DenseAttention: No-Compromise Exact All $N \times N$ Interactions Algorithm with $O(N)$ Space and Time Complexity

Andrew Argatkiny
VK
a.argatkin@vk.team

Abstract

The Transformer architecture, ubiquitous for its advantageous ability to model all pairwise interactions between tokens in the self-attention block, suffers from two main bottlenecks: 1) low computational and high memory efficiency, leading to suboptimal hardware utilization, and 2) quadratic time complexity with respect to sequence length N, making it slow and costly for large data contexts. We propose a novel DenseAttention neural network architecture, a straightforward simplification of the standard Transformer block that addresses these issues and serves as a drop-in replacement for language modeling tasks. We eliminate memory-bound components in DenseAttention, including Softmax, masking, one skip connection, and both LayerNorms, as well as key, value, and output projection matrices, as they become redundant. Despite these removals, it maintains exact $N \times N$ pairwise interactions between tokens. By exploiting the associativity of matrix multiplications, DenseAttention can be computed with $O(N^2d)$ or $O(Nd^2)$ time and space complexity, depending on the context. To handle the absence of Softmax and prevent numerical instability, we introduce MaxNormActivation at both ends of the Transformer block and control select matrices to have a fixed L_∞ norm during pre-training. We validate DenseAttention by pre-training an encoder model on sequences up to 16K in length, which performs similarly or better than baseline BERT-large in the MLM objective for both small and large sequences, while significantly improving speed and memory efficiency. DenseAttention competes with FlashAttention 2 on small sequences and outperforms it by orders of magnitude on large contexts.

1 Introduction

Transformer architecture [46] has become ubiquitous in neural networks across many domains and modalities, (NLP [14], images [17] and video [2], speech, and even tabular data [1]). But most notably, it's the core component of Large Language Models [6, 44], which demonstrate surprisingly good abilities in natural language understanding, comprehension and reasoning tasks. Many of those models that were recently released reach near-human or super-human level of performance on a variety of benchmarks [10].

The most prominent feature which distinguishes a Transformer layer from other architectures is the attention mechanism which allows for all of the inputs to simultaneously interact with each other. However, it's also the source of its limitations: $O(N^2)$ time and space complexity w.r.t context length N, and computational inefficiency of the constituents which make the architecture work seamlessly. As reported in [19], matrix multiplications account for 99.8% of total FLOPs during BERT pretraining and only 61% of runtime, the discrepancy being caused by low arithmetic intensity of memory bound operations, namely, LayerNorms, Softmaxs and other activations as well as elementwise operations.

Preprint. Under review.

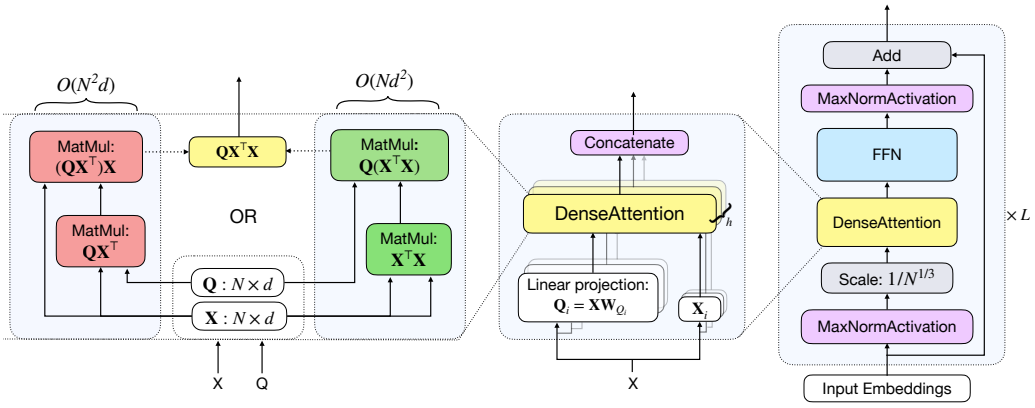

Figure 1: DenseAttention architecture. Left: DenseAttention mechanism; center: multi-head interpretation; right: the entire DenseAttention Network

Numerous architectures and modifications to the standard Transformer have been proposed in the recent years to alleviate the restrictive $O(N^2)$ complexity. However, as these architectures in general rely on non-linear, memory-intensive and sparse operations to a much greater degree than traditional attention mechanism, their throughput in terms of tokens per second and hardware utilization are subpar in comparison with the latter on all but large sequence lengths ([12, 42]). Besides, some report [41, 43, 51], that their modeling capabilities may be limited in comparison with full-rank exact attention while their conceptual complexity and incompatibility with standard architectures prevents their widespread adoption.

Thus, we aim to achieve 3 main goals:

1. To create hardware efficient yet hardware-agnostic architecture with the arithmetic intensity ratio as high as possible. An ideal algorithm should contain merely matrix multiplications with no activations, normalizations and residual connections. However, while possible in principle, it remains a challenging task due to numerical instabilities occurring both in forward and backward pass and lagging performance of such architectures [4, 30, 38]
2. To create an algorithm which would efficiently process long sequences, preferrably with $O(N)$ time and space complexity.
3. To make the resulting architecture as simple as possible and closely resembling original Transformer architecture so it can serve as a drop-in replacement for the former and be easily adopted by both research and practitioners communities.

We accomplished all of these goals with DenseAttention and DenseAttention Network (DANet) blocks (Fig. 1. This architecture is a straight-forward simplification of the traditional Transformer architecture which does not introduce any additional elements and complexities to the module and can be freely swapped with it. On the contrary, we develop DenseAttention by removing all computationally inefficient elements of the original architecture: biases in all linear layers, masks, dropout, residual connection between attention and FFN. Most importantly, we remove Softmax inside self-attention. It results in the whole scaled dot-product attention mechanism becoming just a composition of matrix multiplications, which can be done in any order by associative property of matrix multiplication. This duality allows to calculate DenseAttention using either $O(N^2d)$ or $O(Nd^2)$ FLOPs, and the second option has linear time and space complexity w.r.t sequence length.

We remove LayerNorms and instead use a new MaxNormActivation, which scales token representations by their l_∞ norm. We place it at both ends of the DANet block. We also remove all projection matrices except W_Q in the self-attention module as they become redundand in the absence of non-linearities between attention and FFN. To empirically validate the architecture, we replace Transformer modules jn BERT-large model [14] with DenseAttention Network modules and pre-train it from scratch on sequences up to 16k tokens. The model achieves better quality metrics than the original BERT while enjoying faster training and inference both in $O(N^2)$ and $O(N)$ regimes (Fig. 2.

