Less is More!
A slim architecture, optimal for language tasks

Luca Herranz-Celotti[1] **Ermal Rrapaj**[2]
[1]NECOTIS, Université de Sherbrooke, Canada
[2] NERSC, Lawrence Berkeley National Laboratory, Berkeley, California 94720, USA
luca.celotti@usherbrooke.ca ermalrrapaj@lbl.gov

Abstract

The softmax attention has emerged as a noteworthy development in the field of Deep Learning, building on the successes of Transformer-based architectures. Their ever increasing sizes need increasing computational memory, that limits their usage. We propose QgV, a sigmoid gate that significantly boosts performance without increasing architecture size. We also leverage Tensor Chains to identify and prune the excess parameters. We find that such excess resides primarily within the embedding layer, and not in the output linear layer. To further improve performance and reduce parameters, we introduce H-SoftPOS, a hierarchical embedding layer. Remarkably, on the WMT14 English-German validation set, our approach yields a threefold reduction in perplexity, surpassing the current state-of-the-art, while reducing parameter counts also by a factor of 3. When we further reduce the number of parameters up to sevenfold, we can still achieve a 21% decrease in perplexity with respect to the baseline Transformer. To test generalization capabilities, we conduct experiments on the 7 language pairs of the WMT17 dataset. Our model, Anthe, outperforms existing techniques in terms of test loss while simultaneously halving the number of parameters. Moreover, we observe a 70 times reduction in variance with respect to the prior state-of-the-art. We make the code publicly available[1]. In conclusion, our proposed method yields significant improvements in performance at lower memory cost.

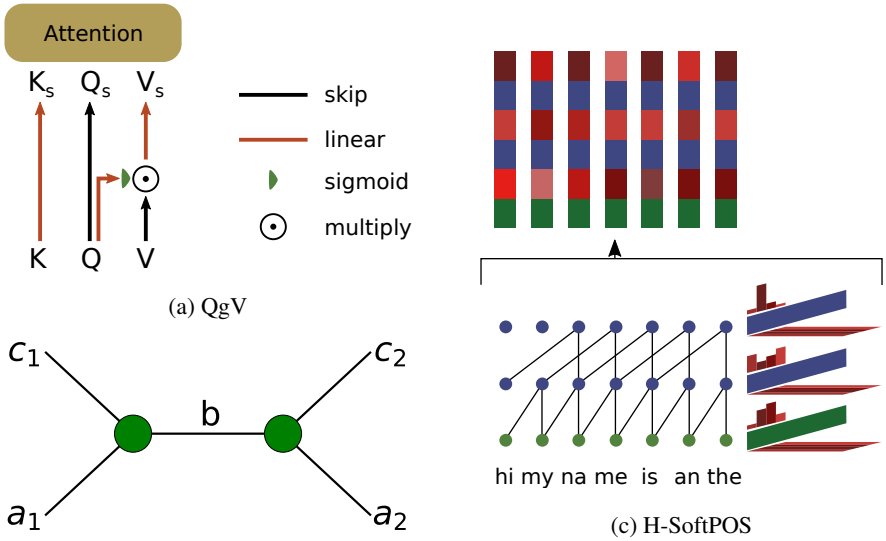
1 Introduction

The Transformer [1] has led to major breakthroughs in Artificial Intelligence on a wide range of tasks, such as language modeling [2], translation [1], speech recognition [3], and protein folding [4]. These architectures have become increasingly wider [2] and deeper [5], leading to a massive increase in the number of parameters. ChatGPT-3 has 175 billion parameters [6, 2], surpassing previous models by orders of magnitude. To address their computational demands, especially in handling long sequences, researchers have proposed approximate attention mechanisms, such as sparse-approximation [7, 8], low-rank approximation [9, 10, 11], their combination [12, 13, 14], and I/O optimization techniques for additional speed-up [15]. However, it often seems that a reduction in parameters leads to degraded performance [16].

In this article we propose a novel gating mechanism before the softmax attention that significantly improves performance. Additionally, we show that removing weight-sharing between the output projection and the embeddings can also improve performance, with a parameter increase of 43%. To mitigate this increase in parameters without compromising accuracy, we introduce Hierarchical Soft Part of Speech (H-SoftPOS) and Tensor Chain (TC). H-SoftPOS is based on the observation

[1] https://github.com/LuCeHe/anthe_official

ENLSP-III Workshop in the 37th Conference on Neural Information Processing Systems (NeurIPS 2023).



