Appendix

A. Proof of Expectation for Bilinear Form

This section provides the proof for the expectation of the bilinear form used in Section 3.

A.1. Derivation

Let $x_A, x_B \in \mathbb{R}^D$ be two random vectors with means $\mu_A = \mathbb{E}[x_A]$ and $\mu_B = \mathbb{E}[x_B]$, and a cross-covariance matrix $C_{AB} = \mathbb{E}[(x_A - \mu_A)^\top (x_B - \mu_B)] \in \mathbb{R}^{D\times D}$. Let $W \in \mathbb{R}^{D\times D}$ be a deterministic matrix. We aim to prove the following identity:

$$\mathbb{E}[x_A W x_B^\top] = \mu_A W \mu_B^\top + \mathrm{tr}(C_{AB} W^\top) \tag{23}$$

First, we express the random vectors x_A and x_B in terms of their means and centered random vectors: $x_A = \mu_A + (x_A - \mu_A)$ and $x_B = \mu_B + (x_B - \mu_B)$. Substituting these into the expectation, we get:

$$\mathbb{E}[x_A W x_B^\top] = \mathbb{E}[(\mu_A + (x_A - \mu_A))W(\mu_B + (x_B - \mu_B))^\top]$$

Due to the linearity of the expectation operator, we can take the expectation of each term separately:

$$\mathbb{E}[x_A W x_B^\top] = \mathbb{E}[\mu_A W \mu_B^\top] + \mathbb{E}[\mu_A W (x_B - \mu_B)^\top]$$
$$+\mathbb{E}[(x_A - \mu_A) W \mu_B^\top] + \mathbb{E}[(x_A - \mu_A) W (x_B - \mu_B)^\top]$$

Since μ_A, μ_B, and W are deterministic, they can be pulled out of the expectation. The second and third terms are zero because $\mathbb{E}[x_A - \mu_A] = \mathbf{0}$ and $\mathbb{E}[x_B - \mu_B] = \mathbf{0}$. Now, we focus on the last term. The term $(x_A - \mu_A)W(x_B - \mu_B)^\top$ is a scalar, which is equal to its trace.

$$\mathbb{E}[(x_A - \mu_A) W (x_B - \mu_B)^\top]$$
$$= \mathbb{E}[\mathrm{tr}((x_A - \mu_A) W (x_B - \mu_B)^\top)]$$

Using the cyclic property, $\mathrm{tr}(ABC) = \mathrm{tr}(CAB)$:

$$= \mathbb{E}[\mathrm{tr}((x_B - \mu_B)^\top (x_A - \mu_A) W)]$$

Using the linearity of the expectation:

$$= \mathrm{tr}(\mathbb{E}[(x_B - \mu_B)^\top (x_A - \mu_A) W])$$
$$= \mathrm{tr}(\mathbb{E}[(x_B - \mu_B)^\top (x_A - \mu_A)] W)$$

The term $\mathbb{E}[(x_B - \mu_B)^\top (x_A - \mu_A)]$ is the cross-covariance matrix C_{BA}, which is the transpose of C_{AB}.

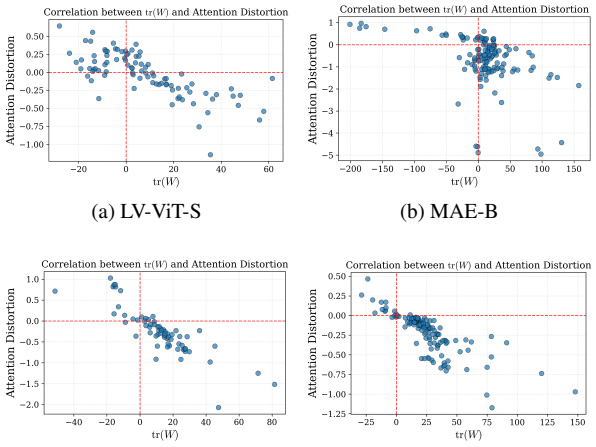
$$= \mathrm{tr}(C_{BA} W) = \mathrm{tr}(C_{AB}^\top W)$$

Combining all the terms, we get the final result:

$$\mathbb{E}[x_A W x_B^\top] = \mu_A W \mu_B^\top + \mathrm{tr}(C_{AB}^\top W)$$

This completes the proof.

B. Validation of Attention Distortion Analysis Across Different Models



(a) LV-ViT-S (b) MAE-B

(c) AugReg-S (d) SAM-B

Figure 4. A plot of the correlation between the trace of the attention projection matrix and attention distortion for models other than DeiT.

