Locality-Sensitive Hashing for Long Context Neural Machine Translation

Frithjof Petrick **Jan Rosendahl** **Christian Herold** **Hermann Ney**
Human Language Technology and Pattern Recognition Group
Computer Science Department
RWTH Aachen University
D-52056 Aachen, Germany
surname@i6.informatik.rwth-aachen.de

Abstract

After its introduction, the Transformer architecture (Vaswani et al., 2017) quickly became the gold standard for the task of neural machine translation. A major advantage of the Transformer compared to previous architectures is the faster training speed achieved by complete parallelization across timesteps due to the use of attention over recurrent layers. However, this also leads to one of the biggest problems of the Transformer, namely the quadratic time and memory complexity with respect to the input length. In this work we adapt the locality-sensitive hashing approach of Kitaev et al. (2020) to self-attention in the Transformer, we extended it to cross-attention and apply this memory efficient framework to sentence- and document-level machine translation. Our experiments show that the LSH attention scheme for sentence-level comes at the cost of slightly reduced translation quality. For document-level NMT we are able to include much bigger context sizes than what is possible with the baseline Transformer. However, more context does neither improve translation quality nor improve scores on targeted test suites.

1 Introduction

After its introduction in 2017, the Transformer architecture (Vaswani et al., 2017) quickly became the gold standard for the task of neural machine translation (NMT) (Ott et al., 2018). Furthermore, variants of the Transformer have since been used very successfully for a variety of other tasks such as language modeling (LM) (Irie et al., 2019), natural language understanding (NLU) (Devlin et al., 2019; Liu et al., 2019), speech translation (ST) (Vila et al., 2018), automatic speech recognition (ASR) (Zeyer et al., 2019; Mohamed et al., 2019) and image processing (Parmar et al., 2018).

A major advantage of the Transformer compared to previous architectures is the faster training speed achieved by complete parallelization across timesteps. However, this also leads to one of the biggest problems of the Transformer, namely the quadratic time and memory complexity of attention layers with respect to the sequence length. For sentence-level NMT this is not a big issue as most of the time the length of sequences is relatively short and can be handled efficently, even if subword segmentation is applied (Sennrich et al., 2016; Kudo, 2018). However, this drastically changes when moving towards character-level (Gupta et al., 2019) or document-level (Tiedemann and Scherrer, 2017) NMT. Especially for the latter, speed and memory issues are one of the biggest roadblocks towards 'true' document level systems (Junczys-Dowmunt, 2019). This leads to the situation where most works make do with including just a few sentences as a form of 'local' context information (Tiedemann and Scherrer, 2017; Jean et al., 2017; Bawden et al., 2018) or heavily compressing the document information (Tu et al., 2018; Kuang et al., 2018; Morishita et al., 2021).

More recently research focus has been shifting towards more efficient attention calculation for longer input sentences in several LM and NLU tasks (Tay et al., 2020). Among these works is the approach by Kitaev et al. (2020), in which the authors propose to make the attention matrix sparse by pre-selecting the relevant positions. They report good results on the LM objective while at the same time drastically reducing computational complexity. In this work we take the approach of Kitaev et al. (2020) as a starting point to improve the efficiency of (document-level) NMT systems.

Our contribution is three-fold:

- We adapt the locality-sensitive hashing (LSH) approach of Kitaev et al. (2020) to self-attention in the Transformer NMT framework.[1]

[1]The source code is available at https://github.com/rwth-i6/returnn-experiments/tree/master/2022-lsh-attention.

- We expand the concept of LSH to encoder-decoder cross-attention and provide insights on how this concept affects the behavior of the system.

- We use this more memory-efficient NMT framework to conduct experiments on document-level NMT with more context information as would be possible with the baseline architecture.

2 Related Work

The problem of quadratic time and memory complexity of the attention framework has received increasing attention since the success of the Transformer architecture (Vaswani et al., 2017).

For ASR, ST and image processing the complexity can be reduced with relative ease by reducing the size of the time dimension with convolutional (Gulati et al., 2020) or pooling layers (Zeyer et al., 2019). Furthermore, it is possible to restrict the attention to a few neighboring positions (Parmar et al., 2018). However, this is not optimal for text input, as neighboring input words do not necessarily have the same strong correlation as neighboring audio frames or image pixels.

