Hard-Coded Gaussian Attention for Neural Machine Translation

Weiqiu You*, Simeng Sun*, Mohit Iyyer
College of Information and Computer Sciences
University of Massachusetts Amherst
{wyou,simengsun,miyyer}@cs.umass.edu

Abstract

Recent work has questioned the importance of the Transformer's multi-headed attention for achieving high translation quality. We push further in this direction by developing a "hard-coded" attention variant without *any* learned parameters. Surprisingly, replacing all learned self-attention heads in the encoder and decoder with fixed, input-agnostic Gaussian distributions minimally impacts BLEU scores across four different language pairs. However, additionally hard-coding cross attention (which connects the decoder to the encoder) significantly lowers BLEU, suggesting that it is more important than self-attention. Much of this BLEU drop can be recovered by adding just a *single* learned cross attention head to an otherwise hard-coded Transformer. Taken as a whole, our results offer insight into which components of the Transformer are actually important, which we hope will guide future work into the development of simpler and more efficient attention-based models.

1 Introduction

The Transformer (Vaswani et al., 2017) has become the architecture of choice for neural machine translation. Instead of using recurrence to contextualize source and target token representations, Transformers rely on multi-headed attention mechanisms (MHA), which speed up training by enabling parallelization across timesteps. Recent work has called into question how much MHA contributes to translation quality: for example, a significant fraction of attention heads in a pretrained Transformer can be pruned without appreciable loss in BLEU (Voita et al., 2019; Michel et al., 2019), and self-attention can be replaced by less expensive modules such as convolutions (Yang et al., 2018; Wu et al., 2019).

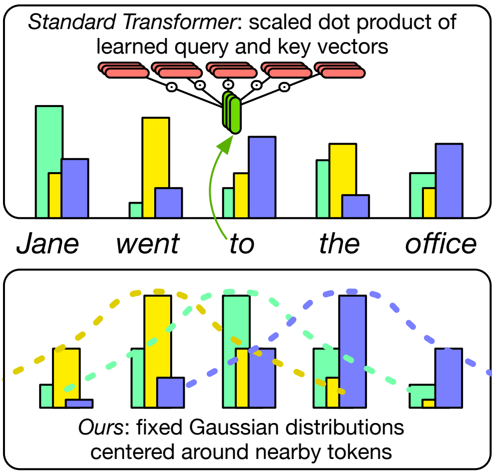
Standard Transformer: scaled dot product of learned query and key vectors
Jane
went
to
the
office
Ours: fixed Gaussian distributions centered around nearby tokens

Figure 1: Three heads of learned self-attention (top) as well as our hard-coded attention (bottom) given the query word "to". In our variant, each attention head is a Gaussian distribution centered around a different token within a local window.

In this paper, we take this direction to an extreme by developing a variant of MHA without *any* learned parameters (Section 3). Concretely, we replace each attention head with a "hard-coded" version, which is simply a standard normal distribution centered around a particular position in the sequence (Figure 1).[1] When we replace all encoder and decoder self-attention mechanisms with our hard-coded variant, we achieve almost identical BLEU scores to the baseline Transformer for four different language pairs (Section 4).[2]

These experiments maintain fully learned MHA *cross attention*, which allows the decoder to condition its token representations on the encoder's outputs. We next attempt to additionally replace cross attention with a hard-coded version, which results in substantial drops of 5-10 BLEU. Motivated to find the minimal number of learned attention

* Authors contributed equally.

[1] In Figure 1, the hard-coded head distribution centered on the word "to" (shown in green) is $[0.054, 0.24, 0.40, 0.24, 0.054]$.

[2] Our code is available at https://github.com/fallcat/stupidNMT

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 7689–7700
July 5 - 10, 2020. ©2020 Association for Computational Linguistics

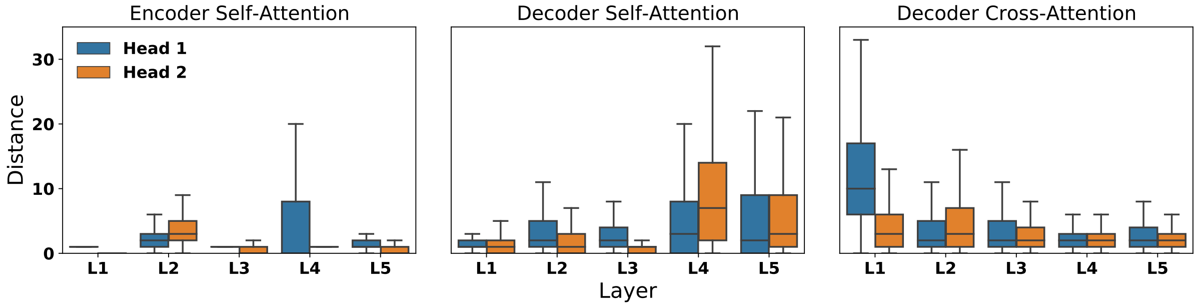

Figure 2: Most learned attention heads for a Transformer trained on IWSLT16 En-De focus on a local window around the query position. The x-axis plots each head of each layer, while the y-axis refers to the distance between the query position and the argmax of the attention head distribution (averaged across the entire dataset).

parameters needed to make up this deficit, we explore configurations with only *one* learned cross attention head in total, which performs just slightly worse (1-3 BLEU) than the baseline.

