Cost-Effective Attention Mechanisms for Low Resource Settings: Necessity & Sufficiency of Linear Transformations

Anonymous ACL submission

Abstract

From natural language processing to vision, Scaled Dot Product Attention (SDPA) is the backbone of most modern deep learning applications. Unfortunately, its memory and computational requirements can be prohibitive in low-resource settings. In this paper, we improve its efficiency without sacrificing its versatility. We propose three attention variants where we remove consecutive linear transformations or add a novel one, and evaluate them on a range of standard NLP and vision tasks. Our proposed models are substantially lighter than standard SDPA (and have 25-50% fewer parameters). We show that the performance cost of these changes is negligible relative to size reduction and that in one case (Super Attention) we succeed in outperforming SPDA by up to 10% while improving its speed and reducing its parameters by 25%.

1 Introduction

Few ideas have had as profound an effect on the field of *Artificial Intelligence* (*AI*) as the *attention mechanism* (Bahdanau et al., 2015). Introduced as a method to improve machine translation, the attention mechanism revolutionized the way neural networks process and interpret data. It mimics a form of cognitive attention in humans by allowing models to focus on specific parts of the input while disregarding irrelevant information. This enhanced the capability and efficiency of Language Models (LM) and paved the way for the development of advanced AI architectures like the Transformer model (Vaswani et al., 2017).

These advances have had far-reaching impacts, extending beyond Natural Language Processing (NLP) to areas such as image recognition (Dosovitskiy et al., 2021), autonomous systems (Mott et al., 2019), healthcare (Choi et al., 2016), and multi-modal application (Xu et al., 2023).

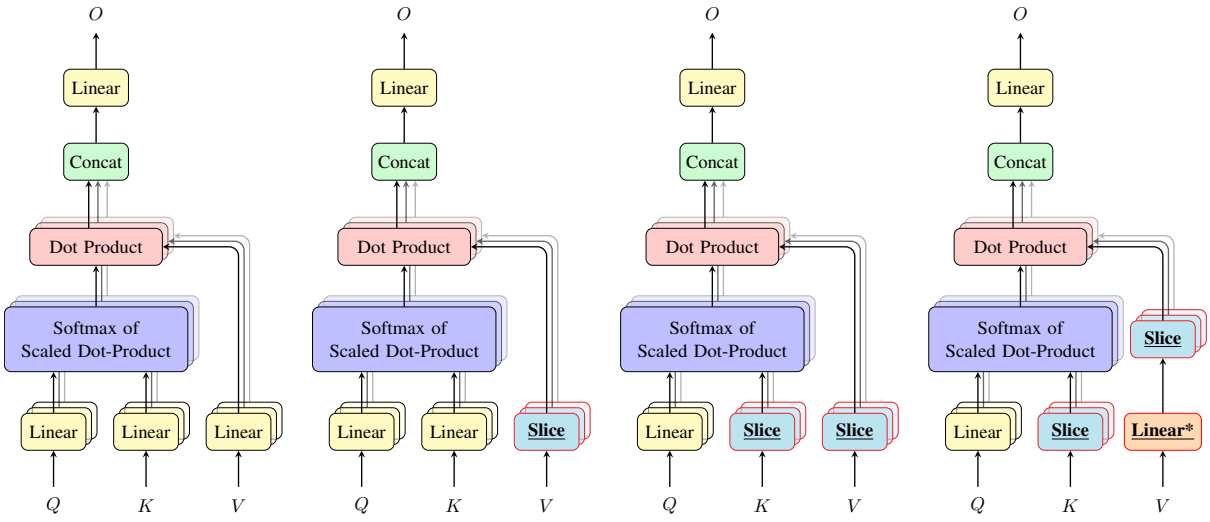
The formulation of SDPA in all these domains has undergone very little change compared to the original formulation of Vaswani et al. (2017). Instead, the prevailing maxim has been "the bigger the better", and Large Language Models (LLM), such as Llama 3 (Touvron et al., 2023a,b), GPT-4 (Achiam et al., 2023), and Gemini (Anil et al., 2023) have demonstrated unprecedented capabilities in multi-modal domains.

The behemothic sizes of these models have introduced numerous challenges. Expensive and slow training and inference have resulted in high carbon emissions (Dhar, 2020); and such models are impossible not only to run but even to store on edge devices such as smartphones, consumer laptops, and even powerful personal workstations.

Numerous attempts have been made to address this problem using post-training techniques, like quantization (Jacob et al., 2018), Low-Rank Adaptation (LoRA) (Hu et al., 2022), Quantized LoRA (QLoRA) (Dettmers et al., 2023), and sparsification (Ashkboos et al., 2024). Others have attempted to optimise the speed and GPU utilization of attention-based models, e.g., Flash Attention 1–3 (Dao et al., 2022; Dao, 2024; Shah et al., 2024). However, all these approaches strive to improve the performance of attention-based models but without altering the attention mechanism.

In this paper, we propose a different approach: modifying the attention mechanism itself. We employ two intuitive principles to design our alternative attention mechanism: **(1)** two consecutive linear transformations do not introduce non-linearity, and **(2)** a learnable linear kernel between each two inputs of SDPA enhances learning. We leverage these two principles to propose 3 SDPA variants:

- **Optimized Attention** (§ 3.1, Fig. 1b), replaces W^V linear transformation with a slicing operation (Principle 1), reducing the parameters in the attention layer by 25% and its computational cost by h matrix multiplications, where h is the number of heads. Optimized Attention reduces



(a) Stn. Attention (b) Opt. Attention (c) Eff. Attention (d) Sup. Attention

Figure 1: Standard multi-head scaled dot product attention (1a) alongside the proposed variations: Optimized Attention (1b), Efficient Attention (1c), and Super Attention (1d). The "Linear" block denotes a linear transformation right while "Linear*" denotes a linear transformation from left.

the inference time by 2.5–7.5%, with little or no performance degradation (§4).

- ◇ **Efficient Attention** (§3.2, Fig. 1c) replaces W^V and W^K linear transformations by slicing operations (Principle 1). This reduces the parameters in the attention layer by 50% and its computational cost by $2h$ matrix multiplications, where h is the number of heads. Efficient Attention reduces the inference time by 5–15%, with no/little performance degradation (§4).

- ◇ **Super Attention** (§3.3, Fig. 1d) introduces a new linear operation W^A (Principle 2), which transforms the values V from the left. Super Attention can be used on top of standard or Optimized attentions (i.e., without replacing W^V and W^K). For simplicity, we build Super Attention on top of Efficient Attention. Super Attention reduces the attention layer's size by $\sim 25\%$ (depending on the attention's context length) and its computational cost by h matrix multiplications. Super Attention outperforms standard attention by 2–10% in NLP and vision tasks and reduces the training and inference time by 2.5–10% (§4).

Our evaluation is comprehensive and compares our proposed attention models with SDPA in the *self-attention* setting in transformers for multiple datasets and for 4 different tasks, including: **(1)** *Natural Language Sentiment Classification* on IMDB and Amazon Reviews datasets; **(2)** *Machine Translation* (*NMT*) on the combined Europarl and Anki English-to-Spanish translation dataset; **(3)** *Generative Language Modeling and Natural Language Inference (NLI)* using NanoGPT (Karpathy, 2022) on the OpenWebText dataset; and to show how these architectural changes generalize to transformers for other modalities, we do complementary experiments for **(1)** *image classification* on MNIST, CIFAR100, and ImageNet datasets;

2 Preliminaries

We start by introducing the notation we use throughout the paper. For natural numbers $d_m, d_k \in \mathbb{N}$, we denote the d_m-dimensional real *vectors space* by $\mathbb{R}^{d_m}$ and the set of all real $d_m \times d_k$ *matrices* by $\mathbb{R}^{d_m \times d_k}$, noting that all matrices can be regarded as 2D *tensors* and vice versa. Given a set $\mathcal{A} \subseteq \mathbb{R}^{d_m}$, we denote the smallest real vector space containing $\mathcal{A}$ by $\text{span}(\mathcal{A})$. Similarly, given a matrix $W \in \mathbb{R}^{d_m \times d_k}$, we denote the smallest real vector space containing the columns of W's by $\text{span}(W)$. For a *subspace* $\mathcal{S} \leq \mathbb{R}^{d_m}$, the *dimension* of $\mathcal{S}$, denoted $\dim(\mathcal{S})$, is the size of the largest *linearly independent* set in $\mathcal{S}$. The *rank* of a matrix $W \in \mathbb{R}^{d_m \times d_k}$, denoted $\text{rank}(W)$, is the number of linearly independent columns (or rows) in W. The rank-nullity theorem implies that $\text{rank}(W) = \dim(\text{span}(W))$ and $\text{rank}(W) \leq \min(d_m, d_k)$.[1]

We use the widely-adopted definition of SDPA as implemented in SotA open-source models such as Llama-3 and Mistral, and machine learning frameworks like Torch and JAX. For consistency, we use the same notation as (Vaswani et al., 2017).

[1] For details see (Meyer, 2023, Chapters 2 & 4).

