christian Federmann, Mark Fishel, Markus Freitag, Thamme Gowda, Roman Grundkiewicz, Barry Haddow, Marzena Karpinska, Philipp Koehn, Benjamin Marie, Christof Monz, Kenton Murray, Masaaki Nagata, Martin Popel, Maja Popovic, and 3 others. 2024. Findings of the WMT24 general machine translation shared task: The LLM era is here but MT is not solved yet. In *Proceedings of the Ninth Conference on Machine Translation, WMT 2024, Miami, FL, USA, November 15-16, 2024*, pages 1–46. Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of*

[5] `https://github.com/Unbabel/COMET`

the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain, pages 388–395. ACL.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018: System Demonstrations, Brussels, Belgium, October 31 - November 4, 2018*, pages 66–71. Association for Computational Linguistics.

Daniel Lee, Harsh Sharma, Jieun Han, Sunny Jeong, Alice Oh, and Vered Shwartz. 2025. Team ack at semeval-2025 task 2: Beyond word-for-word machine translation for english-korean pairs. *Preprint*, arXiv:2504.20451.

Ye Lin, Yanyang Li, Tong Xiao, and Jingbo Zhu. 2021. Bag of tricks for optimizing transformer efficiency. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4227–4233, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Ye Lin, Xiaohui Wang, Zhexi Zhang, Mingxuan Wang, Tong Xiao, and Jingbo Zhu. 2023. MobileNMT: Enabling translation in 15MB and 30ms. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 368–378, Toronto, Canada. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2017. Mixed precision training. *CoRR*, abs/1710.03740.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Z. Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, and 2 others. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 8024–8035.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 186–191. Association for Computational Linguistics.

Ricardo Rei, Craig Stewart, Ana C. Farinha, and Alon Lavie. 2020. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 2685–2702. Association for Computational Linguistics.

Stefan Riezler and John T. Maxwell III. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization@ACL 2005, Ann Arbor, Michigan, USA, June 29, 2005*, pages 57–64. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM*, 63(12):54–63.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7881–7892. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics.

Dimitar Shterionov and Eva Vanmassenhove. 2023. The ecological footprint of neural machine translation systems. In Helena Moniz and Carla Parra Escartín, editors, *Towards Responsible Machine Translation - Ethical and Legal Considerations in Machine Translation*, volume 4, pages 185–213. Springer.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 3645–3650. Association for Computational Linguistics.

Yehui Tang, Yunhe Wang, Jianyuan Guo, Zhijun Tu, Kai Han, Hailin Hu, and Dacheng Tao. 2024. A survey on transformer compression. *CoRR*, abs/2402.05964.

Jörg Tiedemann. 2012. Parallel data, tools and interfaces in OPUS. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 2214–2218. European Language Resources Association (ELRA).

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT - building open translation services for the world. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, EAMT 2020, Lisboa, Portugal, November 3-5, 2020*, pages 479–480. European Association for Machine Translation.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, pages 5998–6008. Curran Associates, Inc.

Patrick Wilken and Evgeny Matusov. 2019. Novel applications of factored neural machine translation. *CoRR*, abs/1910.03912.

Yilin Yang, Liang Huang, and Mingbo Ma. 2018. Breaking the beam search curse: A study of (re-)scoring methods and stopping criteria for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 3054–3059. Association for Computational Linguistics.

Biao Zhang, Deyi Xiong, and Jinsong Su. 2018. Accelerating neural transformer via an average attention network. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1789–1798, Melbourne, Australia. Association for Computational Linguistics.

Beam Size	BLEU	BLEURT	WPS	kJ
32	27.8	57.3	22	1645
16	27.9	57.5	38	977
12	27.9	57.4	58	649
8	28.1	57.6	90	413
4	28.1	57.5	151	245
2	27.8	57.4	196	184
1	27.3	56.9	287	133
greedy	27.3	56.9	315	122
+ quantize	**27.1**	**56.8**	**785**	**60**

Table 7: Various beam sizes of En-Ko system with a 12-layer encoder and a 12-layer decoder. We also investigate greedy search and quantization.

Enc	Dec	BLEU	BLEURT	WPS	kJ
12	12	27.2	56.8	785	60
23	3	27.3	57.2	1.3k	49
21	1	26.6	56.3	1.7k	47
12	1	26.3	55.9	2.4k	44
6	1	25.7	55.4	3.3k	42
5	**1**	**25.6**	**55.3**	**3.6k**	**42**
4	1	25.1	55.0	3.7k	42
3	1	24.9	54.7	4.0k	41
2	1	24.1	53.6	4.2k	42
1	1	22.4	51.5	4.8k	41

Table 8: Comparison of En-Ko systems with varying encoder and decoder depth. All investigated systems utilize greedy search with quantization.

