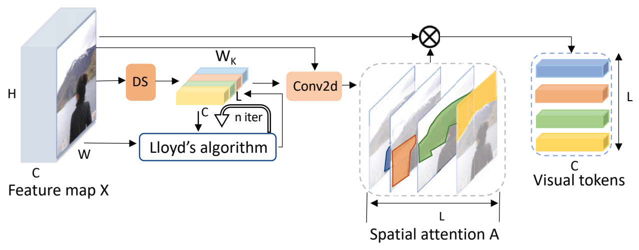


S

Figure 7: Cluster-based tokenizer that group pixels using the K-Means centroids of the pixels in the semantic space.

A. Stage-wise model description of VT-ResNet

In this section, we provide a stage-wise description of the model configurations for VT-based ResNet (VT-ResNet). We use three hyper-parameters to control a VT module: channel size of the output feature map C, channel size of visual token CT, and the number of visual tokens L. The model configurations are described in Table 11.

B. More visualization results

We provide more visualization of the spatial attention on images from the LIP dataset in Figure 8.

C. Clustering-based tokenizer

To address this limitation of filter-based tokenizers, we devise a content-aware $\mathbf{W}_K$ variant of $\mathbf{W}_A$ to form semantic groups from $\mathbf{X}$. The module is shown in Fig. 7. Our insight is to extract concepts present in the current image by clustering pixels, instead of applying the same filters regardless of the image content. First, we treat each pixel as a sample $\{\mathbf{X}_p\}_{p=1}^{HW}$, and apply k-means to find L centroids, which are stacked to form $\mathbf{W}_K \in \mathbb{R}^{C\times L}$. Each centroid represents one semantic concept in the image. Second, we replace $\mathbf{W}_A$ in Equation (1) with $\mathbf{W}_K$ to form L semantic groups of channels:

$$\begin{aligned} \mathbf{W}_K &= \text{KMEANS}(\mathbf{X}), \\ \mathbf{T} &= \text{SOFTMAX}_{HW}\left(\mathbf{X}\mathbf{W}_K\right)^T \mathbf{X}. \end{aligned} \tag{6}$$

Pseudo-code for our K-means implementation is provided in Listing 1, and can be summarized as: Normalize all pixels to unit vectors, initialize centroids with a spatially-downsampled feature map, and run Lloyd's algorithm to produce centroids.

```

def kmeans(X_nchw, L, niter):
  # Input:
  #   X_nchw - feature map
  #   L - num token
  #   niter - num iters of Lloyd's
  N, C, H, W = X_nchw.size()
  # Initialization as down-sampled X
  U_ncl = downsample(X).view(N, C, L)
  X_ncp = X_nchw.view(N, C, H*W)  # p = h*w
  # Normalize to unit vectors
  U_ncl = U_ncl.normalize(dim=1)
  X_ncp = X_ncp.normalize(dim=1)
  for _ in range(niter):  # Lloyd's algorithm
    dist_npl = (
      X_ncp[..., None] - U_ncl[:, :, None, :]
    ).norm(dim=1)
    mask_npl = (dist_npl == dist_npl.min(dim=2)
    U_ncl = X_ncp.MatMul(mask_npl)
    U_ncl = U_ncl / mask_npl.sum(dim=1)
    U_ncl = U_ncl.normalize(dim=1)
  return U_ncl  # centroids
```

Listing 1: Pseudo-code of K-Means implemented in PyTorch-like language

Although this tokenizer efficiently models only concepts in the current image, the drawback is that it is not designed to choose the most discriminative concepts.

Stage	Resolution	VT-R18	VT-R34	VT-R50	VT-R101
1	56×56	conv7×7-C64-S2 → maxpool3×3-S2			
2	56×56	BB-C64 ×2	BB-C64 ×3	BN-C256 ×3	BN-C256 ×3
3	28×28	BB-C128 ×2	BB-C128 ×4	BN-C512 ×4	BN-C512 ×4
4	14×14	BB-C256 ×2	BB-C256 ×6	BN-C1024 ×6	BN-C1024 ×23
5	16	VT-C512-L16 -CT1024 ×2	VT-C512-L16 -CT1024 ×3	VT-C1024-L16 -CT1024 ×3	VT-C1024-L16 -CT1024 ×3
head	1	avgpool-fc1000			

Table 11: Model descriptions for VT-ResNets. VT-R18 denotes visual-transformer-ResNet-18. "conv7×7-C64-S2" denotes a 7-by-7 convolution with an output channel size of 64 and a stride of 2. "BB-C64×2" denotes a basic block [14] with an output channel size of 64 and it is repeated twice. "BN-C256×3" denotes a bottleneck block [14] with an output channel size of 256 and it is repeated by three times. "VT-C512-L16-CT1024 ×2" denotes a VT block with a channel size for the output feature map as 512, channel size for visual tokens as 1024, and the number of tokens as 16. We leave exploration of other clustering methods like agglomerative clustering to future work.