By replacing MHA with hard-coded attention, we improve memory efficiency (26.4% more tokens per batch) and decoding speed (30.2% increase in sentences decoded per second) without significantly lowering BLEU, although these efficiency improvements are capped by other more computationally-expensive components of the model (Section 5). We also perform analysis experiments (Section 6.2) on linguistic properties (e.g., long-distance subject-verb agreement) that MHA is able to better model than hard-coded attention. Finally, we develop further variants of hard-coded attention in Section 6.3, including a version without any attention weights at all.

Our hard-coded Transformer configurations have intuitively severe limitations: attention in a particular layer is highly concentrated on a local window in which fixed weights determine a token's importance. Nevertheless, the strong performance of these limited models indicates that the flexibility enabled by fully-learned MHA is not as crucial as commonly believed: perhaps attention is not *all* you need. We hope our work will spur further development of simpler, more efficient models for neural machine translation.

2 Background

In this section, we first briefly review the Transformer architecture of Vaswani et al. (2017) with a focus on its multi-headed attention. Then, we provide an analysis of the learned attention head distributions of a trained Transformer model, which motivates the ideas discussed afterwards.

2.1 Multi-headed Transformer attention

The Transformer is an encoder-decoder model formed by stacking layers of attention blocks. Each encoder block contains a self-attention layer followed by layer normalization, a residual connection, and a feed-forward layer. Decoder blocks are identical to those of the encoder except they also include a cross attention layer, which connects the encoder's representations to the decoder.

To compute a single head of self-attention given a sequence of token representations $\boldsymbol{t}_{1\ldots n}$, we first project these representations to queries $\boldsymbol{q}_{1\ldots n}$, keys $\boldsymbol{k}_{1\ldots n}$, and values $\boldsymbol{v}_{1\ldots n}$ using three different linear projections. Then, to compute the self-attention distribution at a particular position i in the sequence, we take the scaled dot product between the query vector $\boldsymbol{q}_i$ and all of the key vectors (represented by matrix $\mathbf{K}$). We then use this distribution to compute a weighted average of the values ($\mathbf{V}$):

$$\text{Attn}(\boldsymbol{q}_i, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\boldsymbol{q}_i\mathbf{K}^\top}{\sqrt{d_k}})\mathbf{V} \qquad (1)$$

where d_k is the dimensionality of the key vector.

For MHA, we use different projection matrices to obtain the query, key, and value representations for each head. The key difference between self-attention and cross attention is that the queries and keys come from different sources: specifically, the keys are computed by passing the encoder's final layer token representations through a linear projection. To summarize, MHA is used in three different components of the Transformer: encoder self-attention, decoder self-attention, and cross attention.

2.2 Learned heads mostly focus on local windows

The intuition behind MHA is that each head can focus on a different type of information (e.g., syntactic or semantic patterns). While some heads have been shown to possess interpretable patterns (Voita et al., 2019; Correia et al., 2019), other work has cautioned against using attention patterns to explain a model's behavior (Jain and Wallace, 2019). In our analysis, we specifically examine the behavior of a head with respect to the current query token's position in the sequence. We train a baseline Transformer model (five layers, two heads per layer) on the IWSLT 2016 En→De dataset, and compute aggregated statistics on its learned heads.

Figure 2 shows that outside of a few layers, most of the model's heads focus their attention (i.e., the argmax of the attention distribution) on a local neighborhood around the current sequence position. For example, both self-attention heads in the first layer of the encoder tend to focus on just a one to two token window around the current position. The decoder self-attention and cross attention heads show higher variability, but most of their heads are still on average focused on local information. These results beg the question of whether replacing self-attention with "hard-coded" patterns that focus on local windows will significantly affect translation quality.

3 Hard-coded Gaussian attention

While learned attention enables model flexibility (e.g., a head can "look" far away from the current position if it needs to), it is unclear from the above analysis how crucial this flexibility is. To examine this question, we replace the attention distribution computation in Equation 1 (i.e., scaled dot product of queries and keys) with a fixed Gaussian distribution.[3] In doing so, we remove *all* learned parameters from the attention computation: the mean of the Gaussian is determined by the position i of the current query token, and the standard deviation is always set to 1.[4] As Transformers contain both self-attention and cross attention, the rest of this section details how we replace both of these components with simplified versions. We will refer to experimental results on the relatively small IWSLT16 English-German dataset throughout this section to contextualize the impact of the various design decisions we describe. Section 4 contains a more fleshed out experimental section with many more datasets and language pairs.

3.1 Hard-coded self-attention

In self-attention, the queries and keys are derived from the same token representations and as such have the same length n. The baseline Transformer (BASE) computes the self-attention distribution at position i by taking the dot product between the query representation $\boldsymbol{q}_i$ and all of the key vectors $\boldsymbol{k}_{1\ldots n}$. We instead use a fixed Gaussian distribution centered around position $i-1$ (token to the left), i (the query token), or $i+1$ (token to the right). More formally, we replace Equation 1 with

$$\text{Attn}(i, \mathbf{V}) = \mathcal{N}(f(i), \sigma^2)\mathbf{V}. \tag{2}$$

The mean of the Gaussian $f(i)$ and its standard deviation σ^2 are both hyperparameters; for all of our experiments, we set σ to 1 and $f(i)$ to either $i-1$, i or $i+1$, depending on the head configuration.[5] Note that this definition is completely agnostic to the input representation: the distributions remain the same regardless of what sentence is fed in or what layer we are computing the attention at. Additionally, our formulation removes the query and key projections from the attention computation; the Gaussians are used to compute a weighted average of the value vectors.[6]

Instead of learning different query and key projection matrices to define different heads, we simply design head distributions with different means. Figure 1 shows an example of our hard-coded self-attention for a simple sentence. We iterate over different configurations of distribution means $f(i)$ on the IWSLT16 En-De dataset, while keeping the cross attention learned.[7] Our best validation result with hard-coded self-attention (HC-SA) replaces encoder self-attention with distributions centered around $i-1$ and $i+1$ and decoder self-attention with distributions centered around $i-1$ and i. This

[3] Yang et al. (2018) implement a similar idea, except the mean and standard deviation of their Gaussians are learned with separate neural modules.