A Detailed Results on Secondary Task

We verify the effectiveness of the proposed optimizations on the English to Korean task.

The procedure here is exactly the same as for English to German and investigate the beam size first as shown in Table 7. We choose the beam size of 8 as a baseline, as it gives the best BLEU and BLEURT. Then we follow the recipe as described for English to German. We then start with the inference optimization (greedy search and quantization) in Table 7. Then we optimize the architecture, i.e. the depth and width shown in Tables 8, and 9 respectively. We then replace the decoder with our LSTM implementation, and apply knowledge distillation, as shown in Table 10. In each step, the chosen hyperparameters correspond to the same ones as for the English to German task.

Decoder	BLEU	BLEURT	WPS	kJ
'big'	25.6	55.3	3.6k	42
'base'	25.1	54.8	4.7k	41
'small'	**24.9**	**54.2**	**5.4k**	**41**
'tiny'	24.3	53.4	5.6k	40

Table 9: Comparison of En-Ko system with various decoder width. All investigated system utilize a 5-layer Transformer 'big' encoder, a single layer Transformer decoder and greedy search with quantization.

Decoder	BLEU	BLEURT	WPS	kJ
Transformer	24.9	54.2	5.4k	41
+ KD	25.4	54.9	5.5k	40
Interl. LSTM	24.7	54.2	5.8k	40
+ KD	25.7	55.4	5.9k	40

Table 10: Impact of training optimization on quality, speed and energy for the En→Ko quantized 5/1 system with a 'small' decoder and greedy search.

B Training Details

The statistics of our training and test data is presented in Table 11. The exact training corpora are stated in Tables 12a and 12b. We apply some filtering to our training data based on a set of rules, as well as similarity based on LaBSE embeddings (Feng et al., 2022).

All models, independently of the configuration, are trained for 250 sub-epochs of 1M samples. Our optimizer is AdamW (Loshchilov and Hutter, 2019) with $\beta = (0.9, 0.98)$, weight decay 0.01 and a learning rate of $3 \cdot 10^{-4}$, which after a warmup of ten epochs is reduced by factor 0.9 if the validation perplexity plateaus. We use 16-bit mixed precision training (Micikevicius et al., 2017) as provided by PyTorch lighting, and an effective batch size of up to 120k source plus target tokens. We apply a training dropout of 0.1 and label smoothing of 0.1.

For both language pairs, we compile a heldout validation set that approximately equally represent the four test sets, and use this validation set to select the best checkpoint after each sub-epoch by computing validation BLEURT.

Our En→De baseline system has 485M trainable parameters and was trained on two NVIDIA RTX A6000 GPUs, which took around 103 hours to complete. Other models train faster due to their reduced complexity. All datasets and tools that this work is based on are publicly available.

Dataset	Sentences	Total Words		Vocabulary Words	
		English	German	English	German
Training Data	90.6M	1.6B	1.4B	11.3M	22.9M
Test Data (total)	8984	154.0k	142.7k	29.6k	36.2k
WMT newstest2019	1997	42.0k	42.1k	10.6k	12.4k
TED tst2018	1978	38.0k	35.1k	6.7k	8.4k
Europarl ST	2631	60.4k	52.3k	8.2k	11.2k
OpenSubtitles 2018	2378	13.6k	13.1k	4.0k	4.3k

(a) English to German task

Dataset	Sentences	Total Words		Vocabulary Words	
		English	Korean	English	Korean
Training Data	27.9M	265M	201M	183.2M	3.4M
Test Data (total)	10483	123.6k	83.4k	16.0k	77.7k
FBIS test 2013	676	21.8k	13.0k	4.7k	12.6k
Korean English treebank	3883	51.5k	36.7k	5.9k	33.7k
TED tst2015	1214	21.2k	15.1k	4.9k	14.4k
OpenSubtitles 2018	4710	29.1k	18.6k	6.0k	16.9k

(b) English to Korean task

Table 11: Total training and test set sizes.

