
Quantizable Transformers: Removing Outliers by Helping Attention Heads Do Nothing

Yelysei Bondarenko, Markus Nagel, Tijmen Blankevoort*

Qualcomm AI Research*

Amsterdam, The Netherlands

{ybond, markusn, tijmen}@qti.qualcomm.com

Abstract

Transformer models have been widely adopted in various domains over the last years, and especially large language models have advanced the field of AI significantly. Due to their size, the capability of these networks has increased tremendously, but this has come at the cost of a significant increase in necessary compute. Quantization is one of the most effective ways to reduce the computational time and memory consumption of neural networks. Many studies have shown, however, that modern transformer models tend to learn strong outliers in their activations, making them difficult to quantize. To retain acceptable performance, the existence of these outliers requires activations to be in higher bitwidth or the use of different numeric formats, extra fine-tuning, or other workarounds. We show that strong outliers are related to very specific behavior of attention heads that try to learn a “no-op” or just a partial update of the residual. To achieve the exact zeros needed in the attention matrix for a no-update, the input to the softmax is pushed to be larger and larger during training, causing outliers in other parts of the network. Based on these observations, we propose two simple (independent) modifications to the attention mechanism - *clipped softmax* and *gated attention*. We empirically show that models pre-trained using our methods learn significantly smaller outliers while maintaining and sometimes even improving the floating-point task performance. This enables us to quantize transformers to full INT8 quantization of the activations without any additional effort. We demonstrate the effectiveness of our methods on both language models (BERT, OPT) and vision transformers.

1 Introduction

Quantization has been one of the most impactful ways to reduce the computational complexity of transformer networks. Previous work has shown that quantizing networks to 4-bit weights is possible without losing too much accuracy [63, 66]. Some research even shows 4-bit weights might be optimal when trading off model size and bit-width [12].

However, quantizing transformers is not always trivial. When quantizing the activations of a transformer, significant problems arise with outliers in specific layers. This has been noted by several researchers that suggest fixes to transformers after training to ameliorate their effect [13, 64]. These methods are frequently tedious and either require retraining the network, require implementing specific hardware for input-channel quantization [13] or require parts of the activations to still be in higher bit-widths, reducing the effectiveness of the activation quantization [64].

In this paper, we set out to solve the transformer outlier problem entirely by changing the architecture of the network itself. We hope to make transformers easy to quantize from the get-go without needing any post-processing. To do so, we thoroughly analyze why these outliers appear. Previous work

*Qualcomm AI Research is an initiative of Qualcomm Technologies, Inc.

has found the existence of these outliers [4, 13], but in our work, we come to a fuller understanding of these outlying values. We find that the outliers occur because attention heads are trying not to update the hidden state, and in the process, strong outliers appear due to the softmax function. This happens for language and vision transformers and different specific transformer architectures. This understanding is the foundation for two new tweaks we suggest to transformer architectures that can remove the problem of the outliers entirely.

2 Background and related work

In this section, we briefly cover the basics of neural network quantization and discuss why modern transformers are difficult to quantize.

Quantization One of the most powerful ways to decrease the computational time and memory consumption of neural networks is quantization, which uses low-bit representations for the weights and activation tensors. On top of that, using low-bit fixed-point representations, such as INT8, one can further reduce energy consumption since the fixed-point operations are more efficient than their floating-point counterparts [22, 56].

We simulate the quantization process in floating-point according to Jacob et al. [25]. We use the following definition of the quantization function:

$$\hat{\mathbf{x}} := q(\mathbf{x}; s, z, b) = s \cdot \left(\text{clip} \left(\left\lfloor \frac{\mathbf{x}}{s} \right\rfloor + z; 0, 2^b - 1 \right) - z \right), \quad (1)$$

where \mathbf{x} denotes the quantizer input (i.e., network weights or activations), $s \in \mathbb{R}_+$ the scale factor or the step-size, $z \in \mathbb{Z}$ the zero point, and $b \in \mathbb{N}$ the bitwidth. $\lfloor \cdot \rfloor$ denotes the round-to-nearest-integer operator. This quantization scheme is called *uniform affine* or *asymmetric* quantization [23, 31, 73] and it is one of the most commonly used quantization schemes because it allows for efficient implementation of fixed-point arithmetic. In the case of *symmetric* quantization, we restrict the quantization grid to be symmetric around $z = 0$.

In this work, we focus on *post-training quantization* (PTQ) methods, which take a pre-trained FP32 network and convert it directly into a fixed-point network without the need for the original training pipeline [2, 5, 7, 24, 31, 34, 39, 41, 42, 72]. These methods require either no data or only a small calibration dataset and are easier to use compared to *quantization-aware training* (QAT, Bhalgat et al. 3, Esser et al. 16, Gupta et al. 20, Jacob et al. 25, Krishnamoorthi 31) methods that have you train the entire network for more epochs. For more details on neural network quantization, we refer the reader to [18, 43].

Outliers in Transformers Multiple studies have shown that modern transformer-based language models tend to learn outliers in weights and activations [4, 13, 30]. These outliers are present only in a small fixed set of embedding dimensions, but they appear regularly and consistently across multiple layers and data sequences. It was also shown that those outliers play a crucial role in the model predictions and clipping them or by setting to zero the corresponding parameters significantly degrades the model task performance [30, 46]. The strongest in magnitude outliers typically appear at the output of the feed-forward network, FFN, although Dettmers et al. [13] showed that for big enough transformer-based language models they start appearing after every linear layer, including query, key, and value projection layers. This phenomenon holds for many tasks, training objectives and models (both encoder and decoder transformers), including BERT [14], RoBERTa [35], DistilBERT [50], MobileBERT [52], ELECTRA [9], BART [32], XLNet [65], GPT-2 [47], and OPT [71].

Because of these strong outliers, applying per-tensor PTQ for the FFN’s output and the residual sum will likely cause a notable error because of the following trade-off between the range and the precision. On the one hand, using a large quantization range for small-ranged values leads to a loss in representation (high rounding error). On the other hand, a small quantization range for large values leads to a very high clipping error. For the case of significant transformer outliers, frequently, no good trade-off can be found between the rounding and clipping error, resulting in an overall high error.

There have been numerous attempts to fix the issue of transformer quantization [4, 12, 13, 17, 26, 27, 48, 51, 59, 60, 66, 68]. Most of these approaches resort to finer quantization granularity (row-wise, channel-wise, group-wise weight and activation quantization), use higher bitwidth and/or different

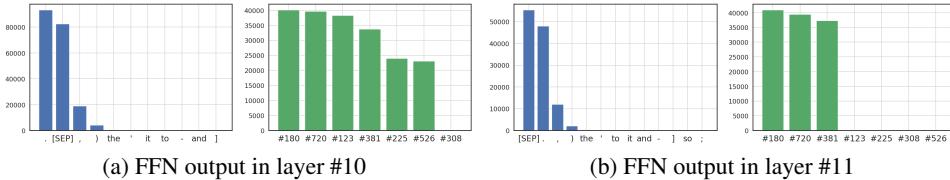


Figure 1: Histograms of outlier counts vs. token positions (blue) and hidden dimensions (green), recorded from the MNLI-m validation set on BERT-base. We use zero-based indexing for dimensions.

numeric format to represent those outliers better or require extra fine-tuning (in the form of QAT and/or knowledge distillation). In other words, they adapt quantization to work with outliers, which often comes at the expense of general applicability or extra inference overhead.

In contrast, in this work, we want to address the root cause of the problem and understand why outliers are learned in the first place and suggest a new pre-training protocol that significantly reduces the magnitude of outliers yielding way more quantization-friendly models that can be effortlessly quantized using PTQ without strong degradation of performance.

3 Outlier analysis

Outliers in BERT models In Section 2 we discussed that outliers are present only in a few designated embedding dimensions but they appear regularly and consistently across multiple layers and data sequences. We also discussed that the strongest magnitude outliers in BERT typically appear at the output of FFN in the last encoder layers.

We start by taking the pre-trained *BERT-base-uncased* checkpoint from HuggingFace [62] and fine-tune it on MNLI dataset from the well-known GLUE benchmark [58] (see experimental details in C.1). To identify the outlier dimensions, we pass the MNLI-m validation set through the network and record all outliers¹ at the FFN output in layers #10 and #11². As we can see in Figure 1, there are indeed only a few hidden dimensions where outliers ever occur. We also notice that the majority of outliers ($> 97\%$) correlate with the position of delimiter tokens – [SEP], ., and ,.

To better understand the role of those outliers, we analyze the attention patterns of the corresponding attention heads. BERT-base uses multi-head attention with $n_{\text{heads}} = 12$ and each head operating on a consecutive subset of $d_{\text{head}} = 64$ features. Therefore, the hidden dimension #180, which happens to have the highest outlier count in both layers #10 and #11, corresponds to attention head #3. In Figure 2 (and more examples in Appendix A.1) we show examples of the attention matrices, values and their product for that head.

A common pattern we found is that the attention head assigns almost all of its probability mass to [SEP] tokens, and other less informative tokens like dots/commas, while these tokens also have small values in V associated with those tokens. This results in a small magnitude product between the two (see Figure 2a). This effectively corresponds to a (soft) *no-update* of the hidden representation, where only small noise is added after the residual. In other cases (Figure 2b and 2c), we observe that a significant portion of attention probability is still spent on delimiter tokens. However, by allocating some of the probability mass on other tokens (together with the small values for the delimiter tokens), this results in a (soft) *selective* update of the hidden representation.

These patterns in self-attention seem to be a learned “workaround” for the limitations of having the softmax and the residual connections in cases where the attention head does not want to update the representation of some or all of the tokens. These observations are in line with Clark et al. [8], Kovaleva et al. [29] that also argued that attending exclusively or almost exclusively to delimiter tokens such as [SEP], periods/commas acts as a “no-op” when the attention head’s function is not applicable.

¹We follow Bondarenko et al. [4] and consider outliers as values that exceed 6 standard deviations from the mean of the corresponding activation tensor.

²We use 1-based indexing for encoder layers and attention heads throughout the paper.