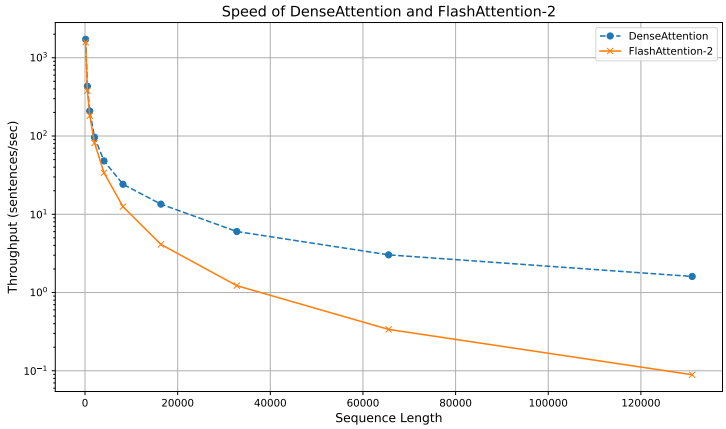

Figure 2: Comparison of speed between DenseAttention and FlashAttention2 ([11]) models across sequence lengths on a NVIDIA A100 40GB. Both models are used with the torch.compile() module.

To the best of our knowledge, we are the first to successfully train an NLP language model with no Softmax or any replacement/approximation for it in the attention layer. However, for vision tasks, such as object detection and instance segmentation, [54] propose two variations of attention, one without Softmax and the other with two softmaxes applied individually to Key and Query projections. However, they conduct experiments and report results only with second architecture. Recently, [23] instead scale Queries and Keys separately by their l_1 norm which allows them to successfully train a vision Transformer on ImageNet1K [13] and MS-COCO [25] datasets for different tasks with linear time complexity.

2 Background

Here we give a brief exposition of essential elements of Transformer architecture and their variations, and discuss concepts of hardware efficiency in relation to Deep Learning.

2.1 Transformer Architecture

Standard Transformer block consists of self-attention and feed-forward-network (FFN) sub-blocks [46]. Let $\mathbf{X} \in \mathbb{R}^{N \times d}$, where N is the sequence length and d is an embedding dimension of one token. Define $\mathbf{Q} = \mathbf{X}\mathbf{W}_Q$ as queries, $\mathbf{K} = \mathbf{X}\mathbf{W}_K$ as keys, and $\mathbf{V} = \mathbf{X}\mathbf{W}_V$ as values, where $\mathbf{W}_Q, \mathbf{W}_K, \mathbf{W}_V \in \mathbb{R}^{d \times d_h}$ are learnable parameters. Then the *Scaled Dot-Product Attention* is formulated as:

$$\text{Attention}\left(\mathbf{X}\right) = \text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d_h}} + \mathbf{M}\right)\mathbf{V}, \tag{1}$$

with Softmax applied row-wise and mask $\mathbf{M} \in \mathbb{R}^{N \times N}$ with values 0 or $-\infty$ which effectively disables some positions from calculation to account for causal sequence processing or to conceal 'PAD' token used for batch processing of sequences with different lengths. If $\mathbf{X}$ in $\mathbf{Q}$ gets replaced for some $\mathbf{X}' \in \mathbb{R}^{N' \times d}$, self-attention becomes cross-attention between two sequences. The form of the attention mask matrix and the presence of cross-attention module additionally to self-attention defines type of architecture and model usage: encoder, decoder (with causal mask), and encoder-decoder (with both causal mask end cross-attention module in some of the layers).

Default implementation in some Transformer-based models (e.g. [14]) use biases in $\mathbf{Q}, \mathbf{K},$ and $\mathbf{V}$ projection layers.

Essentially, all transformer-based models use some form of Multi-Head Attention which has h heads. Projection dimension of a head i $\mathbf{Q}_i, \mathbf{K}_i,$ and $\mathbf{V}_i$ is $\frac{d}{h}$ and typically equals 64 for smaller NLP language models like BERT, 256 for Google's PaLM [7], and 128 for most others in the billions-parameters range, including LLAMA model family [44, 45], Mistral [20] and Mixtral 8x7B [21], and

GPT-3 [6]. Attention 1 is calculated for each head independently and the results are concatenated along the embedding dimension and projected back to full block's output dimension by a matrix $\mathbf{W}_O \in \mathbb{R}^{d \times d_{out}}$:

$$\text{MultiHeadAttn}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{Concat}(\text{head}_i, \ldots, \text{head}_H)\mathbf{W}_O \tag{2}$$

Feed-Forward Network which follows self-attention is composed of two linear layers and an activation in between. Intermediate inner dimension between the two layers is usually chosen to be 4x larger than input/ output dimension but some works deviate from this convention [44]. Original architecture features ReLU as activation but more recent models use a variety of other activations instead, such as GeLU, SiLU, GLU or SwiGLU, all of them being more computationally and memory demanding than ReLU.

Finally, a LayerNorm layer and a residual connection are applied around both blocks, their relative positions dictated by PreNorm or PostNorm architectural choice [50]. The formulation of the whole Transformer layer l with PreNorm is:

$$\mathbf{X}'_l = \mathbf{X}_l + \text{Attention}(\text{LayerNorm}(\mathbf{X}_l))$$
$$\mathbf{X}_{l+1} = \mathbf{X}'_l + \text{FFN}(\text{LayerNorm}(\mathbf{X}'_l))$$

Thus, each full Transformer block has two LayerNorms and two residual connections.

Depending on the implementation, dropout [40] might also be used in various parts of the block, specifically after FFN and attention sub-blocks as in original Transformer, and in attention matrix before softmax as in BERT,

2.2 Hardware Efficiency

All calculations performed by a hardware accelerator such as a NVIDIA GPU are either compute-bound or memory-bound [48]. It depends on whether the operation in question spends the majority of time directly on computation or on data movements between High-Bandwidth Memory (HBM) and processing units. Customary unit of measurement for computational performance is TeraFLOPs (TFLOPs) per second and for memory it's bandwidth (throughput) in TB/s. Arithmetic intensity unifies them both and is calculated as $^{\text{number of FLOPs}}/_{\text{number of bytes accessed}}$. It can be attributed both to hardware accelerator (usually referred to as *ops:byte ratio* in this case) and to a computational kernel, e.g. layer of neural network, and it's necessary but not sufficient for the kernel to maintain the arithmetic intensity higher than the accelerator in order to be computationally intensive [15]. Otherwise, processing units stay idle part of the time waiting for the data to be brought from or written to HBM.

