Multi-Head Rotation - a Multi-Head Attention Utilisation Using Parameters Rotation and Linear Mixing

Guy Azov
Tel-Aviv University
guyazov@mail.tau.ac.il

Omer Landau
Tel-Aviv University
omerlandau1@mail.tau.ac.il

1 Introduction

Being a key component in various NLP solutions, the Transformer model Vaswani et al. (2017) has been extensively studied in order to unveil it's underlying features and behaviours. While many works have tried to improve the original Transformer setup, some focused on challenge and debunk it's components. One of these components is the popular Multi-Head Attention mechanism, an apparatus which supposed to better capture contextual information from a sequence using N_h parallel different attention "Heads". Michel et al. (2019) and Voita et al. (2019) Showed that many of the attention heads can be pruned during inference with no severe effect on overall results. Fan et al. (2019) also did it, but with an entire layer, not degrading performances as well.

Some works also tried to exploit this over-parameterization of the MHA. Li et al. (2018) demonstrated that it is possible to improve the Transformer performances by encouraging discrepancy between the attention heads. Zhou et al. (2020) proposed a regularization method in order to prevent the MHA from being dominated by a small portion of attention heads, Peng et al. (2020) also suggested a method to activate different heads on different inputs.

Stepping into a very active research area, in this article we focus on two main problems regarding to Multi-Head Attention mechanism:

- Unbalanced importance distribution of different Self-Attention/Encoder-Attention heads locally within a layer.
- Unbalanced importance distribution of different Encoder-Attention heads across all layers of the decoder.

We shortly elaborate on these issues in the method section, and offer couple of main strategies for tackling them in addition to bettering utilisation of attention heads in general.

All of our strategies are based on information sharing between different heads during training, using what we call rotation of head parameters and linear mixture of attention heads. We achieve impressive improvements in total BLEU score comparing to our Vanilla setup and provide some analysis for the given results.[1]

2 Background

2.1 Attention

Brief reminder, the most popular form of attention is the scaled bilinear attention (Luong et al., 2015) which can be described as a weighted sum of values (V) where the weights are given by a function of Queries (Q) and Keys (K). Hence:

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

And d_k is the dimension of K and Q.

2.2 Multi Head Attention

Every layer of the transformer has (at least one) attention mechanism called **MHA**. Given a matrix $\mathbf{X} = (x_1, ..., x_n) \in R^{n \times d_{model}}$, every head of the MHA has three trainable matrices:

$W^Q, W^K \in R^{d_{model} \times d_k}$ and $W^V \in R^{d_{model} \times d_v}$ Vaswani et al. (2017) defines a head by:

$$Z_i = softmax(\frac{Q_i K_i^T}{\sqrt{d_k}})V_i \quad (2)$$

where $Q_i = \mathbf{X} \cdot W_i^Q$, $K_i = \mathbf{X} \cdot W_i^K$, $V_i = \mathbf{X} \cdot W_i^V$. In addition to W^Q, W^K every MHA has another

[1]Our source code is available in https://github.com/omerlandau/MHR-NLP-Transformers

trainable parameters matrix $W^O \in R^{(h \times d_v) \times d_{model}}$. Finalizing the computation of Multi-Head Attention we have:

$$MHA(\mathbf{X}) = [Z_1, ..., Z_h] \cdot W^O. \tag{3}$$

2.3 Trace Norm (Nuclear Norm)

The trace norm of a matrix $\mathbf{A} \in R^{mxn}$ is defined by :

$$||\mathbf{A}||_* = trace(\sqrt{\mathbf{A}^* \mathbf{A}}) = \sum_{j=1}^{min(m,n)} \sigma_j(\mathbf{A}) \tag{4}$$

Where $\sigma_j(\mathbf{A})$ is the j-th singular value of $\mathbf{A}$ and $\mathbf{A}^*\mathbf{A}$ is a pseudo semi-definite matrix. The trace norm, $||\mathbf{A}||_*$ is a convex envelope and approximation of the rank($\mathbf{A}$) function.

2.4 Confidence

Presented by Voita et al. (2019) the "Confidence" is a naive heuristic for testing importance of a certain attention head within a transformer layer. A confident head is considered to be one that frequently assigns high proportion of it's attention to a single token. Confidence is defined as the average of the maximal attention weight that the head assigns for each word, excluding the **eos** token. Formally:

$$\mathbf{Confidence} = \mathbf{avg}(\mathbf{max}_{per-word}(attn_{weights})) \tag{5}$$

3 Method

3.1 Multi-Head Rotation (MHR)

Equipped with the need for better utilisation of attention heads, first we offer a naive approach for sharing information gathered by attention heads during training. We call it Multi-Head Rotation (MHR), MHR is basically the process of swapping between pairs of attention heads during different parts of training phase.

Swapping process is quite simple, given a chosen pair of attention heads in the Transformer model and a phase in training for executing the swap. We basically replace the parameters of each specific W^Q, W^V, W^K tensors that are responsible for the calculation of chosen attention heads.

For example (see fig 1), say we want to swap head number 5 in layer 2 of Encoder's self attention and head number 1 in layer 3 of Encoder's self attention. Then for this assignment we would take the parameters of $\mathbf{W^Q_{2,5}}, \mathbf{W^V_{2,5}}, \mathbf{W^K_{2,5}}$ and swap them with the parameters of $\mathbf{W^Q_{3,1}}, \mathbf{W^V_{3,1}} \mathbf{W^K_{3,1}}$ respectively. This way when the model uses W^Q, W^V, W^K parameters for outputting the latter head attentions, it'll perform the computation using the updated swapped weights instead of prior original ones.

