SATA: Spatial Autocorrelation Token Analysis for Enhancing the Robustness of Vision Transformers

Supplementary Material

1. Supplementary material

1.1. ImageNet-1K SOTA

We compare the proposed SATA models against state-of-the-art image classification models. In addition to the methods discussed in the main text, we include several current state-of-the-art models such as VOLO [7], MOAT [6], CoAtNet [2], and MaxViT [5]. We compare their top-1 accuracy and efficiency in terms of GFLOPs. Notably, as shown in Figure 1, our proposed SATA and SATA* models significantly enhance the performance of standard DeiT and ViT models, establishing a new state-of-the-art.

1.2. Spatial Autocorrelation distribution across ViT's blocks

Figure 2 shows the distribution of spatial autocorrelation scores ($\mathbf{s}$) for patches (tokens) generated by different blocks of DeiT-Base/16 on the ImageNet-1K validation set. The spatial autocorrelation of tokens decreases through the blocks of the vision transformer, confirming the trends of the upper and lower bands through the vision transformer layers as discussed and demonstrated in Section 4 and Figure 3 of the main text.

1.3. Robustness on Individual Corruption Type

In this experiment, we compare the corruption error on each individual corruption type of ImageNet-C between the baseline DeiT-Base [4], FAN-B-ViT [8], and our SATA-T. As shown in Figure 3, our SATA model achieves lower corruption errors than the other two models across all corruption types, except for the snow weather corruption. Notably, despite not utilizing any training or patch noise augmentation, the proposed method demonstrates improved robustness and generalizes well to different types of corruption.

1.4. More Visualisation

To create the visualizations in Figure **??**(b) and Figure 4, we followed the method described in [1]. We traced each token of Sets $\mathbb{A}$ and $\mathbb{B}$ (described in Eq.**??**, subsection**??**) back to its original input patches. For each token in Set $\mathbb{A}$, we coloured its input patches using the average colour of the tokens it merged with. To distinguish different tokens, we assigned a random border colour to each of the merged tokens.

Moreover, we visualize the comparison of class token attention maps and spatial autocorrelation score maps across three layers—representing early, middle, and later blocks—of the proposed SATA-B (pre-trained Deit-Base/16+SATA) for various images from ImageNet-1K, ImageNet-C, ImageNet-R, ImageNet-A, and ImageNet-SK. As shown in Figure 5, spatial autocorrelation scores exhibit greater consistency across the Transformer layers compared to the corresponding attention scores, suggesting that the use of spatial autocorrelation can provide a more stable and reliable feature representation throughout the network.

1.5. Implementation

The following Figure 7 is an implementation of our "Spatial Autocorrelation Token Analysis" (SATA) in PyTorch [3]. Detailed settings for model evaluation across different datasets are provided in Table 1.

1.6. Results Verification

We believe that further investigation of our reported results is essential for a fair evaluation of our submission. Therefore, we have included the source code for the proposed SATA on the ImageNet2012-1K evaluation in the supplementary material. To run the code, please execute the `main-val.py` file using the commands and settings in 6:

References

[1] Daniel Bolya, Cheng-Yang Fu, Xiaoliang Dai, Peizhao Zhang, Christoph Feichtenhofer, and Judy Hoffman. Token merging: Your vit but faster. In *ICLR*, 2023. 1

[2] Zihang Dai, Hanxiao Liu, Quoc V Le, and Mingxing Tan. Coatnet: Marrying convolution and attention for all data sizes. In *NeurIPS*, pages 3965–3977, 2021. 1

