Analyzing Multi-Head Self-Attention: Specialized Heads Do the Heavy Lifting, the Rest Can Be Pruned

Elena Voita[1,2] David Talbot[1] Fedor Moiseev[1,5] Rico Sennrich[3,4] Ivan Titov[3,2]

[1]Yandex, Russia [2]University of Amsterdam, Netherlands
[3]University of Edinburgh, Scotland [4]University of Zurich, Switzerland
[5]Moscow Institute of Physics and Technology, Russia
{lena-voita, talbot, femoiseev}@yandex-team.ru
rico.sennrich@ed.ac.uk ititov@inf.ed.ac.uk

Abstract

Multi-head self-attention is a key component of the Transformer, a state-of-the-art architecture for neural machine translation. In this work we evaluate the contribution made by individual attention heads in the encoder to the overall performance of the model and analyze the roles played by them. We find that the most important and confident heads play consistent and often linguistically-interpretable roles. When pruning heads using a method based on stochastic gates and a differentiable relaxation of the L_0 penalty, we observe that specialized heads are last to be pruned. Our novel pruning method removes the vast majority of heads without seriously affecting performance. For example, on the English-Russian WMT dataset, pruning 38 out of 48 encoder heads results in a drop of only 0.15 BLEU.[1]

1 Introduction

The Transformer (Vaswani et al., 2017) has become the dominant modeling paradigm in neural machine translation. It follows the encoder-decoder framework using stacked multi-head self-attention and fully connected layers. Multi-head attention was shown to make more efficient use of the model's capacity: performance of the model with 8 heads is almost 1 BLEU point higher than that of a model of the same size with single-head attention (Vaswani et al., 2017). The Transformer achieved state-of-the-art results in recent shared translation tasks (Bojar et al., 2018; Niehues et al., 2018). Despite the model's widespread adoption and recent attempts to investigate the kinds of information learned by the model's encoder (Raganato and Tiedemann, 2018), the analysis of multi-head attention and its importance for translation is challenging. Previous analysis of multi-head attention considered the average of attention weights over all heads at a given position or focused only on the maximum attention weights (Voita et al., 2018; Tang et al., 2018), but neither method explicitly takes into account the varying importance of different heads. Also, this obscures the roles played by individual heads which, as we show, influence the generated translations to differing extents. We attempt to answer the following questions:

- To what extent does translation quality depend on individual encoder heads?
- Do individual encoder heads play consistent and interpretable roles? If so, which are the most important ones for translation quality?
- Which types of model attention (encoder self-attention, decoder self-attention or decoder-encoder attention) are most sensitive to the number of attention heads and on which layers?
- Can we significantly reduce the number of attention heads while preserving translation quality?

We start by identifying the most important heads in each encoder layer using layer-wise relevance propagation (Ding et al., 2017). For heads judged to be important, we then attempt to characterize the roles they perform. We observe the following types of role: positional (heads attending to an adjacent token), syntactic (heads attending to tokens in a specific syntactic dependency relation) and attention to rare words (heads pointing to the least frequent tokens in the sentence).

To understand whether the remaining heads perform vital but less easily defined roles, or are simply redundant to the performance of the model as

[1] We release code at https://github.com/lena-voita/the-story-of-heads.

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5797–5808
Florence, Italy, July 28 - August 2, 2019.

measured by translation quality, we introduce a method for pruning heads based on Louizos et al. (2018). While we cannot easily incorporate the number of active heads as a penalty term in our learning objective (i.e. the L_0 regularizer), we can use a differentiable relaxation. We prune attention heads in a continuous learning scenario starting from the converged full model and identify the roles of those which remain in the model. These experiments corroborate the findings of layer-wise relevance propagation; in particular, heads with clearly identifiable positional and syntactic functions are pruned last and hence shown to be most important for the translation task.

Our key findings are as follows:

- Only a small subset of heads are important for translation;
- Important heads have one or more specialized and interpretable functions in the model;
- The functions correspond to attention to neighbouring words and to tokens in specific syntactic dependency relations.

2 Transformer Architecture

In this section, we briefly describe the Transformer architecture (Vaswani et al., 2017) introducing the terminology used in the rest of the paper.

The Transformer is an encoder-decoder model that uses stacked self-attention and fully connected layers for both the encoder and decoder. The encoder consists of N layers, each containing two sub-layers: (a) a multi-head self-attention mechanism, and (b) a feed-forward network. The multi-head attention mechanism relies on scaled dot-product attention, which operates on a query Q, a key K and a value V:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

where d_k is the key dimensionality. In self-attention, queries, keys and values come from the output of the previous layer.

The multi-head attention mechanism obtains h (i.e. one per head) different representations of (Q, K, V), computes scaled dot-product attention for each representation, concatenates the results, and projects the concatenation through a feed-forward layer. This can be expressed in the same notation as Equation (1):

$$\text{head}_i = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V) \quad (2)$$

$$\text{MultiHead}(Q, K, V) = \text{Concat}_i(\text{head}_i)W^O \quad (3)$$

where the W_i and W^O are parameter matrices.

The second component of each layer of the Transformer network is a feed-forward network. The authors propose using a two-layer network with a ReLU activation.

Analogously, each layer of the decoder contains the two sub-layers mentioned above as well as an additional multi-head attention sub-layer. This additional sub-layer receives the output of the encoder as its keys and values.