In latest generations of GPUs, FLOPs count rapidly grows but memory bandwidth progression falls behind, which results in latest generations of GPUs having much higher arithmetic intensity. Thus, it's increasingly hard for existing Deep Learning primitives to achieve hardware efficiency. Most operations besides matrix-matrix multiplications are inherently memory limited even on older GPUs. For example, the arithm. intensity of ReLU is 0.25 FLOPS/B, and for LayerNorm it's < 10 FLOPS/B on NVIDIA V100 as stated in [16]. Moreover, GPUs feature fast Tensor Cores (312 TFLOPs for half-precision formats in NVIDIA A100) specialized for matrix multiplications, and general purpose cores with significantly lower throughput (19.5 TFLOPS in NVIDIA A100) which in turn process non-MatMul operations even slower as reported in [18].

So, from the view of computational efficiency, all activations, elementwise operations and reductions are detrimental to high ratios of hardware utilization.

3 Designing DenseAttention

3.1 Dissecting Inefficiencies in Transformer

Non-linearities, namely Softmax, LayerNorms, activation in FFN, dropouts, and skip-connections, which are present in Transformer architecture, indeed contribute majorly to its computational inefficiency, as documented in [19, 32, 34]. But other affine or linear transformations might also require

further exploration. Consider two matrices $\mathbf{A} \in \mathbb{R}^{M \times N}$and $\mathbf{B} \in \mathbb{R}^{N \times K}$ stored in half-precision floating point format which is common for DL apllications. Each element in the matrices has a size of 2 bytes, and each fused multiply-add (FMA) operation takes 2 FLOPs to compute [15]. Then the arithmetic intensity of matrix multiplication in such setting is:

$$arithm.int._{MatMul} = \frac{M \cdot N \cdot K}{M \cdot N + N \cdot K + M \cdot K} \text{ FLOPs/B}, \tag{3}$$

as factors of 2 in the numerator and denominator both cancel out.

If there are no biases, then the two linear transformations in Transformer's FFN with model dimension d and standard inner dimension $4d$ have arithm. int. of $\frac{4Nd}{5N+4d}$ which equals $\frac{4d}{5}$ as $N \to \infty$. N dimension can accumulate both batch size b and sequence length s dimensions, and for BERT-large size model with $d = 1024, s = 512,$ and $b = 128$ arithm. int. is approx. 809 FLOPs/B. For largest LLaMA 2 70B model with $d = 8192, s = 4096,$ and $b = 1$ theoretical arithm. int. without using tensor parallelism [29] would be 2520 FLOPs/B. It's far greater than even NVIDIA H100 ops:byte ratio in both cases. Therefore, linear layers in the FFN are the most computationally efficient component of the Transformer and should be preserved in any hardware-aware architecture.

Similar argument may be applied to K, Q, V projection layers in the self-attention, whose matrices can be concatenated together to yield $\frac{3d}{4}$ asymptotic arithm. intensity, and to the output projection by W_O matrix in 2 ($\frac{d}{2}$ asymptotic arithm. int.). However, it follows from 3 that both products $\mathbf{S} = \mathbf{Q}\mathbf{K}^\top \in \mathbb{R}^{N \times N},$ and $\mathbf{O} = \mathbf{P}\mathbf{V} \in \mathbb{R}^{N \times d},$ where $\mathbf{P} = \text{Softmax}(\mathbf{S} / \sqrt{d_h} + \mathbf{M})$ have arithmetic intensity $\frac{N \cdot d_h}{N + 2d_h}$ with limit d_h. Also, batch and sequence dimensions cannot be fused for these operations because they are performed on *per sequence* level as opposed to *per embedding* level in FFN and KQV projections.

Since the most common choice for d_h is 128, the upper bound of arithm. int. of matrix multiplications inside attention mechanism is lower than even *ops:byte ratio* of an older V100 generation GPU. In the case of real-life configurations of BERT and LLaMA 2 from above the values are 32 and 120.5 FLOPs/B correspondingly. Thus, these operations are memory bound and inefficient.

So, from the computational perspective it would be beneficial to change number of heads in the attention to fewer or even single head with larger dimension d_h. Furthermore, it would keep the total number of flops constant because for all heads in total it equals $h \cdot N^2 \frac{d}{h} = N^2 d$.

3.2 DenseAttention

Since we aim to achieve as much computationally efficient and simple module as possible, we proceed with eliminating inefficient components of original self-attention and Transformer architectures.

The most straightforward idea which we exploit first is to abstain from using Dropout module anywhere in the model. Even though the module can be removed altogether at inference time, we also do it for the training as we believe it won't slow down the convergence with a large corpora dataset typical for LLM pre-training. Besides, as noted in [9], dropout in attention probabilities might be the reason of redundancies among attention heads. Next, we remove the attention mask before Softmax. Note that if there are no biases in FFN, Query and Output linear layers and FFN activation is ReLU, then for a row vector $\mathbf{0}_d^\top = [0, 0, \ldots, 0]_{1 \times d}$

$$\text{Attention}(\mathbf{0}_d^\top \mathbf{W}_Q, \mathbf{K}, \mathbf{V}) = \mathbf{0}_d^\top \text{ and MultiHeadAttn}(\mathbf{0}_d^\top \mathbf{W}_Q, \mathbf{K}, \mathbf{V}) = \mathbf{0}_d^\top,$$

$$\text{FFN}(\mathbf{0}_d^\top) = \mathbf{0}_d^\top,$$

and

$$\text{LayerNorm}(\mathbf{0}_d^\top) = \mathbf{0}_d^\top,$$

i.e. zero vector stays intact when acted upon by all components of the Transformer module. So we refrain from using biases throughout the new block, fix representation of the "PAD" token at the output of embedding layer to $\mathbf{0}_d^\top$, and remove masking from the self-attention layer.

Subsequently, probably the most important modification that we impose on the old architecture is removal of row-wise Softmax activation from attention. We argue that the primary source of unparalleled modeling power of the original Transformer architecture which made it dominant architecture across multiple domains is the ability for all inputs to directly interact with each other

in multiplicative way. This is the feature that all previous popular architectures like MLPs, CNNs and RNNs lack. We hypothesize that the role of softmax activation is ancillary to multiplicative interactions as it acts as a feature selection tool for the outputs of raw interactions matrix and normalizes them to be in $[0, 1]$ range and to add up to 1.