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 1, 8

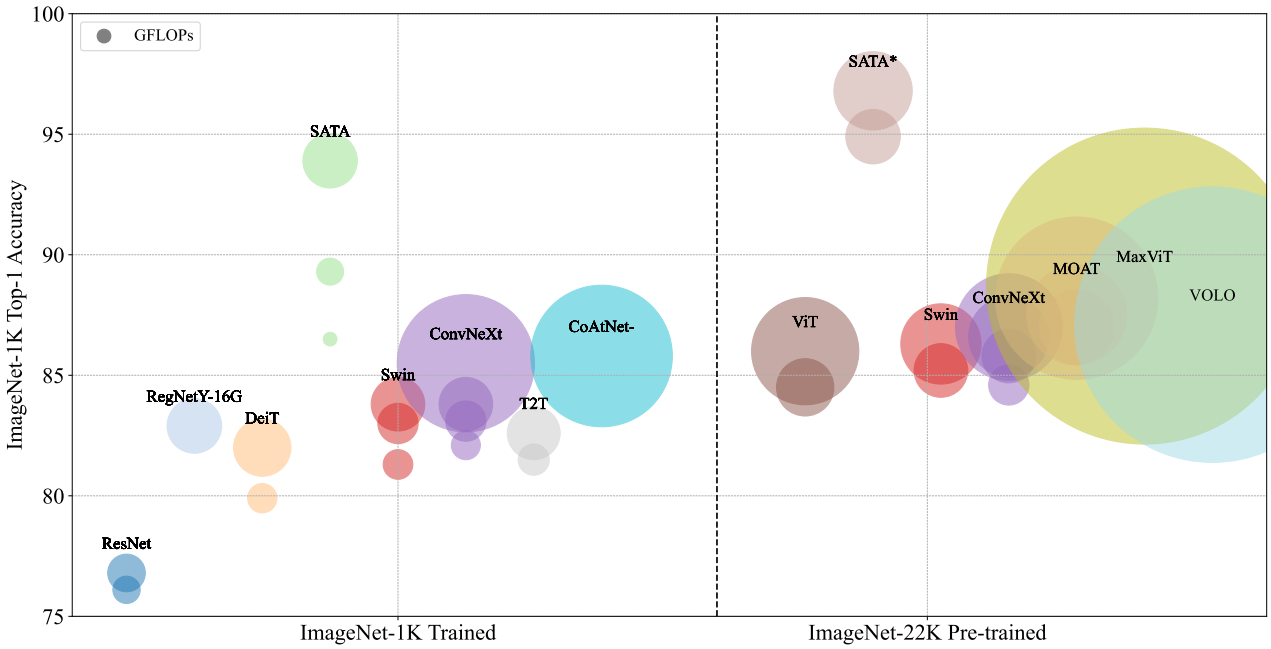

Figure 1. ImageNet-1K classification results for Vision Transformers and ConvNets models. Each bubble's area corresponds to the computational cost (GFLOPs) of a variant within its model family. ImageNet-1K/22K models use 224×224 input image resolutions, respectively. Notably, our proposed SATA and SATA* models significantly enhance the performance of standard DeiT and ViT models, establishing a new state-of-the-art.