Definition 2.1 (Standard Attention)**.** The (*multi-head*) *scaled dot-product attention* on *input* tensors $Q, K, V \in \mathbb{R}^{\ell \times d_m}$ is defined as:

$$O = (H_1\ H_2\ \cdots\ H_h)W^O, \quad (1)$$
$$H_i = S_i V_i', \quad (2)$$
$$S_i = \mathrm{softmax}(\frac{Q_i' K_i'^{\intercal}}{\sqrt{d_k}}), \quad (3)$$
$$V_i' = VW_i^V, \quad (4)$$
$$K_i' = KW_i^K, \quad (5)$$
$$Q_i' = QW_i^Q, \quad (6)$$

where O is the *output*; Q_i', K_i', V_i', S_i, and H_i are the *query*, *key*, *value*, *attention score*, and *head value* of the i-th *head*, respectively. The natural numbers ℓ, d_m and h are the *context length*, *model dimension*, and *number of heads*, respectively. Moreover, $W_i^Q, W_i^K \in \mathbb{R}^{d_m \times d_k}$ and $W_i^V \in \mathbb{R}^{d_m \times d_v}$, where d_k and d_v are the *key* and *value dimensions*, respectively.

Parameters d_m, d_k, d_v and h are often chosen so that $d_k = d_v = d_m/h$, and in recent models, including SotA Transformer models, Q, K, and V are set to X, a single input tensor; whereby, the attention mechanism is called *self-attention*.

3 Revising the Attention Mechanism

We introduce our three proposed attention variants and discuss the motivation behind each of them.

3.1 Optimized Attention: Absorbing W_i^V's into W^0

In standard attention, the output O of the attention layer can be written as

$$\begin{aligned} O &= (H_1\ \cdots\ H_h)W^O \\ &= (S_1 V W_1^V\ \cdots\ S_h V W_h^V) \begin{pmatrix} W_1^O \\ \vdots \\ W_h^O \end{pmatrix} \\ &= S_1 V W_1^V W_1^O + \cdots + S_h V W_h^V W_h^O, \end{aligned} \quad (7)$$

where W_i^O is the matrix with rows $(i-1)d_v + 1, \ldots, id_v$ of W^O for $i = 1, 2, \ldots, h$. By the rank-nullity theorem, for each head, we have that:

$$\begin{aligned} &\dim(\mathrm{span}(VW_i^V W_i^O)) \\ &\quad = \mathrm{rank}(VW_i^V W_i^O) \le \mathrm{rank}(W_i^V W_i^O), \\ &\quad \le \min(\mathrm{rank}(W_i^V), \mathrm{rank}(W_i^O)) \\ &\quad = \min(d_m, d_v) = d_v. \end{aligned}$$

That is, $VW_i^V W_i^O$ has at most d_v independent columns, and the linear function $V \mapsto VW_i^V W_i^O$ maps the columns of V into a d_v-dimensional subspace of $\mathbb{R}^{d_m}$. Thus, standard attention uses two consecutive matrix multiplications to embed the columns of V into a d_v-dimensional subspace of $\mathbb{R}^{d_m}$, which does not align with Principle 1.

To address this, in Optimized Attention, we absorb $W_1^V, W_2^V, \ldots, W_h^V$ into W^O in Eqs. (1) and (4), thus reducing the computational cost of the attention layer by h matrix multiplications at a very limited performance cost–which we evaluate in §4.

Optimized Attention uses one slicing and one linear transformation (see Fig. 1b and Def. 3.1), instead of the two consecutive linear transformations (one downscaling and one upscaling). Specifically, instead of multiplying V from the right by W_i^V, we slice V into $V_1, \ldots, V_h$, where V_i consists of columns $(i-1)d_v + 1, \ldots, id_v$ of V, and then, instead of computing $S_i V W_i^V W_i^O$, we compute $S_i V_i W_i^O$, which needs fewer parameters and matrix multiplications (see Rem. 3.2 and § 4.3 for theoretical and empirical evaluations, respectively.)

Definition 3.1 (Optimized Attention)**.** Using the notation of Def. 2.1, *Optimized Attention* is defined as follows:

$$O = (H_1, H_2, \ldots, H_h)W^O, \quad (8)$$
$$H_i = S_i V_i, \quad (9)$$
$$S_i = \mathrm{softmax}(\frac{Q_i' K_i'^{\intercal}}{\sqrt{d_k}}), \quad (10)$$
$$K_i' = KW_i^K, \quad (11)$$
$$Q_i' = QW_i^Q. \quad (12)$$

Remark 3.2. Optimized Attention is more efficient than standard attention, having h matrix multiplication and d_m^2 parameters less than standard attention.

Proof. Compared to Optimized Attention, standard attention has extra $W_1^V, W_2^V, \ldots, W_h^V$, which are multiplied from the right to V. This amounts to a total of $d_m d_v h = d_m^2$ parameters and h matrix multiplications. □

3.2 Efficient Attention: Absorbing W^K into W^Q

In § 3.1, we discussed our motivation behind dropping W^V. Here, we do the same for W^K to further reduce the computational cost of the attention mechanism. Before this, we note that for the pre-

softmax attention scores for each head, we have:

$$\begin{aligned}&\dim(\text{span}(\frac{QW_i^Q W^{K\intercal}{}_i K^\intercal}{d_k})\\&= \text{rank}(QW_i^Q W^{K\intercal}{}_i K^\intercal) \leq \text{rank}(W_i^Q W^{K\intercal}{}_i),\\&\leq \min(\text{rank}(W_i^Q), \text{rank}(W_i^K))\\&= \min(d_m, d_k) = d_k.\end{aligned}$$

More precisely, here two linear kernels, W_i^Q and $W^{K\intercal}{}_i$, are stacked– this opposes Principle 1. Thus, following the same approach as in Optimized Attention, we merge $W^{K\intercal}{}_i$ into W_i^Q and replace the W_i^K linear transformation by slicing as depicted in Fig. 1c and defined in Def. 3.3.

Definition 3.3 (Efficient Attention)**.** Using the same notation as Def. 3.1, we define *Efficient Attention* with the following equations:

$$\begin{aligned}O &= (H_1, H_2, \ldots, H_h)W^O, && (13)\\H_i &= S_i V_i, && (14)\\S_i &= \text{softmax}(\frac{Q'_i K_i^\intercal}{\sqrt{d_k}}), && (15)\\Q'_i &= QW_i^Q, && (16)\end{aligned}$$

where K_i denotes the subtensor consisting of $(i-1)d_k+1, \ldots, id_k$ rows from K.

Remark 3.4. Efficient Attention is more efficient than standard and Optimized Attention as it has h matrix multiplication and d_m^2 parameters less than Optimized Attention and $2h$ multiplication and $2d_m^2$ parameters fewer than standard attention.

Proof. In Efficient Attention, we do not have $W_1^K, W_2^K, \ldots, W_h^K$, which are applied to K from left. Hence, we reduce the number of matrix multiplications by h and parameters by d_m^2, compared to Optimized Attention. From this and Rem. 3.2, it follows that Efficient Attention has $h+h=2h$ matrix multiplication and $d_m^2+d_m^2=2d_m^2$ parameters fewer than standard attention. □

3.3 Super Attention: Introducing W^A

Looking at the Eqs. (1-6), we observe that in SDPA, there are learnable parameters between Q and K; however, there is no such parameter between K and V (even though a softmax is applied to the term containing K). Following Principle 2, we introduce a new learnable parameter W^A that linearly transforms the values from the left. To better observe this, let us write the equation for one head in one of the attention variants, e.g., Efficient Attention by combining Eqs. (14–16):

$$H_i = \text{softmax}(\frac{QW_i^Q K_i^\intercal}{d_m})V_i W^O. \quad (17)$$

As we see in Eq. (17), there are no learnable parameters between $K^\intercal$ and V, and the attention scores S_i are directly applied to the values V_i. The intuition behind directly applying S_i to V_i is that the attention scores in S_i determine "how much attention is paid" to each of the features of each token in V_i. Despite this intuition, we found that in practice the model can benefit from an additional kernel which comes in between the scores S_i and values V_i. Specifically, with the introduction of W^A, Eq. (17) changes to

$$H_i = \text{softmax}(\frac{QW_i^Q K_i^\intercal}{d_m})W^A V_i W^O. \quad (18)$$

The role of W^A is to mix and align the values vertically (token-wise). Thus, to prevent "look ahead" in the attention mechanism for use in causal language modelling, we can constrain W^A to be lower triangular, so that future tokens do not influence the current one in W^A. Note that we use the same W^A for all heads. The reason here is that we want to improve the model performance while keeping the model size as small as possible. Thus, in a more general formulation, one can use different W^A for each head to perhaps gain even better performance, but at the cost of increasing the number of parameters, and thereby the model size.

Definition 3.5 (Super Attention)**.** Using the notation of Def. 3.3, we define *Super Attention* with the following equations:

$$\begin{aligned}O &= (H_1, H_2, \ldots, H_h)W^O, && (19)\\H_i &= S_i V'_i, && (20)\\S_i &= \text{softmax}(\frac{Q'_i K_i^\intercal}{\sqrt{d_k}}), && (21)\\V'_i &= W^A V_i, && (22)\\Q'_i &= QW_i^Q, && (23)\end{aligned}$$

where $W^A \in \mathbb{R}^{\ell\times\ell}$ is the *alignment kernel*, which vertically (i.e., for values corresponding to different tokens) aligns and mixes the values before the attention scores are applied to them.

