

# Visual Transformers: Token-based Image Representation and Processing for Computer Vision

Bichen Wu<sup>1</sup>, Chenfeng Xu<sup>3</sup>, Xiaoliang Dai<sup>1</sup>, Alvin Wan<sup>3</sup>, Peizhao Zhang<sup>1</sup>  
 Zhicheng Yan<sup>2</sup>, Masayoshi Tomizuka<sup>3</sup>, Joseph Gonzalez<sup>3</sup>, Kurt Keutzer<sup>3</sup>, Peter Vajda<sup>1</sup>  
<sup>1</sup> Facebook Reality Labs, <sup>2</sup> Facebook AI, <sup>3</sup> UC Berkeley  
 {wbc, xiaoliangdai, stzp, zyan3, vajdap}@fb.com  
 {xuchenfeng, alvinwan, tomizuka, jegonzal, keutzer}@berkeley.edu

## Abstract

*Computer vision has achieved remarkable success by (a) representing images as uniformly-arranged pixel arrays and (b) convolving highly-localized features. However, convolutions treat all image pixels equally regardless of importance; explicitly model all concepts across all images, regardless of content; and struggle to relate spatially-distant concepts. In this work, we challenge this paradigm by (a) representing images as semantic visual tokens and (b) running transformers to densely model token relationships. Critically, our **Visual Transformer** operates in a semantic token space, judiciously attending to different image parts based on context. This is in sharp contrast to pixel-space transformers that require orders-of-magnitude more compute. Using an advanced training recipe, our VTs significantly outperform their convolutional counterparts, raising ResNet accuracy on ImageNet top-1 by **4.6 to 7 points** while using fewer FLOPs and parameters. For semantic segmentation on LIP and COCO-stuff, VT-based feature pyramid networks (FPN) achieve 0.35 points higher mIoU while reducing the FPN module’s FLOPs by **6.5x**.*

## 1. Introduction

In computer vision, visual information is captured as arrays of pixels. These pixel arrays are then processed by convolutions, the *de facto* deep learning operator for computer vision. Although this convention has produced highly successful vision models, there are critical challenges:

**1) Not all pixels are created equal:** Image classification models should prioritize foreground objects over the background. Segmentation models should prioritize pedestrians over disproportionately large swaths of sky, road, vegetation etc. Nevertheless, convolutions uniformly process all image patches regardless of importance. This leads to spatial inefficiency in both computation and representation.

**2) Not all images have all concepts:** Low-level features such as corners and edges exist in all natural images, so applying low-level convolutional filters to *all* images is appropriate. However, high-level features such as ear shape exist in specific images, so applying high-level filters to all images is computationally inefficient. For example, dog features may not appear in images of flowers, vehicles, aquatic animals etc. This results in rarely-used, inapplicable filters expending a significant amount of compute.

**3) Convolutions struggle to relate spatially-distant concepts:** Each convolutional filter is constrained to operate on a small region, but long-range interactions between semantic concepts is vital. To relate spatially-distant concepts, previous approaches increase kernel sizes, increase model depth, or adopt new operations like dilated convolutions, global pooling, and non-local attention layers. However, by working within the pixel-convolution paradigm, these approaches at best mitigate the problem, compensating for the convolution’s weaknesses by adding model and computational complexity.

To overcome the above challenges, we address the root cause, the pixel-convolution paradigm, and introduce the *Visual Transformer* (VT) (Figure 1), a new paradigm to represent and process high-level concepts in images. Our intuition is that a sentence with a few words (or visual tokens) suffices to describe high-level concepts in an image. This motivates a departure from the fixed pixel-array representation later in the network; instead, we use spatial attention to convert the feature map into a compact set of semantic tokens. We then feed these tokens to a transformer, a self-attention module widely used in natural language processing [37] to capture token interactions. The resulting visual tokens computed can be directly used for image-level prediction tasks (e.g., classification) or be spatially re-projected to the feature map for pixel-level prediction tasks (e.g., segmentation). Unlike convolutions, our VT can better handle the three challenges above: 1) judiciously allocating com-