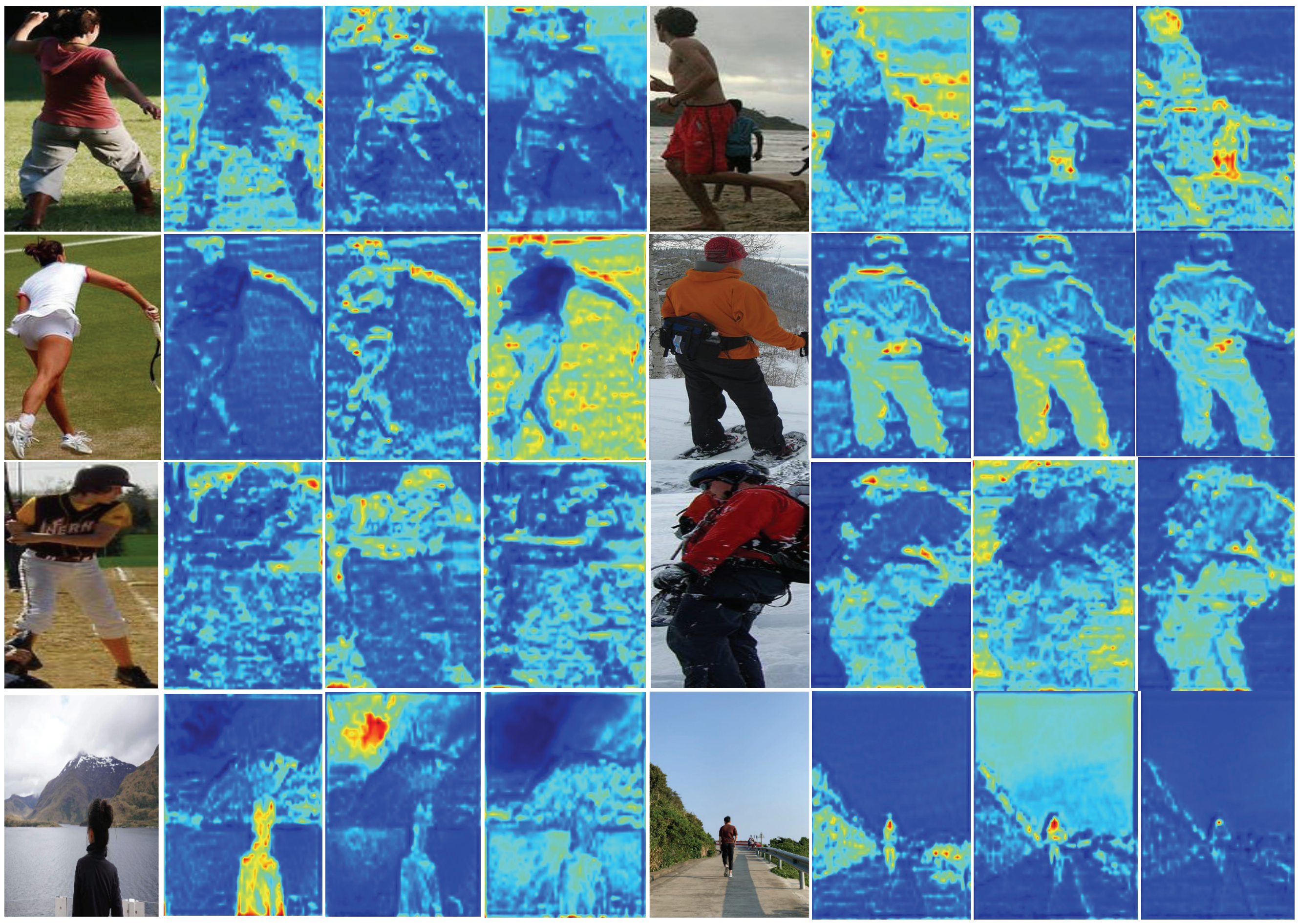
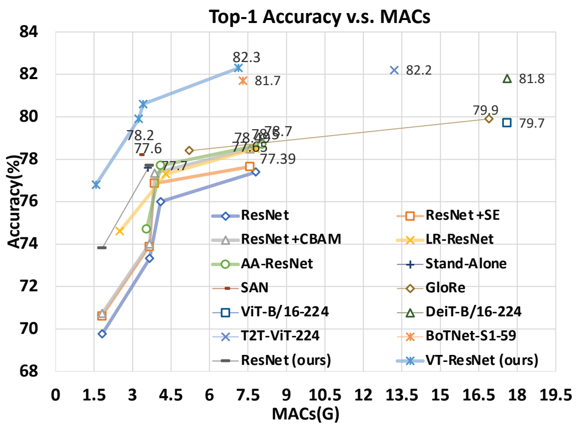


Figure 8: Visualization of the spatial attention generated by a filter-based tokenizer. Red denotes higher attention values and color blue denotes lower. Without any supervision, visual tokens automatically focus on different areas of the image that correspond to different semantic concepts.



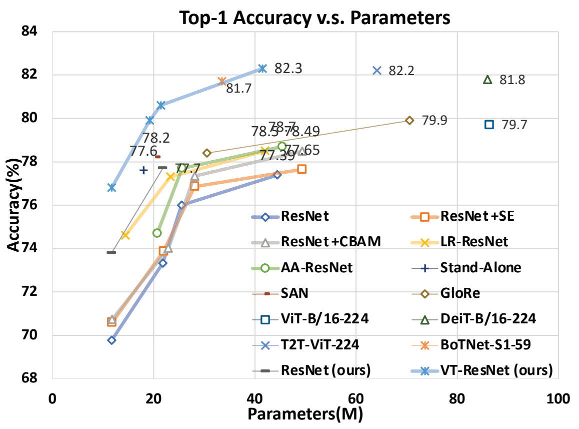


Figure 9: Our VTs (blue) significantly outperform baselines in the accuracy-parameter and accuracy-macs tradeoff.