Existing work on improving the text processing complexity of the Transformer mainly focuses on the case where all attention inputs come from the same embedding space, e.g. language modeling: Dai et al. (2019) and Rae et al. (2019) utilize a segment-level recurrence mechanism similar to what has been used in recurrent architectures. Wang et al. (2020) project the time dimension of key and value down to a smaller, fixed-size dimension while leaving the queries untouched. Directly altering the attention computation, Child et al. (2019), Sukhbaatar et al. (2019) and Qiu et al. (2020) limit the attention to a local neighborhood or a fixed stride while Zaheer et al. (2020) and Beltagy et al. (2020) combine multiple sparse attention masks. In a more flexible approach, matching positions can be pre-selected using a locality-sensitive hashing function (Kitaev et al., 2020) or clustering (Roy et al., 2021). In the present work, we pick one of the most efficient and best performing approaches up to date, namely the approach by Kitaev et al. (2020) and apply it to the task of machine translation. We confirm that the concepts can work for the self-attention in NMT systems and expand the framework for the case of cross-attention.

Most work related to document-level NMT limit the inter sentence context to few neighboring sentences. The simplest approach which we also follow in the present work, is to concatenate consecutive sentences using a special sentence separator token (Tiedemann and Scherrer, 2017). There exist more sophisticated approaches which utilize separate encoders for the context information (Jean et al., 2017; Bawden et al., 2018) but later work seems to suggest that these approaches do not significantly outperform the simpler concatenation approach (Huo et al., 2020; Lopes et al., 2020).

In the realm of NMT, not so much work exists regarding improving the efficiency of the system and the work that exists mainly focuses on document-level NMT. Morishita et al. (2021) propose to compress the context into a single vector which then can be attended to as an additional token embedding. Tu et al. (2018) and Kuang et al. (2018) utilize a cache that holds context information. Zhang et al. (2020) and Bao et al. (2021) mask out the attention energies between tokens from different sentences, showing that the full context is not necessary to achieve good translation performance. Raganato et al. (2020) and You et al. (2020) replace most attention heads with fixed patterns but only for sentence-level NMT and only for self-attention as they report a severe degradation when doing the same for the cross-attention.

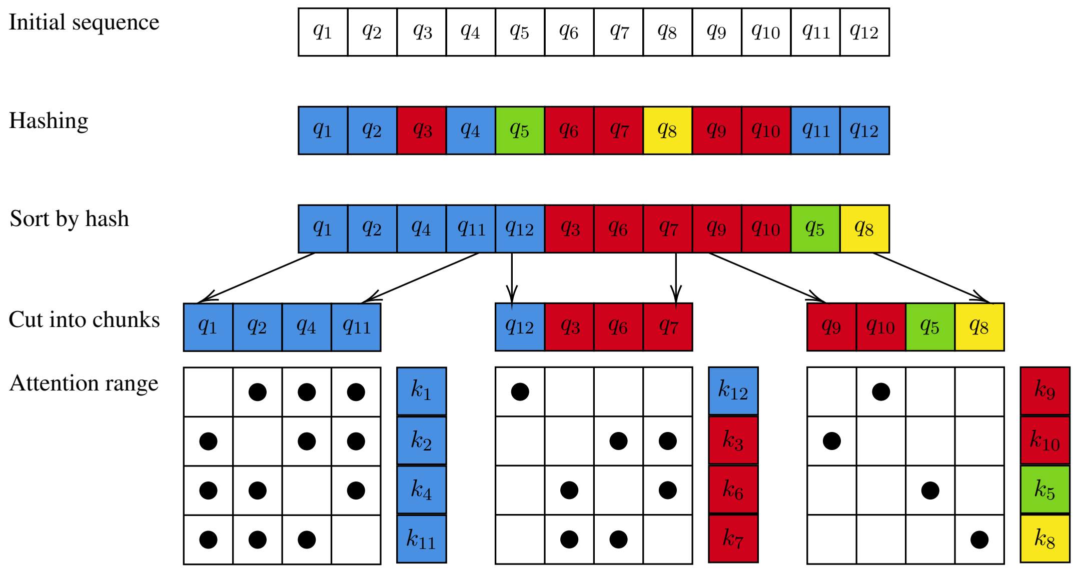
There exist several different ways to implement LSH (Paulevé et al., 2010). The LSH scheme used by Kitaev et al. (2020) and consecutively in this work was proposed by Andoni et al. (2015). LSH has also been successfully applied to efficiently calculate pairwise embedding similarity for information retrieval (Ture et al., 2011; Zhao et al., 2015). Shi and Knight (2017) use LSH to pre-select embeddings in the softmax operation of an NMT system to speed up the decoding process.