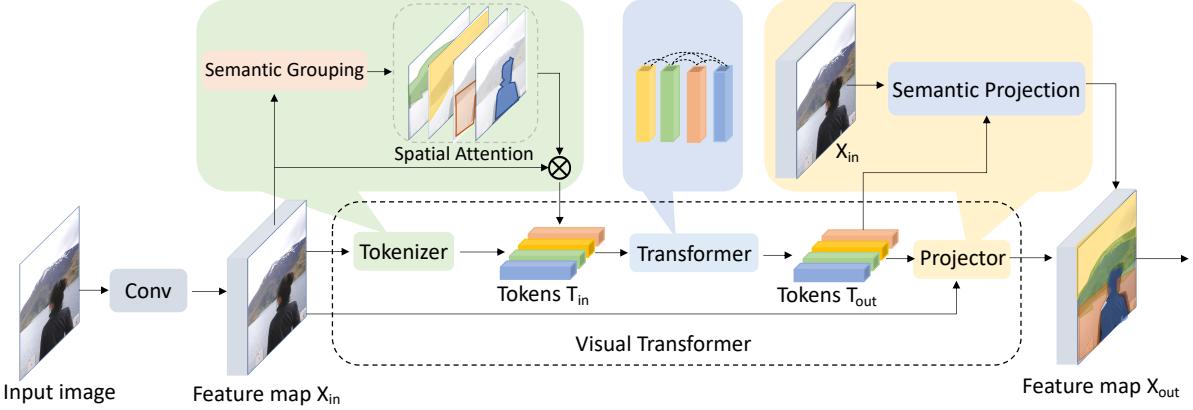


Figure 1: Diagram of a Visual Transformer (VT). For a given image, we first apply convolutional layers to extract low-level features. The output feature map is then fed to VT: First, apply a *tokenizer*, grouping pixels into a small number of *visual tokens*, each representing a semantic concept in the image. Second, apply transformers to model relationships between tokens. Third, visual tokens are directly used for image classification or projected back to the feature map for semantic segmentation.

putation by attending to important regions, instead of treating all pixels equally; 2) encoding semantic concepts in a few visual tokens relevant to the image, instead of modeling all concepts across all images; and 3) relating spatially-distant concepts through self-attention in token-space.

To validate the effectiveness of VT and understanding its key components, we run controlled experiments by using VTs to replace convolutions in ResNet, a common test bed for new building blocks for image classification. We also use VTs to re-design feature-pyramid networks (FPN), a strong baseline for semantic segmentation. Our experiments show that VTs achieve higher accuracy with lower computational cost in both tasks. For the ImageNet[11] benchmark, we replace the last stage of ResNet[14] with VTs, reducing FLOPs of the stage by 6.9x and improving top-1 accuracy by **4.6 to 7 points**. For semantic segmentation on COCO-Stuff [2] and Look-Into-Person [25], VT-based FPN achieves 0.35 points higher mIOU while reducing regular FPN module’s FLOPs by 6.4x.

## 2. Relationship to previous work

**Transformers in vision models:** A notable recent and relevant trend is the adoption of transformers in vision models. Dosovitskiy *et al.* propose a Vision Transformer (ViT) [12], dividing an image into  $16 \times 16$  patches and feeding these patches (i.e., tokens) into a standard transformer. Although simple, this requires transformers to learn dense, repeatable patterns (e.g., textures), which convolutions are drastically more efficient at learning. The simplicity incurs an extremely high computational price: ViT requires up to 7 GPU years and 300M JFT dataset images to outperform competing convolutional variants. By contrast, we leverage the respective strengths of each operation, using convolutions for extracting low-level features and transformers for

relating high-level concepts. We further use spatial attention to focus on important regions, instead of treating each image patch equally. This yields strong performance despite orders-of-magnitude less data and training time.

Another relevant work, DETR[3], adopts transformers to simplify the hand-crafted anchor matching procedure in object detection training. Although both adopt transformers, DETR is not directly comparable to our VT given their orthogonal use cases, i.e., insights from both works could be used together in one model for compounded benefit.

**Graph convolutions in vision models:** Our work is also related to previous efforts such as GloRe [6], Latent-GNN [47], and [26] that densely relate concepts in latent space using graph convolutions. To augment convolutions, [26, 6, 47] adopt a procedure similar to ours: (1) extracting latent variables to represent in graph nodes (analogous to our visual tokens) (2) applying graph convolution to capture node interactions (analogous to our transformer), and (3) projecting the nodes back to the feature map. Although these approaches avoid spatial redundancy, they are susceptible to concept redundancy: the second limitation listed in the introduction. In particular, by using fixed weights that are not content-aware, the graph convolution expects a fixed semantic concept in each node, regardless of whether the concept exists in the image. By contrast, a transformer uses content-aware weights, allowing visual tokens to represent varying concepts. As a result, while graph convolutions require hundreds of nodes (128 nodes in [4], 340 in [25], 150 in [48]) to encode potential semantic concepts, our VT uses just 16 visual tokens and attains higher accuracy. Furthermore, while modules from [26, 6, 47] can only be added to a pretrained network to augment convolutions, VTs can replace convolution layers to save FLOPs and parameters, and support training from scratch.