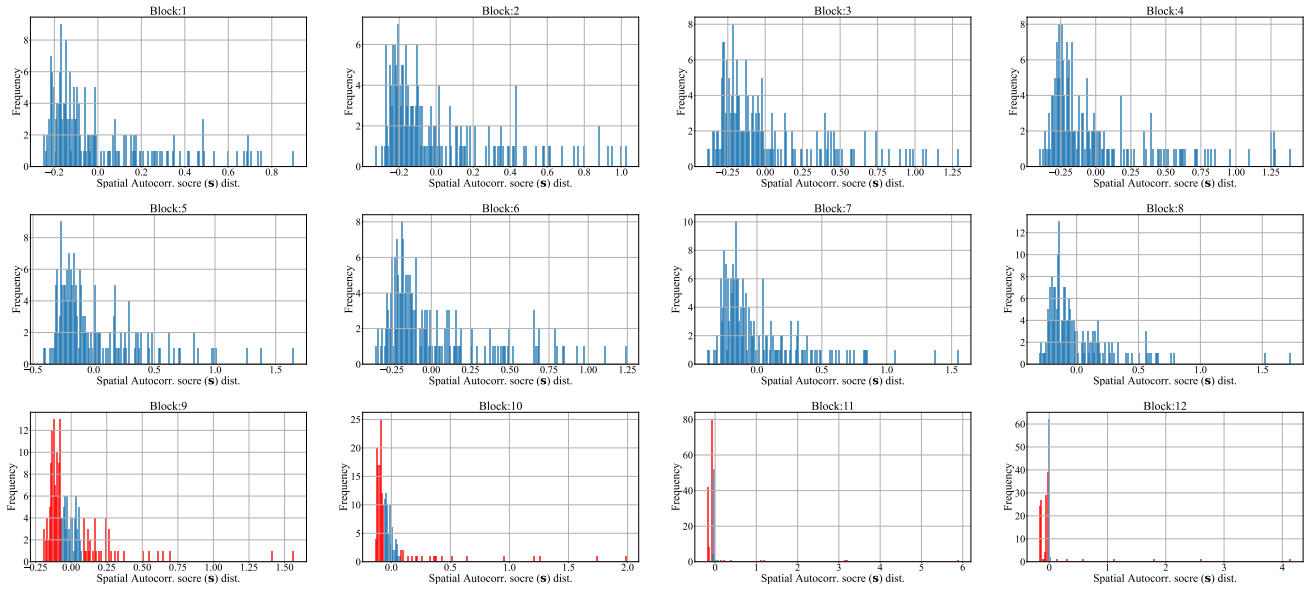

Figure 2. Visualising the distribution of spatial autocorrelation scores ($\mathbf{s}$) for patches (tokens) generated by various blocks of Deit-Base/16 on the ImageNet-1K validation set. In the last four blocks, tokens with $\mathbf{s}$ scores falling outside of the lower bound ($\mu_{\mathbf{s}} - |\hat{\mathbf{s}}|$) and upper bound ($\mu_{\mathbf{s}} + |\hat{\mathbf{s}}|$) are highlighted in red for the SATA process.

