Diversified Attention on Transformers

Michael Xiong mjx2000@columbia.edu
Industrial Engineering and Operations Research
MS in Financial Engineering
Columbia University

Daniel Loredo dl3476@colubia.edu
Industrial Engineering and Operations Research
MS in Operations Research
Columbia University

**Equal contribution on this work. Both developed the necessary code for the empirical experiments and both worked on the presented paper*

1. Introduction

The Transformer Model is one of the most revolutionary ideas from the last decade in the Deep Learning environment (Vaswani et al., 2017). By relying on attention mechanism, the Transformer obtains SOTA results in record time due to its parallelizable structure. Transformers are being widely used, from tasks such as music generation (Huang et al., 2018) to generation of protein sequences (Rives et al., 2021). Transformers have proven worthy of some of the most frequent areas of machine learning. In natural language processing different authors have used Transformers to study large scale language structures (Luo et al. (2021); Devlin et al. (2018); Child et al. (2019); Chen et al. (2018)). They are also being exploited on images (Lu et al. (2019); Tan and Bansal (2019)), where their results compete against those of some of the most used neural network models.

Although Transformers can perform a task that usually takes 100 days in just one day, their space requirements increase significantly per each extra token that is added to the sequence. For this reason, many innovative architectures and techniques have been proposed to reduce the space requirements of Transformers. Kitaev et al. (2020) improve Transformers efficiency by replacing the dot-product attention with locality-sensitive hashing techniques, reducing the space requirements for L tokens

from $O(L^2)$ to $O(L * d * log(L))$, where d is the dimension of the queries, keys and values representation. Going even further, Choromanski et al. (2020) bring to the scene the most efficient architecture yet, The Performer, which uses FAVOR+ to approximate softmax attention kernels with space requirement $O(L * d * log(d))$. This efficient attention algorithm has been successfully applied to the paradigm of reinforcement learning (Choromanski et al., 2021).

Parallel computation occurs inside the Multi-Head Attention mechanism, which are found inside both, the encoder layers and the decoder layers. However, Transformers can suffer from learned redundancies materialized via different attention heads learning similar patterns. Such redundancies can be pruned to make a more parsimonious model, or the computational resources should be directed to train a better model. Only some research has focused on the attention head redundancies that could arise inside the Transformer architecture. Recently, Bhojanapalli et al. (2021) use eigen analysis of attention matrices to show that the diversity of the attention scores can be captured in a low dimension eigenspace and that these eigenspaces are redundant through the layers of the transformer architecture, giving evidence on head redundancies. However, not many proposals have appeared in literature to address this issue. In this work, we focus on improving the Transformer architecture by pushing for diversity between heads inside the Multi-Head structure. We empirically show that by implementing a regularization parameter that addresses low entropy in attention matrix row distributions, we can reduce redundancy between attention heads. We use the original Transformer architecture but the approach we take can be implemented with any attention matrix algorithm.

2. Background

Transformers are neural network architectures that leverage on the regular scaled dot product attention (Vaswani et al., 2017). The model takes an input sequence x of length L and then performs the mapping of the input into the matrices of queries, keys and values, Q,K,V, where the queries and keys mapping have a dimension d_k and the values matrix has a dimension of d_v. By using this mapping, the model takes the dot product of the query and keys matrices and normalizes the result with the square root of their dimension. Finally, the softmax is computed and is linearly combined with the value matrix. In essence we have the following:

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

The Transformer architecture is comprised by two different stacks, the encoder and the decoder. The encoder stack takes an input embedding sequence after a positional encoding is added to it. The encoder contains two sub-layers, a Multi-Head Attention mechanism, and a classic fully connected layer. A residual connection is used in both sub-layers as well as a layer normalization. The encoder contains a custom number of stacks (N) of these two sub-layers. The Multi-Head Attention sub-layer allows the model to learn in parallel different mappings of queries, keys and values, and in consequence different attention matrices, which makes the transformers time complexity so groundbreaking.