However, removing Softmax proves to be a very challenging task exactly for this reason: without it attention outputs become unbounded which can lead for them to either diverge to ∞ or shrink to 0. We formalize this statement with the following proposition considering simplified version of the new mechanism where $\mathbf{W} = \mathbf{W}_Q\mathbf{W}_K^\top$ and $\mathbf{W}_V = \mathbf{I}$:

Proposition 1. *Let $\mathbf{X} \in \mathbb{R}^{N\times d}$ and $\mathbf{W} \in \mathbb{R}^{N\times d}$ be matrices composed of i.i.d. random variables, respectively X_{ij} with $\mathbb{E}[X_{ij}] = 0$, Var$(X_{ij}) = \sigma_X^2$, and W_{km} with $\mathbb{E}[W_{km}] = 0$, Var$(W_{km}) = \sigma_W^2$. Let X_{ij} and W_{km} also be independent for all i, j, k, m. Then each element of the matrix $\mathbf{Y} = \mathbf{XWX}^\top\mathbf{X} \in \mathbb{R}^{N\times d}$ has zero expectation and variance $\sigma_Y^2 \geq Nd^2\sigma_X^6\sigma_W^2$.*

Essentially, it means that variance of an output grows at least as a cube of an input variance in the new architecture layer. And since σ_Y^2 along with tail probability $\mathbb{P}(|Y_{ij}| \geq t)$ are not bounded from above and depend on the form of an unknown distribution, we can't just fix σ_X^2 e.g. with the help of LayerNorm to ensure numerical stability.

Instead, we enforce $max(|X_{ij}|) \leq a$ for some non-negative a which is equivalent to setting fixed L_∞ norm for the inputs. Consequently, even in worst case scenario where

$$X_{ij} = a \text{ for } \forall\, i, j \tag{4}$$

it holds for $\mathbf{Z} = \mathbf{XX}^\top\mathbf{X} \in \mathbb{R}^{N\times d}$:

$$max(|Z_{ij}|) \leq Nda^3, \tag{5}$$

i.e. L_∞ norm of output values is bounded above. Furthermore, we make the following observation:

Proposition 2. *If elements W_{km} of $\mathbf{W}$ are i.i.d normal variables with mean 0 and variance σ_W^2, independent with $\forall\ X_{ij}$, $Var[(\mathbf{XW})_{pq}] \leq \sigma_W^2 a^2 d$*

It follows from **Prop. 2.** that σ_W and a can be chosen such that $\mathbb{P}[|\mathbf{XW})_{pq}| \geq \epsilon] \leq \delta$ for some $\epsilon > 0, \delta > 0$ depending on σ_W and a. Thus, we can assume that the matrix product $\mathbf{Y} = \mathbf{XWX}^\top\mathbf{X} \in \mathbb{R}^{N\times d}$ will not explode with right selection of priors.

Specifically, we set $a = \frac{1}{N^{\frac{1}{3}}}$, so that 5 becomes $max(|Z_{ij}|) \leq d$. We choose not to downscale inputs by further degree, e.g. by $\sqrt{nd}$ because resulting small values may hurt modeling quality during training in low-precision formats (**fp16** and **bf16**).

We fix each embedding vector $\mathbf{X}_i$ to have constant l_∞ norm of 1 by applying our novel *MaxNormActivation* function:

$$MaxNormActivation(\mathbf{X}_i) = \frac{\mathbf{X}_i}{\max_j(\mathbf{X}_{ij}) + \epsilon},$$

where ϵ is a very small number put to prevent division by 0. Note that similarly to *RMSNorm* [52], *MaxNormActivation* doesn't center its inputs. However, it uses l_∞ norm instead of l_2 and doesn't have *scale* and *bias* parameters as in [3, 52].

After *MaxNormActivation* we scale output by $\frac{1}{N^{\frac{1}{3}}}$. We acknowledge that both these calculations are memory bound but together they incur at most the same memory movement and compute cost as *LayerNorm*. In our ablation experiments any other activation or normalization function or absence thereof would lead to a prompt and unrecoverable numerical instability early on during training.

Consequently, it allows the removal of Softmax, which doesn't only lifts a major computational and memory bottleneck which otherwise could be alleviated mainly with clever low-level algorithms as in [12, 35]. Without Softmax and masking attention mechanism becomes a raw product of three matrices $\mathbf{QK}^\top\mathbf{V}$. Exploiting associative property of matrix multiplication, we can compute the product as

1. either $(\mathbf{QK}^\top)\mathbf{V}$ which yields $2N^2d$ FMA operations,
2. or $\mathbf{Q}(\mathbf{K}^\top\mathbf{V})$ which yields $2Nd^2$ FMA operations and is linear w.r.t N both in time and memory complexity.

We can utilize both methods interchangeably depending on what's more favorable given particular values of N and d. $O(N)$ complexity gives way to processing very large sequences in linear time with the same result as if done in traditional $O(N^2)$ paradigm as it calculates exactly the same all $N \times N$ pairwise interactions but just in another order.

Next, we consider reducing the number of heads in the multi-head attention as they are computationally inefficient. As extensive research efforts have shown [5, 24, 26, 47] significant portion of heads in multi-head attention are redundant, output low-rank representations and can be pruned without decrease in quality in downstream tasks, at least in BERT-sized models. Specifically, [5] find that increasing number of heads past a certain threshold degrades performance in BERT. Motivated by this, we propose increasing d_h from conventional value 128 up to 1024. In case of BERT example from 3.1 it leads to a single-head attention with arithm. int. 204.8 FLOPs/B which makes it computationally efficient even on NVIDIA A100. For LLMs with larger model dimension $d_h = 1024$ would still leave room for multiple heads. We also use $d_h = 256$ in experiments. And asymptotic arithm. int. in $O(N)$-regime is $\frac{d}{2}$ just like in an ordinary $d \times d$ dense layer.

We note that the matrix $\mathbf{W} = \mathbf{W}_Q \mathbf{W}_K^\top$ in the expression $\mathbf{Q}\mathbf{K}^\top = \mathbf{X}\mathbf{W}_Q\mathbf{W}_K^\top\mathbf{X}^\top$ is essentially low-rank as in standard attention $d_h << d$. But in our implementation this rank is much higher, in the extreme case being equal to d. It results in multiplication of two high or full rank matrices. That is a redundant operation from DL perspective because composition of linear maps is just another linear map which could be learned using half of the parameters. Thus, we decide to keep the $\mathbf{W}_Q$ and discard $\mathbf{W}_K$.