The Transformer uses multi-head attention in three different ways: encoder self-attention, decoder self-attention and decoder-encoder attention. In this work, we concentrate primarily on encoder self-attention.

3 Data and setting

We focus on English as a source language and consider three target languages: Russian, German and French. For each language pair, we use the same number of sentence pairs from WMT data to control for the amount of training data and train Transformer models with the same numbers of parameters. We use 2.5m sentence pairs, corresponding to the amount of English–Russian parallel training data (excluding UN and Paracrawl). In Section 5.2 we use the same held-out data for all language pairs; these are 50k English sentences taken from the WMT EN-FR data not used in training.

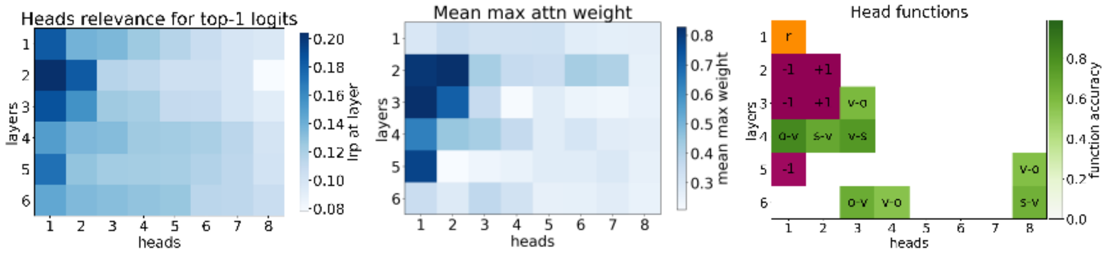
For English-Russian, we perform additional experiments using the publicly available OpenSubtitles2018 corpus (Lison et al., 2018) to evaluate the impact of domains on our results.

In Section 6 we concentrate on English-Russian and two domains: WMT and OpenSubtitles.

Model hyperparameters, preprocessing and training details are provided in appendix B.

4 Identifying Important Heads

Previous work analyzing how representations are formed by the Transformer's multi-head attention mechanism focused on either the average or the maximum attention weights over all heads (Voita et al., 2018; Tang et al., 2018), but neither method explicitly takes into account the varying importance of different heads. Also, this obscures the roles played by individual heads which, as we will show, influence the generated translations to differing extents.

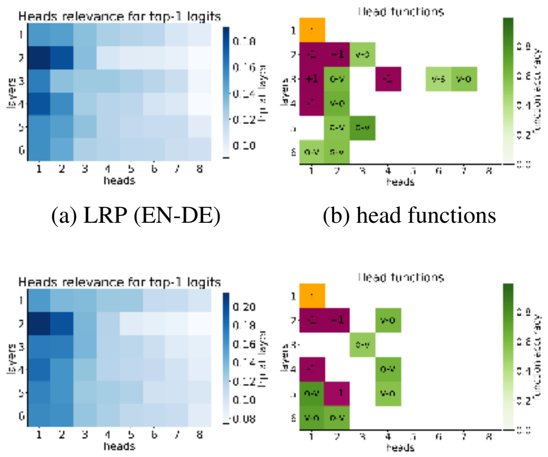


(a) LRP

(b) confidence

(c) head functions

Figure 1: Importance (according to LRP), confidence, and function of self-attention heads. In each layer, heads are sorted by their relevance according to LRP. Model trained on 6m OpenSubtitles EN-RU data.



(a) LRP (EN-DE)

(b) head functions

(c) LRP (EN-FR)

(d) head functions

Figure 2: Importance (according to LRP) and function of self-attention heads. In each layer, heads are sorted by their relevance according to LRP. Models trained on 2.5m WMT EN-DE (a, b) and EN-FR (c, d).

We define the "confidence" of a head as the average of its maximum attention weight excluding the end of sentence symbol,[2] where average is taken over tokens in a set of sentences used for evaluation (development set). A confident head is one that usually assigns a high proportion of its attention to a single token. Intuitively, we might expect confident heads to be important to the translation task.

Layer-wise relevance propagation (LRP) (Ding et al., 2017) is a method for computing the relative contribution of neurons at one point in a network to neurons at another.[3] Here we propose to use LRP to evaluate the degree to which different heads at each layer contribute to the top-1 logit predicted by the model. Heads whose outputs have a higher relevance value may be judged to be more important to the model's predictions.

The results of LRP are shown in Figures 1a, 2a, 2c. In each layer, LRP ranks a small number of heads as much more important than all others.

The confidence for each head is shown in Figure 1b. We can observe that the relevance of a head as computed by LRP agrees to a reasonable extent with its confidence. The only clear exception to this pattern is the head judged by LRP to be the most important in the first layer. It is the most relevant head in the first layer but its average maximum attention weight is low. We will discuss this head further in Section 5.3.

5 Characterizing heads

We now turn to investigating whether heads play consistent and interpretable roles within the model.

We examined some attention matrices paying particular attention to heads ranked highly by LRP and identified three functions which heads might be playing:

1. positional: the head points to an adjacent token,

2. syntactic: the head points to tokens in a specific syntactic relation,

3. rare words: the head points to the least frequent tokens in a sentence.

Now we discuss the criteria used to determine if a head is performing one of these functions and examine properties of the corresponding heads.