$$MultiHeadAtt = Concat(Attention(Q_1, K_1, V_1), ..., Attention(Q_h, K_h, V_h))W \tag{2}$$

On the other hand, the decoder its a little more complicated. It contains three sub-layers, a Masked Multi-Head Attention mechanism, a Multi-Head Attention mechanism and a classic fully connected layer. The Masked Multi-Head takes the embedded and positional encoded output sequence of tokens but prevents a token from attending to future tokens (the prediction can not rely on future predictions). The Multi-Head attention sub-layer its the same as the one we can find in the encoder but with a slight twist, the values and keys mappings that enter the attention mechanism come directly from the encoder output, while the queries come from the output of the previous sub-layer in the decoder. The decoder can also be customized with as many stacks as needed.

3. Diversity on Attention Matrices

Transformers brought unquestionable improvement on the amount of time required for training on sequential data. However, a wide amount of research has focused on improving even further the computational cost of the attention layer. The main results on this trend of research have reduced the amount of memory needed to compute an iteration of the attention matrix (Choromanski et al., 2020).

But transformers, just as any other large neural network model, might suffer from redundancies in the resulting parameters after training. These redundancies could appear mainly in the Multi-Head Attention sub-layer, where heads are trained in parallel.

Bhojanapalli et al. (2021) show empirically that these redundancies in fact happen during training. They use eigen-value decomposition analysis on the variation space of the attention matrices. In other words, they estimate the covariance matrix between attention heads in a layer and the covariance matrix between the attention heads in different layers and perform a spectral analysis on the covariance spaces. Their results show that such spaces are low-dimensional and that they significantly overlap between each other in both cases, different layers, and same layer.

The strategy that we propose is to impose a high level of relative entropy between attention heads on each iteration of training, with the final objective of avoiding redundancies while the model its learning. With this purpose in mind, we leverage on the Bhattacharyya coefficient, a measure of the amount of overlap between distributions.

$$BC(attention_i^{h_1}, attention_j^{h_2}) = \sum_{i=j} \sqrt{p_i^{h_1} p_j^{h_2}} \tag{3}$$

In (3) p_i^h is the distribution obtained on the attention matrix inside head h for token i after the normalization of the dot product on (1).

The Bhattacharyya coefficient lies in the interval $0 \leq BC \leq 1$, where the closer to 0, the heads are attending to different tokens and more diverse. On the other hand, the closer to 1, the greater the overlap that exists between the distributions of the attention matrix rows. It is important to mention that, while the Bhattacharyya coefficient is a measure, it does not follow the triangle inequality. In consequence, we cannot compare between coefficients obtained between different pairs of distributions.

However, it is sufficient for our purposes of reducing the coefficient of each pair with respect to its own initial level.

We use the mean of the Bhattacharyya coefficient on sequence length, number of heads (N) and batchsize, as a diversity penalty. By using the mean we ensure that the resulting penalty is invariant to the magnitude of these parameters. We scale this penalty by a regularization parameter λ and then add it to the loss function that we are trying to minimize, in our case the loss function is the cross entropy loss.

$$loss = CrossEntropyLoss + \lambda * DiversityPenalty \tag{4}$$

With this regularization penalty we force the model to learn a more diversified set of attention matrices with each training step.

We present in Algorithm 1 with detail the full Diversity Penalty implementation through the training of our Transformer model.

Algorithm 1 Calculate Diversity Penalty

diversity_penalty = 0

for $sequence = 1, 2, \ldots, Batchsize$ **do**
  for $layer = 1, 2, \ldots, num\_encoder\_layers$ **do**
    for $head1, head2$ in Combinations(num_heads, 2) **do**
      $temp = head1 \odot head2$ ▷ Hadamard (element-wise) product
      $temp = \sqrt{temp}$ ▷ Square root each entry
      $temp = \text{RowSum}(temp)$ ▷ Sum up each row in the matrix
      $temp = \text{mean}(temp)$ ▷ Average the entries
      $temp = temp$ / $\binom{num\_heads}{2}$ ▷ Divide by number of comparisons
      diversity_penalty += $temp$
    end for
  end for
end for

return diversity_penalty / Batchsize

4. Experiments

We implement our regularization scheme over a pre-existing Transformer implementation on PyTorch. Specifically, we replace the Transformer encoder module with one that allows us to compute the diversity penalty. In our experiments, we only enforce diversified attention on the Multi-Head Attention layer in the encoder stack. We considered that depending on the results of the encoder implementation we could go ahead in the future and modify the decoder module too.