Remark 3.6. Super Attention is more efficient than standard attention whenever the model dimension d_m is greater than or equal to the context length

ℓ. This means that Super Attention has at least h matrix multiplication and d_m^2 parameters fewer than standard attention.

Proof. Looking at Eqs. (13–16) and (19–23), Super and Efficient Attention have the same equations, except that Super Attention has an additional linear transformation in Eq. (22), where V_i's are multiplied by W^A from the left. This amounts to ℓ^2 parameters and h matrix multiplication more than Efficient Attention. From Rem. 3.4, it follows that Super Attention has at least $2d_m^2 - \ell^2 \geq d_m^2$ parameters and $2h - h = h$ matrix multiplications fewer than standard attention. □

4 Evaluation

We evaluate the proposed mechanism on a range of NLP tasks (§4.1 and §B.5); we then show that the approach generalises to other modalities by evaluating them on a number of vision benchmarks (§4.2). We also provided a detailed comparison of the computational costs and edge device performance in § 4.3, B.1, and B.2.

Evaluation Methodology. We evaluate on a range of benchmarks. In each benchmark, we follow the common practices for evaluating the performances. For all benchmarks, (1) we use the same model architecture and iterate between standard, Optimized, Efficient, and Super Attention; (2) we continue training until validation loss flattens or a given computational budget is reached; and (3) for benchmarks on smaller datasets, we report the results by averaging over five runs to ensure fairness.

Experimental Setup. All experiments in § 4.1 and 4.2 are implemented in Keras with JAX backend using `keras.io/examples` with minor dataset-specific adjustments, e.g., modifying the number of classes, layers, etc. The generative language modelling experiment in §4.1 is an adaptation of Andrej Karpathy's NanoGPT (Karpathy, 2022). All the reported results are obtained by training on an Nvidia RTX 4090 GPU (24GB VRAM) or an Nvidia A100 GPU (80GB VRAM); however, we have chosen model and batch sizes to ensure that they run on 24GB VRAM. In each table, we report the train and test loss and accuracy (where relevant), the number of parameters in one attention layer (in the "# Param." column), the average training time (in seconds) of models for one epoch on an RTX 4090 GPU (in the "Epoch Time" column), as well as other related task-specific metrics.

4.1 NLP Benchmarks

In this section, we evaluate the attention variants in Transformer models of different scales for three NLP tasks: sentiment classification, Machine Translation (MT) and generative language modelling (LM) and NLI tasks.

Sentiment Classification. For sentiment classification (Tbl. 1), we use two widely-used benchmarks, IMDB Movie Reviews (Maas et al., 2011) and Amazon Reviews (Ni et al., 2019) datasets. The dataset sizes for these two experiments in this part are 50k and 3.65M, and the model sizes are 650K and 26M parameters, respectively.

Machine Translation (MT) For MT (Tbl. 2), we use the combined Europarl (Koehn, 2005) and Anki (Anki.net) dataset for English-to-Spanish translation. The dataset includes 2 million pairs and the model sizes range from 93-104 million parameters for different architectures.

Generative LM and NLI. For generative language modelling (Tbl. 3), we use the OpenWebText dataset (Gokaslan and Cohen, 2019) for training and the HellaSwag dataset (Zellers et al., 2019) for comparing the common-sense reasoning performance of the trained models. This dataset includes more than 9 Billion tokens and the model sizes range between 110-124 million parameters for different architectures. The context window of the language models is set to 1024 tokens.

NLP Results Analysis. Super Attention outperforms attention variants in terms of validation accuracy (up to $^{(68.10-65.55)}/_{65.55} = 3.89\%$ compared to standard attention on Amazon Reviews) in the sentiment classification task. Similarly, we see for MT as well as generative LM and NLI tasks that the Optimized and Efficient architectures perform closely or on par with the Standard mechanisms. We also observe that standard attention is slower than all other variants (up to $^{(600-523)}/_{523} = 14.72\%$ slower than Efficient Attention in MT) with the highest number of parameters (twice as many parameters per layer compared to Efficient Attention). The generative LM experiment reveals subtle differences in performance among the models in training performance; However, our NLI experiment shows that when evaluated on the HellaSwag benchmark, all three models exhibit comparable performance, achieving accuracy rates between 30% and 31%.

Table 1: Sentiment classification results, averaging over five runs on IMDB and Amazon Reviews datasets. Numbers in parentheses indicate the ranking of each attention variant for a given metric and dataset. Ablation studies on the number of heads for all experiments is available in §B.4. Efficient Attention models have the smallest attention layer size and the Super Attention models perform the best in terms of accuracy and loss.

Dataset	Att.	h	d_m	# Param.	Epoch Time	Acc. (%)	Loss	Val Acc. (%)	Val Loss
IMDB	Stn.	4	32	4,224 (4)	0.315 (4)	95.70 (4)	0.086 (3)	77.62 (4)	0.474 (4)
	Opt.	4	32	3,168 (2)	0.305 (3)	96.31 (3)	0.095 (4)	77.85 (2)	0.472 (2)
	Eff.	4	32	**2,112 (1)**	**0.280 (1)**	96.41 (2)	**0.064 (1)**	77.77 (3)	**0.468 (1)**
	Sup.	4	32	3,168 (2)	0.299 (2)	**97.45 (1)**	0.070 (2)	**78.34 (1)**	0.472 (2)
Amazon	Stn.	4	128	66,048 (4)	66.97 (4)	88.49 (3)	0.25 (3)	65.55 (4)	0.77 (3)
	Opt.	4	128	49,536 (3)	61.75 (3)	**89.56 (1)**	**0.23 (1)**	65.67 (2)	0.75 (2)
	Eff.	4	128	**33,024 (1)**	**56.44 (1)**	86.63 (4)	0.29 (4)	65.58 (3)	0.77 (3)
	Sup.	4	128	42,336 (2)	59.86 (2)	88.56 (2)	0.24 (2)	**68.10 (1)**	**0.71 (1)**

Table 2: Machine translation results, averaging over five runs for English-to-Spanish MT on combined Europarl and Anki translation datasets. Numbers in parentheses indicate the ranking of each attention variant for that metric. Ablation on the number of heads is available in §B.4. Optimized and Efficient Attentions perform similarly to standard attention on most metrics with ½ and ¾ as many attention parameters, respectively. As the Super Attention layer has a fixed context length and the decoder requires a varying context length, using Super Attention would require using a sliding window, which would not be comparable to the full attention used for the other variants.

Att.	h	d_m	d_k	# Param.	Epoch Time	BLEU	Acc.	Loss	Val BLEU	Val Acc.	Val Loss
Stn.	4	1024	256	4.2M (3)	600.0 (3)	23.1 (2)	81.11 (3)	0.83 (3)	**22.8 (1)**	81.41 (3)	0.84 (3)
Opt.	4	1024	256	3.1M (2)	586.8 (2)	**24.5 (1)**	**82.06 (1)**	**0.78 (1)**	22.6 (3)	**81.98 (1)**	**0.80 (1)**
Eff.	4	1024	256	**2.1M (1)**	**523.0 (1)**	22.6 (3)	81.15 (2)	0.82 (2)	22.3 (3)	81.44 (2)	0.83 (2)

4.2 Vision Transformers

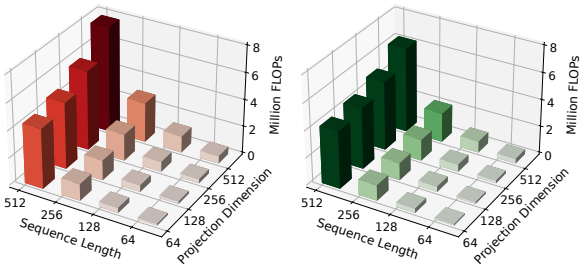
We experiment with three widely adopted vision datasets of varying size and complexity: MNIST (LeCun et al., 2010), CIFAR100 (Krizhevsky, 2009), and ImageNet1K (Russakovsky et al., 2015). For Brevity, we refer to the ImageNet1K dataset throughout the paper as ImageNet. Note that for the reported ImageNet results in Tbl. 4, we first pre-trained the model on the ImageNet21K dataset. We report the training details in §B.3.

ViT Results Analysis. The number of parameters in the models considered for the vision tasks range from 300K (MNIST) to 60M (ImageNet), their context length ranges from 64 (MNIST) to 256 (CIFAR100 and ImageNet), the dataset sizes range from 60K (MNIST) to 1.28M (ImageNet), and the number of classes ranges from 10 (MNIST) to 1K (ImageNet). Similar to text Transformers, ViTs using Super Attention architecture perform better than all other variants despite having fewer parameters than standard attention. Also, Optimized and Efficient Attentions perform comparably to standard attention with fewer parameters.

4.3 Speed and FLOPs Analysis

§B.1 and B.2 are dedicated to studying the computational complexity and inference speed of the considered attention variants. Eq. (24) formulates the computational complexity for each algorithm.

Figs. 2 and 7 visualize a comparison between the required FLOPs for each algorithm based on "sequence length" and "projection dimension". It indicates Efficient Attention requires the least number of FLOPs under all scenarios. From an empirical perspective, Tbl. 5 and Fig. 3 exhibit the faster inference speed (lower latency) of Efficient Attention compared to other variants in all datasets, followed by Optimized and Super variants.