5.1 Positional heads

We refer to a head as "positional" if at least 90% of the time its maximum attention weight is assigned to a specific relative position (in practice either -1 or +1, i.e. attention to adjacent tokens). Such heads are shown in purple in Figures 1c for

[2] We exclude EOS on the grounds that it is not a real token.

[3] A detailed description of LRP is provided in appendix A.

English-Russian, 2b for English-German, 2d for English-French and marked with the relative position.

As can be seen, the positional heads correspond to a large extent to the most confident heads and the most important heads as ranked by LRP. In fact, the average maximum attention weight exceeds 0.8 for every positional head for all language pairs considered here.

5.2 Syntactic heads

We hypothesize that, when used to perform translation, the Transformer's encoder may be responsible for disambiguating the syntactic structure of the source sentence. We therefore wish to know whether a head attends to tokens corresponding to any of the major syntactic relations in a sentence. In our analysis, we looked at the following dependency relations: nominal subject (nsubj), direct object (dobj), adjectival modifier (amod) and adverbial modifier (advmod). These include the main verbal arguments of a sentence and some other common relations. They also include those relations which might inform morphological agreement or government in one or more of the target languages considered here.

5.2.1 Methodology

We evaluate to what extent each head in the Transformer's encoder accounts for a specific dependency relation by comparing its attention weights to a predicted dependency structure generated using CoreNLP (Manning et al., 2014) on a large number of held-out sentences. We calculate for each head how often it assigns its maximum attention weight (excluding EOS) to a token with which it is in one of the aforementioned dependency relations. We count each relation separately and allow the relation to hold in either direction between the two tokens.

We refer to this relative frequency as the "accuracy" of head on a specific dependency relation in a specific direction. Note that under this definition, we may evaluate the accuracy of a head for multiple dependency relations.

Many dependency relations are frequently observed in specific relative positions (for example, often they hold between adjacent tokens, see Figure 3). We say that a head is "syntactic" if its accuracy is at least 10% higher than the baseline that looks at the most frequent relative position for this dependency relation.

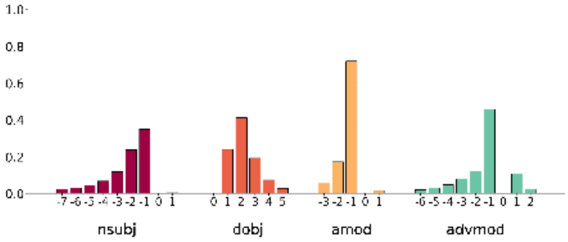


Figure 3: Distribution of the relative position of dependent for different dependency relations (WMT).

dep.	direction	best head / baseline accuracy	
		WMT	OpenSubtitles
nsubj			
	v → s	45 / 35	77 / 45
	s → v	52 / 35	70 / 45
dobj			
	v → o	78 / 41	61 / 46
	o → v	73 / 41	84 / 46
amod			
	noun → adj.m.	74 / 72	81 / 80
	adj.m. → noun	82 / 72	81 / 80
advmod			
	v → adv.m.	48 / 46	38 / 33
	adv.m. → v	52 / 46	42 / 33

Table 1: Dependency scores for EN-RU, comparing the best self-attention head to a positional baseline. Models trained on 2.5m WMT data and 6m OpenSubtitles data.

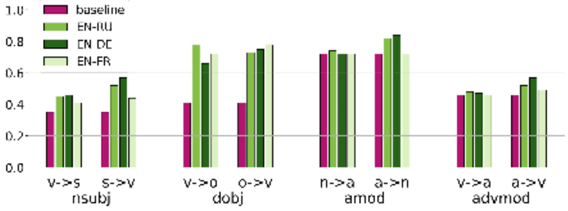
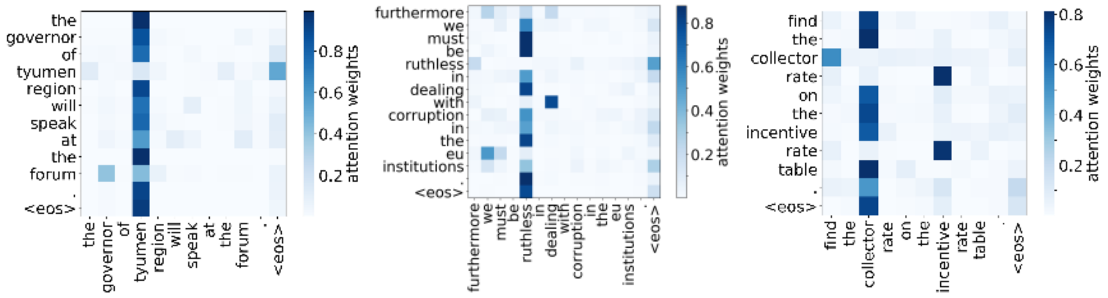


Figure 4: Dependency scores for EN-RU, EN-DE, EN-FR each trained on 2.5m WMT data.

5.2.2 Results

Table 1 shows the accuracy of the most accurate head for each of the considered dependency relations on the two domains for English-Russian. Figure 4 compares the scores of the models trained on WMT with different target languages.

Clearly certain heads learn to detect syntactic relations with accuracies significantly higher than the positional baseline. This supports the hypoth-



(a) (b) (c)

Figure 5: Attention maps of the rare words head. Models trained on WMT: (a) EN-RU, (b) EN-DE, (c) EN-FR

esis that the encoder does indeed perform some amount of syntactic disambiguation of the source sentence.