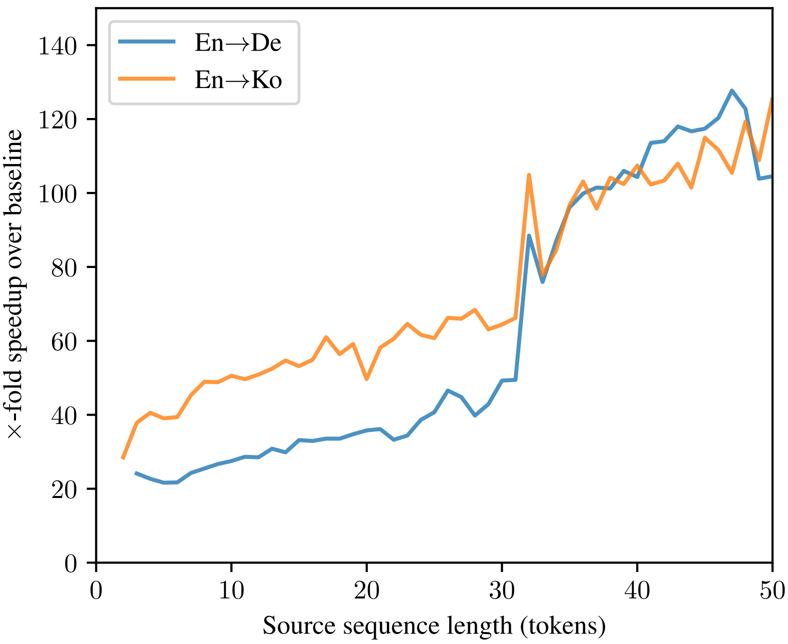


Figure 4: Relative CPU speed up of our optimized model (final row in Table 6) vs. the Transformer 'big' baseline, binned by source sequence length. While the baseline scales poorly with increasingly long sequences, this effect is mitigated with our optimized models. The graph also shows that our models are faster at decoding sentences with 32 or less tokens. This is because longer sequences exhaust hardware parallelization or cache capabilities. These capabilities are exhausted faster with the larger model, thus with the smaller optimized model we see this spike in speedup for larger sequences.

10%	OPUS-OpenSubtitles (16,166,700)
5%	OPUS-TED2020 (162,134)
5%	News-Commentary (317,129), OPUS-GlobalVoices (83,240)
5%	OPUS-Europarl (2,308,549)
65%	pattr (12,183,523), OPUS-CCAligned (10,876,712), OPUS-EuroPat (10,664,245), OPUS-EUbookshop (5,459,744), OPUS-TildeMODEL (3,249,472), OPUS-MultiCCAligned (2,638,152), OPUS-ELRC (2,599,018), OPUS-ParaCrawl (2,551,919), OPUS-DGT (2,240,204), OPUS-JW300 (1,707,885), OPUS-WikiMatrix (1,139,146), OPUS-Wikipedia (1,073,073), rapid (692,934), OPUS-Tatoeba (546,960), CommonCrawl (523,024), OPUS-Tanzil (492,585), WikiTitles (487,528), OPUS-QED (417,637), OPUS-JRC-Acquis (265,780), covost (258,177), OPUS-EMEA (201,860), EUTV (152,233), must-c (115,563), OPUS-KDE4 (100,791), OPUS-MultiUN (63,833), OPUS-ECB (63,277), OPUS-bible-uedin (37,857), OpenOffice (25,980), OPUS-MPC1 (15,794), OPUS-GNOME (12,814), OPUS-Ubuntu (6,971), OPUS-PHP (6,557), OPUS-EUconst (1,928), OPUS-Salome (1,057), OPUS-RF (165)
10%	extracted parallel short phrases and dictionary entries from the above corpora

(a) English to German training data

10%	OPUS-TED2020 (323,188)
1%	fbis (39,867)
14%	OPUS-OpenSubtitles (947,351)
65%	OPUS-NLLB (13,736,682), OPUS-CCMatrix (3,799,459), OPUS-ParaCrawl (2,267,324), OPUS-CCAligned (2,199,281), OPUS-LinguaTools-WikiTitles (1,533,792), systran (576,744), OPUS-MultiCCAligned (475,984), naver (375,119), OPUS-XLEnt (328,552), taus (315,934), subscene (188,012), jaykim (118,297), OPUS-QED (112,298), OPUS-WikiMatrix (88,069), OPUS-Tanzil (62,991), jhe-park (52,850), joongang (47,555), joint-pubs (42,438), OPUS-bible-uedin (40,161), kaist (30,269), OPUS-KDE4 (23,249), OPUS-wikimedia (18,285), osc translated text (15,813), goodneighbor (14,484), various-book1-johanna (13,513), OPUS-Tatoeba (11,403), donga-ilbo (10,728), various-military (9,202), OPUS-Mozilla-I10n (6,791), OPUS-GlobalVoices (6,108), bible world (4,794), sejong (4,632), OPUS-MDN Web Docs (3,655), kgf (3,442), various-unknown-topic (2,871), nvtc (2,673), OPUS-tldr-pages (1,096), OPUS-ELRC (732), usembassy (575), usfkgovplan (204), various-medical (140), OPUS-PHP (126), social-media (92), OPUS-GNOME (74), OPUS-Ubuntu (13)
10%	extracted parallel short phrases and dictionary entries from the above corpora

(b) English to Korean training data

Table 12: Training dataset statistics per weight group. Training data is mostly taken from OPUS (Tiedemann and Thottingal, 2020) and then filtered. We report the number of sentences after filtering here. Before training, we additionally apply sentence deduplication.