(a) Standard Att. (b) Efficient Att.

Figure 2: 3D plots visualizing the number of FLOPs for a forward + backward pass given different sequence lengths and projection dimensions in single-head setting for Efficient and Standard attention. Efficient Att. needs substantially fewer FLOPs for completing a forward + backward pass. Fig. 7 compares all architectures.

Table 3: Averages of different metrics in generative LM using NanoGPT, a widely-referenced re-implementation of GPT-2 124M by Andrej Karpathy, based on different attention variants. The models are trained on the OpenWebText dataset (∼9B training tokens) for one epoch with a batch size of 500 and a micro-batch size of 5 using a single A100 80GB node. The context window is 1024. In addition to the loss and perplexity, we provide the size of each model and the result of NLI on the HellaSwag benchmark. Similarly to the MT task, a fair comparison of Super Att. against other variants is not feasible as NanoGPT uses full attention but Super Att. requires using a sliding window.

Att.	h	d_m	d_k	Layer Size	Model Size	Train Loss	Train PPL	Val Loss	Val PPL	HellaSwag
Stn.	12	768	64	2.36M	124M	2.92	18.5	3.13	22.9	0.31
Opt.	12	768	64	1.77M	117M	2.96	19.3	3.14	23.1	0.31
Eff.	12	768	64	1.18M	110M	3.02	20.5	3.18	24.0	0.30

Table 4: Vision results, averaging over five runs on MNIST and CIFAR100, and one run on ImageNet. Numbers in parentheses indicate the ranking of each mechanism for a given metric and dataset. An ablation study on the number of heads is available in §B.3. An additional ablation study for models of the same size on ImageNet but with different attention mechanisms is provided in §B.3. As expected, Efficient Attention models have the smallest attention layer size, and the Super Attention models achieve the highest accuracy and lowest loss.

Dataset	Att.	h	d_m	# Param.	Epoch Time	Acc. (%)	Loss	Top 5	Val Acc. (%)	Val Loss	Val Top 5
MNIST	Stn.	4	128	66K (4)	8.31 (4)	93.73 (4)	0.209 (4)	N/A	98.12 (4)	0.062 (4)	N/A
	Opt.	4	128	49K (3)	7.68 (3)	95.36 (2)	0.161 (2)	N/A	98.43 (2)	0.046 (2)	N/A
	Eff.	4	128	**33K (1)**	**7.05 (1)**	94.28 (3)	0.197 (3)	N/A	98.27 (3)	0.058 (3)	N/A
	Sup.	4	128	37K (2)	7.58 (2)	**96.96 (1)**	**0.112 (1)**	N/A	**98.62 (1)**	**0.051 (1)**	N/A
CIFAR100	Stn.	8	256	263K (4)	21.19 (4)	72.28 (2)	1.41 (2)	91.02 (2)	48.14 (3)	1.82 (3)	90.22 (4)
	Opt.	8	256	197K (2)	20.39 (3)	72.26 (3)	1.47 (3)	93.01 (3)	48.63 (2)	1.71 (2)	90.99 (2)
	Eff.	8	256	**131K (1)**	**19.22 (1)**	71.96 (4)	1.49 (4)	92.23 (4)	47.95 (4)	1.83 (4)	90.48 (3)
	Sup.	8	256	197K (3)	20.28 (2)	**79.62(1)**	**1.28 (1)**	**94.34 (1)**	**49.28 (1)**	**1.55 (1)**	**91.69 (1)**
ImageNet	Stn.	12	768	2.36M (4)	2572 (4)	92.07 (2)	1.02 (2)	98.41 (2)	74.35 (3)	1.47 (3)	94.10 (4)
	Opt.	12	768	1.77M (3)	2426 (2)	91.78 (3)	1.03 (3)	98.36 (3)	77.12 (2)	1.47 (3)	94.21 (3)
	Eff.	12	768	**1.18M (1)**	**2374 (1)**	90.36 (4)	1.05 (4)	98.37 (4)	75.67 (4)	1.44 (2)	95.46 (2)
	Sup.	12	768	1.22M (2)	2483 (3)	**94.09 (1)**	**0.94 (1)**	**99.32 (1)**	**79.29 (1)**	**1.39 (1)**	**96.37 (1)**

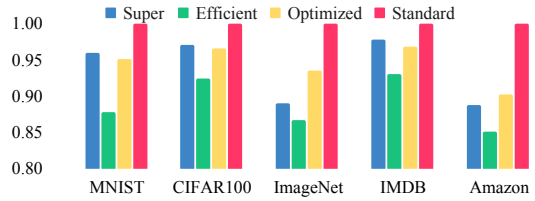


Figure 3: Summary of relative inference latency of models using different attention variants relative to standard attention on different datasets on Edge Device (Apple Laptop M2). Efficient Att. is the fastest (Optimized and Super Att. are also faster than standard attention). More details and numerical results are available in Tbl. 5.

4.4 Scaling Analysis

We analyzed scaling behaviour across three dimensions: attention heads, dataset size, and model size. Head scaling experiments across tasks (Tbls. 5, 6 and 8 to 10) showed consistent performance improvements with increased heads for all architectures. Dataset scaling ranged from IMDB (50K examples) to OpenWebText (9B tokens) for language tasks, and MNIST (60K examples) to ImageNet (1.28M examples) for vision tasks, with our variants maintaining their relative performance advantages across scales. Model scaling experiments on Amazon Reviews (Figs. 4 and 8) demonstrate that as models grow from 5M to 25M parameters, Super Attention consistently outperforms standard attention, while Optimized and Efficient variants match standard's performance with significantly fewer parameters. Notably, standard attention's computational inefficiency becomes more pronounced at larger scales in both training and inference.

5 Related Work

Since their adoption, many research directions have emerged to address various shortcomings of attention mechanisms and Transformer models. Sparse attention, such as Longformer (Beltagy et al., 2020; Zhang et al., 2021a), reduces the computational complexity by focusing on key input parts (Child et al., 2019). Despite handling long sequences efficiently, sparse mechanisms struggle with tasks requiring a comprehensive sequence analysis.

Another line of research focuses on approximating the attention matrix to attain linear complexity. Performer (Choromanski et al., 2021) uses random feature maps and FAVOR+ mechanism; Linformer (Wang et al., 2020) projects keys and values to

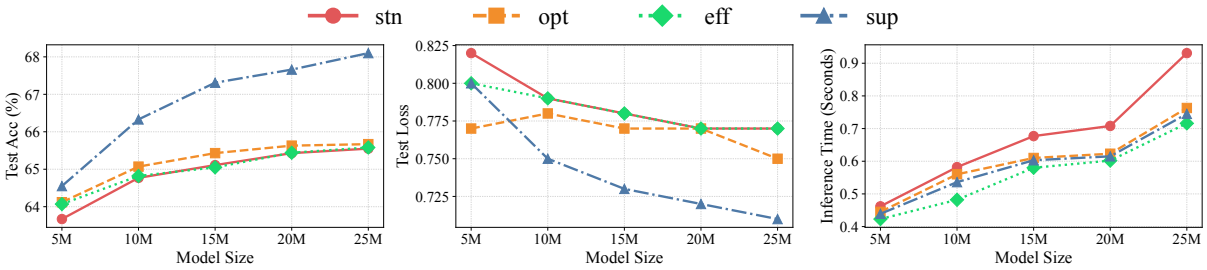


Figure 4: Performance of different architectures on the Amazon Reviews as the size of models grows from 5 Million parameters to 25 Million parameters. In terms of test accuracy and loss, Super Attention shows increasingly better performance compared to all other architectures which are performing on par with each other. In terms of inference speed, all variants (especially Efficient) perform better than the Standard attention.

lower dimensions by exploiting low-rank properties. While these approaches achieve efficiency through approximation, they often compromise model quality. In contrast, our proposed variants achieve efficiency through structural modifications while maintaining or improving model quality.

Recent work has explored architectures that combine transformers' parallel training capabilities with RNNs' inference efficiency, including RWKV (Peng et al., 2023) with linear recurrence and State-Space models like Mamba (Gu and Dao, 2024) and S4 (Gu et al., 2021). While these approaches show promise, they require fundamental architectural changes. Our work instead focuses on optimizing the attention mechanism itself, preserving the proven benefits and versatility of transformer architectures while reducing computational costs.

Several approaches focus on reducing model redundancy. Voita et al. (2019) demonstrate that multi-head SDPA is over-parameterized, leading to collaborative frameworks that reduce projection sizes (Cordonnier et al., 2020). Similarly, sparsification techniques reduce non-zero elements in weights, with recent work achieving 1-10% compression with minimal performance impact (Ashkboos et al., 2024), though potentially affecting robustness (Timpl et al., 2022). While these approaches focus on post-hoc optimization or pruning, our work fundamentally reimagines the attention mechanism's structure to achieve efficiency by design. We discuss further related attempts (including LoRA, Quantization and Flash Attention) for facilitating the deployability of transformers in §C.

6 Discussion and Conclusions

We proposed and evaluated three variants of SDPA that alter the standard arrangement of linear transformations to achieve better performance per computation cost and number of parameters (see Fig. 1 for visualizations). Optimized and Efficient Attention replace one (values) and two (values and keys) linear transformations with slicing, resulting in 25% and 50% size reductions and fewer matrix multiplications, respectively. The third variant, Super Attention, introduces a new linear transformation operating on the values from the left. While Super Attention can be applied to standard, Optimized, or Efficient Attention, we combined it with Efficient Attention, resulting in approximately 25% fewer parameters compared to standard attention.