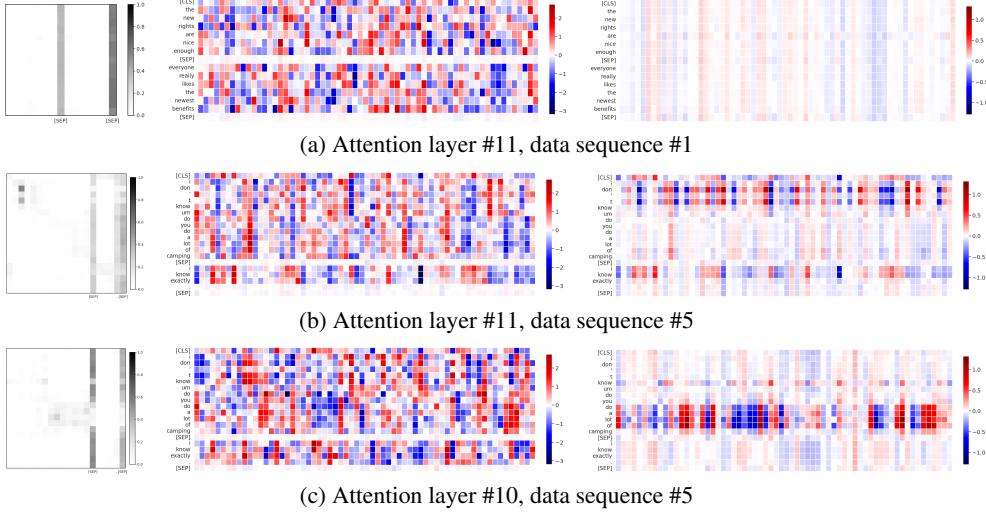


Figure 2: Visualization of the patterns in the self-attention, specifically the attention probabilities, values, and their product (left, middle and right columns, respectively), in attention head #3 for BERT-base, computed on several data sequences from MNLI-m validation set.

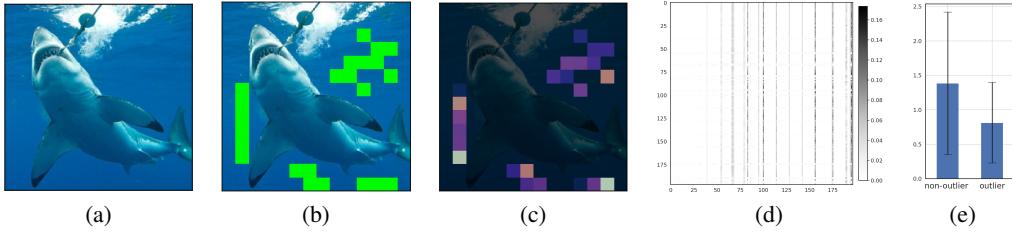


Figure 3: A summary of our outlier analysis for ViT demonstrated on a random image from ImageNet validation set. (a) An input image. (b) Outliers in the output of layer #11. (c) Cumulative attention weight spent on every patch (matrix of attention probabilities summed over rows) in the attention head #1, layer #12. (d) A corresponding matrix of attention probabilities. (e) An average magnitude of values for outlier and non-outlier patches.

Outliers in ViT We conduct a similar analysis for Vision transformer [15] trained on ImageNet [49]. For this study, we use a pre-trained checkpoint following our experimental setup from Section 5.

We highlight our findings in Figure 3 and provide more examples in Appendix A.2. Our analysis shows many similarities to the BERT case. Instead of delimiter tokens, the majority of outliers seem to correlate with some random uninformative patches (e.g., in the background). We also see that the corresponding attention head in the next layer allocates the majority of attention probabilities to the same patches. Finally, those outlier patches on average have a distinctly smaller magnitude of values compared to non-outlier ones, leading to similar no-update behavior. The fact that those values are not as close to zero as it was in the BERT case might be related to the smaller model capacity³, or a relatively shorter training procedure.

Hypothesis Based on these observations, we pose the following hypothesis on how this behavior of attention heads is related to outliers:

1. In order for an attention block to not update a representation of a token on the residual, some attention heads want to allocate most of their attention probability mass to some fixed and common set of tokens that have a low information content (e.g., delimiter tokens or background patches) that can be learned to have a small value function output.

³We use ViT/S-16 configuration that has only 22M parameters.

2. From the definition of the softmax function⁴, it is easy to see that this would require an input of the softmax to have a relatively big dynamic range (Figure 4, ①). In fact, in the limit case where softmax is exactly zero, this would require an infinite dynamic range:

$$\text{softmax}(\mathbf{x})_i = 0 \Leftrightarrow \exists j \neq i, \mathbf{x}_j - \mathbf{x}_i = +\infty \quad (2)$$

3. Since Layer Normalization ([1], ②) normalizes the outliers, the magnitude of the FFN output *in the previous layer* (③) has to be very high to still produce a sufficiently big dynamic range after the LayerNorm. Note, that this is also applicable for the transformer models with LayerNorm applied prior to the self-attention or linear transformations instead, a variant adopted by GPT, OPT, and many vision transformers [15, 36, 54, 55].
4. Finally, as softmax will never output exact zeros, it will always back-propagate a gradient signal to grow bigger outliers⁵. The outliers will thus tend to become stronger in magnitude, the longer the network is trained.

4 Method

In this section, we introduce our proposed modifications for the softmax attention mechanism. Based on our insights from Section 3, the core idea of these modifications is to grant the model the ability to produce very small the magnitude (or even exact zeros) output of attention function, without producing outliers.

Recall that the self-attention [57] is defined as follows:

$$\text{Attention}(\mathbf{x}) := \text{softmax} \left(\frac{\mathbf{Q}(\mathbf{x}) \mathbf{K}(\mathbf{x})^T}{\sqrt{d_{\text{head}}}} \right) \mathbf{V}(\mathbf{x}) \quad (3)$$

where \mathbf{Q} , \mathbf{K} and \mathbf{V} are learnable linear projections of the input \mathbf{x} . Most modern transformer models employ a multi-headed variant of self-attention, where d_{model} features are partitioned into n_{heads} groups of d_{head} features, and the final output is the concatenation of the outputs of (3) applied to each group.

4.1 Clipped softmax

First, we propose to replace softmax function in (3) with the following clipped softmax:

$$\begin{aligned} \text{clipped_softmax}(\mathbf{x}; \zeta, \gamma) := \\ \text{clip}((\zeta - \gamma) \cdot \text{softmax}(\mathbf{x}) + \gamma, 0, 1). \end{aligned} \quad (4)$$

Here \mathbf{x} is the input and $\zeta \geq 1$, $\gamma \leq 0$ are the stretch factors which are hyper-parameters of the method. This formulation was proposed before in [38] in the context of binary stochastic gates. We can view (4) as stretching the output of the softmax from $(0, 1)$ to (γ, ζ) and then clipping back to $(0, 1)$ so that we can represent exact zeros if $\gamma < 0$ and exact ones if $\zeta > 1$. Specifically, the values of the softmax larger than $\frac{1-\gamma}{\zeta-\gamma}$ are rounded to one whereas values smaller than $\frac{-\gamma}{\zeta-\gamma}$ are rounded to zero.

With this drop-in replacement, we can achieve exact zeros (and ones) with a finite range for the softmax input. In addition to that, whenever values are clipped they will not give a gradient, preventing the outliers to grow further.

⁴ $\text{softmax}(\mathbf{x})_i = \exp(\mathbf{x}_i) / \sum_{j=1}^d \exp(\mathbf{x}_j)$

⁵Let $\mathbf{y} = \text{softmax}(\mathbf{x})$. If $\mathbf{y}_i > 0$, then $\frac{\partial \mathbf{y}_i}{\partial \mathbf{x}_j} \neq 0 \forall j$.

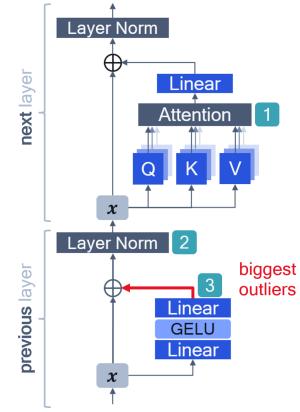


Figure 4: A schematic illustration of the attention layer in BERT. Hidden activation tensor is denoted by \mathbf{x} . \oplus is an element-wise addition. A problematic output of the FFN that generates largest in magnitude outliers is highlighted in red. Notice how those outliers in the *previous layer* influence the behavior in the attention mechanism in the *next layer*.

4.2 Gated attention

An alternative way of architecting the model to have a small attention output without outliers is to equip it with an explicit conditional gating mechanism, as shown in Figure 5. The idea is that the model can use the gating to either keep or nullify the update to the representation of certain tokens and not rely on the attention probabilities and values to achieve the same outcome.

Specifically, we propose the following modification to the attention function:

$$\text{Gated_attention}(\mathbf{x}) := \text{sigmoid}(\mathbf{G}(\mathbf{x})) \odot \text{softmax}\left(\frac{\mathbf{Q}(\mathbf{x})\mathbf{K}(\mathbf{x})^T}{\sqrt{d_{\text{head}}}}\right)\mathbf{V}(\mathbf{x}). \quad (5)$$

Here \mathbf{G} is the gating function, \odot is an element-wise multiplication across the token axis and everything else remains the same as in (3). The gating function \mathbf{G} is parameterized by a small neural network that is learned jointly with the rest of the model. We replace the attention formulation with the proposed variant in every layer on the transformer network.

Gating module design Recall that the input to the attention layer \mathbf{x} has shape (T, d_{model}) that is reshaped into $(n_{\text{heads}}, T, d_{\text{head}})$ for the multi-headed self-attention, where T is the sequence length. We chose to define the gating function on a per-head basis. For each head $i \in \{1, \dots, n_{\text{heads}}\}$, we specify $\mathbf{G}_i : \mathbb{R}^{d_{\text{head}}} \rightarrow \mathbb{R}$ and the output of the gating module is $\pi_i \in \mathbb{R}^T$ that is computed as follows:

$$\hat{\pi}_{i,t} = \mathbf{G}_i(\mathbf{x}_{i,t,:}) \quad \forall t \in \{1, \dots, T\} \quad (6)$$

$$\pi_{i,:} = \text{sigmoid}(\hat{\pi}_{i,:}), \quad (7)$$

note that gating modules are shared between different token positions but not shared across attention heads.

We want our gating module to be as lightweight as possible. To start with, we experiment with \mathbf{G}_i 's parameterized by a single linear layer. This gives us a gating module that is computationally inexpensive and has a memory overhead of just $n_{\text{heads}} \cdot (d_{\text{head}} + 1) \sim d_{\text{model}}$ extra parameters (which is equivalent to 1 extra token) per attention layer⁶. We also investigate the effect of using several other gating functions in Appendix B.1.

5 Experiments

In this section, we evaluate the proposed modifications to self-attention on several language models (BERT, OPT) and the vision transformers (ViT). We first test the different hyperparameters for the methods and provide insight into how they work. Then we set out to test our method in terms of accuracy, and the difference in quantization improvement after training. All detailed hyperparameters of our experiments are in Appendix C.

BERT We experiment with BERT-base-uncased (109M parameters) pre-training using the masked language modeling (MLM) objective. Following [14], we use the concatenation of the training sets of BookCorpus [74] and English Wikipedia⁷. We implement our methods in PyTorch [45] and use training and evaluation pipelines from HuggingFace libraries [19, 33, 62]. We follow closely the pre-training procedure from [14]. To speed up training and experimentation, we train with a maximum sequence length of 128 for the whole duration of the training. We evaluate on Wikipedia validation set and report the MLM perplexity.