**Attention in vision models:** In addition to being used in

transformers, attention is also widely used in different forms in computer vision models [21, 20, 41, 44, 46, 40, 28, 18, 19, 1, 49, 30]. Attention was first used to modulate the feature map: attention values are computed from the input and multiplied to the feature map as in [41, 21, 20, 44]. Later work [46, 33, 39] interpret this “modulation” as a way to make convolution spatially adaptive and content-aware. In [40], Wang *et al.* introduced non-local operators, equivalent to self-attention, to video understanding to capture the long-range interactions. However, the computational complexity of self-attention grows quadratically with the number of pixels. [1] use self-attention to augment convolutions and reduce the compute cost by using small channel sizes for attention. [30, 28, 7, 49, 19] on the other hand restrict receptive field of self-attention and use it in a convolutional manner. Starting from [30], self-attentions are used as a stand-alone building block for vision models. Our work is different from all above since we propose a novel token-transformer paradigm to replace the inefficient pixel-convolution paradigm and achieve superior performance.

**Efficient vision models:** Many recent research efforts have been focusing on building vision models to achieve better performance with lower computational cost. Early work in this direction includes [23, 31, 13, 17, 32, 16, 48, 27, 43]. Recently, people use neural architecture search [42, 10, 38, 9, 36, 35] to optimize network’s performance within a search space that consists of existing convolution operators. The efforts above all seek to make the common convolutional-neural net more computationally efficient. In contrast, we propose a new building block that naturally eliminates the redundant computations in the pixel-convolution paradigm.

### 3. Visual Transformer

We illustrate the overall diagram of a *Visual Transformer* (VT) based model in Figure 1. First, process the input image with several convolution blocks, then feed the output feature map to VTs. Our insight is to leverage the strengths of both convolutions and VTs: (1) early in the network, use convolutions to learn densely-distributed, low-level patterns and (2) later in the network, use VTs to learn and relate more sparsely-distributed, higher-order semantic concepts. At the end of the network, use visual tokens for image-level prediction tasks and use the augmented feature map for pixel-level prediction tasks.

A VT module involves three steps: First, group pixels into semantic concepts, to produce a compact set of *visual tokens*. Second, to model relationships between semantic concepts, apply a transformer [37] to these visual tokens. Third, project these visual tokens back to pixel-space to obtain an augmented feature map. Similar paradigms can be found in [6, 47, 26] but with one critical difference: Previous methods use hundreds of semantic concepts (termed,

“nodes”), whereas our VT uses as few as 16 visual tokens to achieve superior performance.

#### 3.1. Tokenizer

Our intuition is that an image can be summarized by a few handfuls of words, or *visual tokens*. This contrasts convolutions, which use hundreds of filters, and graph convolutions, which use hundreds of “latent nodes” to detect all possible concepts regardless of image content. To leverage this intuition, we introduce a *tokenizer* module to convert feature maps into compact sets of visual tokens. Formally, we denote the input feature map by  $\mathbf{X} \in \mathbb{R}^{HW \times C}$  (height  $H$ , width  $W$ , channels  $C$ ) and visual tokens by  $\mathbf{T} \in \mathbb{R}^{L \times C}$  s.t.  $L \ll HW$  ( $L$  represents the number of tokens).

##### 3.1.1 Filter-based Tokenizer

A filter-based tokenizer, also adopted by [47, 6, 26], utilizes convolutions to extract visual tokens. For feature map  $\mathbf{X}$ , we map each pixel  $\mathbf{X}_p \in \mathbb{R}^C$  to one of  $L$  semantic groups using point-wise convolutions. Then, within each group, we spatially pool pixels to obtain tokens  $\mathbf{T}$ . Formally,

$$\mathbf{T} = \underbrace{\text{SOFTMAX}_{HW}(\mathbf{X}\mathbf{W}_A)^T\mathbf{X}}_{\mathbf{A} \in \mathbb{R}^{HW \times L}} \quad (1)$$

Here,  $\mathbf{W}_A \in \mathbb{R}^{C \times L}$  forms semantic groups from  $\mathbf{X}$ , and  $\text{SOFTMAX}_{HW}(\cdot)$  translates these activations into a spatial attention. Finally,  $\mathbf{A}$  multiplies with  $\mathbf{X}$  and computes weighted averages of pixels in  $\mathbf{X}$  to make  $L$  visual tokens.