Our evaluation spanned a wide range of tasks, including sentiment classification on IMDB and Amazon Reviews, Machine Translation on combined Europarl and Anki datasets, generative LM on OpenWebText dataset and NLI on HellaSwag. We used benchmarks varying in size from 50,000 examples to 9 billion tokens. To verify if these architectural benefits generalize across modalities, we also evaluated all variants for image classification on MNIST, CIFAR100, and ImageNet1K.

The experimental results demonstrate that Optimized and Efficient Attention performed comparably to standard attention across different benchmarks, despite having 25-50% fewer parameters and being faster. Super Attention consistently outperformed standard variant in all applicable benchmarks, achieving improvements of up to 10% on CIFAR100 and 4% on Amazon, while maintaining fewer parameters and faster training and inference.

Our generative LM experiment using a 1.1B Llama-based model in §B.5 provides insight into these variants' performance at larger scales. Yet realizing their true potential requires evaluation at even larger scales, which are beyond our computational resources. The promising results suggest these attention variants could open new pathways for training and deploying capable models on devices with limited computational resources, like smartphones and small personal devices.

Limitations

There are two limitations in this paper. First, Super Attention supports fixed context length due to the fixed size of W^A (see Eq. (22) and Fig. 1d). Nonetheless, these do not affect the advantages of Super Attention in many SotA applications such as in ViT. Moreover, this can be addressed using a sliding window, which is a future work currently in progress. Second, because of limited computational resources, we could only validate our hypotheses on models with up to 124 million (1.1 billion considering the language model trained in §B.5) parameters trained on datasets with up to 9 billion (30 billion considering §B.5) tokens. Further scaling the experiments beyond our computational resources and training large multi-modal and language models using the proposed mechanisms could facilitate a better understanding of their performance on industrial scales.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, et al. 2023. GPT-4 technical report. 1

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, et al. 2023. Gemini: a family of highly capable multimodal models. ArXiv preprint arXiv:2312.11805. 1

Anki.net. `https://ankisrs.net`. 5

Saleh Ashkboos, Maximilian L. Croci, Marcelo Gennari do Nascimento, Torsten Hoefler, and James Hensman. 2024. Slicegpt: Compress large language models by deleting rows and columns. In *12th International Conference on Learning Representations, ICLR*. OpenReview.net. 1, 8

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR*. OpenReview.net. 1

Iz Beltagy, Matthew E. Peters, and Arman Cohan. 2020. Longformer: The long-document transformer. ArXiv preprint arXiv:2004.05150. 7

Shangyu Chen, Wenya Wang, and Sinno Jialin Pan. 2019. Deep neural network quantization via layer-wise optimization using limited training data. In *AAAI Conference on Artificial Intelligence*, pages 3329–3336. AAAI Press. 16

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. ArXiv preprint arXiv:1904.10509. 7

Edward Choi, Mohammad Taha Bahadori, Jimeng Sun, Joshua Kulas, Andy Schuetz, and Walter F. Stewart. 2016. RETAIN: an interpretable predictive model for healthcare using reverse time attention mechanism. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 3504–3512. 1

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. 2021. Rethinking attention with performers. In *9th International Conference on Learning Representations, ICLR*. OpenReview.net. 7

Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. 2020. Multi-head attention: Collaborate instead of concatenate. *arXiv preprint arXiv:2006.16362*. 8

Tri Dao. 2024. Flashattention-2: Faster attention with better parallelism and work partitioning. 1, 16

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. In *Advances in Neural Information Processing Systems, NeurIPS*, volume 35, pages 16344–16359. Curran Associates, Inc. 1, 16

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: Efficient finetuning of quantized llms. ArXiv preprint arXiv:2305.14314. 1, 16

Payal Dhar. 2020. The carbon impact of artificial intelligence. *Nature Machine Intelligence*, 2(8):423–425. 1

Yifu Ding, Haotong Qin, Qinghua Yan, Zhenhua Chai, Junjie Liu, Xiaolin Wei, and Xianglong Liu. 2022. Towards accurate post-training quantization for vision transformer. In *30th ACM International Conference on Multimedia, MM*, pages 5380–5388. ACM. 16

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. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. In *9th International Conference on Learning Representations, ICLR*. OpenReview.net. 1

Aaron Gokaslan and Vanya Cohen. 2019. Openwebtext corpus. `http://Skylion007.github.io/OpenWebTextCorpus`. 5

Albert Gu and Tri Dao. 2024. Mamba: Linear-time sequence modeling with selective state spaces. In *First Conference on Language Modeling*. 8

Albert Gu, Karan Goel, and Christopher Ré. 2021. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*. 8

Kartik Gupta and Thalaiyasingam Ajanthan. 2022. Improved gradient-based adversarial attacks for quantized networks. In *AAAI Conference on Artificial Intelligence*, pages 6810–6818. AAAI Press. 16

Sanghyun Hong, Michael-Andrei Panaitescu-Liess, Yigitcan Kaya, and Tudor Dumitras. 2021. Qu-antization: Exploiting quantization artifacts for achieving adversarial outcomes. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 9303–9316. Curran Associates, Inc. 16

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *10th International Conference on Learning Representations, ICLR*. OpenReview.net. 1, 16

Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew G. Howard, Hartwig Adam, and Dmitry Kalenichenko. 2018. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR*, pages 2704–2713. Computer Vision Foundation / IEEE. 1, 16

Andrej Karpathy. 2022. nanogpt. https://github.com/karpathy/nanoGPT. 2, 5

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Machine Translation Summit, MTSummit*, pages 79–86. 5

Alex Krizhevsky. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto. 6

Yann LeCun, Corinna Cortes, Chris Burges, et al. 2010. Mnist handwritten digit database. http://yann.lecun.com/exdb/mnist. Accessed: 2020-06-13. 6

Zhenhua Liu, Yunhe Wang, Kai Han, Wei Zhang, Siwei Ma, and Wen Gao. 2021. Post-training quantization for vision transformer. In *Advances in Neural Information Processing Systems, NeurIPS*, volume 34, pages 28092–28103. Curran Associates, Inc. 16

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *The 49th Annual Meeting of the Association for Computational Linguistics, ACL*, pages 142–150. The Association for Computer Linguistics. 5

Carl D. Meyer. 2023. *Matrix Analysis and Applied Linear Algebra*, 2nd edition. Other Titles in Applied Mathematics. SIAM. 2

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory F. Diamos, Erich Elsen, David García, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. 2018. Mixed precision training. In *6th International Conference on Learning Representations, ICLR*. OpenReview.net. 16

Alexander Mott, Daniel Zoran, Mike Chrzanowski, Daan Wierstra, and Danilo Jimenez Rezende. 2019. Towards interpretable reinforcement learning using attention augmented agents. In *Advances in Neural Information Processing Systems, NeurIPS*, pages 12329–12338. 1

Markus Nagel, Marios Fournarakis, Yelysei Bondarenko, and Tijmen Blankevoort. 2022. Overcoming oscillations in quantization-aware training. In *39th International Conference on Machine Learning, ICML*, volume 162 of *Proceedings of Machine Learning Research*, pages 16318–16330. PMLR. 16

Jianmo Ni, Jiacheng Li, and Julian McAuley. 2019. Justifying recommendations using distantly-labeled reviews and fine-grained aspects. In *Empirical Methods in Natural Language Processing EMNLP*, pages 188–197. 5

Bo Peng, Eric Alcaide, Quentin Anthony, et al. 2023. RWKV: Reinventing RNNs for the transformer era. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14048–14077. Association for Computational Linguistics. 8

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the limits of transfer learning with a unified text-to-text transformer. ArXiv preprint arXiv:1910.10683. 16

Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. 2015. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision, IJCV*, 115(3):211–252. 6

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. 2024. Flashattention-3: Fast and accurate attention with asynchrony and low-precision. In *Advances in Neural Information Processing Systems, NeurIPS*. Curran Associates, Inc. 1

Lukas Timpl, Rahim Entezari, Hanie Sedghi, Behnam Neyshabur, and Olga Saukh. 2022. Understanding the effect of sparsity on neural networks robustness. ArXiv preprint arXiv:2206.10915. 8

Hugo Touvron, Thibaut Lavril, Gautier Izacard, et al. 2023a. Llama: Open and efficient foundation language models. ArXiv preprint arXiv:2302.13971. 1

Hugo Touvron, Louis Martin, Kevin Stone, et al. 2023b. Llama 2: Open foundation and fine-tuned chat models. ArXiv preprint arXiv:2307.09288. 1

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems, NeurIPS*, pages 5998–6008. Curran Associates, Inc. 1, 2

Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5797–5808. Association for Computational Linguistics. 8

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. 2020. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*. 7

Peng Xu, Xiatian Zhu, and David A. Clifton. 2023. Multimodal learning with transformers: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 45(10):12113–12132. 1

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5

Pengchuan Zhang, Xiyang Dai, Jianwei Yang, Bin Xiao, Lu Yuan, Lei Zhang, and Jianfeng Gao. 2021a. Multi-scale vision longformer: A new vision tansformer for high-resolution image encoding. In *IEEE/CVF International Conference on Computer Vision, ICCV*, pages 2978–2988. IEEE. 7

Zhaoyang Zhang, Wenqi Shao, Jinwei Gu, Xiaogang Wang, and Ping Luo. 2021b. Differentiable dynamic quantization with mixed precision and adaptive resolution. In *38th International Conference on Machine Learning, ICML*, volume 139, pages 12546–12556. Curran Associates, Inc. 16

A Reproducibility Statement

The code for all experiments is provided in the supplementary materials. Publicly available datasets are used, with automatic downloads included in the code, except for the Amazon dataset (link in README). The NanoGPT repository (linked in Experimental Setup) details the generative language modelling experiment. Further implementation details are in Section §4 and §B.3 and B.4.