3 Locality-sensitive Hashing Attention

At the core of the Transformer architecture is the attention mechanism that compares a sequence of queries $q_1, \dots q_I$ to a sequence of key-value pairs $(k_1, v_1), \dots (k_J, v_J)$ via a soft-lookup $\alpha(j|i) = \alpha(q_i, j, k_1^J)$ and maps them to context vectors

$$c_i := \sum_{j=1}^{J} \alpha(j\,|\,i) v_j.$$

To compute the full sequence of context vectors, $\mathcal{O}(IJ)$ operations are required. In the special case



Initial sequence
Hashing
Sort by hash
Cut into chunks
Attention range
q_1 q_2 q_3 q_4 q_5 q_6 q_7 q_8 q_9 q_{10} q_{11} q_{12}
k_1 k_2 k_4 k_{11} k_{12} k_3 k_6 k_7 k_9 k_{10} k_5 k_8

Figure 1: Locality-sensitive hashing for self-attention as presented in Kitaev et al. (2020) with bidirectional context. For self-attention with key and queries shared it holds that $q_i = k_i$. Colors indicate the hash class of the query/key. Note that no position can attend to itself if other attention points are available.

of self-attention, i.e. $I = J$ and $q_i = k_i \ \forall i$, the amount of operations grows quadratically with the sequence length I. Since this can be problematic for long sequences, Kitaev et al. (2020) proposed to use locality-sensitive hashing (LSH) attention.

In the following, we first describe the concept of LSH for self-attention, here we omit the left-to-right masking originally used (Kitaev et al., 2020) and describe the concept for bidirectional self-attention instead. Afterwards, we describe our extension of LSH to cross-attention.

In LSH the context vector for query position i is computed via

$$c_i^{(\text{lsh})} := \sum_{j \in P_i} \hat{\alpha}(j \,|\, i) v_j$$

where a locality-sensitive hashing function h is used to determine

$$P_i := \{j \in \{1, \dots, J\} \setminus \{i\} \,|\, h(j) = h(i)\}$$

and $\hat{\alpha}$ is normalized over P_i instead of $\{1, \dots, J\}$.

The hashing function h maps to a small number of classes $\{1, \dots, n_{\text{hash}}\}$ and is locality-sensitive, i.e. if two vectors are close-by they are likely to get assigned the same hash value. Kitaev et al. (2020) consider the case of self-attention and approximate the set P_i to keep computation efficient. First the original sequence of keys is sorted by their hash value as primary criterion and original sequence order as secondary criterion. The resulting sequence is cut into chunks C_i of fixed size and

$$\hat{P}_i := \{j \in C_i \setminus \{i\} \,|\, h(j) = h(i)\}$$

is used as an approximation to P_i. However, if $\hat{P}_i = \emptyset$ the fallback $\hat{P}_i := \{i\}$ is used. This process is illustrated in Figure 1.

Kitaev et al. (2020) consider only the case of a) self-attention and b) shared query and key transformation matrices within each head. This focus on self-attention leads to several simplifications, in particular that the chunks of the key and query sequence are identical. In order to extend the concept of LSH to cross-attention (i.e. queries and keys are distinct) we need to solve several problems.

How to find an adequate key chunk for each query chunk? Hashing and chunking is done for both the key and the query sequences, resulting in two different chunk sequences. We propose to calculate an alignment from the query chunks to the key chunks. For each query chunk C we find an aligned key chunk $K(C)$ that contains queries with similar hash classes. To do this, the range of hash classes $(h_{\text{min}}, h_{\text{max}})$ of the query chunk C is determined. Next, we enumerate all key chunks $K_1, \dots, K_n$ and search for the first key chunk K_{j_1} that contains an entry hashed to h_{min} and the last key chunk K_{j_2} that corresponds to h_{max}. Then the middle chunk $K_{\lceil \frac{j_2+j_1}{2} \rceil}$ is selected, resulting in