We ran our experiments on the Multi30k data sets, we use the English and Deutsche dictionaries. The model its trained to translate from Deutsche to English. As a baseline, we train a Transformer model with no regularization penalty in order to compare our results on diversity.

In terms of our training framework, we use a batch size of 32 sequences of tokens, an embedding size of 512 and a learning rate of .0003. We implement an encoder layer with 3 stacks, each with 4 heads in the Multi-Head Attention mechanism, the decoder is also composed of 4 attention heads per each of 3 stacks. The dropout parameter is of 0.10. We use Adam optimizer with objective function (4) and a regularization parameter $\lambda = 1.0$. Finally, we perform 30 epochs of training. The selected size of the data sets and the training hyper-parameters are consequence of the computational power available to us. It's an important distinction given that the results might be improved considerably if trained on a much larger transformer.

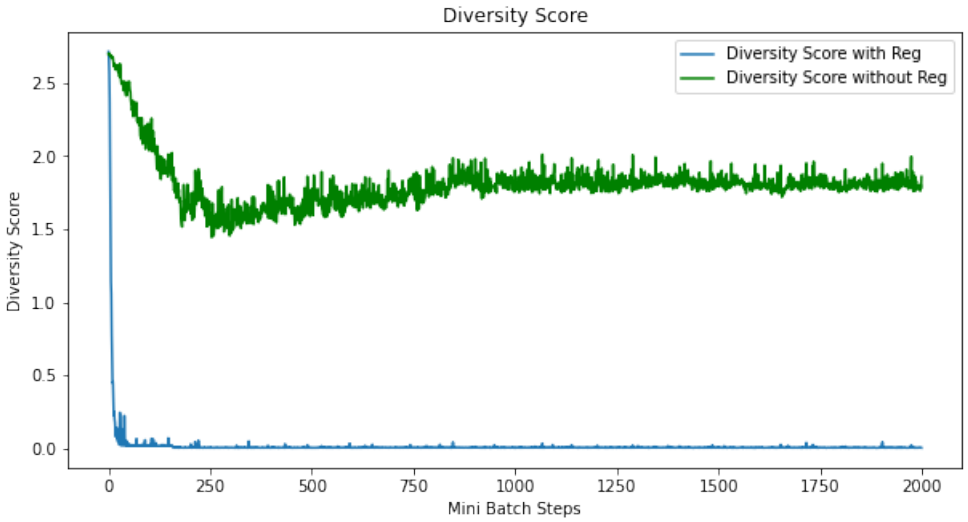

Figure 1: Diversity score contrast

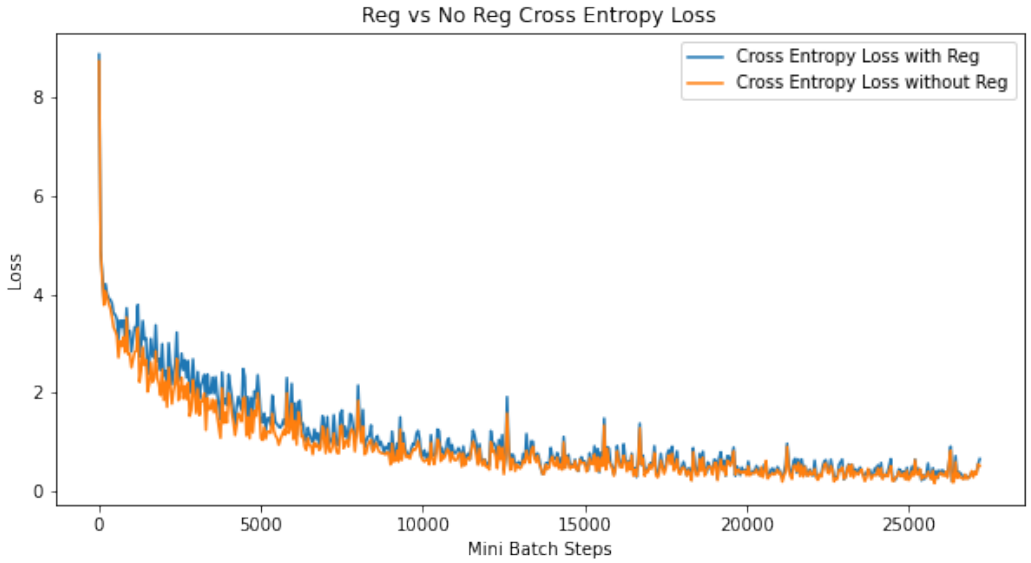

Figure 2: Cross entropy loss contrast

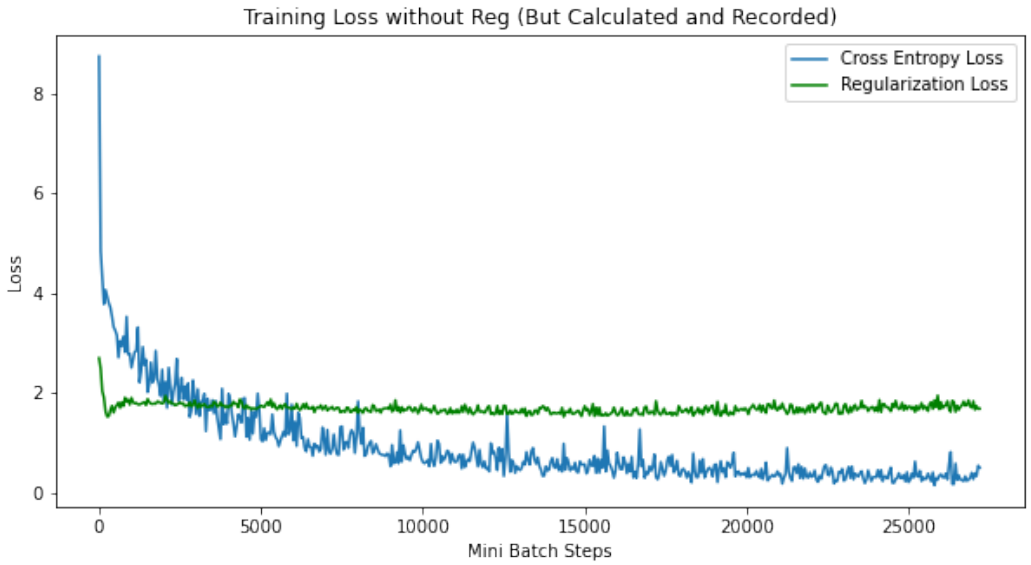

Figure 3: Total loss on non-regularized model

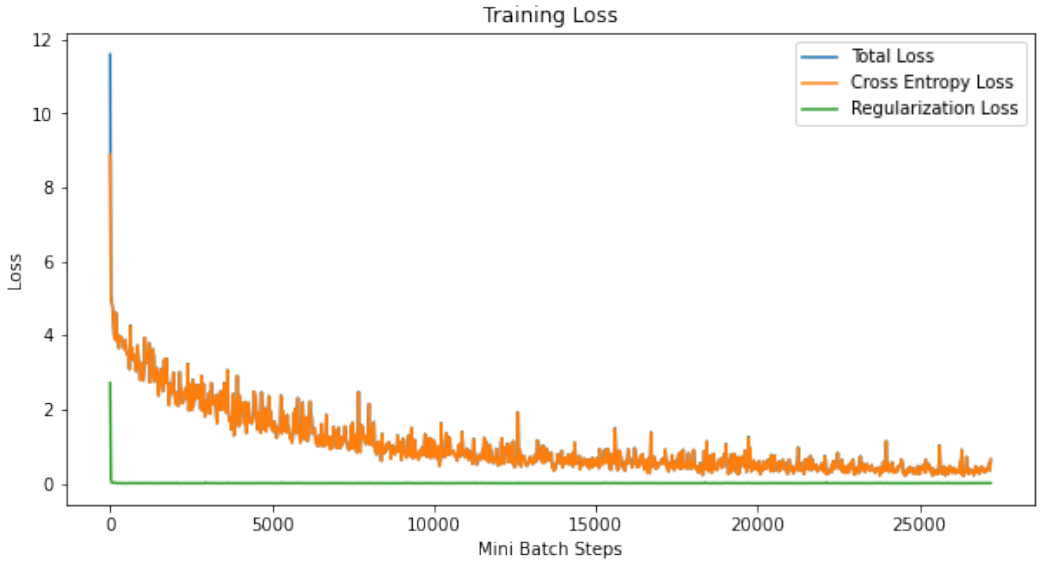

Figure 4: Total loss on regularized model

Our results indicate that we clearly achieve a diversified Multi-Head attention layer. In just a few batch steps of training we achieve to reduce almost completely our diversity penalty in the regularized model, while in the benchmark transformer the diversity penalty, estimated but not implemented on the loss objective, only decreases a small quantity by itself and then fluctuates on a specific level (Figure 1).

Figure 2 shows that increasing diversity with the regularization penalty does not affect the rate at which the model learns from the cross-entropy loss. In both models, the cross entropy loss decreases through the batch steps until it reaches the same level. Only on the first steps the regularized model has a higher cross-entropy loss, which is congruent with the behavior on the diversified penalty on Figure 1.

On Figure 3 we compare the patterns inside the non-regularized model. The cross-entropy loss reduces constantly until it reaches its asymptotic behavior. In contrast, the diversity penalty its not lowered automatically during training. This shows that diversity between the attention matrices is not a consequence of regular architecture.

On the other hand, in Figure 4 we can see that inside the regularized model we have an immediate increase on diversity thanks to the penalty implementation and that this does not negatively affect the learning rate with the cross-entropy loss.

However, even though our model clearly increases diversity between the attention heads it does not increases model accuracy on translation. The regularized model