(a) QgV

(b) Tensor Chain (TC)

(c) H-SoftPOS

Figure 1: **QgV, H-SoftPOS and TC.** (a) QgV uses the Query tensor to gate the Value, before feeding KQV to the attention. (b) Length two Tensor Chain (TC) with bond dimension b and external dimensions $N_a = a_1 \cdot a_2$, $N_c = c_1 \cdot c_2$. (c) H-SoftPOS starts with a smaller matrix embedding (green) that is concatenated with hierarchical convolutions (blue) and their SoftPOS (red).

that language elements, such as sub-words, words or sentences, can have a limited set of context dependent roles. Hence, we assign a learnable Part of Speech (SoftPOS) to each language element, improving performance while decreasing the embedding parameter count. TC represents a matrix as a tensor product contraction, drastically reducing the overall parameter count. Originally proposed in physics to characterize the short-range entanglement in one-dimensional quantum systems [17, 18], it has since found many other applications [19]. We name the resulting architecture the *Anthe* for ***G**ates, a**n**d **T**C and **H**ierarchical SoftPOS for Att**e**ntion*. Our contributions are:

- we introduce the QgV, a gate between Values and Keys in the attention, Sec. 2;
- we introduce H-SoftPOS to reduce embedding parameters without loss in performance by accounting for the limited roles language elements can play in speech, Sec. 2.1;
- we introduce the TC to represent any matrix as a product of small tensors to drastically reduce the amount of trainable parameters in Sec. 2.1;
- we report improvements of *Anthe* over Transformer first on English-German language translation, and then on other seven language pairs in Sec. 3.

2 Improving performance through gating

Gating mechanisms have been widely used to avoid gradient explosion [20]. The LSTM, GRU, Neural Turing Machine, Differentiable Neural Computer, and Mogrifier LSTM are well-known examples of models with gates [20, 21, 22, 23, 24]. In the Transformer architecture, Queries, Keys and Values, are mapped linearly before the softmax attention. In this work we refer to the weights $\{W_Q, W_K, W_V\}$ as pre-attention, or *patt*. We propose a novel gating mechanism,

$$\begin{aligned} V_s &= W_V V \sigma(W_Q Q),\ Q_s = Q,\ K_s = W_K K, \\ \text{Attention}(Q_s, K_s, V_s) &= \text{softmax}\Big(\frac{Q_s K_s^T}{\sqrt{d_{model}/d_h}}\Big) V_s \end{aligned} \tag{1}$$

where σ is the sigmoid, d_{model} the width and d_h the heads in the attention. QgV does not change parameter count and improved performance more than the alternative combinations, see Tab. 3. In addition, we find that having independent weights for the embeddings and output projection significantly improves performance. We also replace the feed-forward layer with GEGLU [25, 26, 27], as we observe a small but statistically significant improvement.

2.1 Reducing the number of parameters without compromising accuracy

H-SoftPOS. Maintaining high performance with fewer parameters is especially important for huge architectures like recent Transformer-based models. The idea behind Hierarchical Soft Part of Speech (H-SoftPOS) is that sub-words play limited roles in a word (e.g. prefix, suffix, past tense of a verb, etc.). Similarly, words play limited roles in a sentence (e.g. verb, noun, adjective, etc.), and so on hierarchically. Since the role depends on the context, there is a soft aspect to consider. We start with a small embedding d_{emb} and assign a learnable Part of Speech (SoftPOS) to each subword. We use 1D convolutions to convert the sub-word embedding into word and sentence embeddings, and at each level we assign a SoftPOS. Finally, we concatenate the initial small embedding with the convolutions and their SoftPOS, to have a full embedding representation. The matrix $W_{sp} \in \mathbb{R}^{n_{sp} \times d_{sp}}$ represents n_{sp} POS roles, of size d_{sp}. We repeat the process at l_{sp} hierarchical levels. If S are the input sequence integers, then

$$\begin{aligned} \text{Embedding}_{hsp}(d_{model})(S) &= \text{Concat} \bigcup_{l=1}^{l_{sp}} \Big\{ X_l, \text{SoftPOS}(X_l) \Big\} \\ X_1 &= \text{Embedding}(d_{emb})(S) \\ X_l &= \text{Conv1D}(\text{kernel} = 3, \text{dilation} = 2^l, \text{pad} = \text{causal})(X_{l-1}) \\ \text{SoftPOS}(X_l) &= W_{sp} \text{softmax}(X_l[:n_{sp}]) \end{aligned} \tag{2}$$

where we use $l_{sp} = 2$, $d_{sp} = \lfloor d_{model}/2l_{sp} \rfloor$ and $d_{emb} = d_{model} - (2l_{sp} - 1)d_{sp}$. We use $X[:n_{sp}]$ to denote the first n_{sp} elements of the vector, resulting in an embedding of d_{model} width, with four times fewer parameters and same performance as the original version. The embedding X_1 has a matrix of size $d_{emb} \times n_{vocab}$ and sums a non-learnable cosine positional encoding [1].