$$\hat{P}_i := \{j \in K(C_i) \,|\, h(j) = h(i)\}.$$

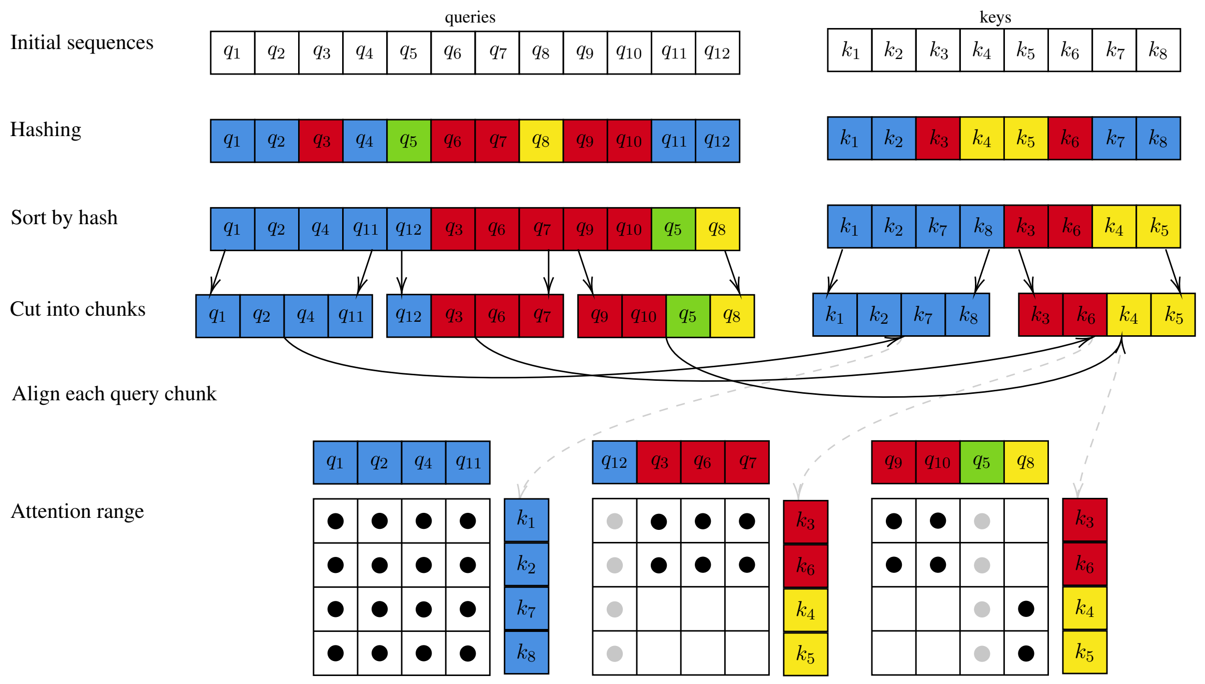


Figure 2: Locality-sensitive hashing for cross-attention. Colors indicate the hash class of the query/key. Greyed out dots in the attention range matrices indicate that attention weights are fixed to $\frac{1}{\ell_{\text{chunk}}} = \frac{1}{4}$, since no possible attention point corresponds to the current hash class.

What happens if a query belongs to a hash class that is not represented in the aligned key chunk? Since no keys are found that are close to the current query q_i, we use the average value of the aligned query chunk. That is, we set $\hat{P}_i := K(C_i)$ and obtain

$$c_i^{(\text{lsh})} := \frac{1}{|K(C_i)|} \sum_{j \in K(C_i)} v_j.$$

Throughout our experiments both key and query chunks are of equal size ℓ_{chunk}. The LSH cross-attention is shown in Figure 2.

To reduce the impact of the chunking we compute attention not only within the aligned chunk but also one chunk to the left and right, similar to Kitaev et al. (2020). This is applied both in self- and cross-attention. For unidirectional attention components, only the left context is considered.

Multi-round LSH Attention

Kitaev et al. (2020) show that multi-round hashing can help to improve the performance of LSH attention systems. For multi-round hashing different hash functions h^r are used to determine the corresponding (chunked) hash classes $\hat{P}_i^r$ and the context vector is calculated over the union

$$c_i^{(\text{lsh})} := \sum_{j \in \bigcup_r \hat{P}_i^r} \hat{\alpha}(j \,|\, i) v_j.$$

with $\hat{\alpha}(j \,|\, i)$ normalized over $\bigcup_r \hat{P}_i^r$. Multi-round hashing can be applied to both self- and cross-attention. For details on an efficient implementation we refer to Kitaev et al. (2020).

4 Experimental Setup

We evaluate our extensions to the attention by training Transformer (Vaswani et al., 2017) models with varying attention mechanisms on four MT tasks: The WMT 2016 news translation Romanian to English data with 612k parallel sentences (Europarl v8 & SE Times), the WMT 2019 English to German data with 329k parallel sentences (News Commentary v14), as well as the IWSLT 2017 English to German and English to Italian data consisting of 232k and 206k parallel sentences (TED talks). The data is pre-processed by applying 20k SPM merge operations (15k for both IWSLT tasks) (Kudo, 2018). The average sentence length for both WMT tasks is 30 subwords and 24 subwords for the IWSLT tasks.

The WMT En→De and the IWSLT En→De and En→It sentences are grouped by document. For document-level systems we utilize this information in a pre-processing step by simply concatenating the k preceding sentences on source and target side to each sentence pair like Tiedemann and Scherrer (2017) do, but experiment with larger

Attention method	Ro→En WMT Bleu	Ro→En WMT Ter	En→De WMT Bleu	En→De WMT Ter	En→De IWSLT Bleu	En→De IWSLT Ter	En→It IWSLT Bleu	En→It IWSLT Ter
Full attention (baseline)	34.2	53.3	32.1	56.7	23.3	68.4	32.8	53.6
LSH self-attention	33.5	54.3	30.5	58.6	22.9	68.6	31.6	54.7
LSH self- & cross-attention	33.3	54.3	29.3	60.0	22.3	69.4	31.9	54.7

Table 1: Translation performance when training models with LSH attention on different sentence-level tasks. We vary where to apply LSH attention: nowhere (baseline), encoder and decoder self-attention, or three-fold. All systems use $n_{\text{hash}} = 4$, $\ell_{\text{chunk}} = 6$ and four hash rounds. Bleu and Ter are given in percentage.

context sizes $k \in \{0, 3, 9, 12\}$. In particular $k = 0$ yields a sentence-level system without any document context. In between the concatenated sentences we add a special separator token. We do not utilize right side context to ensure source and target have roughly the same length.