It is highly important to mention that we never perform swaps of heads from different type of attention mechanism i.e Encoder-Attention/Self-Attention, nor crossing between encoder and decoder too. This applies to any and all of the following methods, even if not directly mentioned.

3.1.1 How To Decide which Heads to Swap?

Describing the concept of a single swap is fine, but how does one conduct a productive set of swaps which actually contributes model's training? Although the search span for productive swaps is quite large, we've tried to answer this question using the two following approaches.

3.1.2 Manual approach

Based on prior knowledge about the importance of different attention heads. Mainly originated from the work done by Voita et al. (2019) and Michel et al. (2019). We focus on two main guidelines for building our swapping experiments:

In-Layer Swaps
Inspired by Voita et al. (2019) findings, we've tried a set of swapping experiments targeting pairs of Attention-Heads within the same layer (In-Layer). Voita et al. (2019) have found that in most of decoder's and encoder's attentions, only the first few Attention-Heads within the layer (1-2 in an 8 heads setup) are dominant and important to overall BLEU score (tested on WMT Translation tasks).

Thus, we composed quite a few manual swap configurations which are focused just on that. Swapping first Attention-Heads of a certain layer with last ones from the same layer. For example on an 8 Heads setup: swapping Encoder Self Attention's H_1 in Layer 1, H_2 in Layer 1 with H_8 in Layer 1 and H_7 in Layer 1 respectively.

We've tried this in various stages of training and along different layers of the model. These

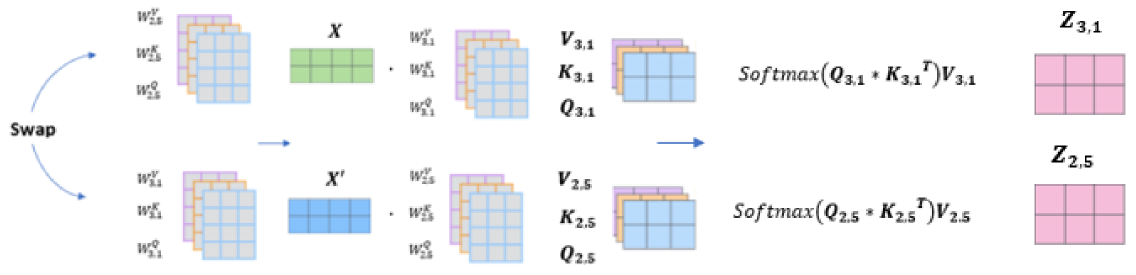

Figure 1: Multi-Head-Rotation

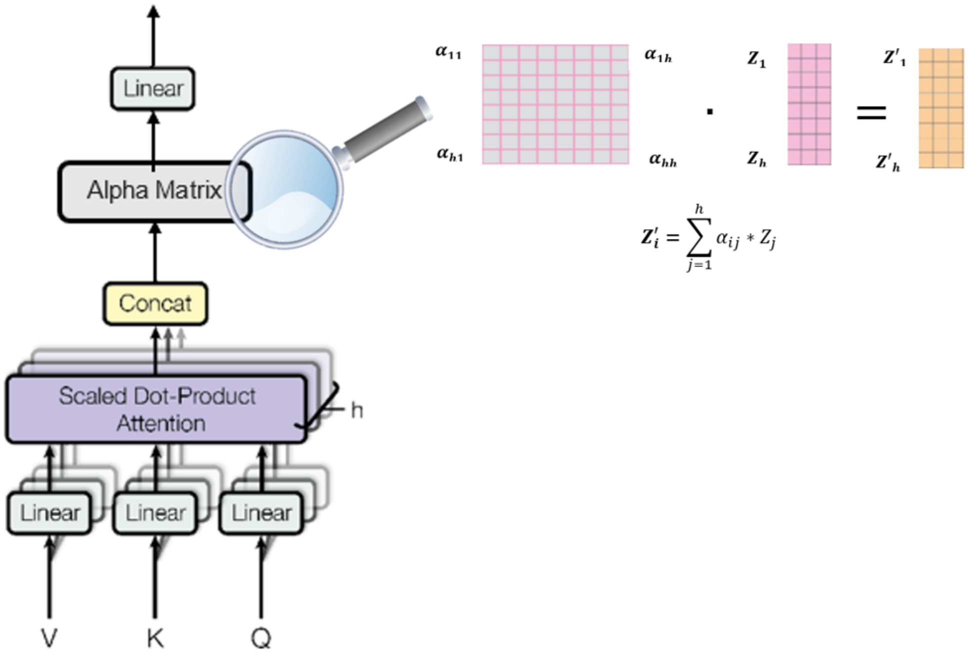

Figure 2: Multi Head Linear Mixing

types of In-Layer swaps certainly did not cause performance degradation, but it has also failed to deliver a significant improvement. Therefore, we didn't find MHR effective for In-Layer Swaps. Yet we do provide a solution for this kind of information sharing later on in our work.