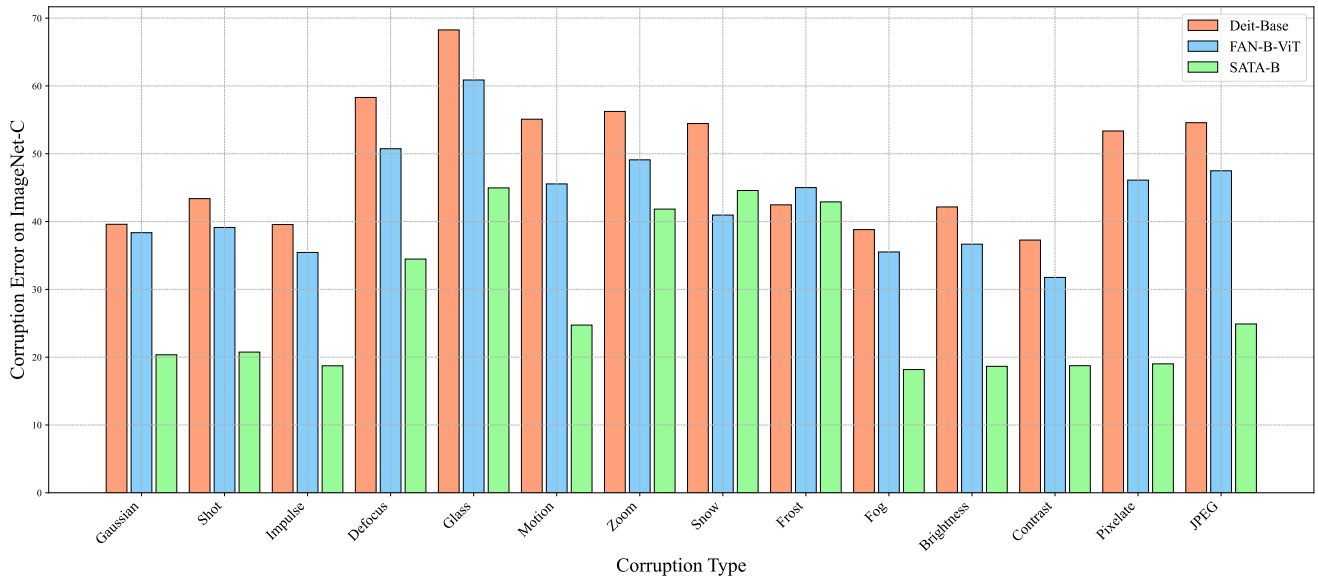

Figure 3. Comparisons of corruption error (the lower, the better) on individual corruption type of ImageNet-C between Deit-Base [4], FAN-B-ViT [8] and our SATA-B. Our SATA model significantly outperforms the other baseline models on all of the corruption types.

Table 1. Detailed Settings for Model Evaluation.† denotes the default settings of the baseline ViT.

Dataset	Image size	Batch size	Mean	Std
ImageNet2012-1K	224×224	256	†	†
IN-C	224×224	256	(0.485, 0.456, 0.406)	(0.229, 0.224, 0.225)
IN-A	224×224	256	(0.5,0.5,0.5)	(0.5,0.5,0.5)
IN-R	224×224	256	(0.5,0.5,0.5)	(0.5,0.5,0.5)
IN-SK	224×224	256	(0.5,0.5,0.5)	(0.5,0.5,0.5)

[4] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, pages 10347–10357. PMLR, 2021. 1, 3

[5] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. In *ECCV*, pages 459–479. Springer, 2022. 1

[6] Chenglin Yang, Siyuan Qiao, Qihang Yu, Xiaoding Yuan, Yukun Zhu, Alan Yuille, Hartwig Adam, and Liang-Chieh Chen. Moat: Alternating mobile convolution and attention brings strong vision models. In *ICLR*, 2022. 1

[7] Li Yuan, Qibin Hou, Zihang Jiang, Jiashi Feng, and Shuicheng Yan. Volo: Vision outlooker for visual recognition. *IEEE TPAMI*, 45(5):6575–6586, 2022. 1

[8] Daquan Zhou, Zhiding Yu, Enze Xie, Chaowei Xiao, Animashree Anandkumar, Jiashi Feng, and Jose M Alvarez. Understanding the robustness in vision transformers. In *ICML*, pages 27378–27394. PMLR, 2022. 1, 3