We also decide to remove LayerNorm and residual connection between attention and FFN sub-blocks as it improves computational efficiency of the architecture and appears not to hinder model performance. This leads to yet another simplification in the model design: $\mathbf{W}_V$ and $\mathbf{W}_V$ also become redundant by similar reasoning as in case of $\mathbf{W}_Q$ because there are no more non-linearities between attention outputs and FFN block.

Finally, the new attention mechanism in the case of a single head is formulated as:

$$\text{DenseAttention}\left(\mathbf{X}\right) = \mathbf{X}\mathbf{W}_Q\mathbf{X}^\top\mathbf{X} \in \ \mathbb{R}^{N\times d}$$

And in the case of multiple heads it slightly changes:

$$\text{DenseAttention}_i\left(\mathbf{X}\right) = \mathbf{X}\mathbf{W}_{Q_i}\mathbf{X}_i^\top\mathbf{X}_i \in \mathbb{R}^{N\times d_h}$$

We call our attention algorithm "DenseAttention" and the entire block as "DenseAttention Network" or DANet (spelled "dah-net") because it basically consists of dense matrix multiplications with little else. We notice that DenseAttention in multi-head setting resembles populat multi-query attention designs from [39] as it also has different representations only for Queries.

To complete the DenseAttention Network, we apply *MaxNormActivation* and residual connection to outputs of FFN. Final architecture can be summarized as follows:

$$\mathbf{X}'_l = \text{DenseAttention}(\text{MaxNormActivation}(\mathbf{X}_l) \cdot N^{-\frac{1}{3}})$$
$$\mathbf{X}_{l+1} = \mathbf{X}_l + \text{MaxNormActivation}(\text{FFN}(\mathbf{X}'_l))$$

4 Experiments

To prove the viability of DenseAttention architecture, we pre-train an encoder model with the approximately same number of parameters as in BERT-large [14]. We keep model dimension $d = 1024$ as in original work but increase number of layers from 24 to 32 to keep parity in number of parameters. We use the same MLM (Masked Language Modelling)+ NSP (Next Sentence Prediction) combination of training objectives and pre-train on the same datasets, namely Wikipedia and BookCorpus [53].

We pre-train two models: one with single head of size $d = 1024$ and the other with 4 heads of size $d = 256$. The are 4 training stages, each one resuming from the last checkpoint of the previous: first with approximately 850 million samples of sequence length 128, second with 150 mil. samples of seq. len 512, third with 80 mil. samples of seq. len 1024, and the last stage with 27 mil. samples of sequence length 16384 conducted exclusively with single head model. The single head model was trained in $O(N^2)$ regime with context sizes 128, 512 and, partially, 1024, and in $O(N)$ for the rest

Table 1: Evaluations MLM loss and accuracy for DenseAttention models w.r.t to BERT on C4 dataset. N is the maximum sequence length with which a model was train or/and evaluated.

Model	N=128		N=512		N=1024	
	MLM Loss	Acc.	MLM Loss	Acc.	MLM Loss	Acc.
BERT-large	2.67	0.561	2.42	0.59	-	-
DenseAttention (1 head, N=128)	**2.13**	**0.577**	-	-	-	-
DenseAttention (1 head, N=512)	2.19	0.572	1.92	**0.603**	-	-
DenseAttention (1 head, N=1024)	2.19	0.572	**1.91**	**0.606**	2.51	0.545
DenseAttention (4 heads, N=128)	2.19	0.568	-	-	-	-
DenseAttention (4 heads, N=512)	2.27	0.558	2.05	0.582	-	-
DenseAttention (4 heads, N=1024)	2.3	0.554	2.04	0.584	**2.08**	**0.575**

of the run with 1k and 16k contexts. The 4 heads model utilized the $O(N)$ regime for all sequence lenghts.

Then we validate and compare the results with BERT-large, using Google's original pretrained checkpoint available from Hugging Face's Transformers library [49]. We evaluate the models on out of domain texts of C4 dataset's subset "RealNewsLike" [36] for all contexts lengths besides 16k because train/test splits for wiki + books dataset are almost surely different for our model and BERT training procedures. We use MLM loss which can be interpreted as logarithmic perplexity, and MLM accuracy as evaluation metrics. The results are presented in Table 1

Key highlights. DenseAttention models uniformly outperform baseline in terms of MLM loss by a large margin. Perhaps, this may be contributed partially to dampening output logits which lead to probabilities more calibrated to the ambiguity of natural language. Nevertheless, single-headed DenseAttention models also uniformly outperform standard BERT in terms of accuracy, although the difference is not so pronounced, as with log-perplexity.

The models with 4 heads are inferior in both metrics to single head ones which supports out hypothesis that larger head sizes lead to better quality. The only exception is the performance of the models trained with context length 1024 on sequences of the same size, where 4 heads DenseAttention model variant outputs significantly better metrics. This might hint that it's easier for several heads to comprehend long sequences than for one. Note that the original BERT wasn't trained with sequence length 1024, so we couldn't compare it with our models on this setup.

Table 2: Quality metrics for single-head DenseAttention model trained on the context up to 16k tokens. Books dataset contains > 98% of sequences with length > 1024, and for each tested max. seq. len. it's guaranteed to contain approx. 80% of sequences with such length. C4 dataset for max seq. lengths 1024 and 2048 has at least 9.5% sequences with context size ≥ 1024.

max seq. len.	**Books**		**C4**	
	MLM loss	**acc.**	**MLM loss**	**acc.**
16384	2.76	0.451	-	-
8192	2.64	0.482	-	-
4096	2.45	0.511	-	-
2048	2.21	0.549	2.4	0.545
1024	-	-	2.59	0.506
512	-	-	2.55	0.513

The models which continued training with seq. len. 1k slightly outperform their counterparts which stopped after seq. len. 512. on sequences with this same size which indicates that training on longer contexts is indeed beneficial for modeling quality. However, performance degrades when models trained on $N = 512$ or $N = 1024$ get tested on seq. len. 128 which is a consequence of the models specializing on the longer sequences.

