PAY ATTENTION WHEN REQUIRED

A PREPRINT

Swetha Mandava
NVIDIA
smandava@nvidia.com

Szymon Migacz
NVIDIA
smigacz@nvidia.com

Alex Fit Florea
NVIDIA
afitflorea@nvidia.com

September 11, 2020

ABSTRACT

Transformer-based models consist of interleaved feed-forward blocks - that capture content meaning, and relatively more expensive self-attention blocks - that capture context meaning. In this paper, we explored trade-offs and ordering of the blocks to improve upon the current Transformer architecture and proposed PAR Transformer. It needs 35% lower compute time than Transformer-XL achieved by replacing 63% of the self attention blocks with feed-forward blocks, and retains the perplexity on WikiText-103 language modelling benchmark. We further validated our results on text8 and enwiki8 datasets, as well as on the BERT model.

1 Introduction

The seminal work in [10] introduced the Transformer architecture. Since its introduction, it profoundly influenced algorithms for Question Answering, Text Classification, Translation, Language Modeling, and practically all of the Natural Language Processing tasks. A transformer layer consists of interleaved self attention and feed forward blocks and is used in state of the art models, like Transformer-XL [1], BERT [7], Megatron [8], and other large-scale language models.

As corresponding model sizes and compute requirements continue to become increasingly more demanding, it becomes important to optimize the Transformer-based architectures. Several optimization approaches, that used pruning [26], and distillation [25], were able to achieve better run-time performance for an accuracy trade-off.

Our optimization approach investigates the trade-off between the self-attention and feed-forward building blocks. We start with the intuition that attention blocks provides context meaning while being comparatively more expensive, and feed-forward blocks provide content meaning. We then ask the fundamental questions of what are the saturation points when using one block versus the other, and how accuracy depends on the relative number of blocks of each type as well as on their ordering. To answer these questions, we employed architecture search.

While recent works such as [3, 4, 6] explored using differential neural architecture search for designing ConvNets automatically to significantly improve accuracies and/or latencies, similar work for transformer models is limited. Recent works, however, explored using random search [2] and evolutionary search [12, 13] for designing transformer models. However, even with a search space of three options per layer (Self Attention, Feed Forward, Identity), the design space becomes intractable for 32 layers as it is combinatorial (=3^{32}). For this reason, we explored the use of differential neural architecture search that has linear complexity to redesign the transformer architecture in this paper.

In order to analyze the transformer architecture, we studied the search on Transformer-XL Base with WikiText-103 dataset. We initialized a supernet, as shown in Figure 1, where each layer has multiple block choices. We then trained the supernet as a whole and studied the final sampled architecture after training. From our study, we make two fundamental observations:

- Self-attention layers are only needed among the former two-third layers of the network
- Self-attention layers to feed-forward layers ratio of 1:k is sufficient, with k=5 being optimal for Transformer-XL.

We propose **P**ay **A**ttention when **R**equired Transformer (or **PAR** transformer), a new model based on the above observations, that uses 63% fewer self-attention blocks while retaining test accuracies. Further, we validated our hypothesis with results on the BERT model with PAR BERT.

2 Differential Neural Architecture Search Algorithm

Our baseline, the Transformer-XL model, has equal number of self-attention and feed forward blocks in an interleaved design pattern as visualized in Figure 3. Sandwich transformers [2], on the other hand, have the first k sublayers consisting of self-attention, the last k sublayers consisting of feed forward layers with both sandwiched between the classic interleaving pattern of self-attention and feed forward blocks.

In this section, we attempt to optimize the transformer architecture by selecting one of the three options - Self Attention, Feed Forward, or Identity – for each of the 32 layers with differential neural architecture search.