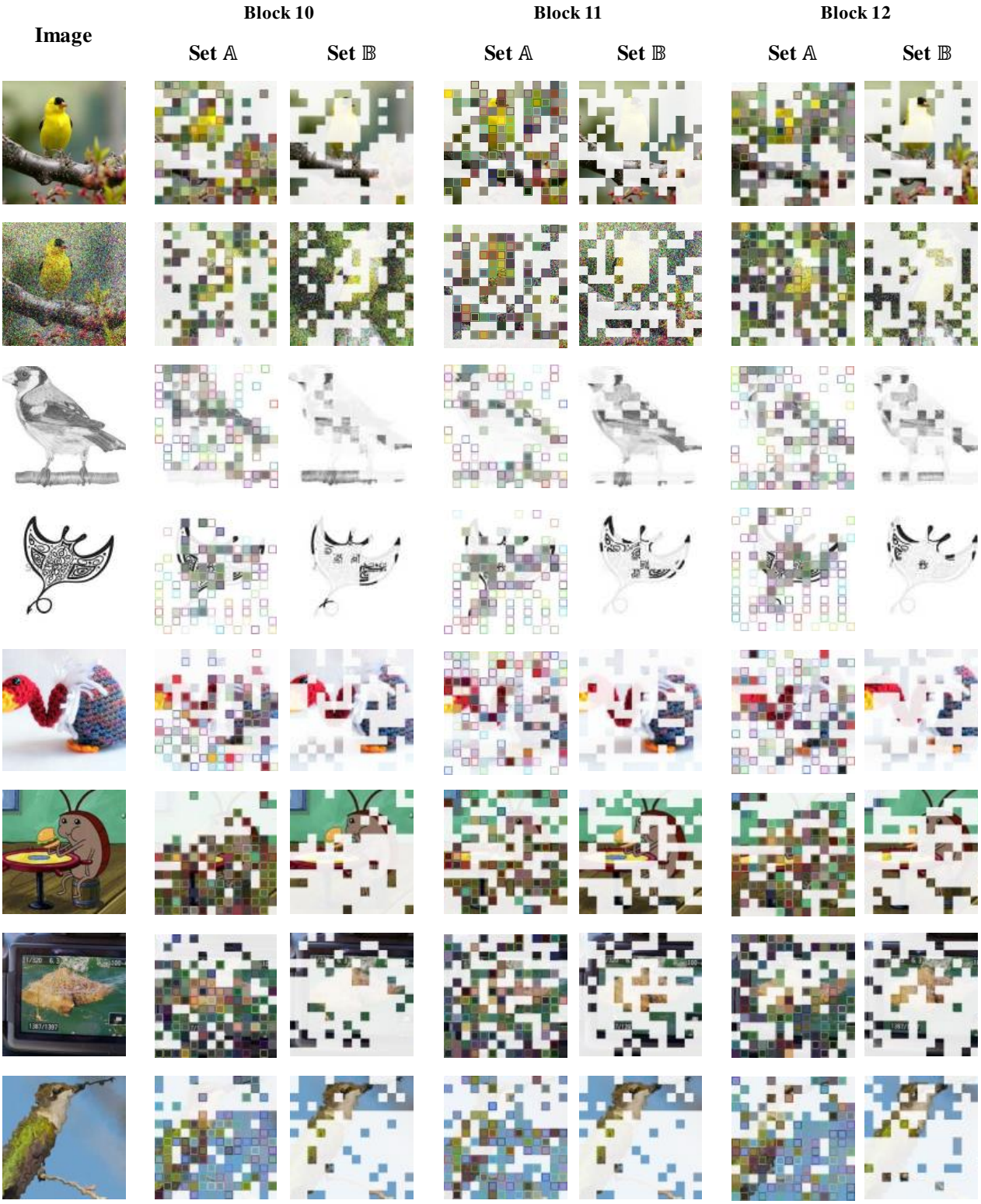

Figure 4. Visualisation of token splitting in Blocks 10 to 12 of SATA-B for images from ImageNet-1K, ImageNet-C, ImageNet-R, ImageNet-A, and ImageNet-SK.