This property gets even more noticeable with the single head model trained on 16k context (Table 2). The pre-training was performed on the dataset which contains 26% of sequences with max size of 16k tokens, and 45% with size ≥ 1024. Therefore, the model is adapted to long-context and performs much better in terms of evaluation metrics on the datasets and context lenghts with greater maximum size. We argue that quality metrics of the model on this maximum context size, while inferior to the metrics of the smaller-context checkpoints on their respective lengths, is actually quite impressive, as it correctly finds the right word out of 35 thousand vocabulary options 45% of the time for approximately 2000 masked tokens for a single sequence of size 16k. And with the decrease of the context length to 2048 tokens, the model quality becomes almost equal to the smaller-context models evaluated with their native sequence sizes.

Table 3: Throughput, sequences per second, of single head DenseAttention model in $O(N)$ and $O(N^2)$ regimes in comparison with BERT, and BERT with FlashAttention 2 across various sequence lengths. FLOPs ratio is total MatMul FLOPs of forward pass of DenseAttention BERT implementation in $O(N^2)$ regime divided by total MatMul FLOPs of forward pass of standard BERT. All experiments were conducted on a single NVIDIA A100 40Gb GPU.

Seq. Len.	**DenseAttention**		**BERT**	**BERT with FlashAttn 2**	**bs**	**FLOPs ratio**
	$O(N)$	$O(N^2)$		$O(N^2)$		$O(N^2)$
128	1403	**1721**	1450	1584	512	1.01
512	400.1	**431.8**	304.9	379.5	256	1.03
1024	**208.9**	208.8	117.9	181.6	128	1.05
2048	**96.42**	85	-	81.69	64	1.08
4096	**48.09**	33.38	-	33.93	32	1.13
8192	**24.18**	11.81	-	12.52	16	1.19
16384	**13.47**	4.1	0.943	4.12	8	1.24
32768	**6.02**	0.985	-	1.224	4	1.28
65536	**3.03**	0.378	-	0.338	2	1.30
131072	**1.604**	-	-	0.089	1	1.32

We also evaluate (Table 3)DenseAttention single head model speed, as measured by throughput, in comparison with standard BERT model and with highly-optimized, low-level FlashAttention-2 implementation which is the fastest conventional kernel for attention computation as of mid 2024 [11]. All evaluations are performed using torch.compile() directive. As expected, DenseAttention model vastly outperforms even FlashAttention-2 algorithm with either quadratic or linear regime, depending on the sequence length. But, surprisingly, we also observed that with the increase of the sequence length the performance of the DenseAttetntion in the $O(N^2)$ regime is slightly worse or even similar to FlashAttention-2 despite being written in high-level language and having more FLOPs per iteration than a standard model with comparable size. It leads to conclusion that the DenseAttention indeed achieves very high computational intensity and FLOPs utilization in comparison with the alternatives.

Moreover, we observe that quality evaluation metrics stay the same if the regime gets switched from $O(N)$ to $O(N^2)$ or vice versa regardless of the mode and sequence length with which a DenseAttention model have been trained. This invariance property holds even for the model trained on 16k context and applied to sequence length 128. Thus, we can train the models with DenseAttention on very large contexts and then use it both short and long sequences with optimal speed with equal quality.

5 Acknowledgements

We thank Nikolay Anokhin for his continuous support, constructive feedback and directing *attention* to the relevant papers; Aleksandr Tarakanov for interesting and useful discussions, suggesting ideas and giving advice; Andrey Kuznetsov for providing the necessary computational resources and constantly inspiring by his talks on LLMs; and Vadim Gurov for starting the research group initiative and making this piece of work possible.

References

[1] Sercan Ömer Arik and Tomas Pfister. Tabnet: Attentive interpretable tabular learning. *CoRR*, abs/1908.07442, 2019. URL http://arxiv.org/abs/1908.07442.

[2] Anurag Arnab, Mostafa Dehghani, Georg Heigold, Chen Sun, Mario Lučić, and Cordelia Schmid. Vivit: A video vision transformer. In *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 6816–6826, 2021. doi: 10.1109/ICCV48922.2021.00676.

[3] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.

[4] David Balduzzi, Marcus Frean, Lennox Leary, J. P. Lewis, Kurt Wan-Duo Ma, and Brian McWilliams. The shattered gradients problem: If resnets are the answer, then what is the question? In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 342–350. PMLR, 06–11 Aug 2017. URL https://proceedings.mlr.press/v70/balduzzi17b.html.

[5] Srinadh Bhojanapalli, Chulhee Yun, Ankit Singh Rawat, Sashank Reddi, and Sanjiv Kumar. Low-rank bottleneck in multi-head attention models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pages 864–873. PMLR, 13–18 Jul 2020. URL https://proceedings.mlr.press/v119/bhojanapalli20a.html.

[6] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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, 2020.

[7] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, Sasha Tsvyashchenko, Joshua Maynez, Abhishek Rao, Parker Barnes, Yi Tay, Noam Shazeer, Vinodkumar Prabhakaran, Emily Reif, Nan Du, Ben Hutchinson, Reiner Pope, James Bradbury, Jacob Austin, Michael Isard, Guy Gur-Ari, Pengcheng Yin, Toju Duke, Anselm Levskaya, Sanjay Ghemawat, Sunipa Dev, Henryk Michalewski, Xavier Garcia, Vedant Misra, Kevin Robinson, Liam Fedus, Denny Zhou, Daphne Ippolito, David Luan, Hyeontaek Lim, Barret Zoph, Alexander Spiridonov, Ryan Sepassi, David Dohan, Shivani Agrawal, Mark Omernick, Andrew M. Dai, Thanumalayan Sankaranarayana Pillai, Marie Pellat, Aitor Lewkowycz, Erica Moreira, Rewon Child, Oleksandr Polozov, Katherine Lee, Zongwei Zhou, Xuezhi Wang, Brennan Saeta, Mark Diaz, Orhan Firat, Michele Catasta, Jason Wei, Kathy Meier-Hellstern, Douglas Eck, Jeff Dean, Slav Petrov, and Noah Fiedel. Palm: Scaling language modeling with pathways, 2022.

[8] Hyung Won Chung, Thibault Fevry, Henry Tsai, Melvin Johnson, and Sebastian Ruder. Rethinking embedding coupling in pre-trained language models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=xpFFI_NtgpW.

[9] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. What does BERT look at? an analysis of BERT's attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 276–286, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-4828. URL https://aclanthology.org/W19-4828.

[10] OpenAI collective. Gpt-4 technical report, 2024.

[11] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

[12] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems*, volume 35, pages 16344–16359, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/67d57c32e20fd0a7a302cb81d36e40d5-Paper-Conference.pdf.