The general system architecture follows the 'base' configuration of Vaswani et al. (2017) with 6 encoder/decoder layers of feature dimension $d_{\text{model}} = 512$, 8 attention heads and key/value dimension $d_k = 64$. We share the source/target embeddings as well as the transposed projections and employ training dropout of 30 % (20 % for Ro→En). All models are implemented in ReTURNN (Zeyer et al., 2018).

We use the Adam optimizer (Kingma and Ba, 2015) with initial learning rate of 10^{-3}. After training the systems for 200 checkpoints (1/4 of all data for WMT Ro→En, 1/2 for WMT En→De and the full data for both IWSLT tasks), we select the best checkpoint based on the dev perplexity on which we report Bleu using SacreBLEU (Post, 2018) and Ter using TERCom (Snover et al., 2006) on an unseen test set. As systems with larger document-context see more frames in each epoch, we already stop training after 100 checkpoints for $k \geq 9$. We find that the converged document-level systems are able to predict the correct number of target sentences with almost perfect accuracy. We extract the last predicted sentence for each sample and then calculate Bleu and Ter on the sentence-level data.

When deploying LSH in the cross-attention, we found it crucial for training stability to first shuffle the key and query sequences as secondary criterion before sorting by hash classes. This helps during training in cases where the amount of queries/keys with the same hash class exceeds the window size.

5 Experimental Results

5.1 Sentence-level

We first evaluate the impact of our LSH attention approximation on different sentence-level tasks by replacing the self- and/or cross-attention components of the baseline with LSH attention. For LSH we use $n_{\text{hash}} = 4$ hash classes, chunks of size $\ell_{\text{chunk}} = 6$ and four hash rounds. This way the LSH attention could cover sentences of length $n_{\text{hash}} \cdot \ell_{\text{chunk}} = 4 \cdot 6 = 24$ entirely by partitioning it into n_{hash} hash classes of size ℓ_{chunk} (neglecting the forward/backward window and the multiple hash rounds), roughly matching the average sentence length. The results are shown in Table 1. We use LSH both while training and during inference.

Across all tasks the LSH-approximated attention performs worse than full attention. All systems but the WMT En→De system perform at most 1 % Bleu worse then the baseline when using three-fold LSH. For WMT En→De however, the performance degradation is much higher (2.8 % Bleu), suggesting that LSH does not work equally well across different tasks and language pairs.

In general, approximating the cross-attention is more damaging than LSH in the self-attention. In an extended analysis we find that the decoder self-attention seems least delicate and can be replaced by LSH attention with almost no decrease in translation capability.

5.2 Document-level

As the sequences in the sentence-level setting are relatively short, employing LSH does not save any memory but instead has a large computational overhead in comparison to the full dot-attention implemented with a few simple matrix multiplications. With increasing document-level context however, the quadratic memory usage of the full attention becomes a limiting factor which is overcome by

Attention method	Context	EN→DE WMT BLEU	EN→DE WMT TER	EN→DE IWSLT BLEU	EN→DE IWSLT TER	EN→IT IWSLT BLEU	EN→IT IWSLT TER	ContraPro Accuracy	Peak Mem. [GB]
Full att. (baseline)	0	32.1	56.7	23.3	68.4	32.8	53.6	42.4	5.5
	3	31.9	57.1	23.6	67.5	31.9	54.7	69.2	7.8
	9	30.8	58.6	OOM		OOM		OOM	9.6
	12	OOM		OOM		OOM		OOM	OOM
LSH self-attention	0	30.2	58.9	22.6	68.8	32.5	53.6	38.4	5.1
	3	30.8	58.5	23.0	68.3	32.5	53.8	50.1	5.7
	9	30.5	58.5	23.2	68.1	32.2	53.6	50.4	6.8
	12	29.8	59.2	23.6	67.6	31.8	53.9	46.3	7.0
LSH self- & cross-att.	0	29.0	60.2	22.5	68.7	31.5	54.7	40.3	9.6
	3	29.4	60.1	22.7	68.4	31.7	55.2	59.8	9.3
	9	27.3	64.8	22.1	69.9	31.4	54.5	51.7	9.0
	12	25.8	62.7	19.8	69.3	29.6	57.6	51.8	9.4

Table 2: Training LSH attention systems with different document-level context sizes. Besides BLEU and TER on the test set, we report the accuracy of the IWSLT EN→DE system on the ContraPro task (Müller et al., 2018). These three metrics are given in percentage. All systems use the same batch size during training, we exemplarily report the memory usage of the WMT EN→DE system. 'OOM' indicates that a system requires too much memory and cannot be trained.

using LSH attention.