Several heads appear to be responsible for the same dependency relation. These heads are shown in green in Figures 1c, 2b, 2d.

Unfortunately, it is not possible to draw any strong conclusions from these results regarding the impact of target language morphology on the accuracy of the syntactic attention heads although relations with strong target morphology are among those that are most accurately learned.

Note the difference in accuracy of the verb-subject relation heads across the two domains for English-Russian. We hypothesize that this is due to the greater variety of grammatical person present[4] in the Subtitles data which requires more attention to this relation. However, we leave proper analysis of this to future work.

5.3 Rare words

In all models (EN-RU, EN-DE, EN-FR on WMT and EN-RU on OpenSubtitles), we find that one head in the first layer is judged to be much more important to the model's predictions than any other heads in this layer.

We find that this head points to the least frequent tokens in a sentence. For models trained on OpenSubtitles, among sentences where the least frequent token in a sentence is not in the top-500 most frequent tokens, this head points to the rarest token in 66% of cases, and to one of the two least frequent tokens in 83% of cases. For models trained on WMT, this head points to one of the two least frequent tokens in more than 50% of such cases. This head is shown in orange in Figures 1c, 2b, 2d. Examples of attention maps for this head for models trained on WMT data with different target languages are shown in Figure 5.

6 Pruning Attention Heads

We have identified certain functions of the most relevant heads at each layer and showed that to a large extent they are interpretable. What of the remaining heads? Are they redundant to translation quality or do they play equally vital but simply less easily defined roles? We introduce a method for pruning attention heads to try to answer these questions. Our method is based on Louizos et al. (2018). Whereas they pruned individual neural network weights, we prune entire model components (i.e. heads). We start by describing our method and then examine how performance changes as we remove heads, identifying the functions of heads retained in the sparsified models.

6.1 Method

We modify the original Transformer architecture by multiplying the representation computed by each head$_i$ by a scalar gate g_i. Equation (3) turns into

$$\mathrm{MultiHead}(Q, K, V) = \mathrm{Concat}_i(g_i \cdot \mathrm{head}_i) W^O.$$

Unlike usual gates, g_i are parameters specific to heads and are independent of the input (i.e. the sentence). As we would like to disable less important heads completely rather than simply down-weighting them, we would ideally apply L_0 regularization to the scalars g_i. The L_0 norm equals the number of non-zero components and would push the model to switch off less important heads:

$$L_0(g_1, \dots, g_h) = \sum_{i=1}^{h} (1 - [[g_i = 0]]),$$

[4] First, second and third person subjects are encountered in approximately 6%, 3% and 91% of cases in WMT data and in 32%, 21% and 47% of cases in OpenSubtitles data.

where h is the number of heads, and $[[\quad]]$ denotes the indicator function.

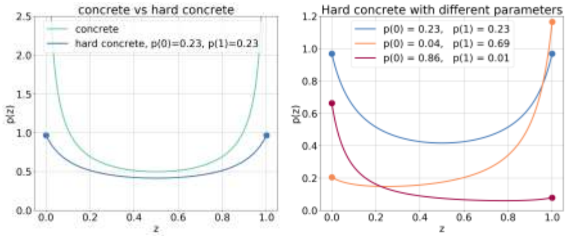
Unfortunately, the L_0 norm is non-differentiable and so cannot be directly incorporated as a regularization term in the objective function. Instead, we use a stochastic relaxation: each gate g_i is now a random variable drawn independently from a head-specific distribution.[5] We use the Hard Concrete distributions (Louizos et al., 2018), a parameterized family of mixed discrete-continuous distributions over the closed interval $[0, 1]$, see Figure 6a. The distributions have non-zero probability mass at 0 and 1, $P(g_i = 0|\phi_i)$ and $P(g_i = 1|\phi_i)$, where ϕ_i are the distribution parameters. Intuitively, the Hard Concrete distribution is obtained by stretching the binary version of the Concrete (aka Gumbel softmax) distribution (Maddison et al., 2017; Jang et al., 2017) from the original support of $(0, 1)$ to $(-\epsilon, 1 + \epsilon)$ and then collapsing the probability mass assigned to $(-\epsilon, 1]$ and $[1, 1 + \epsilon)$ to single points, 0 and 1, respectively. These stretching and rectification operations yield a mixed discrete-continuous distribution over $[0, 1]$. Now the sum of the probabilities of heads being non-zero can be used as a relaxation of the L_0 norm:

$$L_C(\phi) = \sum_{i=1}^{h} (1 - P(g_i = 0|\phi_i)).$$

The new training objective is

$$L(\theta, \phi) = L_{xent}(\theta, \phi) + \lambda L_C(\phi),$$

where θ are the parameters of the original Transformer, $L_{xent}(\theta, \phi)$ is cross-entropy loss for the translation model, and $L_C(\phi)$ is the regularizer described above. The objective is easy to optimize: the reparameterization trick (Kingma and Welling, 2014; Rezende et al., 2014) can be used to backpropagate through the sampling process for each g_i, whereas the regularizer and its gradients are available in the closed form. Interestingly, we observe that the model converges to solutions where gates are either almost completely closed (i.e. the head is pruned, $P(g_i = 0|\phi_i) \approx 1$) or completely open ($P(g_i = 1|\phi_i) \approx 1$), the latter not being explicitly encouraged.[6] This means that at test time we can treat the model as a standard Transformer and use only a subset of heads.[7]



(a) (b)

Figure 6: Concrete distribution: (a) Concrete and its stretched and rectified version (Hard Concrete); (b) Hard Concrete distributions with different parameters.