B Additional Experiments

B.1 Edge Device Performance

Our main motivation for introducing Optimized, Efficient, and Super Attention is to allow running more capable models on edge devices. We calculated the inference times of the Transformer models, we trained before, on a MacBook Pro with an M2 Chip for each task/attention mechanism in Tbl. 5. As expected, Efficient models are the fastest. Also, Super Attention and Optimized Attention models are faster than their standard counterparts with the same number of heads while performing equally well as we discussed before.

B.2 Speed and Efficiency Comparison

In the main body and other sections of the Appendix, we present comprehensive theoretical comparisons and rigorous experiments on Vision and NLP classification tasks as well as for English-to-Spanish translation to compare the attention algorithms. Optimized Attention and Efficient Attention perform on par with standard attention with 25% and 50% less parameters respectively. In addition, Super Attention outperformed all other algorithms significantly while having 25% fewer parameters compared to standard attention.

As mentioned in the main body, according to the definitions of our proposed algorithms, Efficient, Optimized, and Super Attention mechanisms perform 2,1, and 1 fewer matrix multiplication per head compared to standard attention respectively. Here, we further analyze and compare the required number of FLOPs for completing a single forward and backward pass for all algorithms under study to gain further insight into the efficiency of the proposed algorithms.

FLOPs Versus Projection Dim. As depicted in Fig. 5, we compare the number of required FLOPs by each attention algorithm when we fixate the

Table 5: Total inference times (in seconds) for each attention mechanism/dataset pair on an Apple M2 chip over 5,000 samples.

Name	h	MNIST	CIFAR100	ImageNet	IMDB	Amazon
Standard	1	4.43	34.84	299.26	0.114	0.53
	4	5.27	46.06	323.84	0.183	0.87
	8	6.89 (4)	62.08 (4)	341.69 (4)	0.266 (4)	1.34 (4)
Optimized	1	4.19	33.36	281.14	0.109	0.47
	4	5.22	44.17	301.30	0.176	0.76
	8	6.37 (2)	60.63 (2)	320.49 (3)	0.262 (2)	1.21 (2)
Efficient	1	3.78	31.50	259.71	0.101	0.44
	4	4.71	42.16	276.15	0.170	0.72
	8	**6.10 (1)**	**58.60 (1)**	**301.24 (1)**	**0.256 (1)**	**1.14 (1)**
Super	1	4.21	33.69	264.99	0.112	0.46
	4	5.07	44.47	284.49	0.178	0.74
	8	6.65 (3)	60.73 (3)	309.72 (2)	0.264 (3)	1.19 (2)

sequence length (denoted as ℓ) and vary the projection dimension. Even though the number of FLOPs scales linearly with the projection dimension for all algorithms, the slope of this increase differs significantly for each algorithm. Specifically, for Efficient Attention, the slope of the line is equal to 9ℓ while for both Optimized and Super Attention this is equal to 12ℓ compared to 15ℓ for standard attention. This means that as we scale the projection dimension the FLOPs required for finishing a forward and backward pass using Efficient Attention increases 3/5 as fast as standard attention.

FLOPs Equation. The number of FLOPs required for finishing a forward and backward pass for each of the attention mechanisms is calculated according to the following equation:

$$\text{FLOPs} = C_{\text{Attn}}\ell d_m + 15h\ell^2 \tag{24}$$

where C_{Attn} is the attention algorithm constant which is 15 for standard attention, 12 for Optimized and Super Attention, and 9 for Efficient Attention, and ℓ, d_m, and h represent the sequence length, projection dimension, and number of heads consistent with the notation used throughout the paper.

Fig. 2 shows the 3D plot summarizing the number of FLOPs for each attention algorithm under varying sequence length and projection dimension in the single head setting. As evident in Fig. 2 and Eq. (24), our proposed algorithms need fewer FLOPs as sequence length increases, which is an important consideration for use in LLMs.

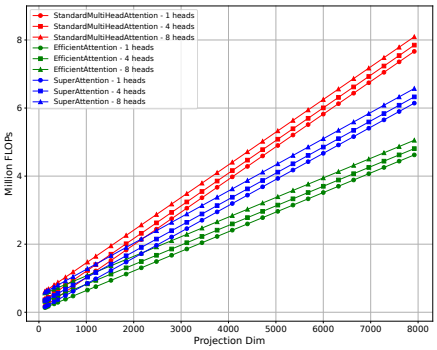


Figure 5: Number of Flops required to complete a single forward plus backward pass for each attention mechanism. While the complexity and therefore, the number of FLOPs increases linearly as the projection dimension increases for all attention mechanisms, the slope of the increase varies significantly as depicted in this plot. Efficient Attention and Super Attention (Optimized Attention is not shown as it is exactly similar to Super Attention) require significantly fewer FLOPs as the projection dimension increases compared to standard attention. Here sequence length is set to 64 ($\ell = 64$). Trying different values for ℓ changes the scale of the y-axis but the chart looks the same.

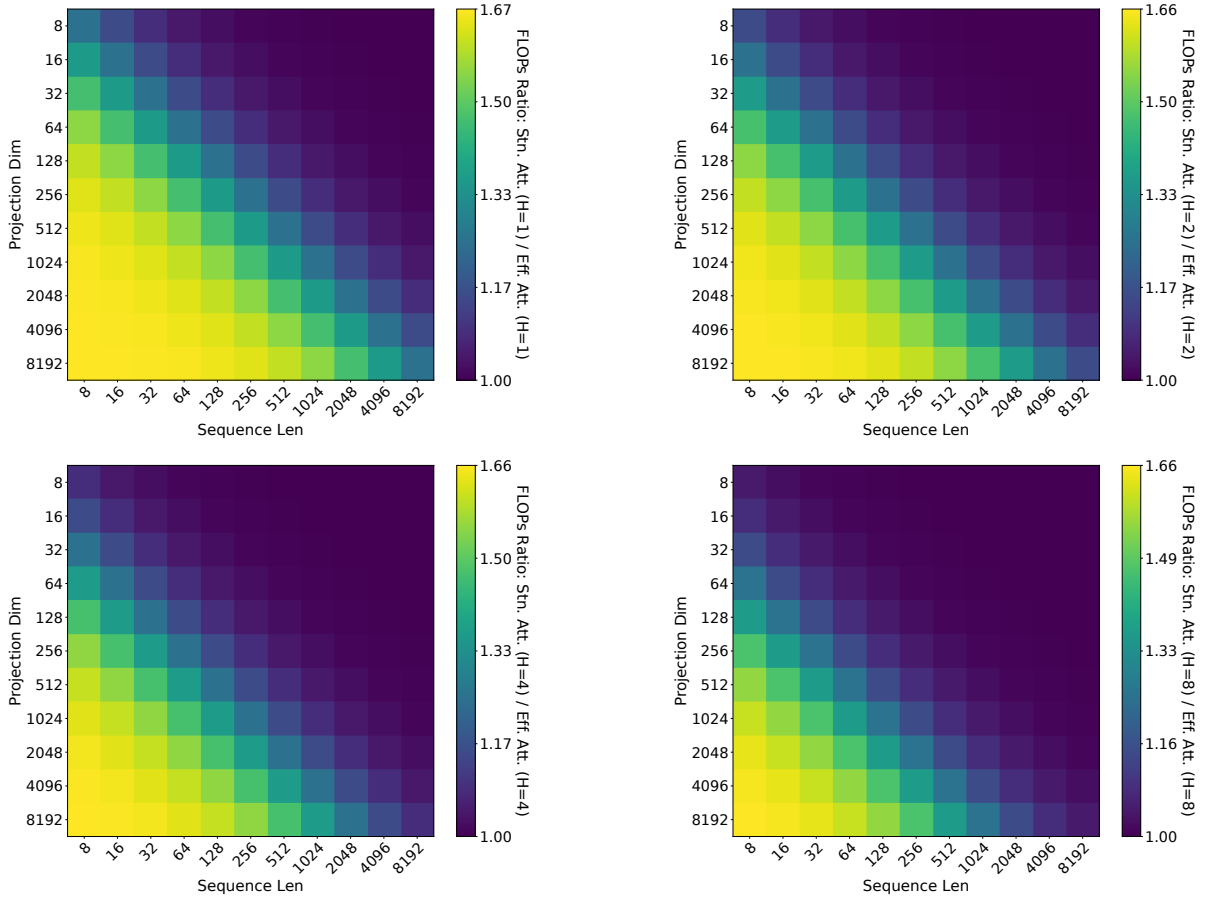


Figure 6: Heatmaps showing the ratio of FLOPs Standard Attention requires compared to the Efficient Attention in 1, 2, 4, and 8 attention head settings. Standard attention requires up to 67% more FLOPs to complete a single forward and backward pass. On average, standard attention requires 30%, 25%, 20%, and 16% more FLOPs than Efficient Attention when using 8, 4, 2, and 1 heads respectively.