Tensor Chain. The Tensor Chain (TC) is a method to represent a linear map as a product and contraction of smaller tensors [28, 29]. It is used in quantum many-body systems for a compact representation of the exponential degrees of freedom in the system[30, 31]. In Deep Learning, [19] replaced the linear layers with TCs in convolutional models without any loss of prediction accuracy. A weight matrix W_{N_a,N_c} of size $\mathbb{R}^{N_a \times N_c}$ can be decomposed as

$$W_{N_a,N_c} = \text{Reshape} \left\{ \text{Tr}_b \left[w^{(1)}_{a_1,b,c_1} \left(\prod_{i=2}^{n-1} w^{(i)}_{a_i,b,b,c_i} \right) w^{(n)}_{a_n,b,c_n} \right], N_a \times N_c \right\}, \tag{3}$$

where, $N_a = \prod_{i=1}^{n} a_i$, $N_b = \prod_{i=1}^{n} b_i$, w_{abc}, w_{abbc} are the weight tensors, and n is the length of the chain. The trace is taken over the index b, the bond. This internal index can vary between consecutive tensors, but we chose it to be the same for simplicity. It can be implemented through the einsum function. For illustration, a TC of length 3 is,

$$\text{einsum}\Big(a_1 b_1 c_1,\ a_2 b_1 b_2 c_2,\ a_3 b_2 c_3 \to a_1 a_2 a_3 c_1 c_2 c_3, \left[w^{(1)}_{a_1,b_1,c_1},\ w^{(2)}_{a_2,b_1,b_2,c_3},\ w^{(3)}_{a_3,b_2,c_3} \right] \Big). \tag{4}$$

As we apply TC to different parts of the network, we use the notation $\text{TC}_{where:r}$ to indicate *where* TC is used, to reduce the parameter count by a ratio r. We use *emb*, for embedding, *ff* for feed-forward or GEGLU, *patt* for pre-attention linear layers, *layer* for *patt* and *ff* simultaneously, and *output* for the last linear layer.The bond parameter is the solution to the equation $b(a_1c_1 + a_nc_n) + b^2 \sum_{i=2}^{n-1} a_i b_i = rN_aN_c$, for a given selection of parameters (a_i, c_i). The external dimensions a_i and c_i are chosen to be close to the n-th root of N_a and N_c. We implement TC during training, and use $n = 2$ unless stated otherwise. If the parameter reduction has minimal effect on loss, then many of the parameters in the original linear layer were of no functional importance.

3 Results

Here we report on our ablation study on the language translation task WMT14 English to German. We trained for a maximum of two days on a single GPU and a batch size of 32, and early stopping on the validation loss with a patience of 10 epochs. Our baseline Transformer has a $d_{model} = 512$ width, $N = 6$ layers, $d_h = 8$ heads, a dropout probability of $p_{dropout} = 0.1$, and a width for the feed-forward layer of $d_{ff} = 4d_{model}$. We used the same tokenizer for all the languages, which is a byte-pair encoding [32] with 32K sub-words, trained on the WikiText-103 dataset [33]. We

used the Adam optimizer [34] with a constant learning rate $lr = 3.16e^{-5}$ which was chosen after a grid search for optimal performance of the baseline Transformer. To account for statistical fluctuations, we report the mean and standard deviation over 4 random seeds. Notice that our Transformer implementation results in 60M parameters because we used 32K sub-words, while the original 37K sub-words for the English-German pair, results in 63M parameters, closer to the 65M reported in [1]. We introduce our innovations sequentially in Tab. 1, on the WMT14 DE/EN validation split. We use

	params	dev PPL
Anthe = B' + QgV + H-SoftPOS + TC$_{ff:.005,\ att:.07}$	**<u>30M</u>**	**3.5674 ± 0.0130**
B' + QgV + H-SoftPOS + TC$_{ff:.1}$	46M	2.7690 ± 0.0048
B' + QgV + H-SoftPOS	**68M**	**<u>1.2627</u> ± 0.0018**
B' + QgV + TC$_{emb:.2}$	**67M**	**1.4146 ± 0.0799**
B' + QgV	93M	1.2642 ± 0.0035
B' = B + GEGLU	93M	2.5665 ± 0.0055
B = Transformer + no-shared embeddings	93M	2.5987 ± 0.0155
Transformer + shared embeddings	60M	3.8245 ± 0.0670
Transformer 512 [1]	65M	4.66
Transformer 1024 [1]	213M	4.33