We conduct a series of experiments with varying document-level context sizes, concatenating up to 13 sentences at once. For each context size, we train models with a) full attention everywhere, b) LSH in the encoder- and decoder-self-attention, and c) LSH in all three attention components.

In all LSH components we fix the LSH chunk size to $\ell_{\text{chunk}} = 10$, meaning each query can only attend to a constant number regardless of how many context sentences the system utilizes. We set the number of hash classes equal to the number of concatenated sentences (i.e. $k + 1$, but rounded to an even number which is required by Kitaev et al. (2020)'s hash function). The systems trained with LSH only in the self-attention use single rounded hashing as this is more memory-efficient. For the three-fold LSH systems we use four hash rounds.

Table 2 shows the results in BLEU and TER as well as the peak memory consumption on a GTX 1080 which fits about 10 GB. All systems are trained with a batch size of 3133 subwords. Additionally, we report the accuracy on the EN→DE contrastive pronoun resolution test set ContraPro (Müller et al., 2018). To resolve the pronouns properly context of up to three sentences is necessary.

With increasing context size, the full attention systems drastically use more memory as the computation of the full attention matrix scales quadratically in the sequence length. The memory usage of the LSH attention on the contrary only scales linearly in the sequence length and therefore is constant w.r.t. a fixed batch size. When the context size is too large, all full attention systems crash during training as a single training batch no longer fits into the 10 GB GPU memory. Replacing the self-attention with LSH is not only in absolute numbers more memory-efficient than the baseline but also scales much more softly in the document-level context size, making it possible to easily train a system with 12 sentences context where all full attention systems crash. Also, replacing the cross-attention with LSH finally means that the memory consumption remains constant w.r.t. the document-level context size, as it scales fully linearly in the number of tokens. Note however that because we use multi-round hashing here, it requires more memory than full attention when used on short sequences.

In terms of translation quality, we see similar results as in Table 1 when comparing the three different system architectures in the sentence-level setting: Employing LSH in the self-attention decreases BLEU by 0.3–0.9 % BLEU. Three-fold LSH performs 0.8 and 1.3 % BLEU worse than the baseline for the IWSLT EN→DE and EN→IT tasks respectively, but 3.1 % BLEU worse on WMT

Hash classes	Class size range	LSH inference Bleu	LSH inference Ter	Full inference Bleu	Full inference Ter	Full attention covered by LSH
1 (baseline)		35.7	51.4	35.7	51.4	100.0
2	49.7 – 50.3	35.6	51.6	35.4	51.6	64.5
4	24.1 – 25.7	35.2	51.9	35.1	51.9	42.4
8	11.0 – 13.4	34.6	52.2	34.6	52.2	29.5

Table 3: WMT Ro→En sentence-level systems trained with single-round LSH cross-attention and full self-attention. We set the chunk size large enough to always cover the entire sequence and vary the number of hash classes. For each system, we aggregate the hash class distribution of all queries/keys on the dev set and report the size of the smallest and largest class in percentage. We report Bleu and Ter on the dev set a) using LSH and b) using full attention not restricted to the same hash class. Further we average the sum of all attention weights of the full attention inference that would have been covered by LSH attention and report it in percent.

En→De as also observed before.

While increasing the document-level context slightly worsens Bleu and Ter for the full attention systems, the accuracy on the ContraPro test set increases significantly from 42.4 % to 69.2 % when including the three previous sentences as this task requires knowledge of the last few sentences.

Both the system with LSH in self-attention only and the three-fold LSH system perform equally well as the sentence-level systems even for high context sizes. Only for very large sizes ($k = 12$), performance starts to decrease.

6 Extended Analysis

6.1 Hash Quality

To evaluate the impact of approximating the full attention LSH we train systems with varying number of hash classes n_{hash} in the cross attention. As described in Section 3, queries may only attend to keys of the same hash class. The results for this are shown in Table 3. We explain the different columns in the following paragraphs.

In a first step we want to answer the question whether LSH attention actually makes use of different hash classes. Otherwise, if one hash class is over- or underrepresented, the chunk size used by the system will not be large enough to actually attend to all relevant keys. To verify this, we extract the distribution of all key and query vectors the system generated on the development set and count the sizes of all hash classes. We find that indeed the hash classes are approximately equally distributed, i.e. all have a size close to $\frac{1}{n_{\text{hash}}}$.

Increasing the number of hash classes decreases the number of keys each query can attend to. This also decreases translation performance in terms of Bleu and Ter, but only minorly: The system using 8 hash classes, i.e. only attending to one eighth of all keys per query, only performs 1.1 % Bleu worse than the baseline when also using LSH during inference.