When applying this regularizer, we start from the converged model trained without the L_C penalty (i.e. parameters θ are initialized with the parameters of the converged model) and then add the gates and continue training the full objective. By varying the coefficient λ in the optimized objective, we obtain models with different numbers of heads retained.

6.2 Pruning encoder heads

To determine which head functions are most important in the encoder and how many heads the model needs, we conduct a series of experiments with gates applied only to encoder self-attention. Here we prune a model by fine-tuning a trained model with the regularized objective.[8] During pruning, the parameters of the decoder are fixed and only the encoder parameters and head gates are fine-tuned. By not fine-tuning the decoder, we ensure that the functions of the pruned encoder heads do not migrate to the decoder.

6.2.1 Quantitative results: BLEU score

BLEU scores are provided in Figure 7. Surprisingly, for OpenSubtitles, we lose only 0.25 BLEU when we prune all but 4 heads out of 48.[9] For the more complex WMT task, 10 heads in the encoder are sufficient to stay within 0.15 BLEU of the full model.

[5] In training, we resample gate values g_i for each batch.

[6] The 'noise' pushes the network not to use middle values. The combination of noise and rectification has been previously used to achieve discretization (e.g., Kaiser and Bengio (2018)).

[7] At test time, gate values are either 0 or 1 depending on which of the values $P(g_i = 0|\phi_i)$, $P(g_i = 1|\phi_i)$ is larger.

[8] In preliminary experiments, we observed that fine-tuning a trained model gives slightly better results (0.2–0.6 BLEU) than applying the regularized objective, or training a model with the same number of self-attention heads, from scratch.

[9] If all heads in a layer are pruned, the only remaining connection to the previous layer is the residual connection.

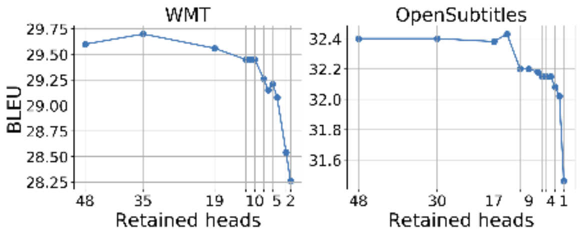


Figure 7: BLEU score as a function of number of retained encoder heads (EN-RU). Regularization applied by fine-tuning trained model.

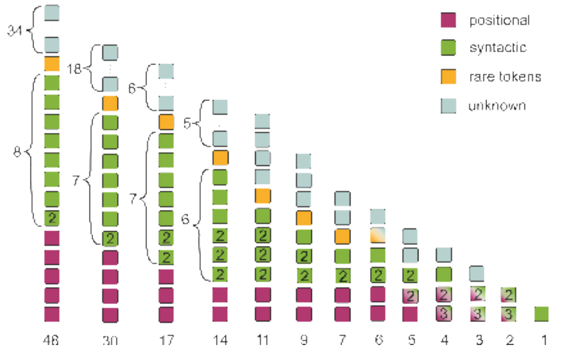


Figure 8: Functions of encoder heads retained after pruning. Each column represents all remaining heads after varying amount of pruning (EN-RU; Subtitles).

6.2.2 Functions of retained heads

Results in Figure 7 suggest that the encoder remains effective even with only a few heads. In this section, we investigate the function of those heads that remain in the encoder during pruning. Figure 8 shows all heads color-coded for their function in a pruned model. Each column corresponds to a model with a particular number of heads retained after pruning. Heads from all layers are ordered by their function. Some heads can perform several functions (e.g., $s \to v$ and $v \to o$); in this case the number of functions is shown.

First, we note that the model with 17 heads retains heads with all the functions that we identified in Section 5, even though ⅔ of the heads have been pruned.

This indicates that these functions are indeed the most important. Furthermore, when we have fewer heads in the model, some functions "drift" to other heads: for example, we see positional heads starting to track syntactic dependencies; hence some heads are assigned more than one color at certain stages in Figure 8.

	attention heads (e/d/d-e)	BLEU from trained	BLEU from scratch
WMT, 2.5m			
baseline	48/48/48	**29.6**	
sparse heads	14/31/30	29.62	29.47
	12/21/25	29.36	28.95
	8/13/15	29.06	28.56
	5/9/12	28.90	28.41
OpenSubtitles, 6m			
baseline	48/48/48	**32.4**	
sparse heads	27/31/46	32.24	32.23
	13/17/31	32.23	31.98
	6/9/13	32.27	31.84

Table 2: BLEU scores for gates in all attentions, EN-RU. Number of attention heads is provided in the following order: encoder self-attention, decoder self-attention, decoder-encoder attention.

6.3 Pruning all types of attention heads

We found our pruning technique to be efficient at reducing the number of heads in the encoder without a major drop in translation quality. Now we investigate the effect of pruning all types of attention heads in the model (not just in the encoder). This allows us to evaluate the importance of different types of attention in the model for the task of translation. In these experiments, we add gates to all multi-head attention heads in the Transformer, i.e. encoder and decoder self-attention and attention from the decoder to the encoder.