[4] Preliminary experiments with other standard deviation values did not yield significant differences, so we do not vary the standard deviation for any experiments in this paper.

[5] The Gaussian distribution is cut off on the borders of the sentence and is not renormalized to sum to one.

[6] Preliminary models that additionally remove the value projections performed slightly worse when we hard-coded cross attention, so we omit them from the paper.

[7] See Appendix for a table describing the effects of varying $f(i)$ on IWSLT16 En-De BLEU score. We find in general that hard-coded heads within each layer should focus on different tokens within the local window for optimal performance.

model achieves slightly *higher* BLEU than the baseline Transformer (**30.3** vs **30.0** BLEU).

3.2 Alternatives to cross attention

We turn next to cross attention, which on its face seems more difficult to replace with hard-coded distributions. Unlike self-attention, the queries and keys in cross attention are not derived from the same token representations; rather, the queries come from the decoder while the keys come from the encoder. Since the number of queries can now be different from the number of keys, setting the distribution means by position is less trivial than it is for self-attention. Here, we describe two methods to simplify cross attention, starting with a fully hard-coded approach and moving onto a minimal learned configuration.

Hard-coded cross attention: We begin with a simple solution to the problem of queries and keys having variable lengths. Given a training dataset, we compute the length ratio γ by dividing the average source sentence length by the average target sentence length. Then, to define a hard-coded cross attention distribution for target position i, we center the Gaussian on positions $\lfloor \gamma i - 1 \rfloor$, $\lfloor \gamma i \rfloor$, and $\lfloor \gamma i + 1 \rfloor$ of the source sentence. When we implement this version of hard-coded cross attention and also hard-code the encoder and decoder self-attention as described previously (HC-ALL), our BLEU score on IWSLT16 En-De drops from **30.3** to **21.1**. Clearly, cross attention is more important for maintaining translation quality than self-attention. Michel et al. (2019) notice a similar phenomenon when pruning heads from a pretrained Transformer: removing certain cross attention heads can substantially lower BLEU.

Learning a single cross attention head: Prior to the advent of the Transformer, many neural machine translation architectures relied on just a single cross attention "head" (Bahdanau et al., 2015). The Transformer has many heads at many layers, but how many of these are actually necessary? Here, we depart from the parameter-free approach by instead removing cross attention at all but the final layer of the decoder, where we include only a single learned head (SH-X). Note that this is the only learned head in the entire model, as both the encoder and decoder self-attention is hard-coded. On IWSLT16 En-De, our BLEU score improves from **21.1** to **28.1**, less than 2 BLEU under the BASE Transformer.

	Train	Test	Len SRC	Len TGT
IWSLT16 En-De	196,884	993	28.5	29.6
IWSLT17 En-Ja	223,108	1,452	22.9	16.0
WMT16 En-Ro	612,422	1,999	27.4	28.3
WMT14 En-De	4,500,966	3,003	28.5	29.6
WMT14 En-Fr	10,493,816	3,003	26.0	28.8

Table 1: Statistics of the datasets used. The last two columns show the average number of tokens for source and target sentences, respectively.

4 Large-scale Experiments

The previous section developed hard-coded configurations and presented results on the relatively small IWSLT16 En-De dataset. Here, we expand our experiments to include a variety of different datasets, language pairs, and model sizes. For all hard-coded head configurations, we use the optimal IWSLT16 En-De setting detailed in Section 3.1 and perform no additional tuning on the other datasets. This configuration nevertheless proves robust, as we observe similar trends with our hard-coded Transformers across all of datasets.[8]

4.1 Datasets

We experiment with four language pairs, English↔{German, Romanian, French, Japanese} to show the consistency of our proposed attention variants. For the En-De pair, we use both the small IWSLT 2016[9] and the larger WMT 2014 datasets. For all datasets except WMT14 En→De and WMT14 En→Fr,[10] we run experiments in both directions. For English-Japanese, we train and evaluate on IWSLT 2017 En↔Ja TED talk dataset. More dataset statistics are shown in Table 1.

4.2 Architectures

Our BASE model is the original Transformer from Vaswani et al. (2017), reimplemented in PyTorch (Paszke et al., 2019) by Akoury et al. (2019).[11] To implement hard-coded attention, we only modify the attention functions in this codebase and keep everything else the same. For the two small IWSLT datasets, we follow prior work

[8] Code and scripts to reproduce our experimental results to be released after blind review.

[9] We report BLEU on the IWSLT16 En-De dev set following previous work (Gu et al., 2018; Lee et al., 2018; Akoury et al., 2019). For other datasets, we report test BLEU.

[10] As the full WMT14 En→Fr is too large for us to feasibly train on, we instead follow Akoury et al. (2019) and train on just the Europarl / Common Crawl subset, while evaluating using the full dev/test sets.