Table 1: **Anthe on the English-to-German translation development set WMT14.** We use the same hyper-parameters as [1], for $d_{model} = 512$, and we report at the bottom their two best results for $d_{model} = 512, 1024$. Our QgV results in better performance, while H-SoftPOS slightly improves performance while reducing the number of parameters. TC drastically reduces the number of parameters while retaining a better performance than the Transformer. The reduction of parameters with respect to Transformer 1024 is sevenfold, while retaining an improvemed performance.

as a baseline B the Transformer without weight sharing, and B' when we change the feed-forward module by GEGLU in B. Removing shared weights improves performance with an increment of $33M$ parameters, a 43%. Instead, GEGLU and QgV improve performance without an increase in parameters. In experiments not reported in the table, perplexity improves over the Transformer with weight sharing from 3.8245 ± 0.0670 to 3.4310 ± 0.0179 with only the addition of the QgV. H-SoftPOS improves minimally performance but improves significantly parameter count. Finally, TC brings down the number of parameters below the original Transformer with better performance. We note that drastic reductions in parameters through TC eventually degrade performance, and we only report the best combinations. In App. A we study the impact of TC length, various gating alternatives, and search for excess parameters through TC. We find length 2 TC gives the best results, QgV gating provides better scores than other options, and the embedding layer has a high number of excess parameters while the output linear layer does not have any. In App. B we confirm these improvements persist on 7 new language pairs from WMT17, with Anthe decresing perplexity by two, with only half the parameters. Finally, in App. C we also confirm improvements in language modeling tasks, e.g PTB [35] and WK2 [36], with respect to the baseline. Due to the non symmetric connections between encoder and decoder in the language translation architecture, QgV is not equivalent to KgV, but in the decoder only architecture in this appendix, we verify they are equivalent.

4 Discussion and Conclusion

Introducing QgV, a sigmoid gating mechanism, as well as H-SoftPOS, a hierarchical embedding layer, and TC, tensor chain representation, we reduced the parameter count required while enhancing performance over the Transformer as we showed experimentally on WMT14 and WMT17, and for language modeling on PTB and WK2 datasets. We call Anthe the resulting architecture. We found that an excess of parameters existed through the use of H-SoftPOS and TC. In fact, the embedding layer can be significantly pruned without major losses in performance, while the output linear layer needs all its parameters. We also see that the feed-forward layer can be pruned more than the pre-attention linear maps. Surprisingly, our Anthe has more than half of all its parameters in the linear readout layer. In light of these findings, we believe that our approach holds great promise for further advancing the field of Artificial Intelligence research in language tasks.

5 Acknowledgements

We thank Professor Jean Rouat, for feedback on the manuscript, and Digital Research Alliance of Canada and Compute Canada for the High Performance Computing hardware used for some of the experiments. Part of this research also used resources of the National Energy Research Scientific Computing Center (NERSC), a U.S. Department of Energy Office of Science User Facility

References

[1] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[3] Alec Radford, Jong Wook Kim, Tao Xu, Greg Brockman, Christine McLeavey, and Ilya Sutskever. Robust speech recognition via large-scale weak supervision. *arXiv preprint arXiv:2212.04356*, 2022.

[4] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[5] Hongyu Wang, Shuming Ma, Li Dong, Shaohan Huang, Dongdong Zhang, and Furu Wei. DeepNet: Scaling Transformers to 1,000 Layers. *arXiv e-prints*, page arXiv:2203.00555, March 2022.

[6] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

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

[8] Valerii Likhosherstov, Krzysztof Choromanski, Jared Davis, Xingyou Song, and Adrian Weller. Sub-linear memory: How to make performers slim. *CoRR*, abs/2012.11346, 2020.

[9] Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamás Sarlós, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, David Belanger, Lucy J. Colwell, and Adrian Weller. Rethinking attention with performers. *CoRR*, abs/2009.14794, 2020.

[10] Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *CoRR*, abs/2006.16236, 2020.