In Section 3, we analyzed the correlation between the attention projection matrix and attention distortion using DeiT-S. However, it remains unclear whether a similar result holds for models trained using other methods. Therefore, in this section, we verify whether the same result can be obtained under same settings for models trained with different methods. Figure 4 shows the results. The figure suggests that the tendency for attention distortion to be negative when tr(W) is positive, and positive when tr(W) is negative, also exists in other models. This result suggests that although our analysis focuses on a part of the attention calculation, that term influences the tendency of attention distortion regardless of the model's training method.

C. Details of Statistical Correction

C.1. Evaluation on other models

In Section 5, we evaluated the effect of statistical correction on DeiT. In this section, we conduct the same evaluation on other models. Table 7 shows the results. First, in LV-ViT-S, the difference between proportional attention and statistical correction was minimal. This result indicates that the effectiveness of addressing attention distortion is limited depending on the model. Moreover, in MAE-B, the method without attention correction achieved the highest accuracy. This result suggests

Model	Method	$r = 8$		$r = 12$		$r = 16$		$r = 20$	
		Acc (%)	Thr (im/s)	Acc (%)	Thr (im/s)	Acc (%)	Thr (im/s)	Acc (%)	Thr (im/s)
LV-ViT-S	Full Model	83.2	2707	83.2	2707	83.2	2707	83.2	2707
	No Corr	82.3	**3457**	78.0	**5622**	69.9	**4872**	46.7	**5495**
	Prop Attn	**83.0**	3376	**82.6**	4061	77.7	4783	58.8	5416
	S-HAC	83.0	3275	82.5	3986	**77.7**	4713	**58.8**	5340
MAE-B	Full Model	83.7	1396	83.7	1396	83.7	1396	83.7	1396
	No Corr	82.7	**1754**	**81.5**	**2088**	**78.0**	**2523**	**66.7**	**3024**
	Prop Attn	**82.7**	1703	81.2	2037	76.5	2471	54.8	2975
	S-HAC	82.7	1675	81.2	2010	76.9	2454	57.9	2955
AugReg-S	Full Model	74.2	3876	74.2	3876	74.2	3876	74.2	3876
	No Corr	73.0	**4753**	70.9	**5629**	65.4	**6725**	50.3	**7991**
	Prop Attn	73.1	4593	71.6	5469	68.1	6579	53.8	7834
	S-HAC	**73.1**	4434	**71.6**	5287	**68.3**	6395	**55.4**	7609
SAM-B	Full Model	75.5	1395	75.5	1395	75.5	1395	75.5	1395
	No Corr	74.6	**1752**	73.6	**2084**	71.2	**2517**	58.8	**3019**
	Prop Attn	75.2	1705	75.0	2034	74.3	2469	70.6	2969
	S-HAC	**75.3**	1683	**75.0**	2013	**74.4**	2442	**70.9**	2949

Table 7. Comparison of no correction (No Corr), proportional attention (Prop Attn), and our statistical HAC (S-HAC) on various models with different token reduction rates (r). Reported are top-1 accuracy (%) on ImageNet and throughput (images/s).

that proportional attention itself is not always optimal choice. On the other hand, in AugReg-S and SAM-B, we confirmed the effect of statistical correction. This result implies that the appropriate approach to attention correction varies depending on the model.

C.2. Implementation Differences

Code 1. PyTorch implementation of proportional attention

```
attn = (q@k.transpose(-2, -1))*self.scale
#attn: (batch_size, num_head, num_token, num_token)
attn = attn+size.log()
attn = attn.softmax(dim=-1)
```

In the evaluation of statistical correction in Section 5, we found that throughput is fastest in the order of no attention correction, proportional attention, and statistical correction. To investigate the cause of this, we discuss the differences in their implementations. We first address the difference between no attention correction and proportional attention. Code 1 shows the PyTorch implementation of proportional attention. Proportional attention adds the logarithm of the size to the attention scores, whereas no attention correction omits this addition. The throughput difference between no attention correction and proportional attention arises from this extra addition.

Code 2. PyTorch implementation of statistical correction

```
attn = (q@k.transpose(-2, -1))*self.scale
attn = attn+size.log()
diag = attn.diagonal(dim1=-2, dim2=-1)
diag += merge_flag*distortion
attn = attn.softmax(dim=-1)
```

Next, we compare the implementations of proportional attention and our proposed statistical correction. Code 2 shows the implementation of statistical correction. The variable `merge_flag` is a Boolean flag indicating whether a token was merged in the previous layer, and `distortion` denotes the expected value of the attention distortion obtained from the training set. Their product is added to the diagonal entries of the attention score. In addition to the size term added in proportional attention, the addition of distortion term is also incorporated, which results in lower throughput for our statistical HAC. To avoid this issue, our learning-based HAC does not perform operations only on the diagonal entries.