[11] `https://github.com/dojoteef/synst`

	BASE	HC-SA	HC-ALL	SH-X
IWSLT16 En-De	30.0	30.3	21.1	28.2
IWSLT16 De-En	34.4	34.8	25.7	33.3
IWSLT17 En-Ja	20.9	20.7	10.6	18.5
IWSLT17 Ja-En	11.6	10.9	6.1	10.1
WMT16 En-Ro	33.0	32.9	25.5	30.4
WMT16 Ro-En	33.1	32.8	26.2	31.7
WMT14 En-De	26.8	26.3	21.7	23.5
WMT14 En-Fr	40.3	39.1	35.6	37.1

Table 2: Comparison of the discussed Transformer variants on six smaller datasets (top)[14] and two larger datasets (bottom). Hard-coded self-attention (HC-SA) achieves almost identical BLEU scores to BASE across all datasets, while a model with only one cross attention head (SH-X) performs slightly worse.

by using a small Transformer architecture with embedding size 288, hidden size 507, four heads,[12] five layers, and a learning rate 3e-4 with a linear scheduler. For the larger datasets, we use the standard Tranformer base model, with embedding size 512, hidden size 2048, eight heads, six layers, and a warmup scheduler with 4,000 warmup steps. For all experiments, we report BLEU scores using SacreBLEU (Post, 2018) to be able to compare with other work.[13]

4.3 Summary of results

Broadly, the trends we observed on IWSLT16 En-De in the previous section are consistent for all of the datasets and language pairs. Our findings are summarized as follows:

- A Transformer with hard-coded self-attention in the encoder and decoder and learned cross attention (HC-SA) achieves almost equal BLEU scores to the BASE Transformer.
- Hard-coding both cross attention and self-attention (HC-ALL) considerably drops BLEU compared to BASE, suggesting cross attention is more important for translation quality.
- A configuration with hard-coded self-attention and a single learned cross attention head in the final decoder layer (SH-X) consistently performs 1-3 BLEU worse than BASE.

These results motivate a number of interesting analysis experiments (e.g., what kinds of phenomena is MHA better at handling than hard-coded attention), which we describe in Section 6. The strong performance of our highly-simplified models also suggests that we may be able to obtain memory or decoding speed improvements, which we investigate in the next section.

5 Bigger Batches & Decoding Speedups

We have thus far motivated our work as an exploration of which components of the Transformer are necessary to obtain high translation quality. Our results demonstrate that encoder and decoder self-attention can be replaced with hard-coded attention distributions without loss in BLEU, and that MHA brings minor improvements over single-headed cross attention. In this section, we measure efficiency improvements in terms of batch size increases and decoding speedup.

Experimental setup: We run experiments on WMT16 En-Ro with the larger architecture to support our conclusions.[15] For each model variant discussed below, we present its memory efficiency as the maximum number of tokens per batch allowed during training on a single GeForce RTX 2080 Ti. Additionally, we provide inference speed as the number of sentences per second each model can decode on a 2080 Ti, reporting the average of five runs with a batch size of 256.

Hard-coding self-attention yields small efficiency gains: Table 7 summarizes our profiling experiments. Hard-coding self-attention and preserving learned cross attention allows us to fit 17% more tokens into a single batch, while also providing a 6% decoding speedup compared to BASE on the larger architecture used for WMT16 En-Ro. The improvements in both speed and memory usage are admittedly limited, which motivates us to measure the maximum efficiency gain if we only modify self-attention (i.e., preserving learned cross attention). We run a set of upper bound experiments where we entirely remove self-attention in the encoder and decoder. The resulting encoder

[12] For hard-coded configurations, we duplicate heads to fit this architecture (e.g., we have two heads per layer in the encoder with means of $i + 1$ and $i - 1$).

[13] SacreBLEU signature: BLEU+case.mixed+lang.LANG +numrefs.1+smooth.exp+test.TEST+tok.intl+version.1.2.11, with LANG $\in$ {en-de, de-en, en-fr} and TEST $\in$ {wmt14/full, iwslt2017/tst2013}. For WMT16 En-Ro and IWSLT17 En-Ja, we follow previous work for preprocessing (Sennrich et al., 2016), encoding the latter with a 32K sentencepiece vocabulary (`https://github.com/google/sentencepiece`) and measuring the de-tokenized BLEU with SacreBLEU.

[15] Experiments with the smaller IWSLT16 En-De model are described in the Appendix.

Model	BLEU	sent/sec	tokens/batch
BASE	33.0	26.8	9.2K
HC-SA	32.9	28.4	10.8K
SH-X	30.3	34.9	11.7K
BASE/-SA	27.0	30.1	11.8K
SH-X/-SA	15.0	37.6	13.3K

Table 3: Decoding speedup (in terms of sentences per second) and memory improvements (max tokens per batch) on WMT16 En-Ro for a variety of models. The last two rows refer to BASE and SH-X configurations whose self-attention is completely removed.

thus just becomes a stack of feed-forward layers on top of the initial subword embeddings. Somewhat surprisingly, the resulting model still achieves a fairly decent BLEU of **27.0** compared to the BASE model's **33.0**. As for the efficiency gains, we can fit 27% more tokens into a single batch, and decoding speed improves by 12.3% over BASE. This relatively low upper bound for HC-SA shows that simply hard-coding self-attention does not guarantee significant speedup. Previous work that simplifies attention (Wu et al., 2019; Michel et al., 2019) also report efficiency improvements of similar low magnitudes.