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

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

[13] Beidi Chen, Tri Dao, Eric Winsor, Zhao Song, Atri Rudra, and Christopher Ré. Scatterbrain: Unifying sparse and low-rank attention approximation. *CoRR*, abs/2110.15343, 2021.

[14] Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontañón, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. *CoRR*, abs/2007.14062, 2020.

[15] Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *arXiv e-prints*, page arXiv:2205.14135, May 2022.

[16] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. arxiv 2019. *arXiv preprint arXiv:1910.01108*, 2019.

[17] F. Verstraete, J. J. García-Ripoll, and J. I. Cirac. Matrix product density operators: Simulation of finite-temperature and dissipative systems. *Phys. Rev. Lett.*, 93:207204, Nov 2004.

[18] B Pirvu, V Murg, J I Cirac, and F Verstraete. Matrix product operator representations. *New Journal of Physics*, 12(2):025012, feb 2010.

[19] Ze-Feng Gao, Song Cheng, Rong-Qiang He, Z. Y. Xie, Hui-Hai Zhao, Zhong-Yi Lu, and Tao Xiang. Compressing deep neural networks by matrix product operators. *Phys. Rev. Res.*, 2:023300, Jun 2020.

[20] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[21] Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International conference on machine learning*, pages 2067–2075. PMLR, 2015.

[22] Alex Graves, Greg Wayne, and Ivo Danihelka. Neural turing machines. *arXiv preprint arXiv:1410.5401*, 2014.

[23] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626):471–476, 2016.

[24] Gábor Melis, Tomáš Kočiskỳ, and Phil Blunsom. Mogrifier lstm. In *International Conference on Learning Representations*, 2019.

[25] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[26] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[27] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. A survey of transformers. *AI Open*, 2022.

[28] I. V. Oseledets. Tensor-train decomposition. *SIAM Journal on Scientific Computing*, 33(5):2295–2317, 2011.

[29] Alexander Novikov, Dmitrii Podoprikhin, Anton Osokin, and Dmitry P Vetrov. Tensorizing neural networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015.

[30] Steven R. White. Density matrix formulation for quantum renormalization groups. *Phys. Rev. Lett.*, 69:2863–2866, Nov 1992.

[31] Guifré Vidal. Efficient classical simulation of slightly entangled quantum computations. *Phys. Rev. Lett.*, 91:147902, Oct 2003.

[32] Philip Gage. A new algorithm for data compression. *C Users Journal*, 12(2):23–38, 1994.

[33] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models, 2016.

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

[35] Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Comput. Linguist.*, 19(2):313–330, jun 1993.

[36] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations*, 2017.

[37] Elias Frantar and Dan Alistarh. Sparsegpt: Massive language models can be accurately pruned in one-shot, 2023.

Supplementary Material

A Further analysis

The performance suffers the most when TC is applied to the output linear map, while performance suffers the least when TC is applied to the embedding, see Tab. 4. This suggests that the embedding has excess parameters that can be pruned, while the output map might be more important than often assumed. Additionally QgV outperforms every other combination of gating mechanisms as explained in the results section, see Tab. 3 for the results. We also explore different TC lengths in Tab. 2, and a length of 2 gives the best results. Finally, completely removing pre-attention improves performance while the removal of *ff* degrades it, see Tab. 5. However, one can aggressively apply TC to *ff* without negative effects, see Tab. 1.

TC length	params	dev PPL
2	33M	3.5592 ± 0.0096
4	29M	3.7582 ± 0.1157
3	33M	3.9060 ± 0.0137

Table 2: **TCs**. The Anthe variant that we use in this study is B' + QgV + H-SoftPOS + $\text{TC}_{\textit{layer}:.1}$. The length of the TC has an impact on performance, the shortest being the best.

gate	dev PPL
B' + QgV	**1.2642** ± 0.0035
B' + VgQ	1.2771 ± 0.0012
B' + KgV	2.5341 ± 0.0058
B'	2.5665 ± 0.0055
B' + VgK	2.6079 ± 0.0066
B' + QgK	2.6113 ± 0.0091
B' + KgQ	2.6315 ± 0.0093

Table 3: **Gatings:** QgV outperforms all the other combinations of gating mechanisms.