However, many high-level semantic concepts are sparse and may each appear in only a few images. As a result, the fixed set of learned weights  $\mathbf{W}_A$  potentially wastes computation by modeling all such high-level concepts at once. We call this a “filter-based” tokenizer, since it uses convolutional filters  $\mathbf{W}_A$  to extract visual tokens.

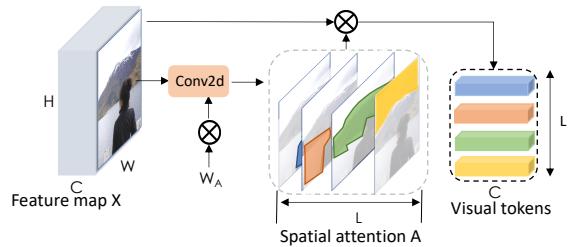


Figure 2: Filter-based tokenizer that use convolution to group pixels using a fixed convolution filter.

##### 3.1.2 Recurrent Tokenizer

To remedy the limitation of filter-based tokenizers, we propose a recurrent tokenizer with weights that are dependent

on previous layer’s visual tokens. The intuition is to let the previous layer’s tokens  $\mathbf{T}_{in}$  guide the extraction of new tokens for the current layer. The name of “recurrent tokenizer” comes from that current tokens are computed dependent on previous ones. Formally, we define

$$\begin{aligned}\mathbf{W}_R &= \mathbf{T}_{in} \mathbf{W}_{T \rightarrow R}, \\ \mathbf{T} &= \text{SOFTMAX}_{HW} (\mathbf{X} \mathbf{W}_R)^T \mathbf{X},\end{aligned}\quad (2)$$

where  $\mathbf{W}_{T \rightarrow R} \in \mathbb{R}^{C \times C}$ . In this way, the VT can incrementally refine the set of visual tokens, conditioned on previously-processed concepts. In practice, we apply recurrent tokenizers starting from the second VT, since it requires tokens from a previous VT.

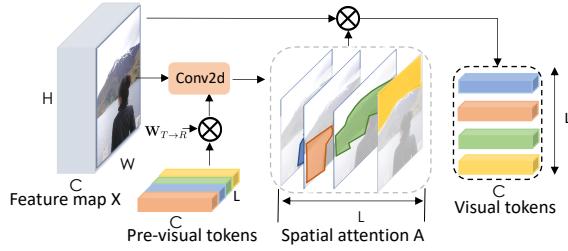


Figure 3: Recurrent tokenizer that uses previous tokens to guide the token extraction in the current VT module.

### 3.2. Transformer

After tokenization, we then need to model interactions between these visual tokens. Previous works [6, 47, 26] use graph convolutions to relate concepts. However, these operations use fixed weights during inference, meaning each token (or “node”) is bound to a specific concept, therefore graph convolutions waste computation by modeling all high-level concepts, even those that only appear in few images. To address this, we adopt transformers [37], which use input-dependent weights by design. Due to this, transformers support visual tokens with variable meaning, covering more possible concepts with fewer tokens.

We employ a standard transformer with minor changes:

$$\mathbf{T}'_{out} = \mathbf{T}_{in} + \text{SOFTMAX}_L ((\mathbf{T}_{in} \mathbf{K})(\mathbf{T}_{in} \mathbf{Q})^T) \mathbf{T}_{in}, \quad (3)$$

$$\mathbf{T}_{out} = \mathbf{T}'_{out} + \sigma(\mathbf{T}'_{out} \mathbf{F}_1) \mathbf{F}_2, \quad (4)$$

where  $\mathbf{T}_{in}, \mathbf{T}'_{out}, \mathbf{T}_{out} \in \mathbb{R}^{L \times C}$  are the visual tokens. Different from graph convolution, in a transformer, weights between tokens are input-dependent and computed as a key-query product:  $(\mathbf{T}_{in} \mathbf{K})(\mathbf{T}_{in} \mathbf{Q})^T \in \mathbb{R}^{L \times L}$ . This allows us to use as few as 16 visual tokens, in contrast to hundreds of analogous nodes for graph-convolution approaches [6, 47, 26]. After the self-attention, we use a non-linearity and two pointwise convolutions in Equation (4), where  $\mathbf{F}_1, \mathbf{F}_2 \in \mathbb{R}^{C \times C}$  are weights,  $\sigma(\cdot)$  is the ReLU function.