OPT We experiment with a 125M sized variant of OPT [71] pre-training using the causal language modeling (CLM) objective. Due to compute constraints, we train the model on the same dataset that was used for BERT pre-training (BookCorpus + Wikipedia) with a maximum sequence length of 512

⁶For instance, in case of BERT-base, this amounts to less than 0.009% of the total model size.

⁷Specifically, we use the English subset of Wiki-40b, <https://huggingface.co/datasets/wiki40b>, that contains cleaned-up text of English Wikipedia and training/validation splits.

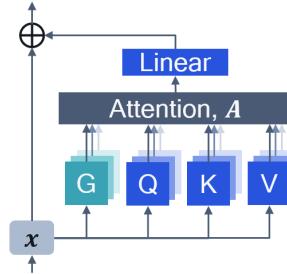


Figure 5: A schematic illustration of our proposed gated attention.

γ	ζ	FP16 ppl. \downarrow	Max inf. norm	Avg. kurtosis	W8A8 ppl. \downarrow
0 (= Vanilla)	1	4.49 ± 0.01	735 ± 55	3076 ± 262	1294 ± 1046
0	1.003	4.48 ± 0.01	715 ± 335	2159 ± 238	451 ± 57
0	1.03	4.49 ± 0.00	741 ± 66	1707 ± 1249	1469 ± 646
-0.003	1	4.46 ± 0.00	688 ± 64	2149 ± 110	636 ± 566
-0.03	1	4.41 ± 0.01	20 ± 1	80 ± 6	4.55 ± 0.01
-0.003	1.003	4.47 ± 0.00	683 ± 23	2494 ± 1205	268 ± 120
-0.03	1.03	4.43 ± 0.03	22 ± 3	73 ± 8	4.56 ± 0.05

Table 1: The impact of clipped softmax hyperparameters on BERT-base.

and batch size of 192. Similar to our BERT experiments, we use training and evaluation pipelines from HuggingFace libraries. We evaluate on Wikipedia validation set and report the CLM perplexity.

ViT Finally, we explore the effectiveness of proposed techniques on vision transformer [15] (*ViT-S/16* configuration, 22M parameters) trained on ImageNet-1K [11, 49]. For these experiments, we adopt the training and validation pipelines from PyTorch Image models library [61]. We report top-1 accuracy on the validation set of ImageNet.

Quantization setup In all experiments, after the model is trained, we apply 8-bit PTQ. We use uniform affine quantization – symmetric weights, asymmetric activations – with the static activation range setting, as discussed in Section 2. We quantize all weights and activations (both input and output), except the final linear layer for BERT and OPT models. We explore several choices of range estimation (see Appendix C.4) and report the best configuration for each experiment, based on the model performance. We repeat each PTQ experiment 3 times with different random seeds⁸ and report mean and standard deviation for accuracy/perplexity.

We train each network two times with different random seeds and report mean and standard deviation. To assess the amount of outliers in the trained model, we use two metrics: the maximum $\|\mathbf{x}\|_\infty$ averaged across the validation set, and *kurtosis* of \mathbf{x} averaged across all layers, where \mathbf{x} is the output of an attention layer. These metrics have been shown to correlate well with the model quantizability [4, 6].

5.1 The impact of clipped softmax hyperparameters (γ and ζ)

We investigate the effect of different values of the clipped softmax stretch parameters and present the results in Table 1. We can see that most of the improvement happens when we use $\gamma < 0$ (clipping at zero). For instance, using the value of $\gamma = -0.03$ leads to a significantly smaller infinity norm, kurtosis, and quantized model perplexity, compared to the baseline. It is also clear that in the limit $|\gamma| \rightarrow 0$ we approach the vanilla softmax attention. Using $\zeta > 1$ (clipping at one) yields similar results to the vanilla softmax. Finally, when we combine both $\gamma < 0$ and $\zeta > 1$, for which the results seem similar to just clipping at 0. We, therefore, conclude that for dampening outliers, only the lower-range clipping allows exact zeros matter. Going forward we use only $\gamma < 0$ and in Appendix B.5 we confirm that $\zeta > 1$ is not required for ViT.

These observations are in line with our hypothesis that by giving the model the mechanism for representing exact zeros in the attention, we don't need to learn the strong outliers.

5.2 Clipped softmax γ vs. sequence length

As having an extra hyper-parameter that needs to be tuned per model or setup is generally not desirable, we study the sensitivity of the stretch factor γ and its relation with the sequence length T . Recall that the matrix of attention probabilities \mathbf{P} has dimensions $T \times T$ and each row sums up to one. Because of that, the average value in \mathbf{P} is $1/T$. It is reasonable to assume that if we define

⁸Different random subsets of training data are used for quantizer range estimation.

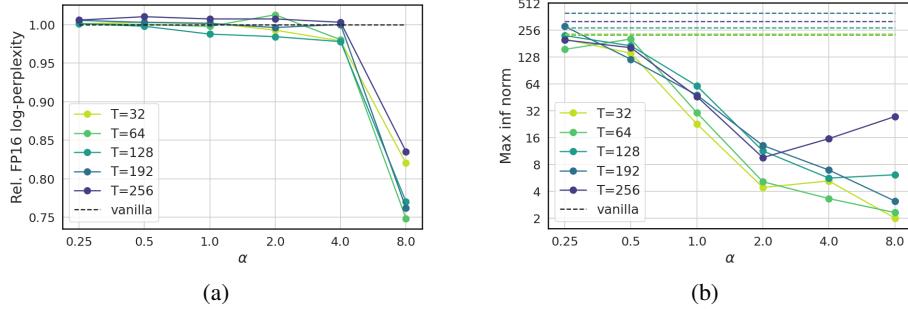


Figure 6: The performance of clipped softmax using $\gamma = -\alpha/T$ parameterization on BERT-6L. (a) Relative (compared to vanilla softmax pre-training) FP16 log-perplexity \uparrow on Wikitext validation set. (b) Maximum infinity norm of the attention layer output (note the logarithmic y-axis).

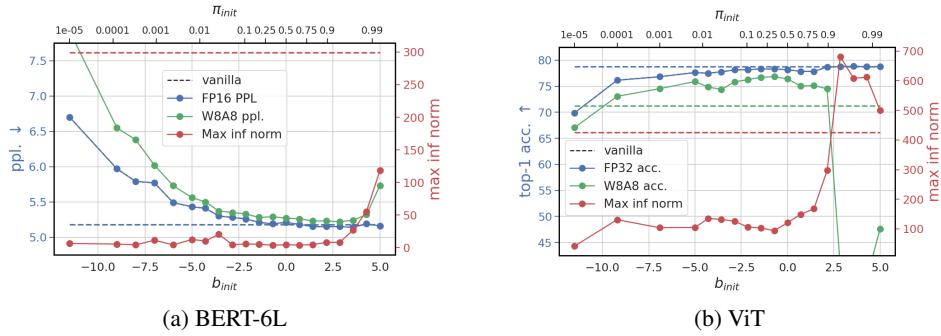


Figure 7: The performance of Linear gated attention using different bias initialization settings.

$\gamma := -\frac{\alpha}{T}$, where $\alpha > 0$ is a new hyperparameter, there might be a set or a range of values of α that works well across different sequence lengths.

To study this, we train a 6-layer variant of BERT-base (BERT-6L) for 500000 steps on WikiText-103 [40] with a batch size of 128 with several values of maximum sequence lengths $T \in \{32, 64, 128, 192, 256\}$ and values of $\alpha \in \{1/4, 1/2, 1, 2, 4, 8\}$. As we can see from Figure 6, using a clipped softmax with $\alpha \in [2, 4]$ significantly dampens the magnitude of outliers while maintaining good FP16 perplexity across all explored sequence lengths.

5.3 The impact of bias initialization in Gated attention

In all our gated attention experiments, we randomly initialize the weights of G , following [21]. By initializing the *bias* to a specific value, however, we can set gates to be more *open* or more *closed* initially. More open at the start means we initialize closer to the original network, but given the exponential nature of the gate it might take many iterations for the gate to learn to close. Similarly, if the gates are all closed at the start, we deviate too far from the original model training, causing a potential decrease in performance. Assuming Linear G_i 's with small initial weights, if we set the bias to the value of b_{init} , then $G_i(\cdot) \approx b_{init}$ and $\pi_i(\cdot) = \text{sigmoid}(G_i(\cdot)) \approx \text{sigmoid}(b_{init}) =: \pi_{init}$, at the start of training.

We study the effect of different values of b_{init} for Linear gated attention on BERT-6L and ViT. We set the bias for all G_i 's to the same value of b_{init} . For BERT-6L, we use the same setup as in Section 5.2, with a fixed sequence length of 128. For ViT, we use the main setup, except we train it for 150 epochs instead of 300.

In Figure 7 we see in both BERT and ViT cases that using bias with very high π_{init} generally performs similarly to the vanilla attention (comparable floating-point performance but strong outliers and poor quantized performance) while setting bias to have very low π_{init} dampens outliers quite well but leads to strong degradation in the floating-point and quantized performance. The reasonable ranges of π_{init}

Model	Method	FP16/32	Max inf. norm	Avg. kurtosis	W8A8
BERT (ppl. \downarrow)	Vanilla	4.49 ± 0.01	735 ± 55	3076 ± 262	1294 ± 1046
	Clipped softmax	4.39 ± 0.00	21.5 ± 1.5	80 ± 6	4.52 ± 0.01
	Gated attention	4.45 ± 0.03	39.2 ± 26.0	201 ± 181	4.65 ± 0.04
OPT (ppl. \downarrow)	Vanilla	15.84 ± 0.05	340 ± 47	1778 ± 444	21.18 ± 1.89
	Clipped softmax	16.29 ± 0.07	63.2 ± 8.8	19728 ± 7480	37.20 ± 2.40
	Gated attention	15.55 ± 0.05	8.7 ± 0.6	18.9 ± 0.9	16.02 ± 0.07
ViT (acc. \uparrow)	Vanilla	80.75 ± 0.10	359 ± 81	1018 ± 471	69.24 ± 6.93
	Clipped softmax	80.89 ± 0.13	73.7 ± 14.9	22.9 ± 1.6	79.77 ± 0.25
	Gated attention	81.01 ± 0.06	79.8 ± 0.5	19.9 ± 0.3	79.82 ± 0.11

Table 2: A summary of results for our proposed methods applied on BERT, OPT, and ViT.

seems to be around $[0.25, 0.9]$ for BERT and $[0.1, 0.5]$ for ViT. The wide range indicates the relative robustness of our method to this hyperparameter.

5.4 Main results

We summarize our main set of results in Table 2. As we can see, in almost all cases, both of our proposed techniques dampen the outliers’ magnitude to a great extent, reduce the kurtosis, and yield models with significantly higher quantized performance, which is close to the original FP16/32 performance. In addition to that, for each model, at least one of our methods also improves the floating-point task performance. We hypothesize this is because the network is helped with learning the “no-op” updates more easily. However, we are cautious about the improved performance as this is not consistent across all hyper-parameters and it is unclear if it generalizes to more architectures and larger models.