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009. doi: 10.1109/CVPR.2009.5206848.

[14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL https://aclanthology.org/N19-1423.

[15] NVIDIA Docs. Matrix multiplication background user's guide, . URL https://docs.nvidia.com/deeplearning/performance/dl-performance-matrix-multiplication/index.html#math-mem.

[16] NVIDIA Docs. Gpu performance background user's guide, . URL https://docs.nvidia.com/deeplearning/performance/dl-performance-gpu-background/index.html#understand-perf.

[17] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

[18] Horace He. Making deep learning go brrrr from first principles. 2022. URL https://horace.io/brrr_intro.html.

[19] Andrei Ivanov, Nikoli Dryden, Tal Ben-Nun, Shigang Li, and Torsten Hoefler. Data movement is all you need: A case study on optimizing transformers. In A. Smola, A. Dimakis, and I. Stoica, editors, *Proceedings of Machine Learning and Systems*, volume 3, pages 711–732, 2021. URL https://proceedings.mlsys.org/paper_files/paper/2021/file/bc86e95606a6392f51f95a8de106728d-Paper.pdf.

[20] Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

[21] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[22] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[23] Soroush Abbasi Koohpayegani and Hamed Pirsiavash. Sima: Simple softmax-free attention for vision transformers. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 2607–2617, January 2024.

[24] Olga Kovaleva, Alexey Romanov, Anna Rogers, and Anna Rumshisky. Revealing the dark secrets of BERT. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural*

Language Processing (EMNLP-IJCNLP), pages 4365–4374, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1445. URL https://aclanthology.org/D19-1445.

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft coco: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham, 2014. Springer International Publishing. ISBN 978-3-319-10602-1.

[26] Paul Michel, Omer Levy, and Graham Neubig. Are sixteen heads really better than one? In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/2c601ad9d2ff9bc8b282670cdd54f69f-Paper.pdf.

[27] Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018. URL https://openreview.net/forum?id=r1gs9JgRZ.

[28] Jishnu Mukhoti, Viveka Kulharia, Amartya Sanyal, Stuart Golodetz, Philip Torr, and Puneet Dokania. Calibrating deep neural networks using focal loss. In *Advances in Neural Information Processing Systems*, volume 33, pages 15288–15299. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/aeb7b30ef1d024a76f21a1d40e30c302-Paper.pdf.

[29] Deepak Narayanan, Mohammad Shoeybi, Jared Casper, Patrick LeGresley, Mostofa Patwary, Vijay Korthikanti, Dmitri Vainbrand, Prethvi Kashinkunti, Julie Bernauer, Bryan Catanzaro, Amar Phanishayee, and Matei Zaharia. Efficient large-scale language model training on gpu clusters using megatron-lm. *International Conference for High Performance Computing, Networking, Storage and Analysis, SC*, 4 2021. doi: 10.1145/3458817.3476209. URL https://arxiv.org/abs/2104.04473v5.

[30] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28 of *Proceedings of Machine Learning Research*, pages 1310–1318, Atlanta, Georgia, USA, 17–19 Jun 2013. PMLR. URL https://proceedings.mlr.press/v28/pascanu13.html.

[31] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. URL http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[32] Suchita Pati, Shaizeen Aga, Nuwan Jayasena, and Matthew D. Sinclair. Demystifying bert: System design implications. In *2022 IEEE International Symposium on Workload Characterization (IISWC)*, pages 296–309, 2022. doi: 10.1109/IISWC55918.2022.00033.

[33] Tiberiu Popoviciu. Sur les équations algébriques ayant toutes leurs racines réelles. *Mathematica*, 9(129-145):20, 1935.

[34] Jacob Portes, Alexander R Trott, Sam Havens, DANIEL KING, Abhinav Venigalla, Moin Nadeem, Nikhil Sardana, Daya Khudia, and Jonathan Frankle. MosaicBERT: A bidirectional encoder optimized for fast pretraining. In *Thirty-seventh Conference on Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/095a6917768712b7ccc61acbeecad1d8-Paper-Conference.pdf.

[35] Markus N. Rabe and Charles Staats. Self-attention does not need o(n^2) memory. *CoRR*, abs/2112.05682, 2021. URL https://arxiv.org/abs/2112.05682.

[36] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv e-prints*, 2019.

[37] Jeff Rasley, Samyam Rajbhandari, Olatunji Ruwase, and Yuxiong He. Deepspeed: System optimizations enable training deep learning models with over 100 billion parameters. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '20, page 3505–3506, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450379984. doi: 10.1145/3394486.3406703. URL https://doi.org/10.1145/3394486.3406703.

[38] Shibani Santurkar, Dimitris Tsipras, Andrew Ilyas, and Aleksander Madry. How does batch normalization help optimization? In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/905056c1ac1dad141560467e0a99e1cf-Paper.pdf.

[39] Noam Shazeer. Fast transformer decoding: One write-head is all you need. *CoRR*, abs/1911.02150, 2019. URL http://arxiv.org/abs/1911.02150.

[40] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[41] Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models, 2024. URL https://openreview.net/forum?id=UU9Icwbhin.

[42] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. Efficient transformers: A survey. *ACM Comput. Surv.*, 55(6), dec 2022. ISSN 0360-0300. doi: 10.1145/3530811. URL https://doi.org/10.1145/3530811.

[43] Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Tran, Dani Yogatama, and Donald Metzler. Scaling laws vs model architectures: How does inductive bias influence scaling? In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12342–12364, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.825. URL https://aclanthology.org/2023.findings-emnlp.825.

[44] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.

[45] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[47] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1580. URL https://aclanthology.org/P19-1580.

[48] Samuel Williams, Andrew Waterman, and David Patterson. Roofline: an insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, apr 2009. ISSN 0001-0782. doi: 10.1145/1498765.1498785. URL https://doi.org/10.1145/1498765.1498785.

[49] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. Transformers: State-of-the-art natural language processing. In Qun Liu and David Schlangen, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://aclanthology.org/2020.emnlp-demos.6.

[50] Ruibin Xiong, Yunchang Yang, Di He, Kai Zheng, Shuxin Zheng, Chen Xing, Huishuai Zhang, Yanyan Lan, Liwei Wang, and Tie-Yan Liu. On layer normalization in the transformer architecture. In *Proceedings of the 37th International Conference on Machine Learning*, ICML'20. JMLR.org, 2020.