Encoder-Attention Multi-Layer Swaps
Michel et al. (2019), have actually reported that decoder's Encoder-Attentions are more reliable on the Multi-Head Attention mechanism. Having said that, they also pointed out that Encoder-Attention in lower layers (closer to output) are significantly more important than upper ones (closer to input). And showed it by testing the effect on the total BLEU score of the WMT translation task, where masking lower layers lead to a degradation in results up to 13 BLEU points comparing to only 0.2 BLEU degradation in upper layers. Determent to better the importance balance of different layers in decoder. We construct couple of experiments for ablating swaps of Encoder-Attention heads from upper and lower layers of decoder. Testing various options along different stages of training. We elaborate more about rotation configurations used in the experiments section.

3.1.3 Dynamic Approach

Since we have limited ability to manually go through the search span and predict different patterns or properties of "good" swaps. Additionally it is hard to plan swaps upfront for an entire delicate training procedure. We provide couple of "out of the box" methods for dynamically composing swaps during training.

In order for the model to "choose" swaps automatically on it's own, we need to establish metrics for determining the contribution and importance of a certain head during training.

The first metric we use is the Confidence heuristic introduced by Voita et al. (2019) 5.
For our second metric we rank attention heads using the average l2 distance of each head from all of it's head "neighbours" within the layer.

$$\mathbf{AVG_{L_2}(H_i)} = \frac{\sum_{j \neq i} ||H_j - H_i||_2}{N_H - 1} \qquad (6)$$

Say a head is relatively closer to most of his neighbours, thus it is less likely to contribute new information within it's layer. There are probably one or more heads which already deduce somewhat similar properties to it's.

It is worth mentioning that both of the metrics are calculated using the validation set after each epoch. This way, model's estimation for head importance over the batch is not entirely biased by sentences it has already seen. Of course we perform swaps based on the same val set at each epcoh, yet it is not as direct as updating head's weights according to them which is the case with sentences in the train set.

Once we have simple methods to detect which attention heads are considered important or neglectable during training. We design 3 nimble algorithms for choosing swaps during training

Hard Algorithm
The "Hard" algorithm is a naive method, which basically swaps k most redundant heads based on the chosen importance metric with k most important heads. Top and Bottom k heads are universally picked from all across the layers of the transformer. Not between different types of transformers i.e encoder/decoder or attentions i.e self/encoder.

Algorithm 1 Hard

```
Require: $k; N_{layers}; N_{heads}$
Ensure: $k \leq \lfloor (N_{layers} * N_{heads})/2 \rfloor$
  Regardless to layer, sort all heads according
to the importance metric chosen:
Swap between $[H_0, ..., H_k]$ and
$[H_{N_{layers}*N_{heads}}, ..., H_{N_{layers}*N_{heads}-k}]$
```

Soft Algorithm
Driven by the relative success of manual swaps of upper and lower layers, we composed a second more delicate algorithm, which looks at heads from upper layers that are considered less important according to the used metric, and swap them with also "redundant" heads of lower layers. The main idea here is to perform more delicate swaps which are on the one hand allowing exchange of information between layers, and on the other hand still preserve in place important heads - for the sake of performance stability.

Algorithm 2 Soft

```
Require: $k$, the number of swaps per layer;
  $t_{max}$, the max participant layers: $t_{max}$;layers:
  L;$N_{heads}$;$N_{layers}$
Ensure: $k \leq \lfloor (N_{heads})/2 \rfloor$,$k \leq \lfloor (N_{layers})/2 \rfloor$
  for $i \leftarrow 1$ to $t_{max}$ do
    Sort $L_i$'s and $L_{NLayers-i}$'s heads according
    to importance metric score (l2 uniqueness/
    confidence).
    Swap between $L_i[0], ..., L_i[k]$
    and $L_{Nlayers-i}[0], ..., L_{Nlayers-i}[k]$
    respectively.
  end for
```

Random Algorithm
We also perform random k swaps for sanity check of our soft and hard methods. The k pairs of heads are universally picked from all across the layers of the transformer, not between different types of transformers or attentions.

Algorithm 3 Random

```
Require: $k$, the number of swaps;$N_{heads}$;$N_{layers}$
Ensure: $k \leq \lfloor (N_{layers} * N_{heads})/2 \rfloor$
  Choose k different pairs
 of heads randomly and swap them.
```

3.2 In-Layer Multi Head Mixing (MHM)

As mentioned above, Voita et al. (2019) pointed out one of the inefficiencies of Multi Head Attention - there are only few dominant heads per layer. Earlier we've tried to balance the magnitude of the heads, by targeting rotations within the same layers. These efforts came up short, in contrast to swaps between different layers of the decoder which benefited quite well. Determent by the failure attempt, we've come up with more subtle idea for information sharing within the same layer, we call it Multi Head Linear Mixing (**MHM**).

MHM is based on an additional small component which we call the Alphas matrix. Alphas matrix is an $N_{heads} X N_{heads}$ matrix which contains alpha coefficient values. We use this matrix to perform replacements of each attention head with a linear combination of all heads within it's layer. In comparison to Multi-Head Rotation, when using the alphas we are not dealing with W^Q, W^V, W^K weight matrices. Instead we directly mixture the final calculated attention heads Z_i 2.

We do it in the following fashion: In the process of producing Multi-Head attention, all attention heads are being calculated as mentioned above using Q, V, K and marked as $Z_1, \ldots, Z_{N_{heads}}$. These Z are then concatenated to a single Z tensor. The Z tensor is in it's turn passed through an out projection layer (W^O) which compresses it back to a single attention mechanism.