Single-headed cross attention speeds up decoding: Despite removing learned self-attention from both the encoder and decoder, we did not observe huge efficiency or speed gains. However, reducing the source attention to just a single head results in more significant improvements. By only keeping single-headed cross attention in the last layer, we are able to achieve 30.2% speed up and fit in 26.4% more tokens to the memory compared to BASE . Compared to HC-SA, SH-X obtains a 22.9% speedup and 8.0% bigger batch size.

From our profiling experiments, most of the speed and memory considerations of the Transformer are associated with the large feed-forward layers that we do not modify in any of our experiments, which caps the efficiency gains from modifying the attention implementation. While we did not show huge efficiency improvements on modern GPUs, it remains possible that (1) a more tailored implementation could leverage the model simplifications we have made, and (2) that these differences are larger on other hardware (e.g., CPUs). We leave these questions for future work.

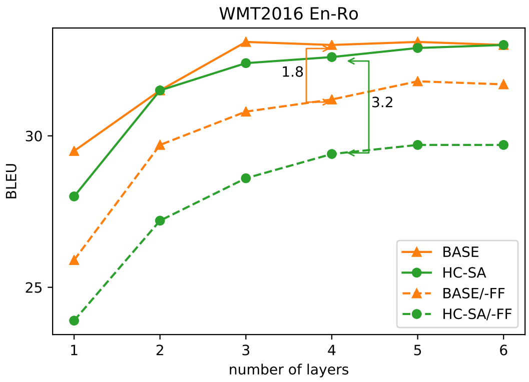

Figure 3: BLEU performance on WMT16 En-Ro before and after removing all feed-forward layers from the models. BASE and HC-SA achieve almost identical BLEU scores, but HC-SA relies more on the feed-forward layers than the vanilla Transformer. As shown on the plot, with a four layer encoder and decoder, the BLEU gap between BASE-FF and BASE is 1.8, while the gap between HC-SA and HC-SA-FF is 3.2.

6 Analysis

Taken as a whole, our experimental results suggest that many of the components in the Transformer can be replaced by highly-simplified versions without adversely affecting translation quality. In this section, we explain how hard-coded self-attention does not degrade translation quality (Section 6.1), perform a detailed analysis of the behavior of our various models by comparing the types of errors made by learned versus hard-coded attention (Section 6.2), and also examine different attention configurations that naturally follow from our experiments (Section 6.3).

6.1 Why does hard-coded self-attention work so well?

Given the good performance of HC-SA on multiple datasets, it is natural to ask why hard-coding self-attention does not deteriorate translation quality. We conjecture that feed-forward (FF) layers play a more important role in HC-SA than in BASE by compensating for the loss of learned dynamic self-attention. To test this hypothesis, we conduct an analysis experiment in which we train four model configurations while varying the number of layers: BASE, BASE without feed-forward layers (BASE/-FF), HC-SA and HC-SAwithout feed-forward layers (HC-SA/-FF). As shown in Figure 3, BASE and HC-SA have similar performance and both -FF models have consistently lower BLEU scores. However, HC-SA without FF layers performs much worse

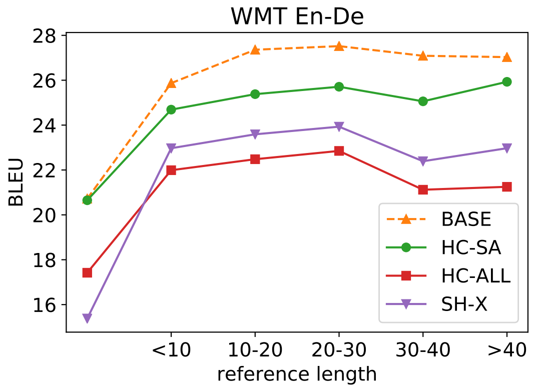

Figure 4: BLEU difference vs. BASE as a function of reference length on the WMT14 En-De test set. When cross attention is hard-coded (HC-ALL), the BLEU gap worsens as reference length increases.

compared to its BASE counterpart. This result confirms our hypothesis that FF layers are more important in HC-SA and capable of recovering the potential performance degradation brought by hard-coded self-attention. Taking a step back to hard-coding cross attention, the failure of hard-coding cross attention might be because the feed-forward layers of the decoder are not powerful enough to compensate for modeling both hard-coded decoder self-attention and cross attention.

6.2 Error analysis of hard-coded models

Is learned attention more important for longer sentences? Since hard-coded attention is much less flexible than learned attention and can struggle to encode global information, we are curious to see if its performance declines as a function of sentence length. To measure this, we categorize the WMT14 En-De test set into five bins by reference length and plot the decrease in BLEU between BASE and our hard-coded configurations for each bin. Somewhat surprisingly, Figure 4 shows that the BLEU gap between BASE and HC-SA seems to be roughly constant across all bins.[16] However, the fully hard-coded HC-ALL model clearly deteriorates as reference length increases.

Does hard-coding attention produce any systematic linguistic errors? For a more fine-grained analysis, we run experiments on LingEval97 (Sennrich, 2017), an English→German dataset consisting of contrastive translation pairs. This dataset measures targeted errors on thirteen different linguistic phenomena such as agreement and adequacy. BASE and HC-SA perform[17] very similarly across all error types (Table 4), which is perhaps unsurprising given that their BLEU scores are almost identical. Interestingly, the category with the highest decrease from BASE for both HC-SA and HC-ALL is *deleted negations*;[18] HC-ALL is 11% less accurate (absolute) at detecting these substitutions than BASE (94% vs 83%). On the other hand, both HC-SA and HC-ALL are actually better than BASE at detecting *inserted negations*, with HC-ALL achieving a robust 98.7% accuracy. We leave further exploration of this phenomenon to future work. Finally, we observe that for the subject-verb agreement category, the discrepancy between BASE and the hard-coded models increases as the distance between subject-verb increases (Figure 5). This result confirms that self-attention is important for modeling some long-distance phenomena, and that cross attention may be even more crucial.