6.3.1 Quantitative results: BLEU score

Results of experiments pruning heads in all attention layers are provided in Table 2. For models trained on WMT data, we are able to prune almost ¾ of encoder heads and more than ⅓ of heads in decoder self-attention and decoder-encoder attention without any noticeable loss in translation quality (sparse heads, row 1). We can also prune more than half of all heads in the model and lose no more than 0.25 BLEU.

While these results show clearly that the majority of attention heads can be removed from the fully trained model without significant loss in translation quality, it is not clear whether a model can be trained from scratch with such a small number of heads. In the rightmost column in Ta-

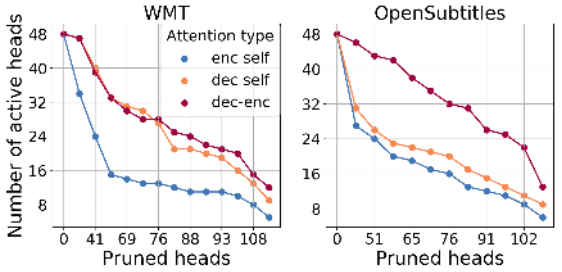


Figure 9: Number of active heads of different attention type for models with different sparsity rate

ble 2 we provide BLEU scores for models trained with exactly the same number and configuration of heads in each layer as the corresponding pruned models but starting from a random initialization of parameters. Here the degradation in translation quality is more significant than for pruned models with the same number of heads. This agrees with the observations made in works on model compression: sparse architectures learned through pruning cannot be trained from scratch to the same test set performance as a model trained with joint sparsification and optimization (Zhu and Gupta, 2017; Gale et al., 2019). In our case, attention heads are less likely to learn important roles when a model is retrained from scratch with a small number of heads.

6.3.2 Heads importance

Figure 9 shows the number of retained heads for each attention type at different pruning rates. We can see that the model prefers to prune encoder self-attention heads first, while decoder-encoder attention heads appear to be the most important for both datasets. Obviously, without decoder-encoder attention no translation can happen.

The importance of decoder self-attention heads, which function primarily as a target side language model, varies across domains. These heads appear to be almost as important as decoder-encoder attention heads for WMT data with its long sentences (24 tokens on average), and slightly more important than encoder self-attention heads for OpenSubtitles dataset where sentences are shorter (8 tokens on average).

Figure 10 shows the number of active self-attention and decoder-encoder attention heads at different layers in the decoder for models with different sparsity rate (to reduce noise, we plot the sum of heads remaining in pairs of adjacent layers). It can be seen that self-attention heads are retained more readily in the lower layers, while decoder-encoder attention heads are retained in the higher layers. This suggests that lower layers of the Transformer's decoder are mostly responsible for language modeling, while higher layers are mostly responsible for conditioning on the source sentence. These observations are similar for both datasets we use.

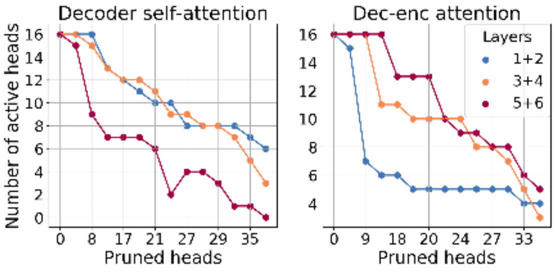


Figure 10: Number of active heads in different layers of the decoder for models with different sparsity rate (EN-RU, WMT)

7 Related work

One popular approach to the analysis of NMT representations is to evaluate how informative they are for various linguistic tasks. Different levels of linguistic analysis have been considered including morphology (Belinkov et al., 2017a; Dalvi et al., 2017; Bisazza and Tump, 2018), syntax (Shi et al., 2016) and semantics (Hill et al., 2017; Belinkov et al., 2017b; Raganato and Tiedemann, 2018).

Bisazza and Tump (2018) showed that the target language determines which information gets encoded. This agrees with our results for different domains on the English-Russian translation task in Section 5.2.2. There we observed that attention heads are more likely to track syntactic relations requiring more complex agreement in the target language (in this case the subject-verb relation).

An alternative method to study the ability of language models and machine translation models to capture hierarchical information is to test their sensitivity to specific grammatical errors (Linzen et al., 2016; Gulordava et al., 2018; Tran et al., 2018; Sennrich, 2017; Tang et al., 2018). While this line of work has shown that NMT models, including the Transformer, do learn some syntactic structures, our work provides further insight into the role of multi-head attention.

There are several works analyzing attention weights of different NMT models (Ghader and Monz, 2017; Voita et al., 2018; Tang et al., 2018;

Raganato and Tiedemann, 2018). Raganato and Tiedemann (2018) use the self-attention weights of the Transformer's encoder to induce a tree structure for each sentence and compute the unlabeled attachment score of these trees. However they do not evaluate specific syntactic relations (i.e. labeled attachment scores) or consider how different heads specialize to specific dependency relations.

Recently Bau et al. (2019) proposed a method for identifying important individual neurons in NMT models. They show that similar important neurons emerge in different models. Rather than verifying the importance of individual neurons, we identify the importance of entire attention heads using layer-wise relevance propagation and verify our findings by observing which heads are retained when pruning the model.