With the purpose of allowing heads from same layer to move and mixture information with other neighbouring heads. **Instead** of directly passing Z to the latter Output layer, we in fact reshape Z to be viewed as $[N_{heads}, BSZ, Tseq, H_{dim}]$ where BSZ, Tseq and H_{dim} are the batch size, the length of the target sequence and the dimensions of a single head respectively. Next we multiply (matrix wise) the reshaped Z with the mentioned above Alphas matrix and deduce a Z' with the same previous dimensions, only that now each attention head in Z' is actually represented by a linear combination of all previous heads within it's layer (see fig 2).

For example Z'_1 would be:
$Z'_1 = \alpha_{1,1} * Z_1 + \ldots \alpha_{1,N_{heads}} * Z_{N_{heads}}$. Where $\alpha_{i,j}, i, j \in 1, \ldots, N_{heads}$ are Alphas matrix's trainable parameters.

We finalise by passing our new Z' to the same output projection layer.

Initialisation
The Alphas matrix is initialised to the Identity matrix so each head starts as it's own and not as a mixture. We also freeze matrix's parameters until a certain percentage P of training is over, the freezing is done in order to allow heads to gather certain unique properties before exposing it to mixtures. This way heads can provide each other prior trained information while Alphas matrix can compose more ”educated” mixtures.

Stepwise Nuclear-Norm Growth Loss
Brief reminder, Voita et al. (2019) and Michel et al. (2019), showed that many heads can be redundant when it comes to multiple heads within the same layer.

A naive implementation of the above Alphas component surprisingly shows a similar thing.

The model rather highlighting couple of dominant heads on the diagonal, to wit in each layer, and sets all other heads in the same layer with relatively smaller Alpha values. Meaning it almost neglects the majority of the heads while not accessing information from different heads nor producing any kind of head mixtures.

This can be viewed by the resulted Alphas matrix which is diagonal 3, pretty sparse and has only couple of significant cells. To sum it up we produce head “picking” instead of “mixtures” and as far as it concerns to evaluation results, improvement is not reached and the results stays almost intact.

Since the entire initial goal of the Alphas approach was to share information and compose mixture - not picking, we introduce a second component which helps the model accomplish that by regularizing Alphas matrix.

Our need is for a regularizer that would not only encourage heads mixtures, but also preserve the unique contribution of each head. We wouldn't want all of the head combinations to be pretty

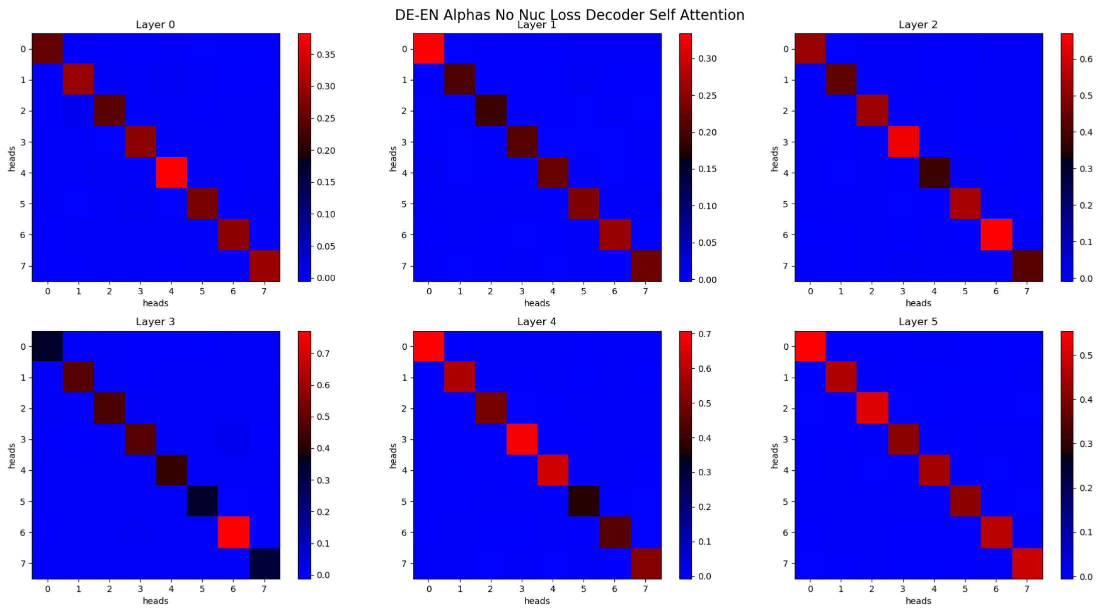

Figure 3: Alphas matrices trained without Nuclear Loss - Decoder Self Attention

similar or converge into a couple of dominant heads.

Looking at the Alphas matrix, it is actually a coefficient matrix where each row represents the alpha coefficients for the applied linear combinations of the heads. For achieving head mixture we want these coefficients to be dominant i.e not close to zero and not concentrated only on the diagonal, we also want the coefficients vectors of the heads i.e Alphas matrix's rows to be almost linearly independent. This will allow us to preserve the uniqueness property that we wanted for the heads.