[51] Wenhan Xiong, Barlas Oguz, Anchit Gupta, Xilun Chen, Diana Liskovich, Omer Levy, Scott Yih, and Yashar Mehdad. Simple local attentions remain competitive for long-context tasks. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1975–1986, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.144. URL https://aclanthology.org/2022.naacl-main.144.

[52] Biao Zhang and Rico Sennrich. Root mean square layer normalization. In *Advances in Neural Information Processing Systems*, volume 32, 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/1e8a19426224ca89e83cef47f1e7f53b-Paper.pdf.

[53] Yukun Zhu, Ryan Kiros, Rich Zemel, Ruslan Salakhutdinov, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. Aligning books and movies: Towards story-like visual explanations by watching movies and reading books. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 19–27, 2015. doi: 10.1109/ICCV.2015.11.

[54] Shen Zhuoran, Zhang Mingyuan, Zhao Haiyu, Yi Shuai, and Li Hongsheng. Efficient attention: Attention with linear complexities. In *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 3530–3538, 2021. doi: 10.1109/WACV48630.2021.00357.

A Proofs

Proof of Proposition 1:

$$Y_{ij} = \sum_{n=1}^{N} \sum_{m=1}^{d} \sum_{k=1}^{d} X_{ik} W_{km} X_{mn}^{\top} X_{nj}$$

Denote $S(i;k;m;n;j) = X_{ik} W_{km} X_{mn}^{\top} X_{nj}$. Since $\mathbb{E}[W_{km}] = 0$ and W_{km} is independent from X, $\mathbb{E}[S(i;k;m;n;j)] = 0$ and $\mathbb{E}[Y_{ij}] = \sum_{k,m,n} \mathbb{E}[S(i;k;m;n;j)] = 0$. Hence, $Var[S(i;k;m;n;j)] = \mathbb{E}[X_{ik}^2 W_{km}^2 (X_{mn}^{\top})^2 X_{nj}^2] - 0$.

As some of the indices i, k, m, n, j can be the same number, there are three possible options for $Var[S(i;k;m;n;j)]$:

1. $\mathbb{E}[x_1^2 x_2^2 x_3^2]\mathbb{E}[w^2] = \sigma_X^6 \sigma_W^2$ by independence of all x and w.
2. $\mathbb{E}[x_1^4 x_2^2]\mathbb{E}[w^2] = \mathbb{E}[x_1^4]\mathbb{E}[x_2^2]\sigma_W^2 \geq \sigma_X^6 \sigma_W^2$, because by Jensen's inequality $\mathbb{E}[g(x^2)] \geq g(\mathbb{E}[x^2])$ and we let $g(f) = f^2$.
3. $\mathbb{E}[x^6]\mathbb{E}[w^2] \geq \sigma_X^6 \sigma_W^2$ by similar reasoning ($g(f) = f^3$ is convex on $(0, \infty)$).

Finally, $Cov(S_p, S_q) = 0$ if the set of indices p is not identically equal to set q because even one distinct index between p and q leads to independent factors inside the covariance operator. Therefore, $Var[Y_{ij}] \geq Nd^2 \sigma_X^6 \sigma_W^2$. □

Proof of Proposition 2: If we let $\mathbf{X}_{ij} = a$ be a degenerate R.V. as in worst case, 4, then $Var[(\mathbf{XW})_{pq}] = \sigma_W^2 a^2 d$ by C.L.T and properties of variance. In all other cases, from $X_{ij} \in [-a, a]$ follows that $\sigma_{X_{ij}}^2 \leq a^2$ by Popoviciu's inequality [33]. Then $Var[X_{pj} W_{jq}] = \sigma_{X_{pj}}^2 \sigma_{W_{jq}}^2 \leq a^2 \sigma_W^2$, and $Var[(\mathbf{XW})_{pq}] = \sum_{j=1}^{d} Var[X_{pj} W_{jq}] \leq \sigma_W^2 a^2 d$ even if some X_{pj} is dependent with some $X_{pj'}$, because $Cov[\sigma_{X_{pj}}^2 \sigma_{W_{jq}}^2; \sigma_{X_{pj'}}^2 \sigma_{W_{j'q}}^2] = 0$ for $j \neq j'$. □

B Details of the training procedure

Besides DenseAttention blocks in place of standard Transformer ones, there are a few minor differences in our implementation:

- We place $HardTanh$ activation at the end of Embeddings layer to ensure that maximum absolute value of the layer outputs is not greater than 1.
- We decouple the embedding layer weight from the vocabulary projection matrix in the output layer. This brings additional $vocab.size \times d$ parameters to the model but makes it more expressive and may bring additional gains in quality as described in [8].
- We modify Standard LayerNorm which is used in original code twice after Transformer layers in the following way: we uncenter it like in *MaxNormActivation*, replace standard deviation with mean absolute value and also remove **bias** parameter. We do it to speed up training and to minimize architectural differences with *MaxNormActivation* but keep its output range unrestricted.
- We add $HardTanh$ activation with parameters $min\_val = -20$ and $max\_val = 2$ to the end of the last layer before Softmax over vocabulary so the model doesn't produce extremely high and low logits. We empirically found that it helps to stabilize gradients and also results in more calibrated probabilities, similar to the effect of focal loss [28].

To ensure numerical stability, we scale weight matrices of FFN layers to have a constant l_∞ norm after each optimizer step during pre-training. fter pretraining, we merge each weight with its final scaling factor so there is no additional overhead at the inference time. The choice of the norm type is motivated largely by the bounds it provides for the layer outputs as in the case with the DenseAttention layer. The scaling factor of a layer is a standalone non-trainable scalar decoupled from its corresponding weight tensor at the train time. This means that the weight itself doesn't gets re-scaled constantly which would otherwise induce tug-of-war dynamics with the direction of

gradient. This way, the weight also has natural proportions compared to ADAM optimizer's ([22]) weight update as it would in the absence of scaling. By employing this technique, we eliminate the need for weight decay and warmup.

We observed that scaling the Queries weight in the DenseAttention hinders loss convergence speed to a certain degree so we proceeded with scaling just FFN layers.

We code the model in plain PyTorch [31] and train it in distributed mode using DeepSpeed [37] in **fp16** precision, using the framework's native implementation which is similar to NVIDIA's AMP [27]. We found out during ablation experiments that training in **bf16** format converges significantly slower, likely because it has less precision bits than **fp16**. **bf16** also has a disadvantage that it doesn't work on older GPUs such as NVIDIA V100.