The only case where our method failed to perform well was the clipped softmax applied to OPT. At the moment, we do not have an explanation of why this is the case and leave it for future work. We list selected hyper-parameters and show extended results in Appendix B.

6 Discussion

“No-op” behavior It is interesting to note that the identified “no-op” behavior is likely not limited to transformers and that convolutional architectures likely learn something similar. We also see that despite the network trying to learn a full “no-op”, still a small amount of noise is added to each residual, which may constitute a form of network regularization. Investigating this further might give us a clue as to why neural networks generalize despite being significantly overparametrized if many parameters are rendered unused by not updating the representation in later layers [69].

Limitations We have not studied the effect of our method on large-scale transformers, as it would require training very expensive models from scratch. Given the fundamental understanding of the issue underlying our solutions, we expect the same effect on large-scale models. We show a very small improvement in FP16/FP32 performance due to our methods, but we do not deem our results exhaustive enough to claim that this will hold in general. Lastly, our methods do have a hyperparameter each, although we show that both methods are relatively robust to its hyperparameter, having one is never optimal.

Impact As our methods help transformers to be more efficient, we expect only positive outcomes of our work. Making neural networks more efficient will help with their high power consumption at inference. It further helps to move inference from the cloud to edge devices which can overcome potential privacy concerns. We cannot fathom any negative impact from our work that is not severely construed.

7 Conclusions

We have thoroughly analyzed the activation outlier problem that makes transformers difficult to quantize. We showed that transformer networks try to learn not to update residuals and that by doing so, through the combination of the softmax, residual connections and LayerNorm, significant outliers appear in transformers. Based on this insight, we proposed two methods to address this at the core – *clipped softmax* and *gated attention*. These structural changes to transformers give similar, if not better, floating-point performance after training but significantly improve the post-training quantization results. We hope that with these two architectural changes to transformers, anyone can train high-performance transformers that are easy to quantize and can benefit from efficient integer inference.

References

- [1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.
- [2] Ron Banner, Yury Nahshan, Elad Hoffer, and Daniel Soudry. Post-training 4-bit quantization of convolution networks for rapid-deployment. *arXiv preprint arXiv:1810.05723*, 2018.
- [3] Yash Bhalgat, Jinwon Lee, Markus Nagel, Tijmen Blankevoort, and Nojun Kwak. Lsq+: Improving low-bit quantization through learnable offsets and better initialization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2020.
- [4] Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Understanding and overcoming the challenges of efficient transformer quantization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7947–7969, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.627. URL <https://aclanthology.org/2021.emnlp-main.627>.
- [5] Yaohui Cai, Zhewei Yao, Zhen Dong, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Zeroq: A novel zero shot quantization framework. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13169–13178, 2020.
- [6] Brian Chmiel, Ron Banner, Gil Shomron, Yury Nahshan, Alex Bronstein, Uri Weiser, et al. Robust quantization: One model to rule them all. *Advances in neural information processing systems*, 33: 5308–5317, 2020.
- [7] Yoni Choukroun, Eli Kravchik, Fan Yang, and Pavel Kisilev. Low-bit quantization of neural networks for efficient inference. In *ICCV Workshops*, pages 3009–3018, 2019.
- [8] 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>.
- [9] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020.
- [10] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020.
- [11] 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. Ieee, 2009.
- [12] Tim Dettmers and Luke Zettlemoyer. The case for 4-bit precision: k-bit inference scaling laws. *arXiv preprint arXiv:2212.09720*, 2022.
- [13] Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. In *Advances in Neural Information Processing Systems*, 2022.

- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *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] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.
- [16] Steven K. Esser, Jeffrey L. McKinstry, Deepika Bablani, Rathinakumar Appuswamy, and Dharmendra S. Modha. Learned step size quantization. In *International Conference on Learning Representations (ICLR)*, 2020.
- [17] Angela Fan, Pierre Stock, Benjamin Graham, Edouard Grave, Rémi Gribonval, Hervé Jégou, and Armand Joulin. Training with quantization noise for extreme model compression. *arXiv preprint arXiv:2004.07320*, 2020.
- [18] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. *arXiv preprint arXiv:2103.13630*, 2021.
- [19] Sylvain Gugger, Lysandre Debu, Thomas Wolf, Philipp Schmid, Zachary Mueller, and Sourab Mangrulkar. Accelerate: Training and inference at scale made simple, efficient and adaptable. <https://github.com/huggingface/accelerate>, 2022.
- [20] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.
- [21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.
- [22] M. Horowitz. 1.1 computing’s energy problem (and what we can do about it). In *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*, pages 10–14, 2014. doi: 10.1109/ISSCC.2014.6757323.
- [23] Itay Hubara, Matthieu Courbariaux, Daniel Soudry, Ran El-Yaniv, and Yoshua Bengio. Quantized neural networks: Training neural networks with low precision weights and activations. *The Journal of Machine Learning Research*, 18(1):6869–6898, 2017.
- [24] Itay Hubara, Yury Nahshan, Yair Hanani, Ron Banner, and Daniel Soudry. Improving post training neural quantization: Layer-wise calibration and integer programming. *arXiv preprint arXiv:2006.10518*, 2020.
- [25] Benoit Jacob, Skirmantas Kligys, Bo Chen, Menglong Zhu, Matthew Tang, Andrew Howard, Hartwig Adam, and Dmitry Kalenichenko. Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2704–2713, 2018.
- [26] Minsoo Kim, Sihwa Lee, Sukjin Hong, Du-Seong Chang, and Jungwook Choi. Understanding and improving knowledge distillation for quantization-aware training of large transformer encoders. *arXiv preprint arXiv:2211.11014*, 2022.
- [27] Sehoon Kim, Amir Gholami, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. I-bert: Integer-only bert quantization. *arXiv preprint arXiv:2101.01321*, 2021.
- [28] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- [29] 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>.
- [30] Olga Kovaleva, Saurabh Kulshreshtha, Anna Rogers, and Anna Rumshisky. Bert busters: Outlier dimensions that disrupt transformers. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3392–3405, 2021.

- [31] Raghuraman Krishnamoorthi. Quantizing deep convolutional networks for efficient inference: A whitepaper. *arXiv preprint arXiv:1806.08342*, 2018.
- [32] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL <https://aclanthology.org/2020.acl-main.703>.
- [33] Quentin Lhoest, Albert Villanova del Moral, Yacine Jernite, Abhishek Thakur, Patrick von Platen, Suraj Patil, Julien Chaumond, Mariama Drame, Julien Plu, Lewis Tunstall, Joe Davison, Mario Sasko, Gunjan Chhablani, Bhavitya Malik, Simon Brandeis, Teven Le Scao, Victor Sanh, Canwen Xu, Nicolas Patry, Angelina McMillan-Major, Philipp Schmid, Sylvain Gugger, Clément Delangue, Théo Matussière, Lysandre Debut, Stas Bekman, Pierrick Cistac, Thibault Goehringer, Victor Mustar, François Lagunas, Alexander Rush, and Thomas Wolf. Datasets: A community library for natural language processing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 175–184, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL <https://aclanthology.org/2021.emnlp-demo.21>.
- [34] Yuhang Li, Ruihao Gong, Xu Tan, Yang Yang, Peng Hu, Qi Zhang, Fengwei Yu, Wei Wang, and Shi Gu. Brecq: Pushing the limit of post-training quantization by block reconstruction. *arXiv preprint arXiv:2102.05426*, 2021.
- [35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.
- [36] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021.
- [37] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.
- [38] Christos Louizos, Max Welling, and Diederik P Kingma. Learning sparse neural networks through l_0 regularization. *arXiv preprint arXiv:1712.01312*, 2017.
- [39] Eldad Meller, Alexander Finkelstein, Uri Almog, and Mark Grobman. Same, same but different: Recovering neural network quantization error through weight factorization. In *International Conference on Machine Learning*, pages 4486–4495. PMLR, 2019.
- [40] Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. *arXiv preprint arXiv:1609.07843*, 2016.
- [41] Markus Nagel, Mart van Baalen, Tijmen Blankevoort, and Max Welling. Data-free quantization through weight equalization and bias correction. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1325–1334, 2019.
- [42] Markus Nagel, Rana Ali Amjad, Mart Van Baalen, Christos Louizos, and Tijmen Blankevoort. Up or down? Adaptive rounding for post-training quantization. In *International Conference on Machine Learning (ICML)*, 2020.
- [43] Markus Nagel, Marios Fournarakis, Rana Ali Amjad, Yelysei Bondarenko, Mart van Baalen, and Blankevoort Tijmen. A white paper on neural network quantization. *arXiv preprint arXiv:2106.08295*, 2021.
- [44] Vinod Nair and Geoffrey E. Hinton. Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814. Omnipress, 2010.
- [45] 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 *Neural Information Processing Systems (NeurIPS)*. 2019.

- [46] Giovanni Puccetti, Alessio Miaschi, and Felice Dell’Orletta. How do BERT embeddings organize linguistic knowledge? In *Proceedings of Deep Learning Inside Out (DeeLIO): The 2nd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures*, pages 48–57, Online, June 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.deelio-1.6. URL <https://aclanthology.org/2021.deelio-1.6>.
- [47] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [48] Bita Rouhani, Daniel Lo, Ritchie Zhao, Ming Liu, Jeremy Fowers, Kalin Ovtcharov, Anna Vinogradsky, Sarah Massengill, Lita Yang, Ray Bittner, Alessandro Forin, Haishan Zhu, Taesik Na, Prerak Patel, Shuai Che, Lok Chand Koppaka, Xia Song, Subhojit Som, Kaustav Das, Saurabh Tiwary, Steve Reinhardt, Sitaram Lanka, Eric Chung, and Doug Burger. Pushing the limits of narrow precision inferencing at cloud scale with microsoft floating point. In *Neural Information Processing Systems (NeurIPS 2020)*. ACM, November 2020.
- [49] 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. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2015.
- [50] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.
- [51] Sheng Shen, Zhen Dong, Jiayu Ye, Linjian Ma, Zhewei Yao, Amir Gholami, Michael W Mahoney, and Kurt Keutzer. Q-bert: Hessian based ultra low precision quantization of bert. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8815–8821, 2020.
- [52] Zhiqing Sun, Hongkun Yu, Xiaodan Song, Renjie Liu, Yiming Yang, and Denny Zhou. MobileBERT: a compact task-agnostic BERT for resource-limited devices. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2158–2170, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.195. URL <https://aclanthology.org/2020.acl-main.195>.
- [53] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1–9, 2015.
- [54] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.
- [55] Hugo Touvron, Matthieu Cord, and Hervé Jégou. Deit iii: Revenge of the vit. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIV*, pages 516–533. Springer, 2022.
- [56] Mart van Baalen, Andrey Kuzmin, Suparna S Nair, Yuwei Ren, Eric Mahurin, Chirag Patel, Sundar Subramanian, Sanghyuk Lee, Markus Nagel, Joseph Soriaga, and Tijmen Blankevoort. Fp8 versus int8 for efficient deep learning inference. 2023.
- [57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 6000–6010, 2017.
- [58] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-5446. URL <https://www.aclweb.org/anthology/W18-5446>.
- [59] Xiuying Wei, Yunchen Zhang, Xiangguo Zhang, Ruihao Gong, Shanghang Zhang, Qi Zhang, Fengwei Yu, and Xianglong Liu. Outlier suppression: Pushing the limit of low-bit transformer language models. *arXiv preprint arXiv:2209.13325*, 2022.
- [60] Xiuying Wei, Yunchen Zhang, Yuhang Li, Xiangguo Zhang, Ruihao Gong, Jinyang Guo, and Xianglong Liu. Outlier suppression+: Accurate quantization of large language models by equivalent and optimal shifting and scaling. *arXiv preprint arXiv:2304.09145*, 2023.
- [61] Ross Wightman. Pytorch image models. <https://github.com/rwightman/pytorch-image-models>, 2019.