Luckily we show that using an adaptation of the Nuclear Norm for matrices can provide us just that. Since the Nuclear Norm of matrix (A) is the convex envelope of the rank (A) function, Many researches have traditionally used it for finding low ranked solutions of convex optimisation problems (Mishra et al., 2013), and it has been a very efficient tool for rank approximation and other purposes. Having said that, we remind that our need is for the Alphas coefficients vectors to be linearly independent i.e we need the Alphas coefficient matrix to actually be high ranked. For accomplishing that we exploit the above property of the Nuclear Norm but we do so opposite to the common direction. We are using the Nuclear Norm with the intention of searching for a high-ranked Alphas matrix.

Instead of naively taking the negative value of Alphas matrix's Nuclear Norm as an additional loss, we introduce a more delicate and adaptive approach for encouraging growth of Alphas matrix's Nuclear Norm. We use a stepwise loss which encourages a relative growth of all Alphas matrices's Nuclear Norms at each step. The growth regularizer loss is calculated in the following way:

$$\mathcal{L}_{nuc} = N_{nuc}(\alpha_{matrix}, \Theta_0) + \Delta_r - N_{nuc}(\alpha_{matrix}, \Theta) \tag{7}$$

Where Θ_0 is the former not updating parameters of Alphas matrix, and Θ is the current updating parameter of Alphas matrix. $\mathcal{L}_{CE}$ is the cross entropy loss and N_{nuc} is the Nuclear Norm. That way we simply encourage each Alphas matrix to grow it's Nuclear Norm by a Δ_r relatively to itself. The radius is an hyperparameter so as γ. γ is set to zero until the training finishes a certain P percentage which is also an hyper-parmter same as freezing P mentioned above.

The total objective function is obtained by :

$$\mathcal{L} = \mathcal{L}_{CE} + \gamma * \mathcal{L}_{nuc} \tag{8}$$

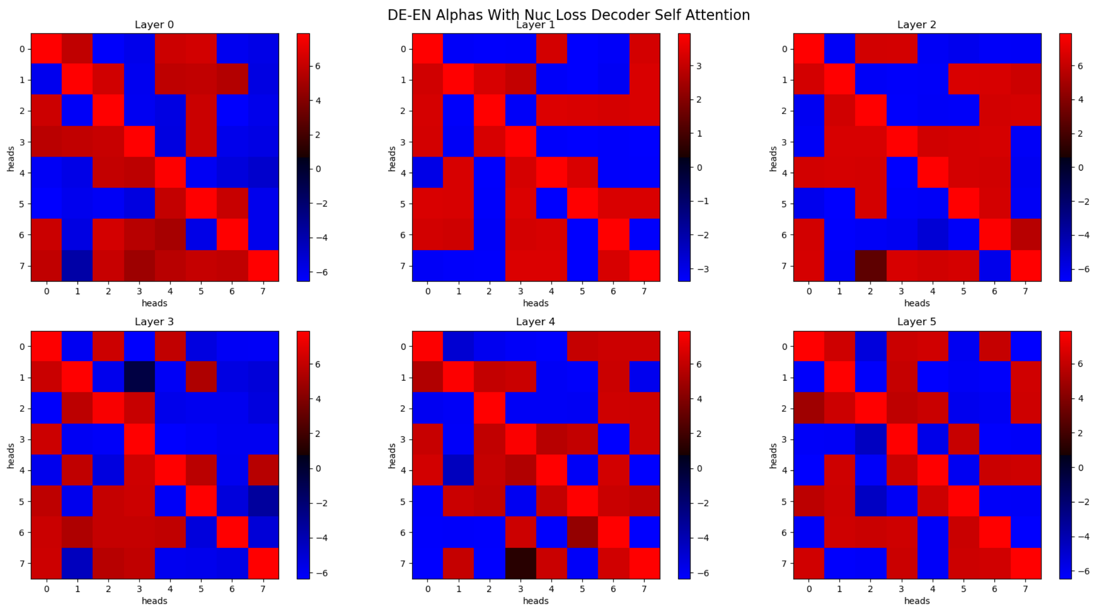

Figure 4: Alphas matrix with Nuclear Loss - Decoder Self Attention. The heads are certainly composed from a combination of neighbouring heads values, that is while the coefficient matrix itself stays quite high ranked. In-fact, a nice observation is that the final Alphas matrices are almost skew-symmetric, neglecting the diagonal which is not zero and up to other some value differences. We couldn't find a direct explanation for that, but we rather think of it as if the different heads are trying to do choose "oppositely" to their neighbouring heads, still not canceling each other.

4 Experiments

4.1 Datasets

We used the Ted based IWSLT'14 (Cettolo et al., 2014) dataset. Translation models were trained using both ES-EN and DE-EN variants 1. We've performed sub-word tokenization, then encoded the data using Byte-pair encoding (BPE).

Data	Train	Test	Val
IWSLT'14 DE-EN	160K	7K	7K
IWSLT'14 ES-EN	169K	6K	7K

Table 1: IWSLT14 Datasetes Statistics

4.2 Models

We have performed three main experiments on both datasets:

- Manually swapping all encoder-decoder attention heads of layers 1 and 5. Addressing it as Encoder-Attention Multi-Layer Swap model.
- Dynamically swapping via soft, hard, and random methods. Using both AVG_{L_2} and Confidence metrics - results are quite close so we show it only for AVG_{L_2}.
- Using the alpha matrix component with different dimensions (according to N_{heads}). Several hyper-parameters were tested - γ coefficient, starting percantge P, growth radius Δ_r and using Alphas component on all or part of the Multi-Head Attentions.

All experiments results can be found in section 5.1.