8 Conclusions

We evaluate the contribution made by individual attention heads to Transformer model performance on translation. We use layer-wise relevance propagation to show that the relative contribution of heads varies: only a small subset of heads appear to be important for the translation task. Important heads have one or more interpretable functions in the model, including attending to adjacent words and tracking specific syntactic relations. To determine if the remaining less-interpretable heads are crucial to the model's performance, we introduce a new approach to pruning attention heads.

We observe that specialized heads are the last to be pruned, confirming their importance directly. Moreover, the vast majority of heads, especially the encoder self-attention heads, can be removed without seriously affecting performance. In future work, we would like to investigate how our pruning method compares to alternative methods of model compression in NMT.

Acknowledgments

We would like to thank anonymous reviewers for their comments. We thank Wilker Aziz, Joost Bastings for their helpful suggestions. The authors also thank Yandex Machine Translation team for helpful discussions and inspiration. Ivan Titov acknowledges support of the European Research Council (ERC StG BroadSem 678254) and the Dutch National Science Foundation (NWO VIDI 639.022.518).

References

Sebastian Bach, Alexander Binder, Grégoire Montavon, Frederick Klauschen, Klaus-Robert Müller, and Wojciech Samek. 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PloS one*, 10(7):e0130140.

Anthony Bau, Yonatan Belinkov, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2019. Identifying and controlling important neurons in neural machine translation. In *International Conference on Learning Representations*, New Orleans.

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017a. What do neural machine translation models learn about morphology? In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 861–872. Association for Computational Linguistics.

Yonatan Belinkov, Lluís Màrquez, Hassan Sajjad, Nadir Durrani, Fahim Dalvi, and James Glass. 2017b. Evaluating layers of representation in neural machine translation on part-of-speech and semantic tagging tasks. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1–10. Asian Federation of Natural Language Processing.

Arianna Bisazza and Clara Tump. 2018. The lazy encoder: A fine-grained analysis of the role of morphology in neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2871–2876, Brussels, Belgium. Association for Computational Linguistics.

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Matthias Huck, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (wmt18). In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 272–307, Belgium, Brussels. Association for Computational Linguistics.

Fahim Dalvi, Nadir Durrani, Hassan Sajjad, Yonatan Belinkov, and Stephan Vogel. 2017. Understanding and improving morphological learning in the neural machine translation decoder. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 142–151. Asian Federation of Natural Language Processing.

Yanzhuo Ding, Yang Liu, Huanbo Luan, and Maosong Sun. 2017. Visualizing and understanding neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1150–1159, Vancouver, Canada. Association for Computational Linguistics.

Trevor Gale, Erich Elsen, and Sara Hooker. 2019. The state of sparsity in deep neural networks. *arXiv preprint*.

Hamidreza Ghader and Christof Monz. 2017. What does attention in neural machine translation pay attention to? In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 30–39. Asian Federation of Natural Language Processing.

Kristina Gulordava, Piotr Bojanowski, Edouard Grave, Tal Linzen, and Marco Baroni. 2018. Colorless green recurrent networks dream hierarchically. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1195–1205. Association for Computational Linguistics.

Felix Hill, Kyunghyun Cho, Sébastien Jean, and Y Bengio. 2017. The representational geometry of word meanings acquired by neural machine translation models. *Machine Translation*, 31.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, Toulon, France.

Łukasz Kaiser and Samy Bengio. 2018. Discrete autoencoders for sequence models. *arXiv preprint arXiv:1801.09797*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *Proceedings of the International Conference on Learning Representation (ICLR 2015)*.

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. In *International Conference on Learning Representations*, Banff, Canada.

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *Transactions of the Association for Computational Linguistics*, 4:521–535.

Pierre Lison, Jörg Tiedemann, and Milen Kouylekov. 2018. OpenSubtitles2018: Statistical Rescoring of Sentence Alignments in Large, Noisy Parallel Corpora. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Christos Louizos, Max Welling, and Diederik P. Kingma. 2018. Learning sparse neural networks through l_0 regularization. In *International Conference on Learning Representations*, Vancouver, Canada.

Chris J. Maddison, Andriy Mnih, and Yee Whye Teh. 2017. The concrete distribution: A continuous relaxation of discrete random variables. In *International Conference on Learning Representations*, Toulon, France.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Jan Niehues, Ronaldo Cattoni, Sebastian Stüker, Mauro Cettolo, Marco Turchi, and Marcello Federico. 2018. The IWSLT 2018 Evaluation Campaign. In *Proceedings of the 15th International Workshop on Spoken Language Translation*, pages 118–123, Bruges, Belgium.

Martin Popel and Ondrej Bojar. 2018. Training Tips for the Transformer Model. pages 43–70.

Alessandro Raganato and Jörg Tiedemann. 2018. An analysis of encoder representations in transformer-based machine translation. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium. Association for Computational Linguistics.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1278–1286, Bejing, China. PMLR.

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.

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, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural mt learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Gongbo Tang, Rico Sennrich, and Joakim Nivre. 2018. An analysis of attention mechanisms: The case of word sense disambiguation in neural machine translation. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 26–35, Belgium, Brussels. Association for Computational Linguistics.

Ke Tran, Arianna Bisazza, and Christof Monz. 2018. The importance of being recurrent for modeling hierarchical structure. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4731–4736. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, Los Angeles.

Elena Voita, Pavel Serdyukov, Rico Sennrich, and Ivan Titov. 2018. Context-aware neural machine translation learns anaphora resolution. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1264–1274, Melbourne, Australia. Association for Computational Linguistics.