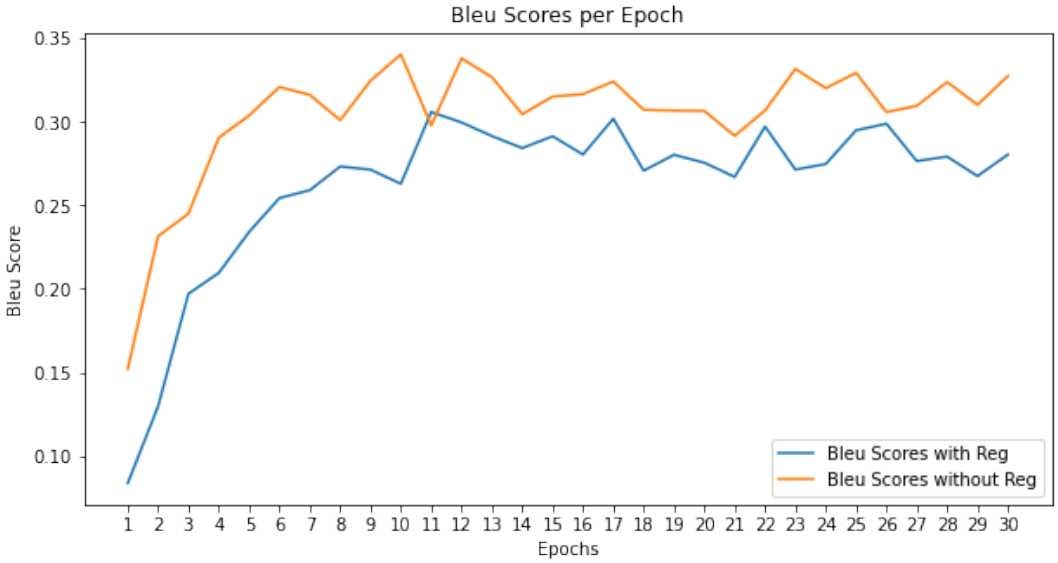

Figure 5: Bleu score contrast

obtains a lower Bleu score during the hole training process. The Bleu score of the regularized model follows the pattern of the benchmark Transformer but never beats it (Figure 5).

5. Conclusion and Future Work

In this work we used the Battacharyya coefficient to promote diversity between the attention matrices inside the Multi-Head Attention Mechanism of the encoder stack in a Transformer model. We added a penalty to the objective function during training that corresponds to the BC coefficient between the rows of the different attention heads. With this strategy we achieved to reduce the redundancies between the attention parameters learned by the model.

However, our experiments show that the diversified Transformer obtains a lower Bleu score than the traditional Transformer. Although this could be discouraging on our line of research, it's important to notice that these results might be affected by the architecture of our model, the size of the used data and the chosen hyper-parameters, all of them consequence of the resources available to us during the experiments. Currently, our model is conformed only by 4 heads per each sub-layer of the encoder. If we implemented the diversity penalty in a larger architecture with known redundancies, we could see better results.

Another possible improvement for future research is the decoder Multi-Head Attention diversification. It's possible that some of the gains in diversity inside the encoder stack are offset by the decoder stack. If better results are possible with this kind of strategy, then the following steps would be to maximize efficiency in our code and try to reduce the time complexity of the Transformer.

Finally, we could take this scheme and try to implement it in other attention-based algorithms, such as performers and reformers.

References

Srinadh Bhojanapalli, Ayan Chakrabarti, Himanshu Jain, Sanjiv Kumar, Michal Lukasik, and Andreas Veit. Eigen analysis of self-attention and its reconstruction from partial computation. *arXiv preprint arXiv:2106.08823*, 2021.

Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Niki Parmar, Mike Schuster, Zhifeng Chen, et al. The best of both worlds: Combining recent advances in neural machine translation. *arXiv preprint arXiv:1804.09849*, 2018.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. *arXiv preprint arXiv:1904.10509*, 2019.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. *arXiv preprint arXiv:2009.14794*, 2020.

Krzysztof Choromanski, Deepali Jain, Jack Parker-Holder, Xingyou Song, Valerii Likhosherstov, Anirban Santara, Aldo Pacchiano, Yunhao Tang, and Adrian Weller. Unlocking pixels for reinforcement learning via implicit attention. *arXiv preprint arXiv:2102.04353*, 2021.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

Nikita Kitaev, Łukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. *arXiv preprint arXiv:2001.04451*, 2020.

Jiasen Lu, Dhruv Batra, Devi Parikh, and Stefan Lee. Vilbert: Pretraining task-agnostic visiolinguistic representations for vision-and-language tasks. *arXiv preprint arXiv:1908.02265*, 2019.

Haoneng Luo, Shiliang Zhang, Ming Lei, and Lei Xie. Simplified self-attention for transformer-based end-to-end speech recognition. In *2021 IEEE Spoken Language Technology Workshop (SLT)*, pages 75–81. IEEE, 2021.

Alexander Rives, Joshua Meier, Tom Sercu, Siddharth Goyal, Zeming Lin, Jason Liu, Demi Guo, Myle Ott, C Lawrence Zitnick, Jerry Ma, et al. Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences. *Proceedings of the National Academy of Sciences*, 118(15), 2021.

Hao Tan and Mohit Bansal. Lxmert: Learning cross-modality encoder representations from transformers. *arXiv preprint arXiv:1908.07490*, 2019.

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