- [62] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierrick Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 conference on empirical methods in natural language processing: system demonstrations*, pages 38–45, 2020.
- [63] Xiaoxia Wu, Cheng Li, Reza Yazdani Aminabadi, Zhewei Yao, and Yuxiong He. Understanding int4 quantization for transformer models: Latency speedup, composability, and failure cases. 2023.
- [64] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *CVPR*, 2022.
- [65] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/dc6a7e655d7e5840e66733e9ee67cc69-Paper.pdf.
- [66] Zhewei Yao, Reza Yazdani Aminabadi, Minjia Zhang, Xiaoxia Wu, Conglong Li, and Yuxiong He. Zeroquant: Efficient and affordable post-training quantization for large-scale transformers. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27168–27183. Curran Associates, Inc., 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/adf7fa39d65e2983d724ff7da57f00ac-Paper-Conference.pdf.
- [67] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019.
- [68] Ofir Zafrir, Guy Boudoukh, Peter Izsak, and Moshe Wasserblat. Q8bert: Quantized 8bit bert. *arXiv preprint arXiv:1910.06188*, 2019.
- [69] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. 2017.
- [70] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.
- [71] Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuhui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*, 2022.
- [72] Ritchie Zhao, Yuwei Hu, Jordan Dotzel, Chris De Sa, and Zhiru Zhang. Improving neural network quantization without retraining using outlier channel splitting. In *International conference on machine learning*, pages 7543–7552. PMLR, 2019.
- [73] Shuchang Zhou, Yuxin Wu, Zekun Ni, Xinyu Zhou, He Wen, and Yuheng Zou. Dorefa-net: Training low bitwidth convolutional neural networks with low bitwidth gradients. *arXiv preprint arXiv:1606.06160*, 2016.
- [74] 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 *The IEEE International Conference on Computer Vision (ICCV)*, December 2015.

Supplementary materials

A Additional graphs from outlier analysis

In this section, we present additional graphs from our outlier investigation in Section 3 for BERT and vision transformer.

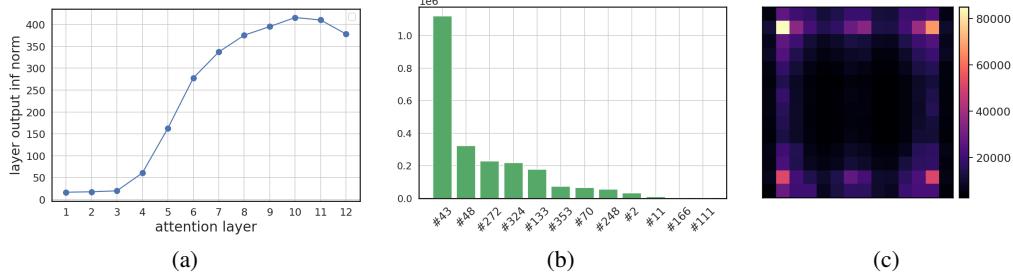


Figure 8: A summary of several outlier statistics recorded from ImageNet validation set on ViT. (a) Average infinity norm of the output of each attention layer. (b) A histogram of outlier counts in attention layer #10 vs. hidden dimensions. We use zero-based indexing for dimensions. (c) A heatmap of outlier counts in attention layer #10 vs. patch positions.

A.1 BERT

Recall from Figure 1 that all the outliers are only present in hidden dimensions #123, #180, #225, #308, #381, #526, #720 (with the majority of them in #180, #720). These hidden dimensions correspond to attention heads #2, #3, #4, #5, #6, #9, and #12. In Figures 9 and 10 we show more examples of the discovered self-attention patterns for attention heads #3 and #12 (\leftrightarrow hidden dim #180 and #720, respectively). We also show self-attention patterns in attention heads and layers which are not associated with the outliers in Figures 11 and 12, respectively.

A.2 ViT

Figure 8 further shows that there are a lot of similarities in the outlier behavior in the vision transformer, compared to BERT. The strongest magnitude outliers generally happen in the later layers, peaking at layers #10 and #11. The majority of outliers ($> 99\%$) are only ever happening in only 10 hidden dimensions, primarily in dimensions #48 and #43, which corresponds to the attention head #1. Finally, averaged across the entire ImageNet validation set, the outliers seem to be concentrated at the boundaries of the image, which suggest a strong correlation with the background (and a negative correlation with the object, which is usually in the center of the image in the ImageNet dataset).

In Figures 13 and 14, we show more examples of outlier and self-attention patterns in the attention head #1 (\leftrightarrow hidden dimensions #48, #43) for a random subset of images from the ImageNet validation set (in layers #10 and #11, respectively).

B Detailed results

In this section, we provide extended results for each model, including the used hyperparameters and other design choices. We also present some additional ablation studies.

B.1 Gating architectures

We investigate the choice of several gating functions, summarized in Table 3. The configuration “MLP” parameterizes each G_i with a feed-forward net with one hidden layer of size n_{hid} and a

Configuration	G	Memory overhead (per attention layer)		
		# extra parameters	# extra tokens	
Linear	$n_{\text{heads}} \times \text{Linear}(d_{\text{head}} \rightarrow 1)$	$n_{\text{heads}}(d_{\text{head}} + 1)$		~ 1
MLP	$n_{\text{heads}} \times \text{MLP}(d_{\text{head}} \rightarrow n_{\text{hid}} \rightarrow 1)$	$n_{\text{heads}}(n_{\text{hid}}(d_{\text{head}} + 2) + 1)$		$\sim n_{\text{hid}}$
All-heads-linear	$\text{Linear}(d_{\text{model}} \rightarrow n_{\text{heads}})$	$n_{\text{heads}}(d_{\text{model}} + 1)$		$\sim n_{\text{heads}}$

Table 3: An overview of the gating function parameterizations explored in this paper and their memory overhead.

ReLU non-linearity [44]. We also explore what happens if we allow the mixing of the representation from different attention heads in the “All-heads-linear” setting, where we use a single linear layer to produce the gating probabilities for all attention heads at once. All three options are tested below. Unless explicitly stated otherwise, we initialize the bias of the gating function to zero (i.e., $b_{\text{init}} = 0 \leftrightarrow \pi_{\text{init}} = 0.5$).

B.2 BERT

Method	FP16 ppl. \downarrow	Max inf norm	Avg. Kurtosis	W8A8 ppl. \downarrow
Vanilla	4.49 ± 0.01	735.0 ± 54.9	3076 ± 262	1294 ± 1046
CS ($\gamma = -0.005$)	4.44 ± 0.02	406.6 ± 35.2	1963 ± 753	75.27 ± 39.57
CS ($\gamma = -0.01$)	4.35 ± 0.01	198.3 ± 78.7	1581 ± 839	7.06 ± 2.37
CS ($\gamma = -0.015$)	4.37 ± 0.01	38.9 ± 7.9	165 ± 34	4.54 ± 0.01
CS ($\gamma = -0.02$)	4.39 ± 0.02	31.7 ± 6.3	90 ± 20	4.56 ± 0.02
CS ($\gamma = -0.025$)	4.39 ± 0.00	21.5 ± 1.5	80 ± 6	4.52 ± 0.01
CS ($\gamma = -0.03$)	4.41 ± 0.01	20.4 ± 0.2	79 ± 6	4.55 ± 0.01
CS ($\gamma = -0.04$)	4.51 ± 0.05	19.8 ± 9.0	85 ± 7	4.65 ± 0.06
GA, Linear ($\pi_{\text{init}} = 0.25$)	4.49 ± 0.00	139.8 ± 62.3	739 ± 412	5.05 ± 0.27
GA, Linear ($\pi_{\text{init}} = 0.5$)	4.48 ± 0.00	177.3 ± 33.2	652 ± 81	5.13 ± 0.15
GA, Linear ($\pi_{\text{init}} = 0.75$)	4.49 ± 0.00	71.4 ± 49.9	262 ± 147	4.88 ± 0.22
GA, Linear ($\pi_{\text{init}} = 0.9$)	4.49 ± 0.00	171.5 ± 8.8	559 ± 141	5.15 ± 0.03
GA, MLP ($n_{\text{hid}} = 4$)	4.45 ± 0.03	39.2 ± 26.0	201 ± 181	4.65 ± 0.04
GA, MLP ($n_{\text{hid}} = 64$)	4.49 ± 0.01	117.0 ± 48.3	507 ± 167	4.77 ± 0.01
GA, All-heads-linear	4.49 ± 0.01	58.3 ± 41.2	334 ± 321	4.67 ± 0.03

Table 4: Main results for our proposed Clipped softmax (CS) and Gated attention (GA) applied to BERT-base. We report the masked language modeling perplexity (ppl. for short) on the English Wikipedia validation set for both the floating-point baseline and W8A8 quantized model. We also report the maximum $\|\mathbf{x}\|_\infty$ averaged across the validation set, and kurtosis of \mathbf{x} averaged across all layers, where \mathbf{x} is the output of an attention layer.

Detailed results for BERT-base are summarized in Table 4. As we can see, across most of the settings, both of our methods significantly dampen the outliers’ magnitude, reduce the kurtosis, drastically improve the quantized performance, while maintaining and sometimes improving the FP16 perplexity.

B.3 OPT

Detailed results for OPT-125m are summarized in Table 5.

In our early experiments on a smaller OPT model, we found that applying the weight decay on LayerNorm weights γ (which isn’t the case, by default) has a strong effect on reducing the outliers’ magnitude while yielding the comparable FP16 performance. Therefore, we present the results of applying our gated attention approach in both cases, with and without applying weight decay on LN γ . As we can see in Table 5, in both cases gated attention (further) dampens the outliers’ magnitude to a great extent, reduces the kurtosis, and yields models with significantly higher quantized performance, which is close to the original FP16 performance.