4.3 Setup

We based our experiments on Fairseq's infrastructure (Ott et al., 2019). We used a Transformer model with 6 layers of both encoder and decoder, whereas $d_{model} = 512$ and $d_v = d_k = 128$ were chosen. We choose Adam as optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9$, $\beta_2 = 0.98$ and $\epsilon = 1e^{-8}$. Training and evaluation were made with a single Nvidia Titan Xp GPU with maximal batch size of 128.

4.4 Evaluation Metrics

The models were evaluated using the BLEU metric (Papineni et al., 2002). A beam search with beam size of 5 was also used for inference.

Layers	Heads	DE-EN	#Params	ES-EN	#Params
6	8	34.51	39469056	40.97	39387136
6	16	34.16	39469056	40.63	39387136

Table 2: Baselines BLEU Scores

Layers	Heads	DE-EN	#Params	ES-EN	#Params
6	8	**35.04**	39470352	**41.21**	39388432
6	16	**34.41**	39469056	**40.93**	39473952

Table 3: BLEU Scores of Alphas models

-	DE-EN	ES-EN
Baseline	34.51	40.97
After Swapping	**34.74**	**41.35**

Table 4: BLEU Score of Manual Swapping Layers 1 and 5 Experiment, 8 Heads Configuration

soft	hard	random	baseline
34.76	**34.66**	34.35	34.51

Table 5: Dynamic Approach DE-EN, 8 Heads configuration

5 Analysis

5.1 Results

We can see that all of our methods leads to improvements in overall BLEU scores, each in it's own way. It's worth pointing out that the most significant overall improvement is deduced by the Alphas component (up to **0.53** BLEU), as shown in table 3. We find the Alphas component pretty useful and efficient addition. It is most certainly resulting improvements to both of our tested sets, while adding a very small amount of parameters, almost neglectable comparing to the total number of parameters in the base architecture. Manual swaps have also gained a nice improvement of **0.38** BLEU on the ES-EN dataset.

And the dynamic approach provided the highest rotation growth on the DE-EN baseline: **0.25** BLEU5.

5.2 Is head importance distribution more balanced now?

In order to validate whether our methods not only improve overall results, but also do it by somehow changing the utilization of Multi-Head Attentions, we ran a simple comparison test. Using the naive Confidence heuristic 5, we calculate the confidence of: Alphas model, Encoder-Attention Swap model and of course the Vanilla setup as a baseline. As visualised in figure 5 each of our alterations reacts as expected. Taking the Alphas model's Decoder Self-Attention for example, it is clearly seen that attention head's confidence is generally Lower, but also much more balanced between different heads comparing to baseline. Same goes with the Encoder-Attention Swap model. This is just a naive heuristic test for general perceiving. But yet it somehow demonstrate a change in the way that our model utilises different heads.

5.3 Is Linear Head Mixing equivalent to encouraging Head discrepancy?

Li et al. (2018) claimed in their research that disagreement regularizations applied to attention heads can lead to improvements in overall results. They've demonstrated this claim simply by encouraging discrepancy between every two attention heads within the same layer, using the cosine distance as their metric. Analysing the improvements of our Alphas component, we wanted to test weather the growth came mostly from mixing the heads, or perhaps from the fact that during the process of regularising the Alphas coefficients, we were indirectly causing the heads to become further apart. Say that our model is causing significant growth in the discrepancy between the heads - comparing to the Vanilla setup, it is fair to assume that we were simply reproducing Li et al. (2018) findings indirectly through our regularisation method. In order to review this issue, we perform the following steps:

- Implement the pairwise cosine similarly loss shown at Li et al. (2018) in our environment