2.1 Search Space

Is interleaved attention and feed forward layers in a transformer really the optimal design pattern? Can we get the same results for smaller, faster or imbalanced networks? In order to test these questions, we used a very simple search space that would allow us to do so, consisting of identity block, feed forward block and self-attention block that modify the input sequence X as follows:

$$F_{attn}(X) = Self\text{-}Attention(LayerNorm(X)) + X \tag{1}$$

$$F_{FF}(X) = Feed\text{-}Forward(LayerNorm(X)) + X \tag{2}$$

$$F_{identity}(X) = X \tag{3}$$

The output of each layer l can be computed using equation 4 where i is the block choice and $m_{l,i}$ is a probability distribution computed by a Gumbel Softmax function [16, 17] on all the choices in a layer from the search space. Once trained, $m_{l,i}$ allows us to study the optimal models. For example, if identity block is the most possible block on a layer, we can hypothesize that there is no benefit from a deeper network. Similarly, the model can also pick different design patterns and faster models.

$$X^l = \sum_{i \in (attn, FF, identity)} m_{l,i}.F_i^l(X^{l-1}) \tag{4}$$

Since the output at each layer is a linear combination of individual search blocks in that layer, the search cost is linear with respect to the number of blocks in the search space. Since the search also consists of training only one supernet consisting of all the search blocks, it is orders of magnitude faster than RL based search algorithms [18, 19] that rely on training individual combinations of search blocks. For our choice of supernet, the search cost was $< 2\times$ the training cost of the baseline. All of our experiments use the same model architecture parameters as the baseline from Table 5 unless otherwise mentioned. Once the search is completed, we sampled the best model from the search by selecting the most probable block at each layer.

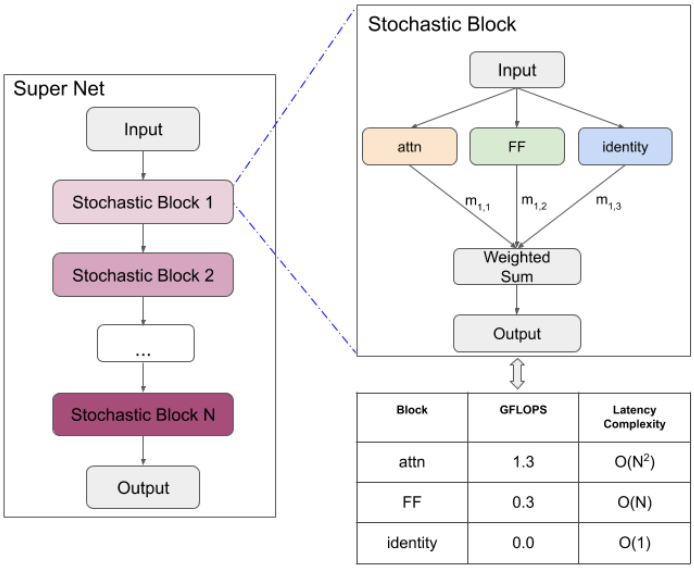
2.2 Search Algorithm and Observations

In order to explore design paradigms of a transformer architecture, we use differential neural architecture search, similar to FBNet Algorithm [3], formulated as a two stage search shown in equation 5 where the goal is to find the architecture a within a search space A, and weights w_a that minimizes the loss function L_{a,w_a} described in equation 6. Within the architecture phase, architecture parameters $m_{l,i}$ are tuned and within the weight phase, weight parameters of the individual search blocks are tuned, to minimize the loss.

$$a, w_a = min_{a \in A} min_{w_a} E_{a \sim P_\theta} \{L_{a,w_a}\} \tag{5}$$

$$L_{a,w_a} = CrossEntropyLoss_{a,w_a}(data, prediction) \tag{6}$$

We run the neural architecture search described above on $16 \times 2 = 32$ layers. On each layer, the search algorithm is able to choose between a feed-forward, self-attention or identity blocks. We run the search algorithm with batch size = 128, architecture update lr = 1e-2, weight decay for architecture parameters = 5e-4, weight update lr = 1e-2, weight decay for block weights = 1e-4. We initialize the architecture parameters uniformly and keep architecture parameters



Block	GFLOPS	Latency Complexity
attn	1.3	$O(N^2)$
FF	0.3	O(N)
identity	0.0	O(1)

Figure 1: Composition of Super Net as a linear combination of search blocks along with the block cost. GFLOPS computed for inference with tgt_len = 64, mem_len=640, $batch\_size$=1. Latency complexity with respect to sequence length.

