SimA: Simple Softmax-free Attention for Vision Transformers (Supplementary material)

A. Appendix

A.1. Inference Time Comparison on GPU:

We compare execution time of SimA and other SOTA methods on edge devices in Figure 1. Additionally, we compare execution time of SimA, XCiT, and DeiT on GPU in Figure A1.

A.2. Simple Pseudocode of SimA:

Since our method is simple, we include the pseudocode of SimA in Algorithm 1.

Visualization: Figure A2 provides more results similar to Figure 3. Please see Section 4.7 for details.

A.3. SimA without LPI:

Although XCiT [2] shows that LPI layer can improve the accuracy by 1.2 point, it limits the application of vanilla transformer (e.g., running masked auto encoder models like MAE [24] is not straightforward). To show that our method is not dependent on LPI, we train our model without LPI. We observe that the accuracy drops by 1.2 point (82.1% vs 80.9%). Hence, although LPI boosts the accuracy, our method has comparable performance without LPI.

A.4. Details of Linear Attention Comparison:

CosFormer with cosine re-weighting requires 4× more FLOPs compared to SimA in multiplying K and V matrices. Since CosFormer is developed for NLP, it assumes one dimensional indexing for the tokens. However, applying it to vision, we need to index the tokens with two indices to take advantage of the induced locality. To do so, one may introduce two cosine weights to Eq 10 of CosFormer [51]: one in x direction and the other one in y direction to come up with:

$$Q_{i,m}K^T_{j,n}cos(i-j)cos(m-n)$$

which can be expanded to:

$$Q_{i,m}K^T_{j,n}\Big(cos(i)cos(j)+sin(i)sin(j)\Big)\Big(cos(m)cos(n)+sin(m)sin(n)\Big)$$

which can be regrouped to:

$$\begin{aligned}
=&\Big(Q_{i,m}cos(i)cos(m)\Big)\Big(K^T_{j,n}cos(j)cos(n)\Big)\\
+&\Big(Q_{i,m}cos(i)sin(m)\Big)\Big(K^T_{j,n}cos(j)sin(n)\Big)\\
+&\Big(Q_{i,m}sin(i)cos(m)\Big)\Big(K^T_{j,n}sin(j)cos(n)\Big)\\
+&\Big(Q_{i,m}sin(i)sin(m)\Big)\Big(K^T_{j,n}sin(j)sin(n)\Big)
\end{aligned}$$

Hence, for every attention value, CosFormer needs 4 dot products between Q and K vectors while our method needs only one dot product. Hence, following Eq. 12 of the CosFormer paper, CosFormer needs 4 times more FLOPS compared to our method in calculating the attention values (multiplying Q, K, and V matrices).

References

Algorithm 1 Pseudocode of SimA (Single Head) in a PyTorch-like style.

```
# self.qkv: nn.Linear(dim, dim * 3, bias=qkv_bias) ; query, key, value projection
# self.proj: nn.Linear(dim, dim, bias=output_proj_bias) ; output projection

def forward(self, x):
   B, N, D = x.shape # B: batch size, N: number of Tokens, D: Dimension of Tokens
   qkv = self.qkv(x).reshape(B, N, 3, D).permute(2, 0, 1, 3) # (3 x B x N x D)
   q, k, v = qkv[0], qkv[1], qkv[2] # split into query (B x N x D), key (B x N x D) and value (B x N x D)

   k = torch.nn.functional.normalize(k, p=1.0, dim=-2) # Normalized key (B x N x D)
   q = torch.nn.functional.normalize(q, p=1.0, dim=-2) # Normalized query (B x N x D)

   if (N/D) < 1:
      x = (q @ k.transpose(-2, -1)) @ v # (B x N x D)
   else:
      x = q @ (k.transpose(-2, -1) @ v) # (B x N x D)

   x = self.proj(x) # Output (B x N x D)
   return x
```