Error type	BASE	HC-SA	HC-ALL
np-agreement	**54.2**	53.5	53.5
subj-verb-agreement	**87.5**	85.8	82.5
subj-adequacy	**87.3**	85.0	80.3
polarity-particle-nicht-del	**94.0**	91.4	83.2
polarity-particle-kein-del	**91.4**	88.3	79.9
polarity-affix-del	**91.6**	90.8	83.1
polarity-particle-nicht-ins	**92.6**	92.5	89.8
polarity-particle-kein-ins	94.8	96.7	**98.7**
polarity-affix-ins	**91.9**	90.6	84.3
auxiliary	**89.1**	87.5	85.6
verb-particle	**74.7**	72.7	70.2
compound	88.1	**89.5**	80.5
transliteration	97.6	**97.9**	93.4

Table 4: Accuracy for each error type in the LingEval97 contrastive set. Hard-coding self-attention results in slightly lower accuracy for most error types, while more significant degradation is observed when hard-coding self and cross attention. We refer readers to Sennrich (2017) for descriptions of each error type.

Do hard-coded models struggle when learned self-attention focuses on non-local information? Since hard-coded models concentrate most of the attention probability mass on local tokens, they might underperform on sentences for which the

[16] We note that gradients will flow across long distances if the number of layers is large enough, since the effective window size increases with multiple layers (van den Oord et al., 2016; Kalchbrenner et al., 2016).

[17] Accuracy is computed by counting how many references have lower token-level cross entropy loss than their contrastive counterparts.

[18] Specifically, when *ein* is replaced with negation *kein*.

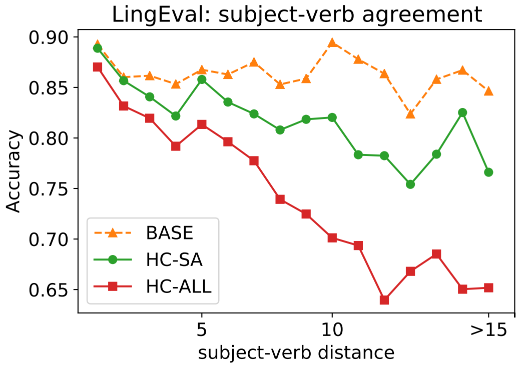

Figure 5: Hard-coded models become increasingly worse than BASE at subject-verb agreement as the dependency grows longer.

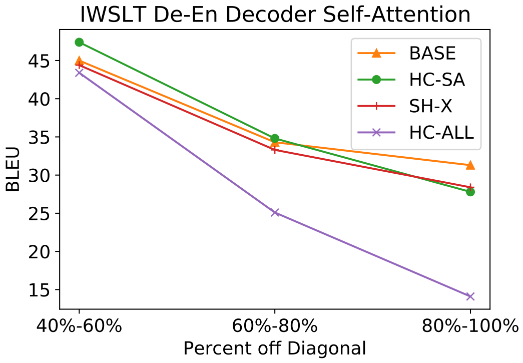

Figure 6: Hard-coded attention performs better for sentences with low off-diagonality (i.e., sentences for which the BASE model's learned attention focuses close to the query position for most of their tokens).

learned heads of the BASE model focus on tokens far from the current query position. We define a token to be "off-diagonal" when the maximum probability of that token's attention is at least two steps away from query position. A sentence's "off-diagonality" is then the proportion of off-diagonal tokens within the sentence. We bin the sentences in IWSLT En-De development set by their off-diagonality and analyze the translation quality of our models on these different bins. Figure 6 shows that for decoder self attention, the BLEU gap between HC-ALL and BASE increases as off-diagonality increases, while the gap between BASE and SH-X remains relatively constant across all bins. HC-SA even outperforms BASE for sentences with fewer off-diagonal tokens.

6.3 Other hard-coded model configurations

Is it important for the Gaussian to span the entire sequence? One natural question about the hard-coded attention strategy described in Section 3 is whether it is necessary to assign some probability to all tokens in the sequence. After all, the probabilities outside a local window become very marginal, so perhaps it is unnecessary to preserve them. We take inspiration from Wu et al. (2019), who demonstrate that lightweight convolutions can replace self-attention in the Transformer without harming BLEU, by recasting our hard-coded attention as a convolution with a hard-coded 1-D kernel. While this decision limits the Gaussian distribution to span over just tokens within a fixed window around the query token, it does not appreciably impact BLEU (second column of Table 5). We set the window size to 3 in all experiments, so the kernel weights become $[0.242, 0.399, 0.242]$.

	Original	Conv (window=3)	Indexing
En-De	30.3	30.1	29.8
En-Ro	32.4	32.3	31.4

Table 5: Comparison of three implementations of HC-SA. Truncating the distribution to a three token span has little impact, while removing the weights altogether slightly lowers BLEU.

Are any attention weights necessary at all? The previous setting with constrained window size suggests another follow-up: is it necessary to have any attention weights within this local window at all? A highly-efficient alternative is to have each head simply select a single value vector associated with a token in the window. Here, our implementation requires no explicit multiplication with a weight vector, as we can compute each head's representation by simply indexing into the value vectors. Mathematically, this is equivalent to convolving with a binary kernel (e.g., convolution with $[1, 0, 0]$ is equivalent to indexing the left token representation). The third column of Table 5 shows that this indexing approach results in less than 1 BLEU drop across two datasets, which offers an interesting avenue for future efficiency improvements.