TC	params	dev PPL
B' + QgV	93M	1.2642 ± 0.0035
B' + QgV + $\text{TC}_{\textit{emb}:.8}$	86M	1.3787 ± 0.0048
B' + QgV + $\text{TC}_{\textit{emb}:.2}$	67M	1.4146 ± 0.0799
B' + QgV + $\text{TC}_{\textit{layer}:.2}$	61M	4.4466 ± 0.0134
B' + QgV + $\text{TC}_{\textit{layer}:.8}$	85M	4.5861 ± 0.0157
B' + QgV + $\text{TC}_{\textit{output}:.2}$	80M	6.8445 ± 0.0985
B' + QgV + $\text{TC}_{\textit{output}:.8}$	89M	9.7301 ± 0.1794

Table 4: **Finding the excess parameters.** Parameters in the linear output layer seem to be much more important, since when reduced by 20% and 80%, it results in the strongest decrease in performance. However, decreasing the number of learnable parameters in the embedding has less of an impact on the performance.

gate	params	dev PPL
Anthe + no *patt*	29M	**3.2351** $\pm$ **0.0186**
Anthe	30M	3.5674 ± 0.0130
Anthe + no *patt* + no *ff*	29M	4.3444 ± 0.0359
Anthe + no *ff*	30M	4.9069 ± 0.0266

Table 5: **Removing or TC?** Removing completely the pre-attention linear layers improves performance with respect to using TC on them, while removing the linear layers in the GEGLU worsens performance with respect to TC.

B Multiple language translation pairs

We consider the 7 WMT17 language pairs in Tab. 6. Both RU/EN and ZH/EN pairs exceed 9G in size, so we limit them to 9G to make better use of our limited resources. We compare Transformer to Anthe with and without TC. Remarkably the small Anthe outperforms the Transformer in all pairs, with only half its parameters, and reduces the results variance up to 70 times. Also, removing *patt* from Anthe causes a small improvement, apart from the LV/EN and TR/EN language pairs.

dataset size	params	CS/EN 1.6G test PPL	DE/EN 8.1G test PPL	FI/EN 3.9G test PPL
Anthe + no *patt*	29M	**4.6167 ± 0.6501**	**3.7891 ± 0.0444**	**4.2269 ± 0.1164**
Anthe	30M	**4.5259 ± 0.5028**	3.9822 ± 0.0131	**4.1558 ± 0.0202**
Anthe$_{noTC}$	68M	5.1874 ± 1.1889	5.3560 ± 0.0480	5.8806 ± 0.1741
Transformer	60M	11.1995 ± 4.4999	6.2168 ± 0.5581	6.8927 ± 1.4090

	params	LV/EN 4.3G test PPL	RU/EN 9G test PPL	TR/EN 306M test PPL	ZH/EN 9G test PPL
Anthe + no *patt*	29M	9.4998 ± 0.6706	**4.2085 ± 0.1171**	4.1760 ± 0.0125	**6.3080 ± 0.2609**
Anthe	30M	**8.8340 ± 0.1074**	**4.2563 ± 0.0838**	**4.1473 ± 0.0046**	**6.4599 ± 0.0456**
Anthe$_{noTC}$	68M	**8.2546 ± 0.1130**	6.6637 ± 0.1169	4.1697 ± 0.0190	9.5552 ± 0.3533
Transformer	60M	10.9181 ± 1.9593	8.2324 ± 0.1390	5.4590 ± 0.0262	15.3343 ± 0.8857

Table 6: **Different Languages.** Test perplexity on the WMT17 pairs. The Anthe outperforms the Transformer in all the language pairs with half the parameters, and improvements in perplexity up to a factor of two, e.g. CS/EN. Removing the pre-attention linear maps remains on par with Anthe.

C Laguage modeling

Results for PTB and WK2 datasets, trained for 100 epochs with batch size 8 on a single GPU, and 4 different seeds. Given the architecture size and simple training procedure, there is no expectation of outperforming GPT-3 and Sparse-GPT [37]; both of which have more than 80 billion paramters, see Tab. 7. With respect to the transformer baseline, small GPT-2, trained in the same fashion, there is significant improvement. Both KgV and QgV achieve the same performance within one standard deviation.

dataset Model	params	PTB test PPL	WK2 test PPL
GPT-2/Anthe (KgV)	66.3M	66.89 ± 0.46	263.27 ± 3.99
GPT-2/Anthe (QgV)	66.3M	66.94 ± 0.37	262.10 ± 2.35
GPT-2	124.4M	105.71 ± 4.87	1740.82 ± 274.11
GPT-3	175B	$20.5 \pm -$	$-$
Sparse-GPT (50% sparsity)	87.5B	$-$	$8.21 \pm -$

Table 7: **Language Modeling.**