The previous results all also use LSH during inference. Alternatively, we also experiment with full attention during inference after training the system with LSH. In this case, performance is almost equal to the LSH-restricted attention, even when using many hash classes. For each sentence pair, we extract the attention weights using full attention and sum over the key positions the LSH system attends to. This is the share of full attention covered by the LSH approximation, which however in the LSH system is renormalized to have a sum of 1 for each query. The average of this over all dev sentences and attention heads is shown in the last column of Table 3. Even though with increasing number of hash classes the share of covered attention decreases drastically, both LSH inference and full inference perform equally well in terms of Bleu and Ter. This indicates that LSH is able to focus on the most important positions.

6.2 Effective Window Size

The number of keys each query can attend to depends on a) the LSH chunk size, b) the number of attention heads used in parallel, and c) the number of hash rounds used in each attention head. Fixing the product of these three factors, which combination leads to the best translation performance?

As shown in Table 4, a larger chunk size or more attention heads do not improve performance. Using two hash rounds increases performance by 0.5 % Bleu. Different hash rounds allow the system to partition the key sequences w.r.t. different

Chunk size	Heads	Rounds	BLEU	TER
6	8	1	35.0	52.1
12	8	1	34.7	52.2
6	16	1	35.0	52.1
6	8	2	35.5	51.7
6	8	4	35.4	51.6

Table 4: WMT RO→EN sentence-level systems trained with LSH encoder self-attention, varying three parameters determining the how many keys each query may attend to. All systems with $\ell_{\text{chunk}} = 6$ use $n_{\text{hash}} = 4$ ($n_{\text{hash}} = 8$ for $\ell_{\text{chunk}} = 12$). We report BLEU and TER on the dev set in percentage.

aspects described by different hash functions. This effect is limited however, as four hash rounds perform equally well as just two.

6.3 Training Time and Memory

While LSH is more memory-efficient than full attention, it requires more operations to compute due to its increased complexity. For example, training for one checkpoint for the sentence-level WMT EN→DE system (Table 2) takes 49 min when using full-attention, 69 min when using single-round LSH in the self-attention, and 120 min when using three-fold LSH with four hash rounds. In particular, the time complexity of LSH scales linearly in the amount of hash rounds.

To still be able to train the full attention systems with large document-level context, a simple option is to reduce the batch size at the cost of a longer training time. With $k = 12$ sentences context, if we reduce the batch size to 2500 subwords, we can run the full attention system at a speed of 165 min / checkpoint. For this however note that we need to remove a few very long sequences no longer fitting into a single batch. In comparison, the self-attention system with a tuned batch size takes about the same time, 163 min / checkpoint.

7 Conclusion

We present a method to make the Transformer NMT architecture more memory-efficient when handling long input sequences. This is achieved by pre-selecting the most relevant candidates in self-attention and cross-attention using an LSH scheme that has been successfully applied for language modeling in previous work. We modify the existing LSH scheme to work in the NMT framework and conduct experiments on both sentence-level and document-level NMT tasks.

Our experiments show that the LSH attention scheme can be used for sentence-level NMT, although the approximation comes at the cost of slightly reduced translation quality. For document-level NMT we are able to include much bigger context sizes than what is possible with the baseline Transformer. However, more context does neither improve translation quality nor improve scores on targeted test suites.

In the future, we plan to use this approach for speech translation where long input sequences are a more pressing issue.

Acknowledgements

This project has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694537, project "SEQCLAS"). The work reflects only the authors' views and the European Research Council Executive Agency (ERCEA) is not responsible for any use that may be made of the information it contains.

References

Alexandr Andoni, Piotr Indyk, Thijs Laarhoven, Ilya Razenshteyn, and Ludwig Schmidt. 2015. Practical and optimal lsh for angular distance. *Advances in neural information processing systems*, 28.

Guangsheng Bao, Yue Zhang, Zhiyang Teng, Boxing Chen, and Weihua Luo. 2021. G-transformer for document-level machine translation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3442–3455.

Rachel Bawden, Rico Sennrich, Alexandra Birch, and Barry Haddow. 2018. Evaluating discourse phenomena in neural machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 1304–1313. Association for Computational Linguistics.

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

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *CoRR*, abs/1904.10509.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime G Carbonell, Quoc Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. 2020. Conformer: Convolution-augmented transformer for speech recognition. *Proc. Interspeech 2020*, pages 5036–5040.

Rohit Gupta, Laurent Besacier, Marc Dymetman, and Matthias Gallé. 2019. Character-based NMT with transformer. *CoRR*, abs/1911.04997.