Where should we add additional cross attention heads? Our experiments with cross attention so far have been limited to learning just a single head, as we have mainly been interested in minimal configurations. If we have a larger budget of cross attention heads, where should we put them? Is it better to have more cross attention heads in the last layer in the decoder (and no heads anywhere else), or to distribute them across multiple layers

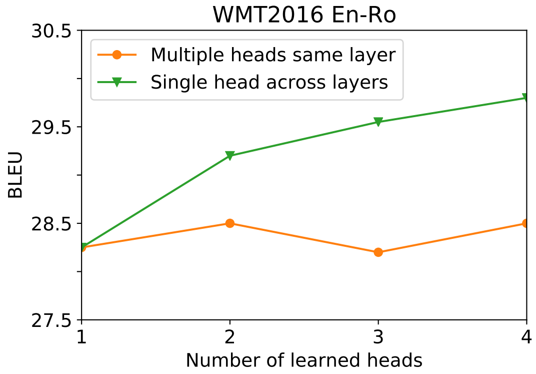

Figure 7: Adding more cross attention heads in the same layer helps less than adding individual heads across different layers.

of the decoder? Experiments on the WMT16 En-Ro dataset[19] (Figure 7) indicate that distributing learned heads over multiple layers leads to significantly better BLEU than adding all of them to the same layer.

7 Related Work

Attention mechanisms were first introduced to augment vanilla recurrent models (Kalchbrenner and Blunsom, 2013; Sutskever et al., 2014; Bahdanau et al., 2015; Luong et al., 2015; Chorowski et al., 2015; Wu et al., 2016; Miceli Barone et al., 2017) but have become the featured component of the state-of-the-art Transformer architecture (Vaswani et al., 2017) for NMT. We review recent research that focuses on analysing and improving multi-headed attention, and draw connections to our work.

The intuitive advantage of MHA is that different heads can focus on different types of information, all of which will eventually be helpful for translation. Voita et al. (2019) find that some heads focus on adjacent tokens to the query (mirroring our analysis in Section 2), while others focus on specific dependency relations or rare tokens. Correia et al. (2019) discover that some heads are sensitive to subword clusters or interrogative words. Tang et al. (2018) shows that the number of MHA heads affects the ability to model long-range dependencies. Michel et al. (2019) show that pruning many heads from a pretrained model does not significantly impact BLEU scores. Similarly, Voita et al. (2019) prune many encoder self-attention heads without degrading BLEU, while Tang et al. (2019) further simplify the Transformer by removing the entire encoder for a drop of three BLEU points. In contrast to existing literature on model pruning, we *train* our models without learned attention heads instead of removing them post-hoc.

There have been many efforts to modify MHA in Transformers. One such direction is to inject linguistic knowledge through auxiliary supervised tasks (Garg et al., 2019; Pham et al., 2019). Other work focuses on improving inference speed: Yang et al. (2018) replace decoder self-attention with a simple average attention network, assigning equal weights to target-side previous tokens.[20] Wu et al. (2019) also speed up decoding by replacing self-attention with convolutions that have time-step dependent kernels; we further simplify this work with our fixed convolutional kernels in Section 6. Cui et al. (2019) also explore fixed attention while retaining some learned parameters, and Vashishth et al. (2019) show that using uniform or random attention deteriorates performances on paired sentences tasks including machine translation. Other work has also explored modeling locality (Shaw et al., 2018; Yang et al., 2018).

8 Conclusion

In this paper, we present "hard-coded" Gaussian attention, which while lacking any learned parameters can rival multi-headed attention for neural machine translation. Our experiments suggest that encoder and decoder self-attention is not crucial for translation quality compared to cross attention. We further find that a model with hard-coded self-attention and just a single cross attention head performs slightly worse than a baseline Transformer. Our work provides a foundation for future work into simpler and more computationally efficient neural machine translation.

Acknowledgments

We thank the anonymous reviewers for their thoughtful comments, Omer Levy for general guidance and for suggesting some of our efficiency experiments, the UMass NLP group for helpful comments on earlier drafts, Nader Akoury for assisting with modifications to his Transformer codebase, and Kalpesh Krishna for advice on the structure of the paper.

[19] We used the smaller IWSLT En-De architecture for this experiment.

[20] In preliminary experiments, we find that using uniform distributions for encoder self-attention decreases BLEU. This result is similar to the indexing implementation we describe in Section 6.3.

References

Nader Akoury, Kalpesh Krishna, and Mohit Iyyer. 2019. Syntactically supervised transformers for faster neural machine translation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 1269–1281, Florence, Italy. Association for Computational Linguistics.

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.

Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio. 2015. Attention-based models for speech recognition. In C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, editors, Advances in Neural Information Processing Systems 28, pages 577–585. Curran Associates, Inc.

Gonçalo M. Correia, Vlad Niculae, and André F. T. Martins. 2019. Adaptively sparse transformers. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2174–2184, Hong Kong, China. Association for Computational Linguistics.

Hongyi Cui, Shohei Iida, Po-Hsuan Hung, Takehito Utsuro, and Masaaki Nagata. 2019. Mixed multi-head self-attention for neural machine translation. In Proceedings of the 3rd Workshop on Neural Generation and Translation, pages 206–214, Hong Kong. Association for Computational Linguistics.