Method	LN	γ	wd	FP16 ppl. \downarrow	Max inf norm	Avg. Kurtosis	W8A8 ppl. \downarrow
Vanilla		\times		$15.84^{\pm 0.05}$	$339.6^{\pm 47.2}$	$1777^{\pm 444.}$	$21.18^{\pm 1.89}$
GA, Linear ($\pi_{\text{init}} = 0.1$)		\times		$15.61^{\pm 0.05}$	$35.6^{\pm 4.5}$	$42.4^{\pm 22.9}$	$16.41^{\pm 0.18}$
GA, Linear ($\pi_{\text{init}} = 0.25$)		\times		$15.50^{\pm 0.04}$	$35.8^{\pm 0.5}$	$59.0^{\pm 48.3}$	$16.25^{\pm 0.08}$
GA, Linear ($\pi_{\text{init}} = 0.5$)		\times		$15.54^{\pm 0.01}$	$46.5^{\pm 5.0}$	$40.6^{\pm 8.9}$	$16.30^{\pm 0.01}$
GA, All-heads-linear		\times		$15.43^{\pm 0.01}$	$32.8^{\pm 1.7}$	$24.2^{\pm 3}$	$16.30^{\pm 0.12}$
Vanilla		\checkmark		$15.96^{\pm 0.03}$	$87.7^{\pm 31.9}$	$2080^{\pm 1460}$	$39.46^{\pm 16.59}$
CS ($\gamma = -1/512$)		\checkmark		$15.99^{\pm 0.02}$	$106.4^{\pm 7.0}$	$5764^{\pm 2150}$	$185.23^{\pm 220.00}$
CS ($\gamma = -2/512$)		\checkmark		$15.90^{\pm 0.02}$	$102.0^{\pm 27.0}$	$11290^{\pm 4372}$	$60.90^{\pm 52.70}$
CS ($\gamma = -4/512$)		\checkmark		$15.86^{\pm 0.01}$	$83.1^{\pm 20.6}$	$17174^{\pm 7791}$	$84.64^{\pm 10.55}$
CS ($\gamma = -8/512$)		\checkmark		$16.13^{\pm 0.09}$	$61.5^{\pm 9.9}$	$19204^{\pm 4284}$	$42.62^{\pm 3.64}$
CS ($\gamma = -12/512$)		\checkmark		$16.29^{\pm 0.07}$	$63.2^{\pm 8.8}$	$19727^{\pm 7479}$	$37.22^{\pm 2.39}$
GA, Linear ($\pi_{\text{init}} = 0.1$)		\checkmark		$15.69^{\pm 0.05}$	$7.3^{\pm 0.4}$	$25.4^{\pm 10}$	$16.23^{\pm 0.08}$
GA, Linear ($\pi_{\text{init}} = 0.25$)		\checkmark		$15.55^{\pm 0.05}$	$8.7^{\pm 0.6}$	$18.9^{\pm 1}$	$16.02^{\pm 0.07}$
GA, Linear ($\pi_{\text{init}} = 0.5$)		\checkmark		$15.63^{\pm 0.00}$	$10.8^{\pm 0.7}$	$42.0^{\pm 19}$	$16.20^{\pm 0.01}$
GA, All-heads-linear		\checkmark		$15.53^{\pm 0.01}$	$7.9^{\pm 0.3}$	$13.8^{\pm 1}$	$16.09^{\pm 0.08}$

Table 5: Main results for our proposed Clipped softmax (CS) and Gated attention (GA) applied to OPT-125m. We report the causal language modeling perplexity (ppl. for short) on the English Wikipedia validation set for both the floating-point baseline and W8A8 quantized model. We also report the maximum $\|\mathbf{x}\|_\infty$ averaged across the validation set, and kurtosis of \mathbf{x} averaged across all layers, where \mathbf{x} is the output of an attention layer.

B.4 ViT

Method	Patch.	Embd.	LN	FP32 acc.	Max inf norm	Avg. Kurtosis	W8A8 acc.
Vanilla		\times		$80.75^{\pm 0.10}$	$358.5^{\pm 81.2}$	$1018.3^{\pm 471.5}$	$69.24^{\pm 6.93}$
CS ($\gamma = -0.003$)		\times		$80.24^{\pm 0.05}$	$69.3^{\pm 20.7}$	$25.6^{\pm 8.6}$	$78.71^{\pm 0.33}$
CS ($\gamma = -0.004$)		\times		$80.38^{\pm 0.01}$	$74.9^{\pm 10.6}$	$30.6^{\pm 4.9}$	$78.66^{\pm 0.49}$
GA, Linear ($\pi_{\text{init}} = 0.25$)		\times		$80.62^{\pm 0.01}$	$86.0^{\pm 8.0}$	$23.4^{\pm 2.7}$	$79.16^{\pm 0.05}$
GA, Linear ($\pi_{\text{init}} = 0.5$)		\times		$80.32^{\pm 0.02}$	$88.4^{\pm 17.9}$	$27.9^{\pm 14.0}$	$78.90^{\pm 0.25}$
GA, MLP ($n_{\text{hid}} = 4$)		\times		$80.62^{\pm 0.05}$	$118.2^{\pm 40.5}$	$47.8^{\pm 29.8}$	$78.79^{\pm 0.29}$
Vanilla		\checkmark		$80.98^{\pm 0.08}$	$81.1^{\pm 2.5}$	$24.5^{\pm 1.8}$	$79.62^{\pm 0.06}$
CS ($\gamma = -0.0001$)		\checkmark		$80.89^{\pm 0.13}$	$73.7^{\pm 14.9}$	$22.9^{\pm 1.6}$	$79.77^{\pm 0.25}$
CS ($\gamma = -0.0003$)		\checkmark		$80.92^{\pm 0.07}$	$78.9^{\pm 5.5}$	$23.8^{\pm 0.5}$	$79.63^{\pm 0.05}$
CS ($\gamma = -0.0005$)		\checkmark		$80.95^{\pm 0.08}$	$72.9^{\pm 11.8}$	$24.4^{\pm 0.7}$	$79.73^{\pm 0.08}$
CS ($\gamma = -0.001$)		\checkmark		$80.95^{\pm 0.16}$	$80.8^{\pm 2.1}$	$24.1^{\pm 0.7}$	$79.69^{\pm 0.03}$
CS ($\gamma = -0.002$)		\checkmark		$80.80^{\pm 0.07}$	$78.0^{\pm 0.5}$	$25.8^{\pm 0.7}$	$79.32^{\pm 0.07}$
CS ($\gamma = -0.003$)		\checkmark		$80.79^{\pm 0.02}$	$75.6^{\pm 7.9}$	$28.1^{\pm 4.0}$	$79.00^{\pm 0.10}$
GA, Linear ($\pi_{\text{init}} = 0.5$)		\checkmark		$81.01^{\pm 0.06}$	$79.8^{\pm 0.5}$	$19.9^{\pm 0.3}$	$79.82^{\pm 0.11}$
GA, Linear ($\pi_{\text{init}} = 0.75$)		\checkmark		$81.01^{\pm 0.05}$	$77.8^{\pm 0.3}$	$21.8^{\pm 1.9}$	$79.80^{\pm 0.08}$
GA, Linear ($\pi_{\text{init}} = 0.9$)		\checkmark		$80.92^{\pm 0.11}$	$70.6^{\pm 8.0}$	$23.2^{\pm 3.7}$	$79.64^{\pm 0.09}$

Table 6: Main results for our proposed Clipped softmax (CS) and Gated attention (GA) applied to ViT-S/16. We report the top-1 accuracy on ImageNet-1K validation set for floating-point baseline and W8A8 quantized model. We also report the maximum $\|\mathbf{x}\|_\infty$ averaged across the validation set, and kurtosis of \mathbf{x} averaged across all layers, where \mathbf{x} is the output of the attention layer.

Detailed results for ViT-S/16 are summarized in Table 6.

After our preliminary experiments on ViT, we noticed that distinct outliers already originate after the patch embeddings. Therefore, we experimented with adding the LayerNorm after the patch embeddings (which was absent in the model definition, by default). As we can see in Table 5, together with this change, both of our proposed methods greatly dampens the outliers’ magnitude, reduces the kurtosis, and yields models with significantly higher quantized performance, which is within 1% of the original FP32 accuracy.

B.5 The impact of clipped softmax hyperparameters (γ and ζ) on ViT

γ	ζ	FP32 acc.	Max inf norm	W8A8 acc.
0 (= Vanilla)	1	78.80 \pm 0.42	426 \pm 69	71.27 \pm 0.88
0	1.001	78.78 \pm 0.29	411 \pm 88	71.24 \pm 0.59
0	1.002	78.90 \pm 0.17	420 \pm 47	70.74 \pm 0.34
0	1.004	78.80 \pm 0.45	377 \pm 67	72.31 \pm 0.06
0	1.01	78.81 \pm 0.30	419 \pm 77	71.35 \pm 0.26
-0.00001	1	78.81 \pm 0.21	432 \pm 76	69.02 \pm 0.19
-0.0001	1	78.81 \pm 0.36	380 \pm 64	64.04 \pm 10.8
-0.001	1	78.42 \pm 0.63	282 \pm 105	68.43 \pm 6.50
-0.003	1	78.26\pm0.06	99\pm36	76.49\pm0.48
-0.01	1	78.10 \pm 0.14	391 \pm 21	75.83 \pm 1.12
-0.03	1	70.26 \pm 1.46	197 \pm 2	65.80 \pm 1.41
-0.001	1.001	78.45 \pm 0.53	283 \pm 82	65.03 \pm 8.54
-0.003	1.003	78.25\pm0.14	119\pm17	76.37\pm0.45

Table 7: The impact of clipped softmax hyperparameters on ViT-S/16.

We investigate the effect of different values of the clipped softmax stretch parameters applied to the vision transformer and present the results in Table 7. To speed up training, for this experiment we trained ViT for 150 epochs instead of the usual 300 epochs. For this experiment, we did not apply LayerNorm after the patch embeddings.

We found similar observations compared to BERT. Specifically, most of the improvement happens when we use $\gamma < 0$ (clipping at zero) whereas using $\zeta > 1$ (clipping at one) yields similar results to the vanilla softmax and combining both $\gamma < 0$ and $\zeta > 1$ yields similar results compared to just clipping at zero.

C Experimental details

C.1 BERT

Fine-tuning on MNLI dataset We use pre-trained checkpoint BERT-base-uncased (109M parameters) from HuggingFace repository. We follow standard fine-tuning practices from [14] and [62]. Each data sequence is tokenized and truncated to the maximum sequence length of 128. Shorter sequences are padded to the same length of 128 using a special [PAD] token. We fine-tune for 3 epochs using Adam [28] with a batch size of 16 and no weight decay. The learning rate is initially set to its maximum value of $2 \cdot 10^{-5}$ and is linearly decayed to zero by the end of fine-tuning.