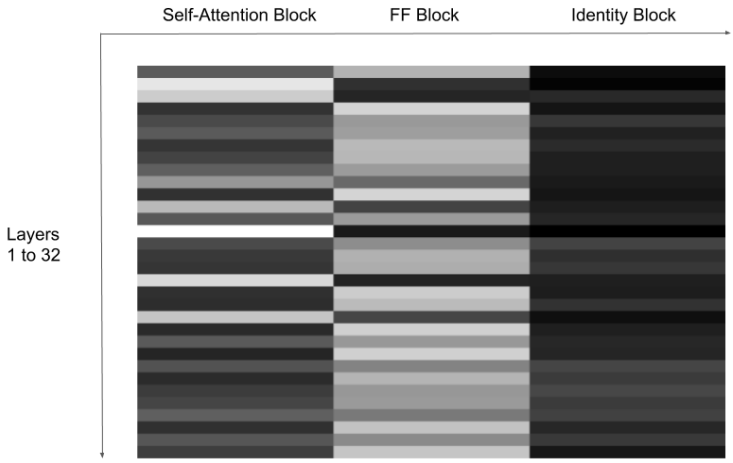


Figure 2: Probability distributions of Self-Attention, Feed forward, and Identity in each layer. Brighter shades indicate higher probability of being selected for the corresponding layer.

constant for the first 10k iterations and then perform architecture update for 20% of an epoch from there on for 40k iterations. We train till the architecture converges i.e. doesn't change in 75% of the architecture tuning stage.

The differential architecture search produces probability distributions $m_{l,i}$ representing the likelihood of block i being the optimal choice for layer l. Figure 2 visualizes weights $m_{l,i}$ at the end of training. The brighter a particular block is depicted on a layer, the higher the probability of getting picked for that layer. We make the following observations from these search results

- Self-attention layers are more probable in the former two-third layers of the network
- Self-attention layers to Feed-Forward layers ratio is much lower than 1:1.

These observations are consistent with previous research [28, 27] on transformer layers that indicate that attention layers are severely over parametrized and that they are more prevalent in the beginning of a network [2]. Based on

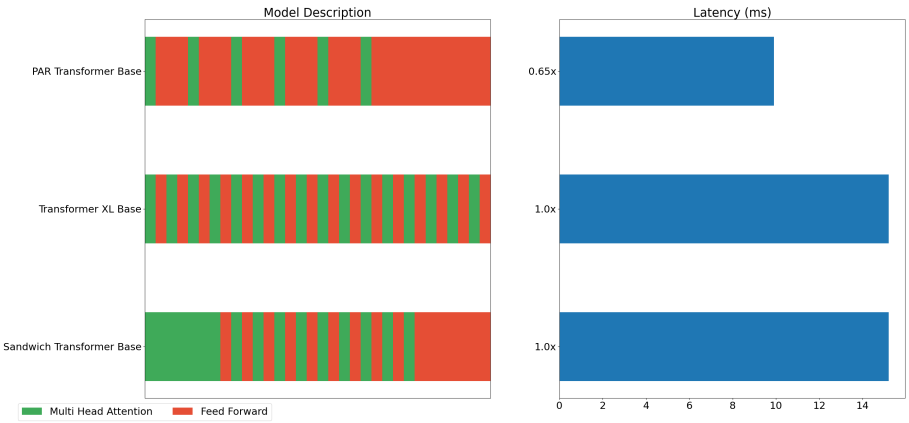


Figure 3: Comparison of Model Architecture and Latency on A100

these observations, we explored using fewer self-attention layers in Table 1. We see that while attention-blocks are essential for contextual meaning, the need is saturated fairly quickly. We observed that we can obtain the same accuracy using a fraction of the self-attention layers and led us to design a new model *PAR Transformer Base* that uses 63% fewer self-attention blocks. The advantage of replacing self-attention blocks with feed-forward blocks is the significant latency benefit we obtain, as shown in Figure 3. The benefits are even more pronounced with higher sequence lengths as self-attention and feed forward blocks have $O(N^2)$ and $O(N)$ complexities per-layer with respect to sequence lengths, respectively.