D. Details of Learning-based Correction

D.1. Raw Data of Figure 3

The L-HAC and Prop Attn in Table 8 correspond to each data point in Figure 3. The results of the full model are obtained from Table 1

D.2. Correcting Only the Diagonal Components

Our proposed learning-based HAC adjusts the entire attention scores, including the off-diagonal components, for better computational efficiency. However, this approach involves elements that fall outside the original focus of our analysis, because correcting the whole matrix

Model	Metric	Method	r						
			8	10	12	14	16	18	20
DeiT-B	Acc (%)	L-HAC	**81.4**	**81.2**	**80.8**	**80.4**	**79.3**	**76.8**	**72.2**
		L-HAC (D)	81.4	81.1	80.7	80.1	78.9	76.5	71.5
		Prop attn	81.2	80.9	80.4	79.5	77.9	74.1	66.7
	Thr (im/s)	L-HAC	**1730**	**1883**	**2063**	**2251**	**2491**	**2743**	**3001**
		L-HAC (D)	1601	1740	1904	2102	2318	2554	2777
		Prop attn	1708	1863	2026	2242	2479	2732	2975
DeiT-S	Acc (%)	L-HAC	**79.5**	**79.4**	**79.2**	**78.9**	**78.3**	77.3	75.2
		L-HAC (D)	**79.5**	79.3	79.1	**78.9**	**78.3**	**77.4**	**75.5**
		Prop attn	79.4	79.3	78.9	78.6	78.1	76.6	74.1
	Thr (im/s)	L-HAC	**4640**	**5042**	**5511**	**6005**	6560	7179	**7837**
		L-HAC (D)	4173	4546	4973	5419	5987	6513	7123
		Prop attn	4596	5009	5471	5940	**6568**	**7191**	7803
DeiT-T	Acc (%)	L-HAC	**71.7**	71.3	**71.1**	70.3	**69.3**	66.3	62.8
		L-HAC (D)	**71.7**	**71.5**	**71.1**	**70.5**	69.2	**67.2**	**63.9**
		Prop attn	71.5	71.2	70.8	69.9	68.6	65.5	61.1
	Thr (im/s)	L-HAC	**9799**	10580	11411	12417	13458	14739	16034
		L-HAC (D)	8756	9488	10318	11216	12258	13424	14563
		Prop attn	9797	**10599**	**11495**	**12506**	**13650**	**14981**	**16282**

Table 8. Comparison of learning-based correction methods on Vision Transformers. Results show accuracy (Acc) and throughput (Thr) for three DeiT models across different token reduction ratios (r). "L-HAC" denotes the original learning-based correction, "L-HAC (D)" represents diagonal-only correction, and "Prop attn" indicates ToMe's proportional attention method. Bold values highlight the best performance for each setting.

also changes a token's attention score to other tokens. Here, we want to explore a learning based approach that specifically focuses on the distortion of a token's attention score to itself. To isolate the impact of correcting off-diagonal components, we evaluated learning-based HAC for diagonal components as follows

$$A^{(h)}_{\text{corr}} = \frac{Q^{(h)} {K^{(h)}}^{\top}}{\sqrt{D_h}} + \mathbf{D}^{(h)} + \log \boldsymbol{s}, \tag{24}$$

where $\mathbf{D}^{(h)}$ is a diagonal matrix whose i-th diagonal element is determined by the output of the function f_h,

$$\mathbf{D}^{(h)}_{ii} = \log\big(\text{softplus}\big(f_h(s_i)\big)\big), \tag{25}$$

and s_i is the size of token i. Since the addition of logarithms corresponds to the logarithm of the product of their arguments, $\text{softplus}\big(f_h(s_i)\big)$ serves as a scale factor for the size.

Table 8 shows the results of learning-based HAC for diagonal components (L-HAC (D)), compared with our original learning-based HAC and ToMe's proportional attention. Two observations can be made from the results. First, correcting only the diagonal components outperforms not only proportional attention but also the original learning-based HAC in some models. This indicates that there are models for which our proposed remedy to the identified issue is particularly effective, while there are also models for which generalized strategies are beneficial. Second, correction applied only to the diagonal components degrades throughput due to diagonal memory access patterns.