Michael Zhu and Suyog Gupta. 2017. To prune, or not to prune: exploring the efficacy of pruning for model compression. *arXiv preprint arXiv:1710.01878*.

A Layer-wise Relevance Propagation

Layer-wise relevance propagation (LRP) was originally designed to compute the contributions of single pixels to predictions of image classifiers (Bach et al., 2015). LRP back-propagates relevance recursively from the output layer to the input layer. We adapt LRP to the Transformer model to calculate relevance that measures the association degree between two arbitrary neurons in neural networks. In the following, we describe the general idea of the LRP method, give the formal definition used in our experiments and describe how to compute a head relevance.

A.1 General idea

Layer-wise relevance propagation in its general form assumes that the model can be decomposed into several layers of computation. The first layer are the inputs (for example, the pixels of an image or tokens of a sentence), the last layer is the real-valued prediction output of the model f. The l-th layer is modeled as a vector $z = (z_d^{(l)})_{d=1}^{V(l)}$ with dimensionality $V(l)$. Layer-wise relevance propagation assumes that we have a Relevance score $R_d^{(l+1)}$ for each dimension $z_d^{(l+1)}$ of the vector z at layer $l+1$. The idea is to find a Relevance score $R_d^{(l)}$ for each dimension $z_d^{(l)}$ of the vector z at the next layer l which is closer to the input layer such that the following equation holds:

$$f = \ldots = \sum_{d\in l+1} R_d^{(l+1)} = \sum_{d\in l} R_d^{(l)} = \cdots = \sum_{d} R_d^{(1)}.$$

This equation represents a *conservation principle*, on which LRP relies to propagate the prediction back without using gradients. Intuitively, this means that total contribution of neurons at each layer is constant. Since we are interested only in heads relevance, we do not propagate till input variables and stop at the neurons of the encoder layer of interest.

A.2 Formal rules

In this section, we provide formal rules for propagating relevance. Here we follow the approach by Ding et al. (2017) previously used for neural machine translation.

Let $r_{u\leftarrow v}$ denote *relevance* of neuron u for neuron v.

Definition 1 Given a neuron u, its incoming neuron set $IN(u)$ comprises all its direct connected preceding neurons in the network.

Definition 2 Given a neuron u, its outcoming neuron set $OUT(u)$ comprises all its direct connected descendant neurons in the network.

Definition 3 Given a neuron v and its incoming neurons $u \in IN(v)$, the *weight ratio* measures the contribution of u to v. It is calculated as

$$w_{u\to v} = \frac{W_{u,v}u}{\sum\limits_{u'\in IN(v)} W_{u',v}u'} \quad \text{if } v = \sum_{u'\in IN(v)} W_{u',v}u',$$

$$w_{u\to v} = \frac{u}{\sum\limits_{u'\in IN(v)} u'} \quad \text{if } v = \prod_{u'\in IN(v)} u'.$$

These equations define weight ratio for matrix multiplication and element-wise multiplication operations.

Redistribution rule for LRP Relevance is propagated using the *local redistribution rule* as follows:

$$r_{u\leftarrow v} = \sum_{z\in OUT(u)} w_{u\to z} r_{z\leftarrow v}.$$

The provided equations for computing weights ratio and the redistribution rule allow to compute the relative contribution of neurons at one point in a network to neurons at another. Note that we follow Ding et al. (2017) and ignore non-linear activation functions.

A.3 Head relevance

In our experiments, we compute relative contribution of each head to the network predictions. For this, we evaluate contribution of neurons in head$_i$ (see equation 1) to the top-1 logit predicted by the model. Head relevance for a given prediction is computed as the sum of relevances of its neurons, normalized over heads in a layer. The final relevance of a head is its average relevance, where average is taken over all generation steps for a development set.

B Experimental setup

B.1 Data preprocessing

Sentences were encoded using byte-pair encoding (Sennrich et al., 2016), with source and target vocabularies of about 32000 tokens. For OpenSubtitles data, we pick only sentence pairs with a relative time overlap of subtitle frames between source and target language subtitles of at least 0.9 to reduce noise in the data. Translation pairs were batched together by approximate sequence length. Each training batch contained a set of translation pairs containing approximately 16000[10] source tokens. It has been shown that Transformer's performance depends heavily on a batch size (Popel and Bojar, 2018), and we chose a large value of batch size to ensure that models show their best performance.

B.2 Model parameters

We follow the setup of Transformer base model (Vaswani et al., 2017). More precisely, the number of layers in the encoder and in the decoder is $N = 6$. We employ $h = 8$ parallel attention layers, or heads. The dimensionality of input and output is $d_{model} = 512$, and the inner-layer of a feed-forward networks has dimensionality $d_{ff} = 2048$.

We use regularization as described in (Vaswani et al., 2017).

B.3 Optimizer

The optimizer we use is the same as in (Vaswani et al., 2017). We use the Adam optimizer (Kingma and Ba, 2015) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\varepsilon = 10^{-9}$. We vary the learning rate over the course of training, according to the formula:

$$l_{rate} = scale \cdot \min(step\_num^{-0.5}, \\ step\_num \cdot warmup\_steps^{-1.5})$$

We use $warmup\_steps = 16000$, $scale = 4$.

[10] This can be reached by using several of GPUs or by accumulating the gradients for several batches and then making an update.