Pre-training from scratch We follow closely the pre-training procedure from [14]. We concatenate, tokenize, and split the training set into sequences of length 128 (to speed up training and experimentation, we do not fine-tune on longer sequences of 512). We use the masked language modeling objective with the probability of masking $p = 0.15$. We train with a batch size of 256 sequences for 10^6 steps, using AdamW optimizer [37] with the maximum learning rate of 10^{-4} , learning rate warm up over the first 10^4 steps, following by a linear decay to zero by the end of training. We use L2 weight decay of 0.01, L2 gradient norm clipping of 1.0, and dropout probability of 0.1 on all layers. We also use FP16 mixed-precision from HuggingFace Accelerate library [19].

C.2 OPT pre-training

To speed up experimentation, we train OPT-125m sized model on the concatenation of Wikipedia and BookCorpus (same as BERT pre-training). We train with a batch size of 48 and 4 gradient accumulation steps (which results in the effective batch size of 192), so that we can perform pre-training on a single A100 80GB GPU. We concatenate, tokenize, and split the training set into sequences of length 512 and train for 125000 steps (500000 forward passes).

We use the rest of the hyper-parameters and follow pre-training practices from [71] and [62]. We initialize weights using a normal distribution with zero mean and a standard deviation of 0.006. All bias terms are initialized to zero. We use AdamW optimizer with $(\beta_1, \beta_2) = (0.9, 0.95)$. We use the linear learning rate schedule, warming up from 0 to the maximum value[†] of $4 \cdot 10^{-4}$ over the first 2000 steps, following by a linear decay to zero by the end of training. We use L2 weight decay of 0.1, L2 gradient norm clipping of 1.0, and dropout probability of 0.1 on all layers. We also use FP16 mixed-precision from HuggingFace Accelerate library [19].

C.3 ViT pre-training

We use the model definition for ViT-S/16 and the training pipeline from PyTorch Image models library [61]. All training is done on resolution 224×224 and 16×16 patches. For data augmentation, we use RandAugment [10], Mixup [70], CutMix [67], random image cropping [53], horizontal flip, label smoothing $\varepsilon = 0.1$, color jitter 0.4, and random (between bilinear and bicubic) interpolation during training.

We train with a batch size of 512 for 300 epochs, using AdamW optimizer and the L2 weight decay of 0.03. We use the cosine learning rate schedule, warming up from 10^{-6} to the maximum value of 10^{-3} over the first 20 epochs, followed by a LR decay by a factor of 10 every 30 epochs, until it reaches the minimum value of 10^{-5} .

C.4 Quantization settings

Weights In all cases, we use symmetric uniform quantization of weights. We use min-max weight quantization for all models except the OPT model, for which we found the MSE estimator to perform better in all cases.

Activations We adopt *static range estimation* approach, which determines quantization parameters for the network by passing a few batches of calibration data through the model before inference. Specifically, we use a running min-max estimator [31], which uses an exponential moving average of the min and max over multiple batches. In all cases, we use running min-max with 0.9 momentum over 16 batches randomly sampled from respective training sets.

For OPT model, we also experiment with using 99.99% and 99.999% percentiles instead of actual min and max. We select the best configuration for each experiment (including baseline), based on the model performance. In almost all cases, we found that setting activation quantization ranges using 99.999% percentiles gives the lowest W8A8 perplexity.

D Compute cost

Model	Vanilla	Clipped softmax	Gated attention (Linear / MLP)
BERT	92.8 ± 1.2	93.6 ± 0.8	$97.7 / 119.1$
OPT	53.6 ± 0.4	54.4 ± 0.4	$55.7 / 64.7$
ViT	101.8 ± 0.3	104.0 ± 0.7	$110.8 / 122.9$

Table 8: An overview of the runtime of the proposed methods, compared to the vanilla pre-training, measured in hours on Nvidia-A100 GPUs.

We compare the runtime of our proposed methods in Table 8. As we can see, the clipped softmax is only marginally more expensive compared to using the vanilla softmax attention. The gated attention using the linear G adds the compute overhead between 3% and 8%, depending on the model. We found that adding weight decay on LayerNorm γ for OPT and adding the LayerNorm after the patch embeddings for ViT had a negligible effect on the runtime.

We estimated that the compute cost of producing the main results in the paper is about 320 GPU days (on A100) and the total cost of the project (including preliminary experiments and ablation studies) to be about 1400 GPU days.

[†]In our experiments, we found this value to perform better compared to the value of $6 \cdot 10^{-4}$ listed in the paper.

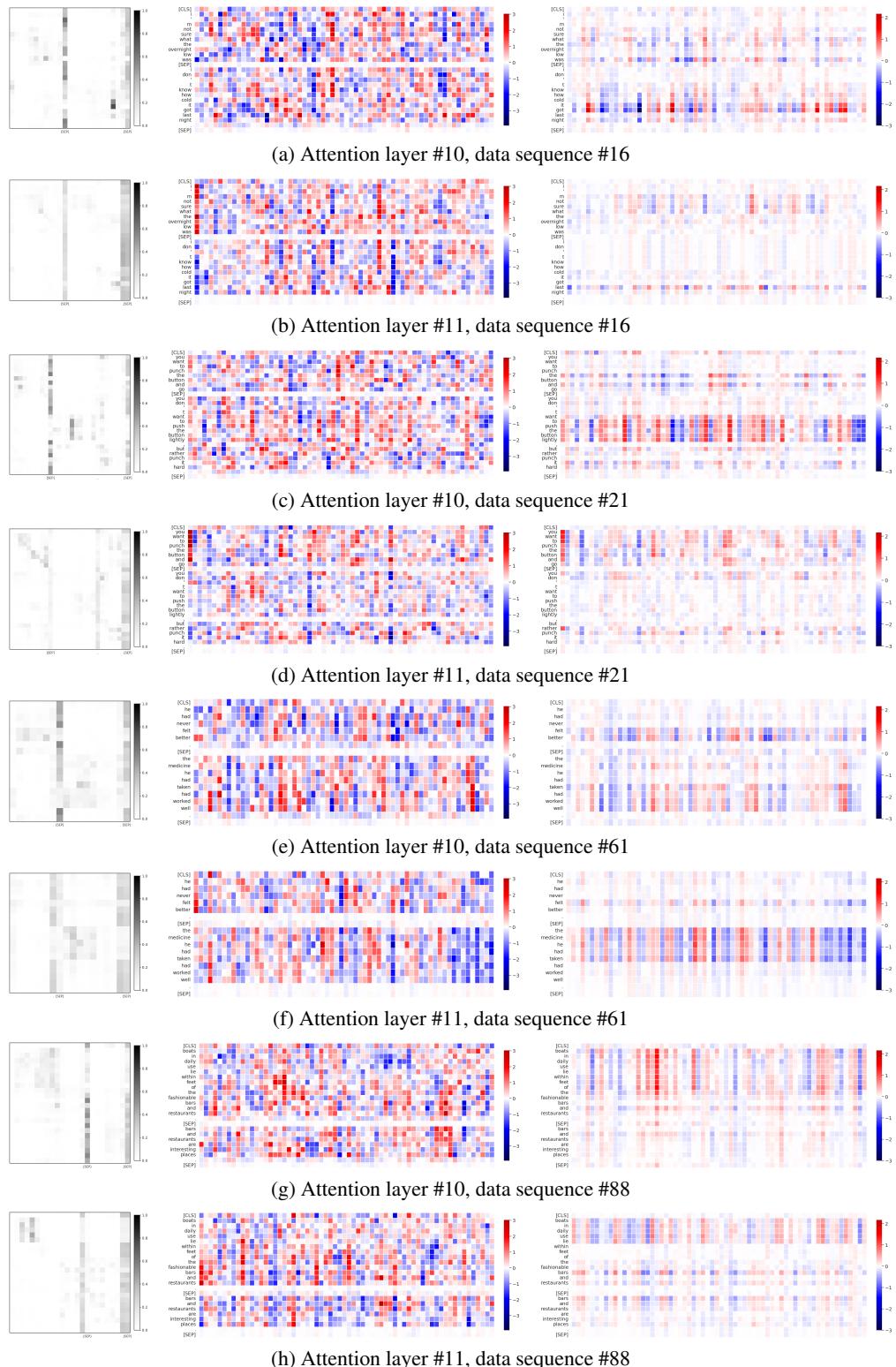


Figure 9: Visualization of the self-attention patterns (attention probabilities, values, and their product in left, middle and right columns, respectively) in attention head #3 (\leftrightarrow channel dim #180) for BERT-base, computed on several random data sequences from MNLI-m validation set.