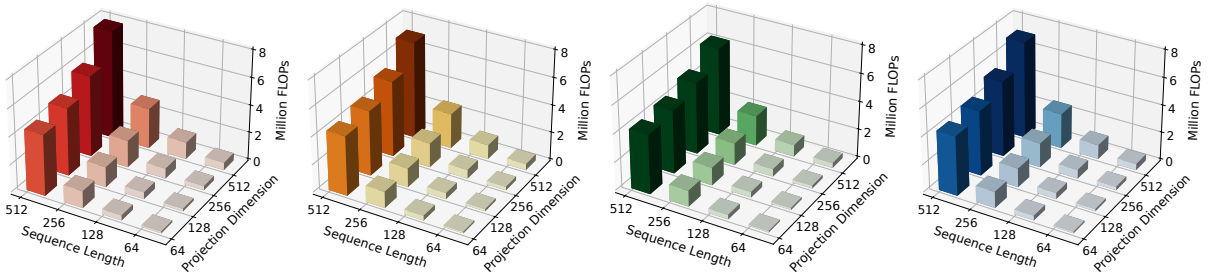
FLOPs Heatmaps. In addition to the previous analyses, in Fig. 6, we compare the ratio of FLOPs required to finish a single forward and backward pass by standard attention to Efficient Attention under different settings (i.e., varying sequence length and projection dimension) for different number of heads. In all scenarios, standard attention requires up to 66% more FLOPs in comparison to Efficient Attention. On average, Standard Efficient requires 30%, 25%, 20%, and 16% more FLOPs in comparison to Efficient Attention when using 1, 2, 4, and 8 heads, respectively.

B.3 Vision Transformers

MNIST. We trained ViT models with different attention mechanisms, all with two attention layers and model dimension $d_m = 128$. As expected, Super Attention outperforms all other architectures, in terms of accuracy, by at least 2.68% and standard attention by 3.23%. The smallest attention layer size belongs to Efficient Attention, which performs on par with standard attention. The complete results are presented in Tbl. 6.

ImageNet. Scaling the vision experiments even further, the ImageNet1k dataset presents much more complexity as the labels comprise 1000 classes. We used a modified ViT-B/16 model architecture, employed different attention mechanisms in its Transformers blocks, and trained the models. Due to our computational constraints, we reduced the number of transformer blocks from 12 to 8, resized the images to 112×112 (instead of the original 224×224) and reduced the patch size from 16 to 8 to enable training on our Nvidia RTX 4090 GPU. Other parameters are similar to the original architecture; specifically, $d_m = 768$ and $h = 12$. Tbls. 4 and 7 present the results of our experiments on the ImageNet dataset.

Val. results in Tbls. 4, 6 and 7 refer to models' performances on the official validation set for ImageNet1K, and the official tests sets for MNIST and



(a) Standard Att. (b) Optimized Att. (c) Efficient Att. (d) Super Att.

Figure 7: 3D plots visualizing the number of FLOPs for each variant in a forward + backward pass given different sequence lengths and projection dimensions in single-head setting. Efficient Att. followed by Super and Optimized Att. needs substantially fewer FLOPs for completing a forward + backward pass compared to standard attention.

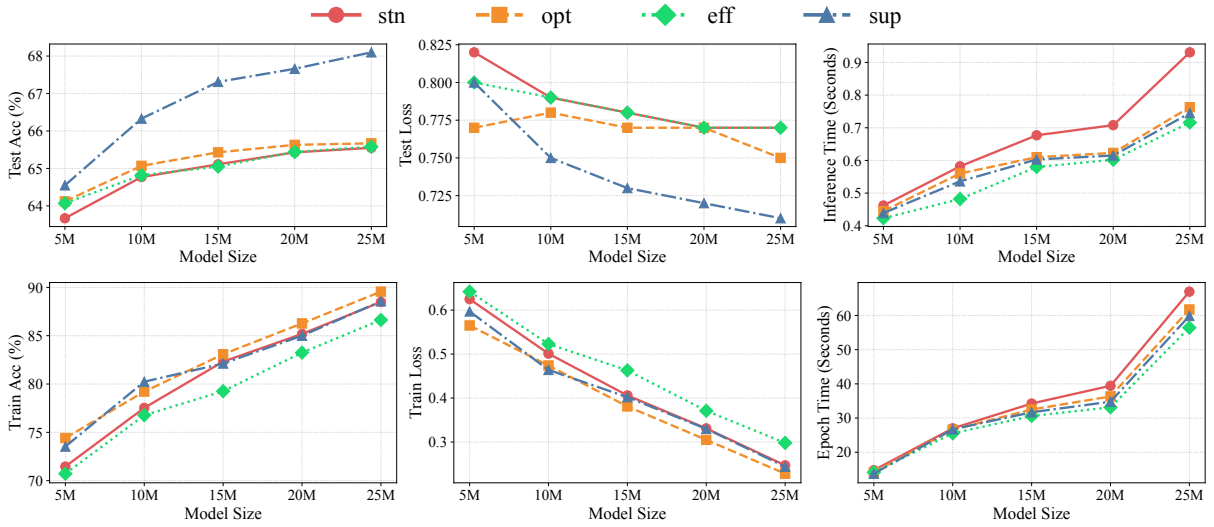


Figure 8: Performance of different architectures on the Amazon Reviews Classification task as the size of the models increases from 5 Million parameters to 25 Million parameters. The results point to the overparameterization of the Standard Attention as it puts an additional computational burden which is not accompanied with better performance in terms of accuracy or loss.

CIFAR100 datasets.

B.4 Natural Language Processing

B.4.1 Transformer for Text Classification

IMDB. The IMDB dataset includes 50,000 reviews with binary labels, indicating negative and positive sentiments. The Transformer models, used in this experiment, all have a single attention layer with model dimension and context length 32. The complete results are presented in Tbl. 8.

Amazon Reviews. The Amazon Reviews dataset poses a different challenge than the IMDB dataset as it is a significantly larger dataset with 3,650,000 reviews, containing a wider range of sentiments in $1, 2, \ldots, 5$; higher values indicate more positive sentiment. The Transformer models, used in this experiment, all have three attention layers with model dimension and context length 64. The complete results are presented in Tbl. 9.

B.4.2 Transformer for Machine Translation

Europarl Parallel Corpus and Anki. Anki dataset for English-Spanish translation consists of more than 118,000 sentence pairs in both English and Spanish languages. While training a model on this dataset enables basic translation, the educational nature and size of the dataset are too simple for training a capable translation model. Therefore, we also add the Europarl Parallel Corpus which has around 2 million examples in both English and Spanish languages and has sentences with much more technical and sophisticated terms to enable training in a powerful English-to-Spanish translation model. We then shuffle the mix of both datasets, and randomly split the dataset into 99.8%, 0.1%, and 0.1% for train, validation, and test splits respectively.

We then train a translation model inspired by the implementation available on the official Keras

Table 6: Averages of different metrics over five runs in the MNIST experiment. The numbers in parentheses indicate the ranking of each mechanism for that metric. An ablation study on the number of heads shows increasing the number of heads enhances the performance of all algorithms. As expected, the Efficient Attention model has the smallest attention layer size and the Super Attention model performs the best in terms of accuracy and loss.

Att.	h	d_m	d_k	# Param.	Avg. Time (s)	Acc. (%)	Loss	Val Acc. (%)	Val Loss
Stn.	1	128	128	66,048	8.15	93.26	0.227	98.02	0.063
	2	128	64	66,048	8.18	95.40	0.161	98.61	0.049
	4	128	32	66,048 (4)	8.31 (4)	93.73 (4)	0.209 (4)	98.12 (4)	0.062 (4)
Opt.	1	128	128	49,536	7.56	91.02	0.299	97.30	0.095
	2	128	64	49,536	7.57	93.70	0.215	97.93	0.071
	4	128	32	49,536 (3)	7.68 (3)	95.36 (2)	0.161 (2)	98.43 (2)	0.046 (2)
Eff.	1	128	128	33,024	6.89	93.29	0.228	97.78	0.073
	2	128	64	33,024	6.99	93.60	0.223	98.11	0.061
	4	128	32	**33,024 (1)**	**7.05 (1)**	94.28 (3)	0.197 (3)	98.27 (3)	0.058 (3)
Sup.	1	128	128	37,184	7.46	96.24	0.136	98.32	0.056
	2	128	64	37,184	7.50	96.59	0.124	98.52	0.050
	4	128	32	37,184 (2)	7.58 (2)	**96.96 (1)**	**0.112 (1)**	**98.62 (1)**	**0.051 (1)**

Table 7: Performance of different architectures on the ImageNet dataset. Since different attention layer architectures in the main ImageNet experiment had different numbers of parameters, an interesting ablation study is comparing these architectures when the total number of parameters is very close. To achieve this, we change some hyperparameters like d_m or the number of attention layers from the previous experiment. The numbers in parentheses indicate the ranking of each mechanism for that metric. We used a modified ViT-B/16 model, plugged in the attention algorithms in the Transformers block, and trained the models. Super Attention significantly outperforms all other algorithms. Unlike the results reported in Tbl. 4 in the main body, the models in this ablation experiment are not pre-trained on ImageNet21K (as such the accuracies and validation accuracies are lower compared to the ones with pre-training).