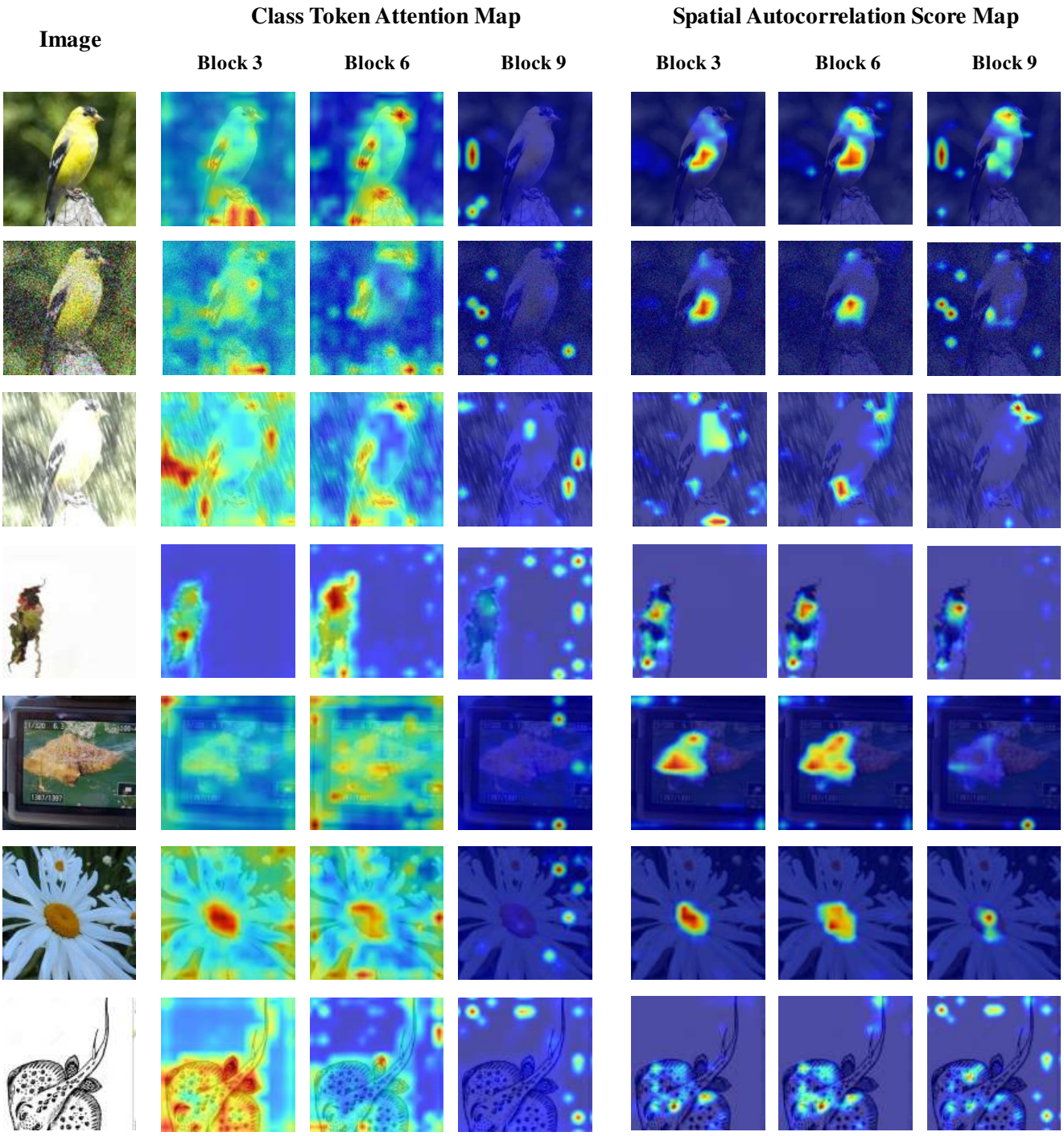

Figure 5. Visual comparison of class token attention maps and spatial autocorrelation score maps across three layers—representing early, middle, and later blocks—of the proposed SATA-B (pre-trained Deit-Base/16+SATA) for different images from ImageNet-1K, ImageNet-C, ImageNet-R, ImageNet-A, and ImageNet-SK.

```
python main-eval.py --model_name "deit_base_patch16_224" --gamma 0.7 --data_path
    ./ImageNet2012/val/ --sata
```