Number of Self-Attention Layers	Architecture	PPL
0	(f)x32	252.9
1	(sfff)x1 (f)x28	31.5
3	(sfff)x3 (f)x20	23.9
4	(sfff)x4 (f)x16	23.4
5	(sfff)x5 (f)x12	23.0
6	(sff)x6 (f)x14	22.9
6	(sfff)x6 (f)x8	22.7
7	(sff)x7 (f)x11	22.7
7	(sfff)x7 (f)x4	22.6
16	(sf)x16	22.7

Table 1: Test Perplexity on WikiText-103 dataset with respect to number of self-attention layers for the Base model. Architecture column explains the composition of the model, with **s** indicating a self-attention block and **f** indicating a feed-forward block.

3 Experiments

In this section, we review our observations from the search on PAR Transformer Base with WikiText-103. We further validate that the same observations generalize to other Transformer-XL models (Large, 24B) and to other datasets (enwiki8, text8). We also validate our observations on BERT models with PAR BERT.

All the models are based on the same code base[1] for training, for an apples-to-apples comparison. We used NVIDIA A100 Tensor Core GPUs for our experiments. The Architecture column explains the composition of the model, with **s** indicating a self-attention block and **f** indicating a feed-forward block. Latencies indicate inference latencies for batch size 1 as is standard. We see similar performance benefits while training as well.

[1]Our code is available at https://github.com/NVIDIA/DeepLearningExamples/tree/master/PyTorch/LanguageModeling

3.1 PAR Transformer

We compare the performance of our PAR Transformer on WikiText 103 dataset [11] in Table 2. WikiText 103 language modelling dataset consists of over 100 million tokens from articles on Wikipedia. It is well suited for testing long term dependencies as it is composed of full articles and with original case, punctuation and numbers. The only difference in the model architectures is the ordering and composition of layers, as visualized in Figure 3 and listed under the Architecture column.

The Transformer-XL Base code is based on the code published by the authors from the Transformer-XL paper but modifies hyperparameters as described in Table 5 for better hardware utilization in base model. Inference latencies are computed using tgt_len = 64, mem_len=640 and $clamp\_len$=400. We validate that we are able to obtain the same perplexities with 0.65x the cost in terms of latency.

Model	Architecture	Latency on A100 (ms)	PPL
Transformer-XL Base	(sf)x16	15.2	22.7
Sandwich Transformer Base	(s)x6 (sf)x10 (f)x6	15.2	22.6
PAR Transformer Base	(sfff)x6 (f)x8	9.9	22.7

Table 2: Latency and Perplexity (PPL) of Transformer-XL Base models on WikiText-103 dataset.

Dataset	Model	Architecture	Latency on A100 (ms)	bpc / PPL
WikiText-103	Transformer-XL Large	(sf)x18	18.9	18.4
	Sandwich Transformer Large	(s)x6 (sf)x12 (f)x6	18.9	18.2
	PAR Transformer Large	(sfff)x7 (f)x8	13.4	18.4
enwiki8	Transformer-XL 24B	(sf)x12	12.5	1.10
	Sandwich Transformer 24B	(s)x6 (sf)x6 (f)x6	12.5	1.11
	PAR Transformer 24B	(sff)x5 (f)x9	8.4	1.11
text8	Transformer-XL 24B	(sf)x12	12.5	1.18
	Sandwich Transformer 24B	(s)x6 (sf)x6 (f)x6	12.5	1.19
	PAR Transformer 24B	(sff)x5 (f)x9	8.4	1.18

Table 3: Bits Per Character (bpc) on enwiki8 and text8 and Perplexity (PPL) on WikiText-103 for Transformer-XL models.