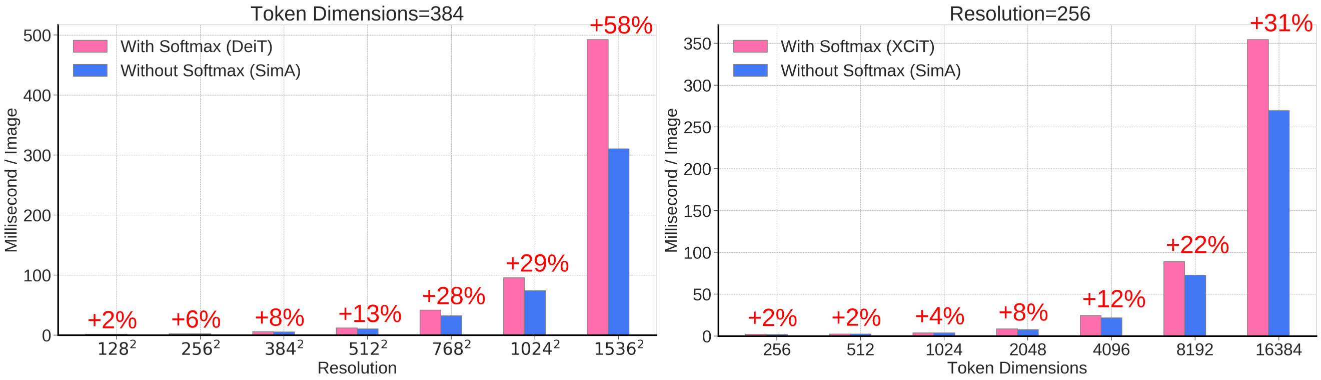

Figure A1. **Effect of Softmax on inference time (GPU):** We evaluate performance of each model on a single RTX 8000 GPU with batch size of 8. When comparing the baseline to our method (SimA), we fix the order of (QK^TV) to have the same dot product complexity as the baseline. For example, when comparing with DeiT, if $N > D$, then it is more efficient to do $\hat{Q}(\hat{K}^TV)$ for our method, but we do $(\hat{Q}\hat{K}^T)V$ to have same complexity as DeiT($O(N^2D)$). We do this to solely evaluate the effect of Softmax on the computation time. **Left:** We fix the token dimension to 384 and increase the image resolution. At 1536×1536 resolution, DeiT is 58% slower than our method due to the overhead of $exp(.)$ function in Softmax. **Right:** We fix the resolution and increase the capacity of the model (dimensions of Q and K). With 8192 dimensions, XCiT is 22% slower due to Softmax overhead.

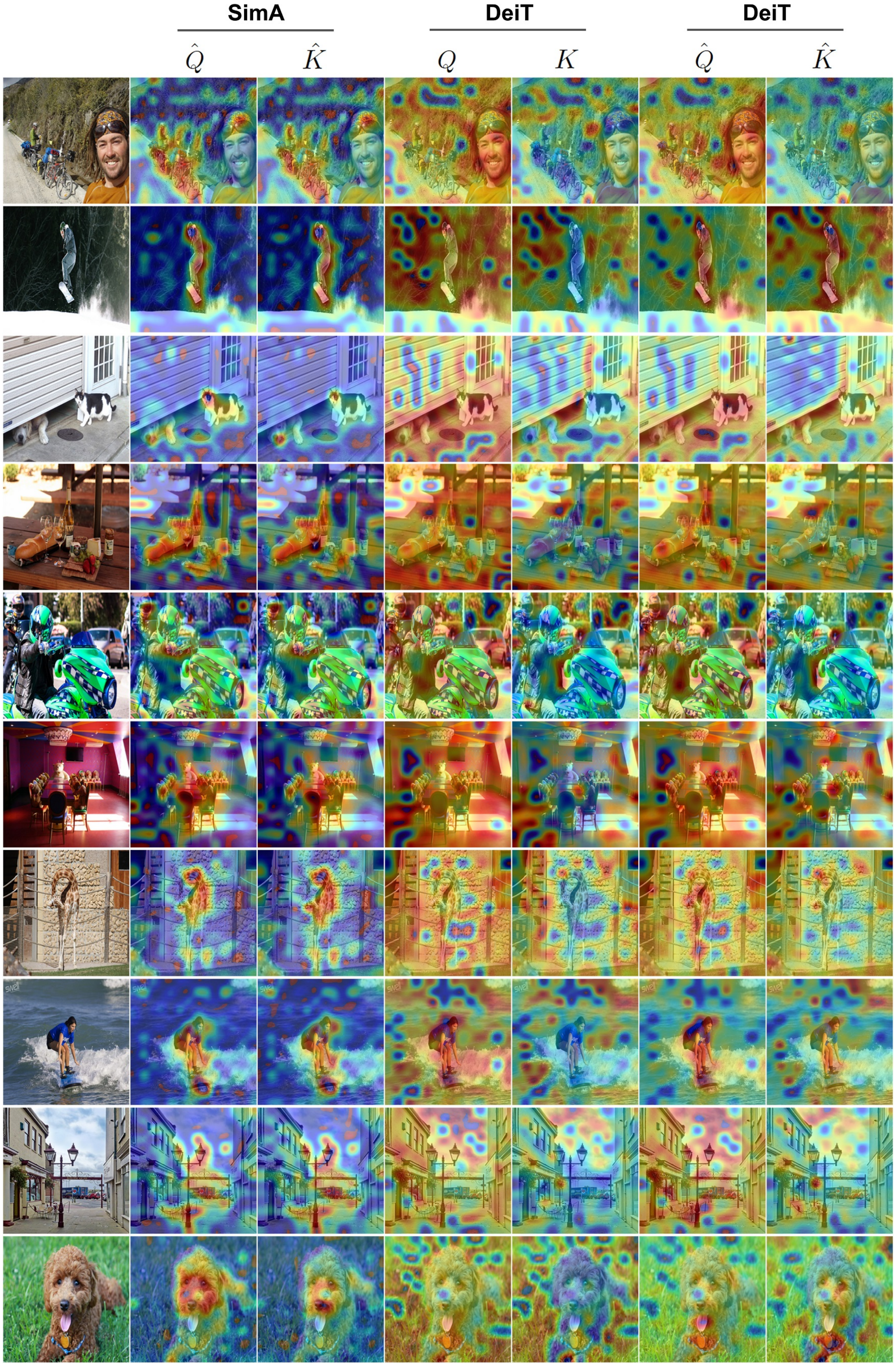

Figure A2. **Our method (SimA):** We extract $\hat{Q}$ and $\hat{K}$ from layer 12 of transformer. We get ℓ_2-norm of each token for $\hat{Q}$ and $\hat{K}$, normalize it to range [0,1] and overlay it as a heatmap on the image. Interestingly, magnitude of tokens represent the significance of tokens in our method. Note that all images are randomly selected from MS-COCO test set without any visual inspection or cherry picking.