Figure 6. Command for verifying results on the ImageNet2012-1K validation set.

```
import torch
import torch.nn.functional as F

def SATA(x: torch.Tensor, M_att: torch.Tensor, alpha:float=1.0):
    """
    x: token embedding tensor  , [ batch_size, tokens (N), channels]
    M_att : attention map , [batch_size, N, N]
    alpha: bound controling factor
    """
    batch_size, num_token, channels = x.shape
    if num_token<2 or alpha<=0:
        return x, None
    ##### Spatial Autocorrelation
    # remove class token
    cls_token=x[...,0,:].view(batch_size,1,-1)
    x = x[...,1:,:]
    M_att = M_att[...,1:,:]
    M_att = M_att[...,:,1:]
    (batch_size, N , dim) = tuple(x.size())
    ###### Local Moran's index
    a = F.adaptive_avg_pool1d(x, 1) # compute global context attribute (Eq.6)
    a = torch.reshape(a, (batch_size, N, -1))
    z = (a-a.mean(1,keepdim=True))/a.std(1, keepdim=True) # Eq. 4

    z_t = z.transpose(-1, -2)
    zxz_t = z@z_t# B x N x N

    I_l = torch.reshape(torch.diagonal(zxz_t@M_att, dim1=1, dim2=2), (batch_size, N
        , 1))  # B x N x 1
    s = (I_l-I_l.mean(1,keepdim=True))/I_l.std(1,keepdim=True) # spatial
        autocorrelation score, Eq. 5 # B x N x 1

    ### Tokens Splitting and Grouping
    output_tokens, residual_tokens = split_group_by_scores(x,s_score=s,alpha=alpha)

    ## add class token
    output_tokens = torch.cat([cls_token, output_tokens],dim=1)

    return  output_tokens, residual_tokens
```

```python

def split_group_by_scores(x, s_score, alpha=1.):

   ####### Tokens Splitting
   # computing lower and upper bounds
    batch_size, num_token, channels = x.shape

    med_score,_ = torch.median(s_score, dim=1,keepdim=True)
    mean_score = torch.mean(s_score,dim=1,keepdim=True)

    lower_bound = (mean_score - torch.abs(med_score))*alpha
    upper_bound = (mean_score + torch.abs(med_score))*alpha

    set_B_mask = (s_score <= upper_bound) & (s_score >= lower_bound) # Eq.8

   #### Unification with regards to the batch_size
   # Step 1: Calculate the unified size for set B with regards to the batch size
    num_B_elements = torch.sum(set_B_mask).item()
    unified_size = int(num_B_elements / batch_size)
    unified_num_B = unified_size * batch_size
    num_B_to_swap = num_B_elements - unified_num_B  # Determine how many elements
       need to be swapped out

    if num_B_to_swap > 0:
       # Step 2: Sort the scores of elements in set_B_mask along with indices
       sorted_scores, sorted_indices = torch.sort(s_score[set_B_mask].view(-1))  #
           Sort the scores and get sorted indices

       # Step 3: Extract num_elements_to_swap highest-scored elements from
          set_B_mask and set to False
       selected_indices = sorted_indices[:num_B_to_swap]
       true_indices = torch.where(set_B_mask)  # Get the indices of true elements
          in set_B_mask
       set_B_mask[true_indices[0][selected_indices], true_indices[1][
          selected_indices],
                  true_indices[2][selected_indices]] = False  # Set the selected
                     elements to False
    ##########################################
    set_A_mask = ~set_B_mask # Eq.7

    set_B = x.masked_select(set_B_mask.expand_as(x)).view(batch_size,-1,channels)
    set_A = x.masked_select(set_A_mask.expand_as(x)).view(batch_size,-1,channels)

    A_num= set_A.size(1)

   ####### Tokens Grouping
    if A_num>2:
        merged_tokens, residual_tokens = token_merging(set_A,channels)
        output_tokens = torch.cat([set_B, merged_tokens], dim=1)
    else:
        residual_tokens = None
        output_tokens = set_B

    return output_tokens, residual_tokens
```

```python
def token_merging(x,token_size):

    batch_size = x.size(0)
    src, dst = x[..., ::2, :], x[..., 1::2, :]
    scores = src @ dst.transpose(-1, -2)

    node_max, node_idx = scores.max(dim=-1)
    src_idx = node_max.argsort(dim=-1, descending=True)[..., None]
    dst_idx = node_idx[..., None].gather(dim=-2, index=src_idx)


    res_tokens = src.gather(dim=-2, index=src_idx.expand(batch_size, -1,
        token_size))
    mrg_tokens = dst.scatter_reduce(-2, dst_idx.expand(batch_size, -1, token_size)
        , src, reduce="mean")
    return mrg_tokens,res_tokens
```

Figure 7. Implementation of the proposed "*Spatial Autocorrelation Token Analysis*" (SATA) in PyTorch [3].