We further validate that our hypotheses generalizes to the PAR Transformer Large Model in Table 3, by maintaining the perplexities with 0.7x the cost. The Large model uses 36 layers, d_model = 1024, d_head = 64, n_head = 16, d_inner = 4096, $tgt\_len = mem\_len$ = 384, $batchsize$ = 128 for training and tgt_len=128, mem_len=1600, $clamp\_len$=1000 for evaluation.

In order to showcase generalizability over different datasets, we validate our results on enwiki8 and text8 datasets [21] in Table 3. Enwiki8 consists of 100M bytes of unprocessed Wikipedia text where as text8 contains 100M characters of preprocessed Wikipedia text. We reuse the same model hyperparameters as in Table 5 with 24 layers. In addition, we use tgt_len = mem_len=512 for training and tgt_len=80, mem_len=2100, $clamp\_len$=820 for evaluation.

3.2 PAR BERT

We further study the observations on BERT models by pre-training on Wikipedia+Books datasets [29] using the NVLAMB optimizer [14] in two phases. Phase 1 is trained with a sequence length of 128, with a batch size of 64k for 7038 steps and phase 2 is trained with a sequence length of 512, with a batch size of 32k for 1563 steps.

Our pre-training loss curves in Figure 4 highlight the on-par performance of PAR BERT and BERT Base with a fraction of the self-attention blocks. We see in Table 4 that using the same architectural design rules results in a 1% accuracy drop on SQuAD v1.1 fine-tuning task even though the pre-training loss and accuracy on MRPC and SST-2 are on track. We hypothesize that performing an architecture search specifically might help bridge the gap. However, incorporating

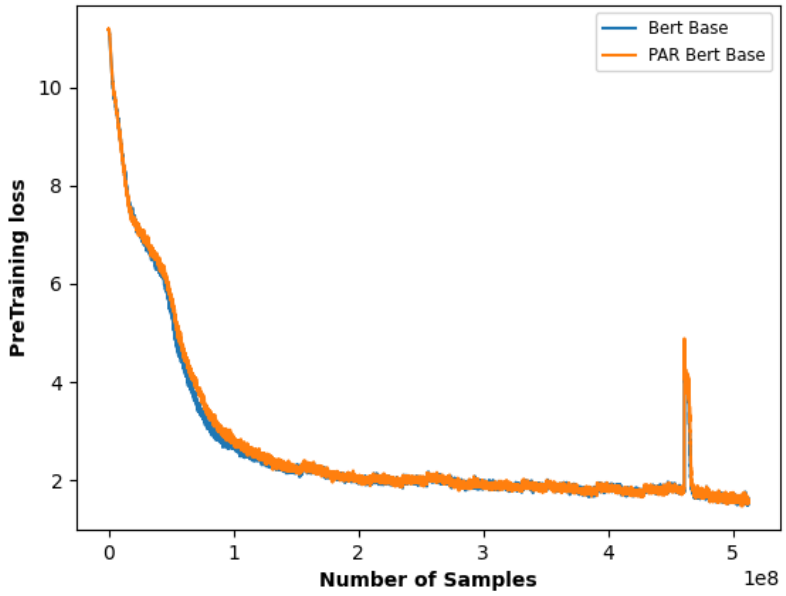


Figure 4: Pretraining Loss curve using NVLamb Optimizer for BERT Base and PAR BERT Base models

the two stage optimization process (pre-training followed by fine-tuning) that is inherent to BERT and other such language models into architecture search is a future research problem.

It is, however, interesting to note that PAR BERT has comparable latencies with respect to DistilBERT even though it uses twice as many layers. PAR BERT also outperforms DistilBERT while having a much simpler training paradigm. Nevertheless, we note that pruning, quantization and distillation are orthogonal to the present work and could be used in conjunction.