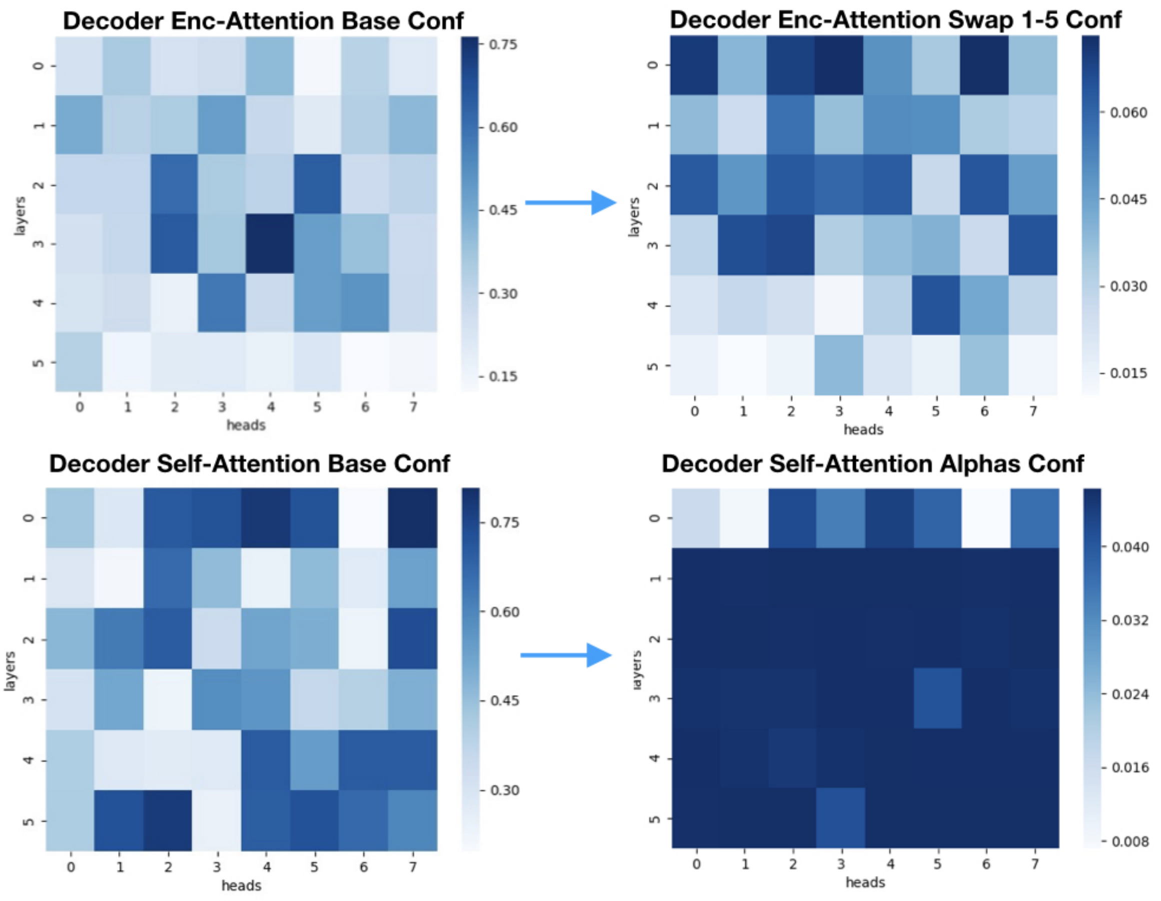

Figure 5: Confidence Comparison

and test their model comparing to our Vanilla setup.

- Calculate the total discrepancy between the various heads across the Alphas model and our version of Li et al. (2018) model then comparing it to the vanilla setup one.

  This was done in the following manner: first, we've calculated and saved for each layer a pairwise distance matrix of size N_{heads} X N_{heads} containing the distance between every two heads. The distance matrices are calculated based on two metrics, cosine similarity and l2 Euclidean distance. This way we test both direction and l2 length, also, checking only cosine similarity would be kind of biased since Li et al. (2018) loss's directly minimise it. We add 1.0 to each cell of the cosine similarity pairwise distance matrix to re-scale it to a positive range. After calculating the distances, we use two measures to define the size of total discrepancy between all heads. First is the $L_{1,1}$ entry-wise norm (L_1 of Vec(pairwise-matrix)) and the second is the average distance of a single head from every other heads (close to what we do in equation 6). Both provide pretty much the same results, so we wont address them separately.

- Next we use the total discrepancy value to measure the percentage of growth/decay in discrepancy of the heads per layer comparing to the Vanilla setup. Meaning we calculate how much our alphas model changed the total discrepancy between the heads comparing to the Vanilla baseline. We do so with cosine similarity loss model inspired by Li et al. (2018) as well.
  The results are quite surprising. Not only that our model doesn't enlarge the discrepancy between the heads measured by cosine difference, it is in fact bringing them closer in terms of L2 norm (see fig 6 and fig7).
  The Cos-Sim loss model which we reproduced delivered growth in both Cosine-Difference and Euclidean distance as expected, also it's

overall BLEU results did not improve the Vanilla setup.

- This hints in our opinion that the Alphas and regularization components are not contributing through indirect enlargement of distances between the heads. Rather the linear mixture leads to performance improvements through utilizing the importance of each head - and not necessarily by ”tearing them apart”.

References

Mauro Cettolo, Jan Niehues, Sebastian Stüker, Luisa Bentivogli, and Marcello Federico. 2014. Report on the 11th iwslt evaluation campaign, iwslt 2014. In *Proceedings of the International Workshop on Spoken Language Translation, Hanoi, Vietnam*, volume 57.

Angela Fan, Edouard Grave, and Armand Joulin. 2019. Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jian Li, Zhaopeng Tu, Baosong Yang, Michael R Lyu, and Tong Zhang. 2018. Multi-head attention with disagreement regularization. *arXiv preprint arXiv:1810.10183*.

Minh-Thang Luong, Hieu Pham, and Christopher D Manning. 2015. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Paul Michel, Omer Levy, and Graham Neubig. 2019. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, pages 14014–14024.

Bamdev Mishra, Gilles Meyer, Francis Bach, and Rodolphe Sepulchre. 2013. Low-rank optimization with trace norm penalty. *SIAM Journal on Optimization*, 23(4):2124–2149.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

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

Hao Peng, Roy Schwartz, Dianqi Li, and Noah A Smith. 2020. A mixture of $h-1$ heads is better than h heads. *arXiv preprint arXiv:2005.06537*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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. *arXiv preprint arXiv:1905.09418*.

Wangchunshu Zhou, Tao Ge, Ke Xu, Furu Wei, and Ming Zhou. 2020. Scheduled drophead: A regularization method for transformer models. *arXiv preprint arXiv:2004.13342*.