Att.	h	d_m	Att. Layers	Tot. # Param.	Acc. (%)	Loss	Top 5	Val Acc. (%)	Val Loss	Val Top 5
Stn.	12	768	8	60.54M (4)	51.18 (4)	2.09 (4)	76.05 (4)	32.74 (4)	3.36 (4)	56.48 (4)
Opt.	12	816	8	60.12M (2)	53.22 (2)	1.98 (2)	77.21 (2)	33.44 (3)	3.23 (3)	57.37 (3)
Eff.	12	804	9	**60.09M (1)**	51.28 (3)	2.06 (3)	76.66 (3)	**35.49 (1)**	**3.13 (1)**	**59.69 (1)**
Sup.	12	804	9	60.44M (3)	**64.98 (1)**	**1.37 (1)**	**87.36 (1)**	34.31 (2)	3.18 (2)	58.70 (2)

website for translation but with 2 decoder blocks and one encoder block for 6 epochs. Additionally, we set the $d_m = 1024$ and try 1, 2, and 4 as the number of heads. We use Sparse Categorical Cross Entropy as our loss metric. The complete analysis of the results is available in Tbl. 10.

All 3 algorithms perform comparably in terms of BLEU score, Accuracy, and Loss. However, the number of attention parameters per encoder/decoder layer is ½ and ¾ of standard attention in Efficient and Optimized Attention respectively. Additionally, Efficient attention is up to $^{(556.5-472.7)}/_{556.6} = 15.06\%$ faster to train in comparison to the standard attention.

B.5 Evaluation For Use in LLMs

In addition to evaluating the standard SDPA and its variants for generative language modelling in a scale of around 125M parameters, we also trained a Language Model (LM) with 1.1B parameters based on Efficient Attention architecture to see the feasibility and scalability of this variant of SDPA in a large scale experiment. This Language Model achieves lower loss than the similarly-sized TinyLlama model, which is based on Standard Attention (details are provided in Tbl. 11 below). We could not train more LMs based on other architectures due to our limited computational resources. The LM based on Efficient Attention was trained using a GPU credit donation that we used to train

Table 8: Averages of different metrics over five runs in the IMDB experiment. Here, varying the number of heads doesn't meaningfully affect the performance of any of the algorithms. As expected, the Efficient Attention model has the smallest attention layer size and the Super Attention model performs the best in terms of accuracy and loss.

Att.	h	d_m	d_k	# Param.	Avg. Time	Acc. (%)	Loss	Test Acc. (%)	Test Loss
Stn.	1	32	32	4,224	0.284	96.09	0.082	78.09	0.461
	2	32	16	4,224	0.297	95.51	0.112	78.14	0.467
	4	32	8	4,224 (4)	0.315 (4)	95.70 (4)	0.086 (3)	77.62 (4)	0.474 (4)
Opt.	1	32	32	3,168	0.283	96.62	0.070	78.00	0.461
	2	32	16	3,168	0.299	96.77	0.073	78.00	0.460
	4	32	8	3,168 (2)	0.305 (3)	96.31 (3)	0.095 (4)	77.85 (2)	0.472 (2)
Eff.	1	32	32	2,112	0.267	96.66	0.080	77.58	0.478
	2	32	16	2,112	0.273	96.86	0.068	77.74	0.473
	4	32	8	**2,112 (1)**	**0.280 (1)**	96.41 (2)	**0.064 (1)**	77.77 (3)	**0.468 (1)**
Sup.	1	32	32	3,168	0.272	97.68	0.063	78.21	0.472
	2	32	16	3,168	0.294	97.84	0.064	78.35	0.454
	4	32	8	3,168 (2)	0.299 (2)	**97.45 (1)**	0.070 (2)	**78.34 (1)**	0.472 (2)

our LM over 8 weeks on 30 billion tokens of C4 dataset (Raffel et al., 2019) using a single A100 with 80GB of GPU.

C Additional Related Work

Flash Attention (Dao et al., 2022) and Flash Attention 2 (Dao, 2024) optimize multi-head attention for modern GPUs without changing its structure, enabling faster processing and reduced memory demands. It's worth mentioning our proposed algorithms also benefit from these optimizations.

With the adoption of LLMs and Foundation Models (FMs), a lot of work has been done to improve their scalability and deployability. LoRA (Hu et al., 2022) adapts pre-trained models with minimal additional parameters, and QLoRA (Dettmers et al., 2023) incorporates quantization to reduce memory and computational demands.

Quantization has revolutionized the adoption of FMs, particularly those based on Transformers. Recent advances include mixed-precision post-training quantization for vision transformers (Liu et al., 2021), quantization-aware training (Jacob et al., 2018; Nagel et al., 2022), mixed-precision training (Micikevicius et al., 2018), dynamic quantization (Zhang et al., 2021b), and layer-wise quantization (Chen et al., 2019).

Moreover, Ding et al. (2022) unveiled a cutting-edge framework enhancing quantized model accuracy without significant performance degradation. However, quantization faces challenges such as potential performance drops and increased vulnerability to adversarial attacks (Hong et al., 2021; Gupta and Ajanthan, 2022).

Table 9: Averages of different metrics over five runs in the Amazon Reviews experiment. An ablation study on the number of heads shows increasing the number of heads helps improve the performance of all algorithms. The Efficient Attention model has the smallest attention layer size and the Super Attention model performs the best in accuracy and loss.

Att.	h	d_m	d_k	# Param.	Avg. Time	Acc.	Loss	Val Acc.	Val Loss
	1	128	128	66,048	42.06	86.76	0.31	62.32	0.87
Stn.	2	128	64	66,048	50.91	87.13	0.30	63.66	0.81
	4	128	32	66,048 (4)	66.97 (4)	88.49 (3)	0.25 (3)	65.55 (4)	0.77 (3)
	1	128	128	49,536	38.68	89.41	0.25	63.03	0.82
Opt.	2	128	64	49,536	47.97	90.48	0.23	65.98	0.75
	4	128	32	49,536 (3)	61.75 (3)	**89.56 (1)**	**0.23 (1)**	65.67 (2)	0.75 (2)
	1	128	128	33,024	34.82	88.56	0.27	63.42	0.81
Eff.	2	128	64	33,024	42.19	88.36	0.27	63.81	0.80
	4	128	32	**33,024 (1)**	**56.44 (1)**	86.63 (4)	0.29 (4)	65.58 (3)	0.77 (3)
	1	128	128	42,336	37.78	89.11	0.26	65.73	0.74
Sup.	2	128	64	42,336	46.24	88.41	0.27	67.22	0.73
	4	128	32	42,336 (2)	59.86 (2)	88.56 (2)	0.24 (2)	**68.10 (1)**	**0.71 (1)**

Table 10: Averages of different metrics over five runs trained on Europarl and Anki English-to-Spanish translation datasets. The numbers in parentheses indicate the ranking of each mechanism for that metric. An ablation study on the number of heads shows increasing the number of heads enhances the performance of all algorithms. Optimized and Efficient Attentions perform on par or better than Standard Attention on most benchmarks with 1/2 and 3/4 as many attention parameters.

Att.	h	d_m	d_k	# Param.	Avg. Time	BLEU	Acc.	Loss	Val BLEU	Val Acc.	Val Loss
	1	1024	1024	4,198,400	556.5	23.2	80.48	0.86	22.1	80.86	0.87
Stn.	2	1024	512	4,198,400	598.7	22.3	81.03	0.84	22.7	81.43	0.84
	4	1024	256	4,198,400 (3)	600.0 (3)	23.1 (2)	81.11 (3)	0.83 (3)	**22.8 (1)**	81.41 (3)	0.84 (3)
	1	1024	1024	3,148,800	552.0	22.5	81.15	0.87	22.6	81.11	0.84
Opt.	2	1024	512	3,148,800	583.8	22.1	81.61	0.82	23.0	81.57	0.82
	4	1024	256	3,148,800 (2)	586.8 (2)	**24.5 (1)**	**82.06 (1)**	**0.78 (1)**	22.6 (3)	**81.98 (1)**	**0.80 (1)**
	1	1024	1024	2,099,200	472.7	22.4	81.13	0.82	22.8	81.43	0.83
Eff.	2	1024	512	2,099,200	498.6	22.3	81.48	0.80	22.9	81.62	0.81
	4	1024	256	**2,099,200 (1)**	**523.0 (1)**	22.6 (3)	81.15 (2)	0.82 (2)	22.3 (3)	81.44 (2)	0.83 (2)

Table 11: A Language Model (Based on Efficient Attention) compared to TinyLlama (Based on Standard Attention) after training on 30 billion tokens of C4 dataset. We set the number of heads to 1 in this LM to make training faster. Despite this, this LM performs favourably (5.8% smaller categorical cross-entropy loss) compared to TinyLlama.

name	# layers	# heads	model dim	intermediate size	loss
TinyLlama	22	32	2048	5632	2.25
Efficient based LM	10	1	3072	8192	2.12