Model	Architecture	Latency on A100 (ms)	SQuAD v1.1	SST-2	MRPC
DistilBERT*	(sf)x6	5.3^{+}	86.9	91.3	87.5
BERT Base	(sf)x12	8.6	88.4	91.5	88.7
PAR BERT Base	(sff)x5 (f)x9	5.7	87.4	91.6	89.2

Table 4: Experimental results of PAR BERT in comparison to BERT Base and DistilBERT.
F1 score for SQuAD v1.1 and accuracy for SST-2 and MRPC reported from a median of 5 runs on dev sets.
Reported Latency for SQuAD inference.
* indicates originally published results.
+ indicates estimated latency as 61% of Bert Base based on DistilBERT paper

4 Conclusion

We used differential neural architecture search to study patterns in the ordering of transformer model sub-layers and made two key observations. One, that we only need attention layers in the former part of the network and two, that we need 63% fewer attention layers to retain the model accuracy. Even though we studied the search results specifically on Transformer-XL Base for the WikiText-103 dataset, the same observations were valid for other transformer models and datasets as well.

We proposed PAR Transformer that needs 35% lower latency and validated accuracy on enwiki8 and text8 datasets as well as on 24B + Large variations. We also validated our results on SQuAD v1.1, MRPC and SST-2 with PAR BERT Base model.

This paper emphasizes the ease of automatically producing optimal architectures using differential neural architecture search, even with very simple search spaces. We hope that our it opens up an avenue for automatic designing for easy optimized deployments.

References

[1] Zihang Dai & Zhilin Yang & Yiming Yang & Jaime Carbonell & Quoc V. Le & Ruslan Salakhutdinov. Transformer-XL: Attentive Language Models Beyond a Fixed-Length Context. arXiv preprint arXiv:1901.02860

[2] Ofir Press & Noah A. Smith & Omer Levy. Improving Transformer Models by Reordering their Sublayers. arXiv preprint arXiv:1911.03864

[3] Bichen Wu & Xiaoliang Dai & Peizhao Zhang & Yanghan Wang & Fei Sun & Yiming Wu & Yuandong Tian & Peter Vajda & Yangqing Jia & Kurt Keutzer. FBNet: Hardware-Aware Efficient ConvNet Design via Differentiable Neural Architecture Search. arXiv preprint arXiv:1812.03443

[4] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, Peter Vajda, Joseph E. Gonzalez. FBNetV2: Differentiable Neural Architecture Search for Spatial and Channel Dimensions. arXiv preprint arXiv:2004.05565

[5] Han Cai, Ligeng Zhu, Song Han. ProxylessNAS: Direct Neural Architecture Search on Target Task and Hardware. arXiv preprint arXiv:1812.00332

[6] Hanxiao Liu, Karen Simonyan, Yiming Yang. DARTS: Differentiable Architecture Search. arXiv preprint arXiv:1806.09055

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv preprint arXiv:1810.04805

[8] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper,Bryan Catanzaro. Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism. arXiv preprint arXiv:1909.08053

[9] Kevin Clark, Minh-Thang Luong, Quoc V. Le, Christopher D. Manning. ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators. arXiv preprint arXiv:2003.10555

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, Illia Polosukhin. Attention Is All You Need. arXiv preprint arXiv:1706.03762

[11] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher
2016. Pointer Sentinel Mixture Models

[12] Hanrui Wang and Zhanghao Wu and Zhijian Liu and Han Cai and Ligeng Zhu and Chuang Gan and Song Han. HAT: Hardware-Aware Transformers for Efficient Natural Language Processing. arXiv preprint arXiv:2005.14187

[13] David R. So and Chen Liang and Quoc V. Le. The Evolved Transformer. arXiv preprint arXiv:1901.11117

[14] Sharath Sreenivas, Swetha Mandava, Chris Forster, Boris Ginsburg. NVLAMB. https://developer.nvidia.com/blog/pretraining-bert-with-layer-wise-adaptive-learning-rates/

[15] Yang You and Jing Li and Sashank Reddi and Jonathan Hseu and Sanjiv Kumar and Srinadh Bhojanapalli and Xiaodan Song and James Demmel and Kurt Keutzer and Cho-Jui Hsieh. Large Batch Optimization for Deep Learning: Training BERT in 76 minutes. arXiv preprint arXiv:1904.00962

[16] E. Jang, S. Gu, and B. Poole. Categorical reparameterization with gumbel-softmax. arXiv preprint arXiv:1611.01144, 2016.