Sarthak Garg, Stephan Peitz, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Jointly learning to align and translate with transformer models. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4452–4461, Hong Kong, China. Association for Computational Linguistics.

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

Sarthak Jain and Byron C. Wallace. 2019. Attention is not Explanation. 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 3543–3556, Minneapolis, Minnesota. Association for Computational Linguistics.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, pages 1700–1709, Seattle, Washington, USA. Association for Computational Linguistics.

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. arXiv preprint arXiv:1610.10099.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 1173–1182, Brussels, Belgium. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Antonio Valerio Miceli Barone, Jindřich Helcl, Rico Sennrich, Barry Haddow, and Alexandra Birch. 2017. Deep architectures for neural machine translation. In Proceedings of the Second Conference on Machine Translation, pages 99–107, Copenhagen, Denmark. Association for Computational Linguistics.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 14014–14024. Curran Associates, Inc.

Aäron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew W. Senior, and Koray Kavukcuoglu. 2016. Wavenet: A generative model for raw audio. In The 9th ISCA Speech Synthesis Workshop, Sunnyvale, CA, USA, 13-15 September 2016, page 125. ISCA.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, Advances in Neural Information Processing Systems 32, pages 8024–8035. Curran Associates, Inc.

Thuong Pham, Dominik Macháček, and Ondřej Bojar. 2019. Promoting the knowledge of source syntax in transformer nmt is not needed. Computación y Sistemas, 23.

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

Rico Sennrich. 2017. How grammatical is character-level neural machine translation? assessing MT quality with contrastive translation pairs. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers, pages 376–382, Valencia, Spain. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. 2018. Self-attention with relative position representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers), pages 464–468, New Orleans, Louisiana. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 27, pages 3104–3112. Curran Associates, Inc.

Gongbo Tang, Mathias Müller, Annette Rios, and Rico Sennrich. 2018. Why self-attention? a targeted evaluation of neural machine translation architectures. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4263–4272, Brussels, Belgium. Association for Computational Linguistics.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2019. Understanding neural machine translation by simplification: The case of encoder-free models. In Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019), pages 1186–1193, Varna, Bulgaria. INCOMA Ltd.

Shikhar Vashishth, Shyam Upadhyay, Gaurav Singh Tomar, and Manaal Faruqui. 2019. Attention interpretability across nlp tasks. arXiv preprint arXiv:1909.11218.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, Advances in Neural Information Processing Systems 30, pages 5998–6008. Curran Associates, Inc.

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5797–5808, Florence, Italy. Association for Computational Linguistics.

Felix Wu, Angela Fan, Alexei Baevski, Yann Dauphin, and Michael Auli. 2019. Pay less attention with lightweight and dynamic convolutions. In International Conference on Learning Representations.

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

Baosong Yang, Zhaopeng Tu, Derek F. Wong, Fandong Meng, Lidia S. Chao, and Tong Zhang. 2018. Modeling localness for self-attention networks. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 4449–4458, Brussels, Belgium. Association for Computational Linguistics.

A Mixed position for hard-coded self-attention works the best

Enc-Config	Dec-Config	BLEU
(l, l)	(l, l)	27.4
(l, l)	(c, c)	27.8
(l, l)	(l, c)	28.1
(l, r)	(l, c)	**30.3**

Table 6: Search for best hard-coded configuration for hard-coded self-attention. 'l' stands for left, focusing on $i-1$, 'r' for $i+1$ and 'c' for i. Middle layers are (l,r) for encoder and (l,c) for decoder. Each cell shows settings we used in the lowest and highest layer.

B Memory efficiency and inference speedups

Table 7 summarizes the results of our profiling experiments on IWSLT16 En-De development set.

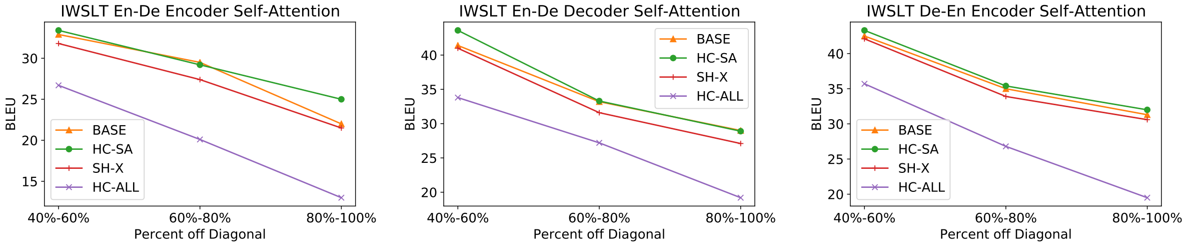

Figure 8: Off-diagonal analysis for IWSLT En-De/De-En self-attention

Model	BLEU	sent/sec	tokens/batch
BASE	30.0	43.1	14.1k
HC-SA	30.3	44.0	15.1k
SH-X	28.1	50.1	16k
BASE/-SA	22.8	46.1	16.1k
SH-X/-SA	14.9	54.9	17k

Table 7: Decoding speedup (in terms of sentences per second) and memory improvements (max tokens per batch) on IWSLT16 En-De for a variety of models. The last two rows refer to BASE and SH-X configurations whose self-attention is completely removed.

C Off-diagonal Analysis

In addition to IWSLT16 De-En decoder self-attention analysis, we provide here the off-diagonal analysis results on IWSLT16 En-De encoder and decoder self-attention, and IWSLT16 De-En encoder self-attention in Figures 8.