Jingjing Huo, Christian Herold, Yingbo Gao, Leonard Dahlmann, Shahram Khadivi, and Hermann Ney. 2020. Diving deep into context-aware neural machine translation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 604–616.

Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney. 2019. Language modeling with deep transformers. *Proc. Interspeech 2019*, pages 3905–3909.

Sébastien Jean, Stanislas Lauly, Orhan Firat, and Kyunghyun Cho. 2017. Does neural machine translation benefit from larger context? *CoRR*, abs/1704.05135.

Marcin Junczys-Dowmunt. 2019. Microsoft translator at wmt 2019: Towards large-scale document-level neural machine translation. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 225–233.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The efficient transformer. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*. OpenReview.net.

Shaohui Kuang, Deyi Xiong, Weihua Luo, and Guodong Zhou. 2018. Modeling coherence for neural machine translation with dynamic and topic caches. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 596–606.

Taku Kudo. 2018. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 66–75. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

António Lopes, M Amin Farajian, Rachel Bawden, Michael Zhang, and André FT Martins. 2020. Document-level neural mt: A systematic comparison. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 225–234.

Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer. 2019. Transformers with convolutional context for ASR. *CoRR*, abs/1904.11660.

Makoto Morishita, Jun Suzuki, Tomoharu Iwata, and Masaaki Nagata. 2021. Context-aware neural machine translation with mini-batch embedding. In *Proceedings of the 16th conference of the European chapter of the association for computational linguistics: main volume*, pages 2513–2521.

Mathias Müller, Annette Rios, Elena Voita, and Rico Sennrich. 2018. A large-scale test set for the evaluation of context-aware pronoun translation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 61–72. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, David Grangier, and Michael Auli. 2018. Scaling neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 1–9, Brussels, Belgium. Association for Computational Linguistics.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. 2018. Image transformer. In *International Conference on Machine Learning*, pages 4055–4064. PMLR.

Loïc Paulevé, Hervé Jégou, and Laurent Amsaleg. 2010. Locality sensitive hashing: A comparison of hash function types and querying mechanisms. *Pattern recognition letters*, 31(11):1348–1358.

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.

Jiezhong Qiu, Hao Ma, Omer Levy, Wen-tau Yih, Sinong Wang, and Jie Tang. 2020. Blockwise self-attention for long document understanding. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2555–2565.

Jack W Rae, Anna Potapenko, Siddhant M Jayakumar, Chloe Hillier, and Timothy P Lillicrap. 2019. Compressive transformers for long-range sequence modelling. In *International Conference on Learning Representations*.

Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. 2020. Fixed encoder self-attention patterns in transformer-based machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 556–568.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

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 (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Xing Shi and Kevin Knight. 2017. Speeding up neural machine translation decoding by shrinking run-time vocabulary. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 574–579.

Matthew G. Snover, Bonnie J. Dorr, Richard M. Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers, AMTA 2006, Cambridge, Massachusetts, USA, August 8-12, 2006*, pages 223–231. Association for Machine Translation in the Americas.

Sainbayar Sukhbaatar, Édouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 331–335.

Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. 2020. Efficient transformers: A survey. *CoRR*, abs/2009.06732.

Jörg Tiedemann and Yves Scherrer. 2017. Neural machine translation with extended context. *DiscoMT 2017*, page 82.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420.

Ferhan Ture, Tamer Elsayed, and Jimmy Lin. 2011. No free lunch: Brute force vs. locality-sensitive hashing for cross-lingual pairwise similarity. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, page 943–952, New York, NY, USA. Association for Computing Machinery.

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

Laura Cross Vila, Carlos Escolano, José AR Fonollosa, and Marta R Costa-Jussa. 2018. End-to-end speech translation with the transformer. In *IberSPEECH*, pages 60–63.

Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *CoRR*, abs/2006.04768.

Weiqiu You, Simeng Sun, and Mohit Iyyer. 2020. Hard-coded gaussian attention for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7689–7700.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. 2020. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, pages 17283–17297. Curran Associates, Inc.

Albert Zeyer, Tamer Alkhouli, and Hermann Ney. 2018. RETURNN as a generic flexible neural toolkit with application to translation and speech recognition. In *Proceedings of ACL 2018, Melbourne, Australia, July 15-20, 2018, System Demonstrations*, pages 128–133. Association for Computational Linguistics.

Albert Zeyer, Parnia Bahar, Kazuki Irie, Ralf Schlüter, and Hermann Ney. 2019. A comparison of transformer and lstm encoder decoder models for asr. In *2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 8–15. IEEE.

Pei Zhang, Boxing Chen, Niyu Ge, and Kai Fan. 2020. Long-short term masking transformer: A simple but effective baseline for document-level neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1081–1087.

Kai Zhao, Hany Hassan, and Michael Auli. 2015. Learning translation models from monolingual continuous representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1527–1536.