[17] C. J. Maddison, A. Mnih, and Y. W. Teh. The concrete distribution: A continuous relaxation of discrete random variables. arXiv preprint arXiv:1611.00712, 2016

[18] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. arXiv preprint arXiv:1611.01578, 2016.

[19] B. Zoph, V. Vasudevan, J. Shlens, and Q. V. Le. Learning transferable architectures for scalable image recognition. arXiv preprint arXiv:1707.07012, 2(6), 2017.

[20] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. 2018. Character-level language modeling with deeper self-attention. arXiv preprint arXiv:1808.04444.

[21] MultiMedia LLC. 2009. Large text compression benchmark.

[22] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. 2016. Pointer sentinel mixture models. arXiv preprint arXiv:1609.07843.

[23] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv preprint arXiv:1606.05250

[24] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, Samuel R. Bowman. 2018. GLUE: A Multi-Task Benchmark and Analysis Platform for Natural Language Understanding. arXiv preprint arXiv:1804.07461

[25] Victor SANH, Lysandre DEBUT, Julien CHAUMOND, Thomas WOLF. 2020. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108

[26] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? NeurIPS, 2019.

[27] Jean-Baptiste Cordonnier, Andreas Loukas, Martin Jaggi. Multi-Head Attention: Collaborate Instead of Concatenate. arXiv preprint arXiv:2006.16362

[28] Olga Kovaleva, Alexey Romanov, Anna Rogers, Anna Rumshisky. Revealing the Dark Secrets of BERT. arXiv preprint arXiv:1908.08593

[29] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In Proceedings of the IEEE international conference on computer vision, pages 19–27.

A Appendix

A.1 Hyper parameter changes to Model

Our Transformer-XL (Base, 24B) baselines are based on the code base published by the authors of the Transformer-XL paper but uses a modified set of model hyper parameters. Our modifications were made to achieve better hardware utilization and to take advantage of Tensor Cores, most commonly by aligning certain hyper parameters with powers of two. They are described in Table 5.

Hyperparameter	Description	Original setting	Our modification
d_{model}	hidden size	410	512
n_{head}	number of attention heads	10	8
d_{head}	size of each attention head	41	64
d_{inner}	hidden size in fully-connected layers	2100	2048
tgt_len	number of tokens to predict during training	150	192
mem_len	number of tokens cached from previous iterations during training	150	192

Table 5: Hyperparameter modifications made to Transformer-XL Base

A.2 #Parameters #Flops with respect to Latency

Model	Architecture	#Params	#GFLOPs	Latency on A100 (ms)
Transformer-XL Base	(sf)x16	192M	27	15.2
Sandwich Transformer Base	(s)x6 (sf)x10 (f)x6	192M	27	15.2
PAR Transformer Base	(sfff)x6 (f)x8	200M	17	9.9

Table 6: Flops and Parameters with respect to Latency for Base Models

Even though literature generally reports number of parameters to estimate efficiency of a model, it is too simplistic, often obscuring performance issues rather than illuminating them. We can see from Table 6 that #Parameters don't actually reflect the latency. While #FLOP count is hardware independent to its merit, latency is less abstract and more indicative of actual performance.

Model	Valid PPL @ 40k	Valid PPL @ 140k
Transformer-XL Base	23.3	22.2
Sandwich Transformer Base	23.4	22.4
PAR Transformer Base	23.3	22.4

Table 7: Latency and Test Perplexity on WikiText-103 dataset with respect to train epochs

A.3 Accuracy with respect to train epochs

Table 7 lists test perplexities at 40k and 140k iterations with a global batch size of 256. As we can see, there is little benefit in training much further after 40k iterations for the base model.