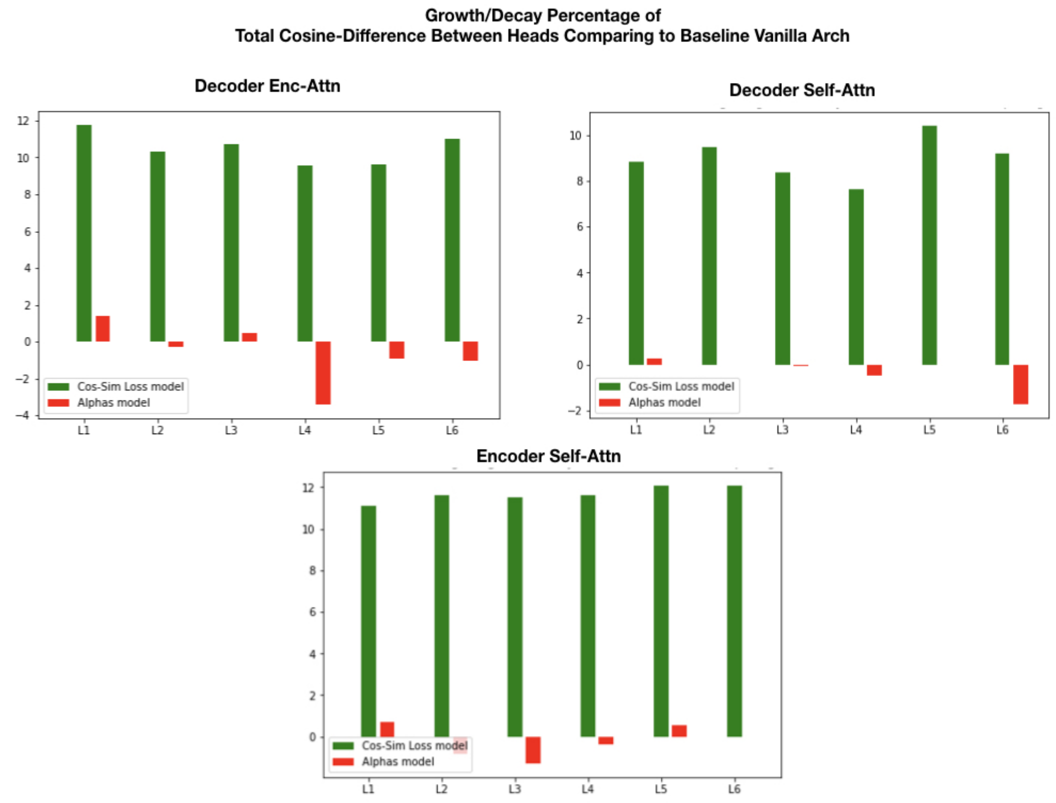

Figure 6: Percentage of growth/decay in Cosine-Difference comparing to Vanilla setup

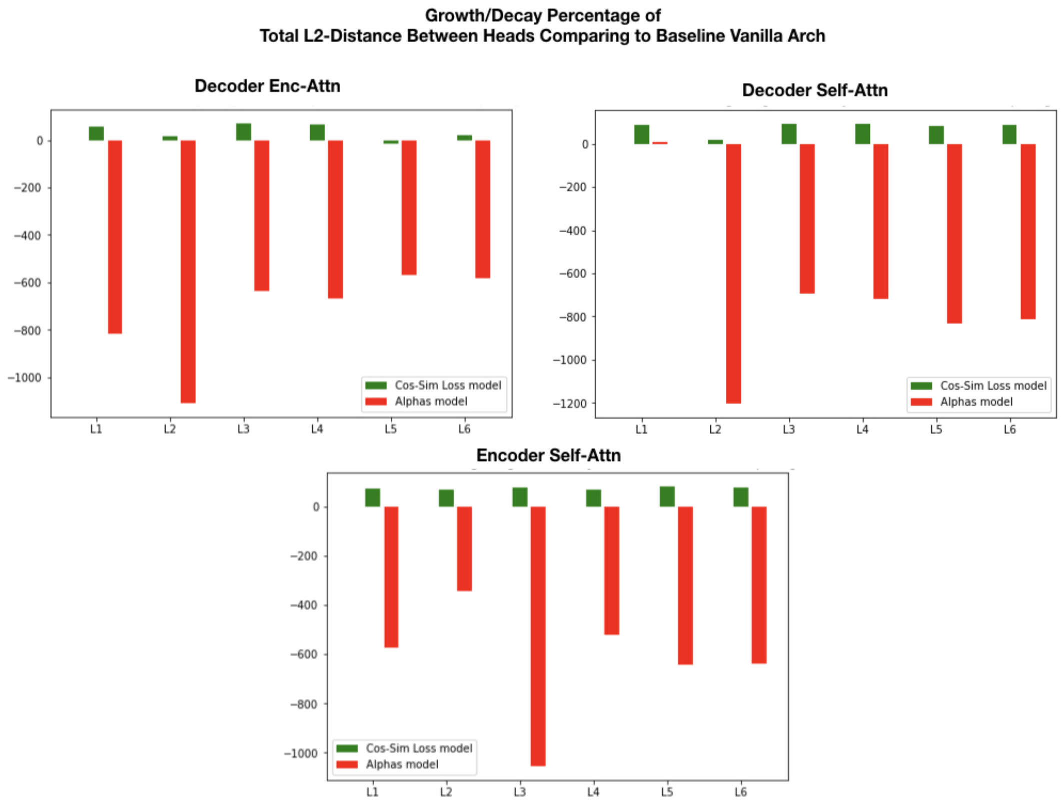

Figure 7: Percentage of growth/decay in L2 Difference comparing to Vanilla setup