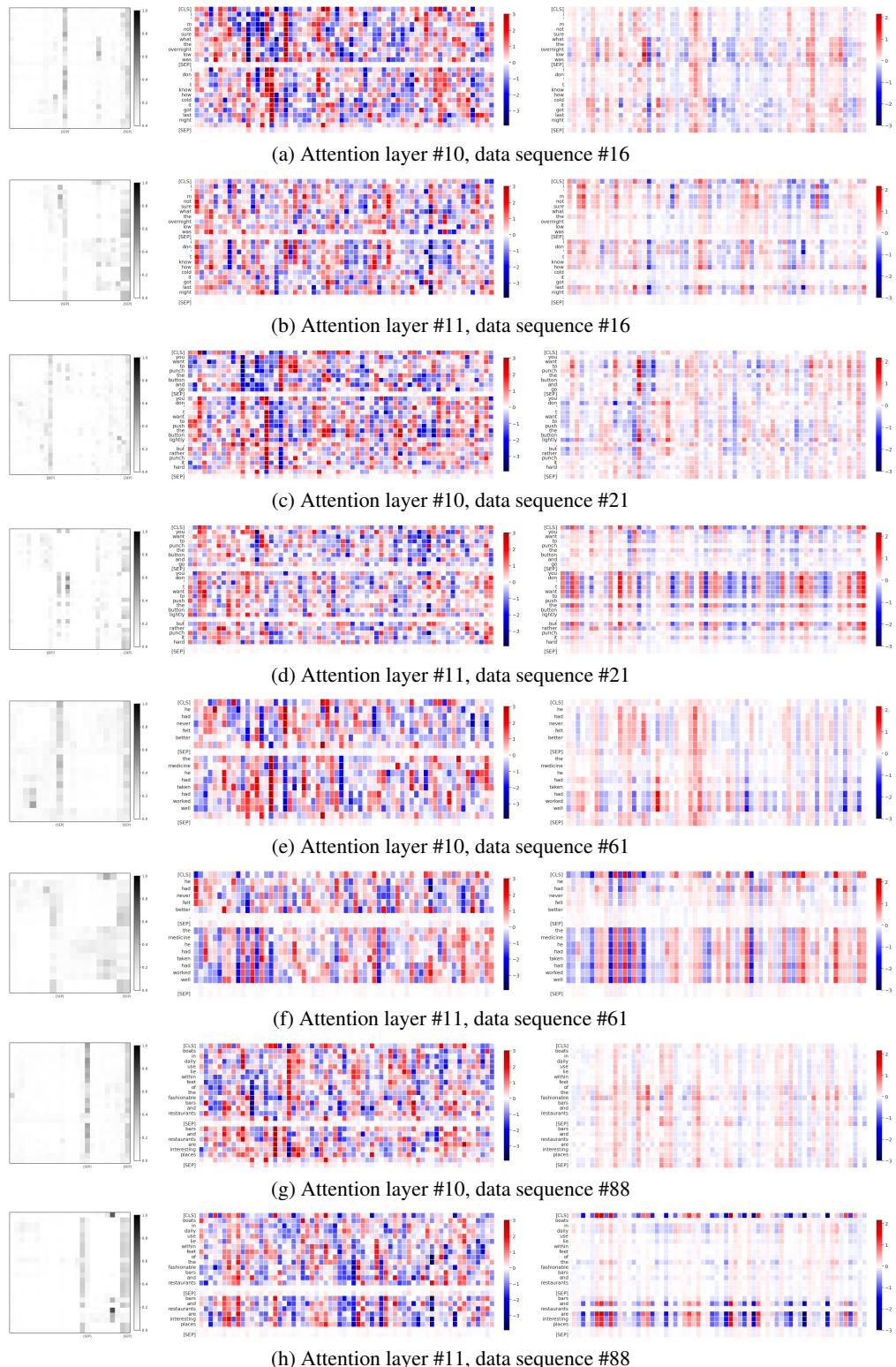


Figure 10: Visualization of the self-attention patterns (attention probabilities, values, and their product in left, middle and right columns, respectively) in attention head #12 (\leftrightarrow channel dim #720) for BERT-base, computed on several random data sequences from MNLI-m validation set.

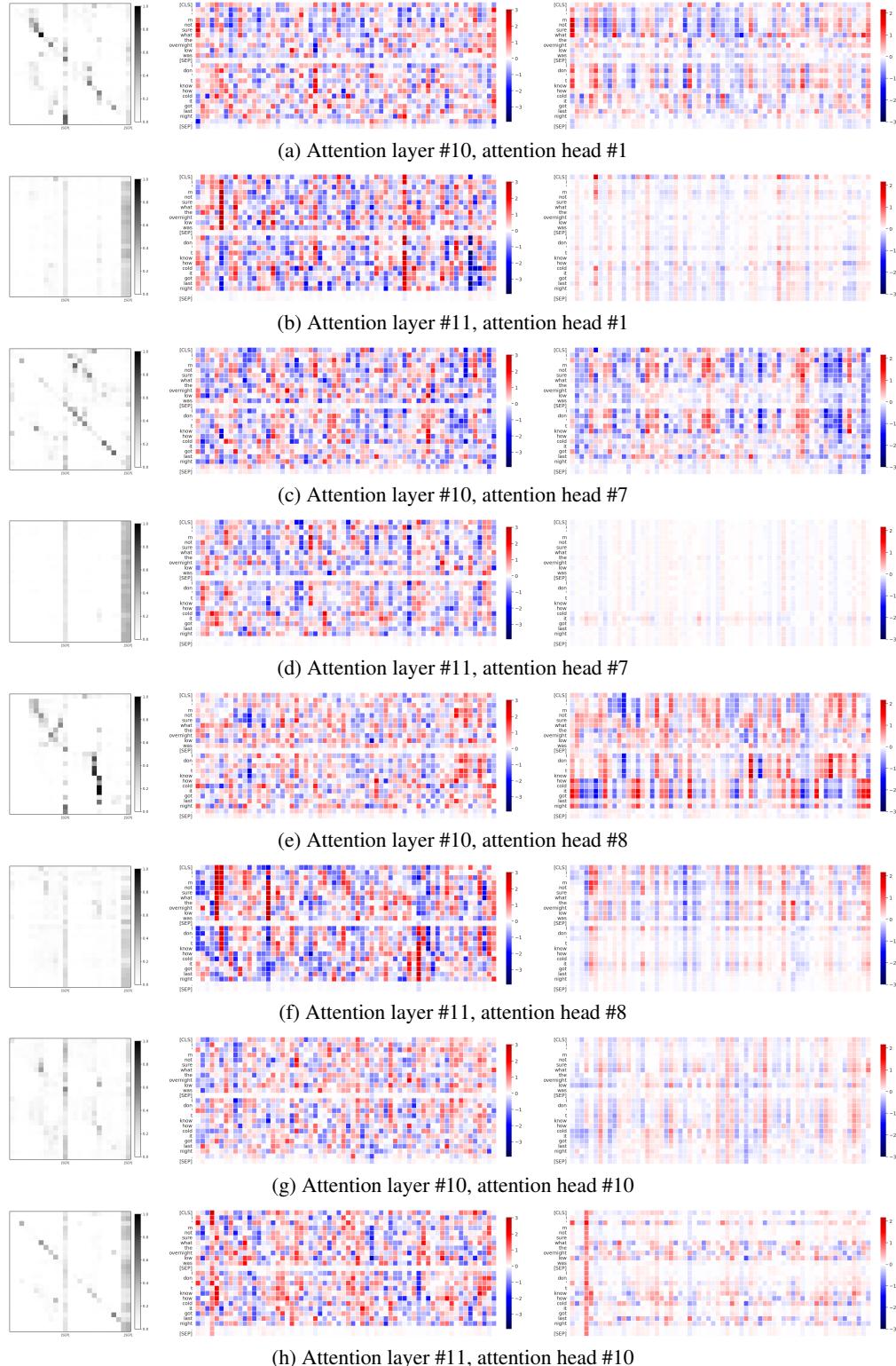


Figure 11: Visualization of the self-attention patterns (attention probabilities, values, and their product in left, middle and right columns, respectively) in attention heads that are not associated with the strong outliers for BERT-base, computed on data sequences #16 from MNLI-m validation set.

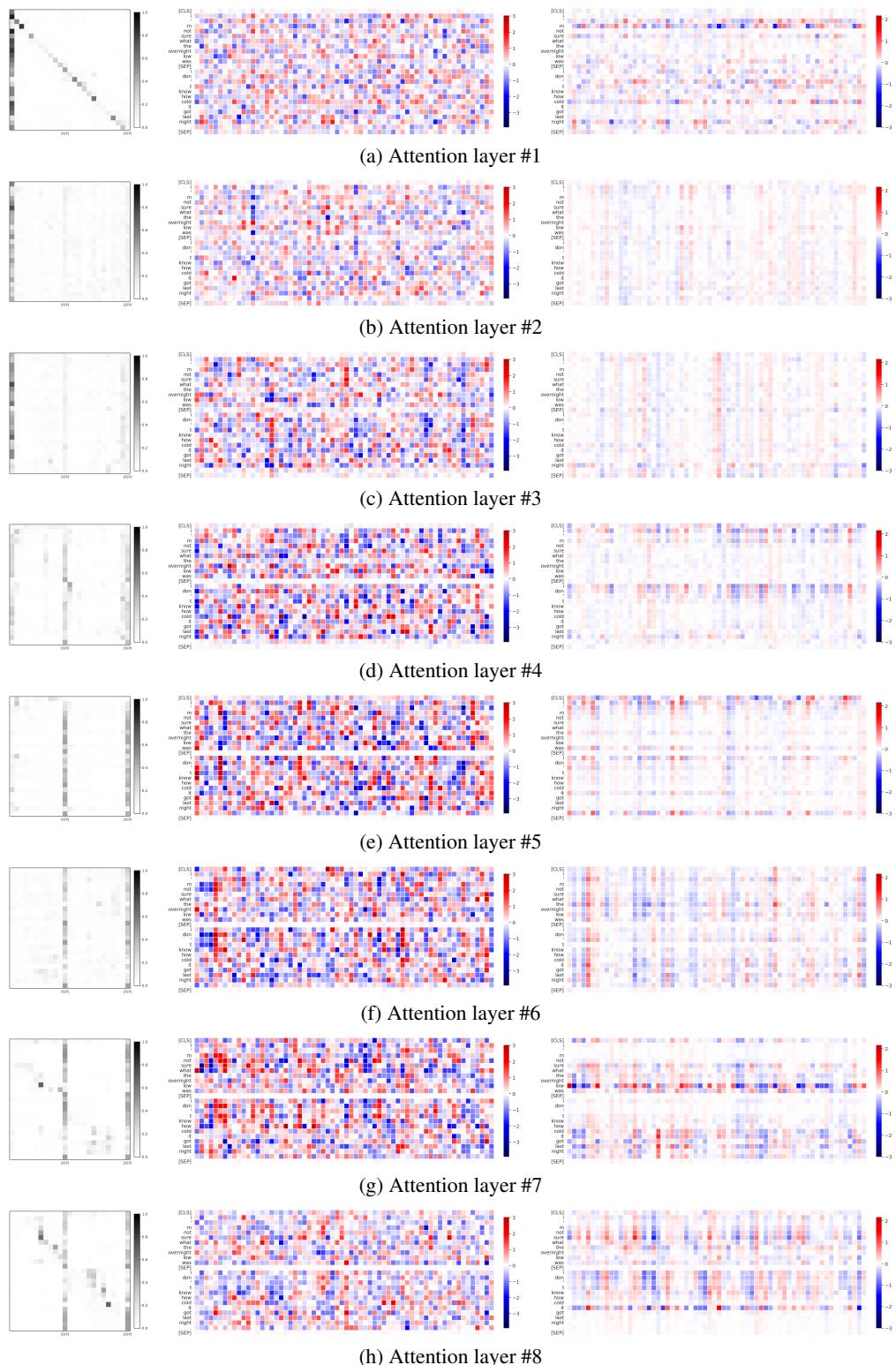


Figure 12: Visualization of the self-attention patterns (attention probabilities, values, and their product in left, middle and right columns, respectively) in attention head #3 (\leftrightarrow channel dim #180) and the first eight layers of BERT-base, computed on data sequences #16 from MNLI-m validation set.



Figure 13: A summary of our outlier analysis for ViT demonstrated on a random subset from ImageNet validation set. (a) An input image. (b) Outliers in the output of layer #10. (c) Cumulative attention weight spent on every patch (matrix of attention probabilities summed over rows) in the attention head #1, in the next layer #11. (d) A corresponding matrix of attention probabilities. (e) An average magnitude of values (V) for outlier and non-outlier patches.



Figure 14: A summary of our outlier analysis for ViT demonstrated on a random subset from ImageNet validation set. (a) An input image. (b) Outliers in the output of layer #11. (c) Cumulative attention weight spent on every patch (matrix of attention probabilities summed over rows) in the attention head #1, in the next layer #12. (d) A corresponding matrix of attention probabilities. (e) An average magnitude of values (\bar{V}) for outlier and non-outlier patches.