### 3.3. Projector

Many vision tasks require pixel-level details, but such details are not preserved in visual tokens. Therefore, we fuse the transformer’s output with the feature map to refine the feature map’s pixel-array representation as

$$\mathbf{X}_{out} = \mathbf{X}_{in} + \text{SOFTMAX}_L ((\mathbf{X}_{in} \mathbf{W}_Q)(\mathbf{T} \mathbf{W}_K)^T) \mathbf{T}, \quad (5)$$

where  $\mathbf{X}_{in}, \mathbf{X}_{out} \in \mathbb{R}^{HW \times C}$  are the input and output feature map.  $(\mathbf{X}_{in} \mathbf{W}_Q) \in \mathbb{R}^{HW \times C}$  is the query computed from the input feature map  $\mathbf{X}_{in}$ .  $(\mathbf{X}_{in} \mathbf{W}_Q)_p \in \mathbb{R}^C$  encodes the information pixel- $p$  requires from the visual tokens.  $(\mathbf{T} \mathbf{W}_K) \in \mathbb{R}^{L \times C}$  is the key computed from the token  $\mathbf{T}$ .  $(\mathbf{T} \mathbf{W}_K)_l \in \mathbb{R}^C$  represents the information the  $l$ -th token encodes. The key-query product determines how to project information encoded in visual tokens  $\mathbf{T}$  to the original feature map.  $\mathbf{W}_Q \in \mathbb{R}^{C \times C}, \mathbf{W}_K \in \mathbb{R}^{C \times C}$  are learnable weights used to compute queries and keys.

## 4. Using Visual Transformers in vision models

In this section, we discuss how to use VTs as building blocks in vision models. We define three hyper-parameters for each VT: channel size of the feature map; channel size of the visual tokens; and the number of visual tokens.

**Image classification model:** For image classification, following the convention of previous work, we build our networks with backbones inherited from ResNet [14]. Based on ResNet-{18, 34, 50, 101}, we build corresponding visual-transformer-ResNets (VT-ResNets) by replacing the last stage of convolutions with VT modules. The last stage of ResNet-{18, 34, 50, 101} contains 2 basic blocks, 3 basic blocks, 3 bottleneck blocks, and 3 bottleneck blocks, respectively. We replace them with the same number (2, 3, 3, 3) of VT modules. At the end of stage-4 (before stage-5 max pooling), ResNet-{18, 34} generate feature maps with the shape of  $14^2 \times 256$ , and ResNet-{50, 101} generate feature maps with the shape of  $14^2 \times 1024$ . We set VT’s feature map channel size to be 256, 256, 1024, 1024 for ResNet-{18, 34, 50, 101}. We adopt 16 visual tokens with a channel size of 1024 for all the models. At the end of the network, we output 16 visual tokens to the classification head, which applies an average pooling over the tokens and use a fully-connected layer to predict the probabilities. A table summarizing the stage-wise description of the model is provided in Appendix A. Since VTs only operate on 16 visual tokens, we can reduce the last stage’s FLOPs by up to 6.9x, as shown in Table 1.

**Semantic segmentation:** We show that using VTs for semantic segmentation can tackle several challenges with the pixel-convolution paradigm. First, the computational complexity of convolution grows with the image resolution. Second, convolutions struggles to capture long-term

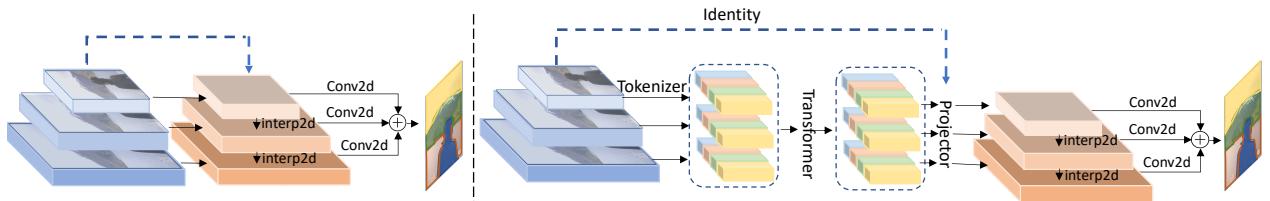


Figure 4: Feature Pyramid Networks (FPN) (left) vs visual-transformer-FPN (VT-FPN) (right) for semantic segmentation. FPN uses convolution and interpolation to merge feature maps with different resolutions. VT-FPN extract visual tokens from all feature maps, merge them with one transformer, and project back to the original feature maps.

	R18	R34	R50	R101
FLOPs	Total	1.14x	1.16x	1.20x
	Stage-5	2.4x	5.0x	6.1x
Params	Total	0.91x	1.21x	1.19x
	Stage-5	0.9x	1.5x	1.26x

Table 1: FLOPs and parameter size reduction of VTs on ResNets by replacing the last stage of convolution modules with VT modules.

interactions between pixels. VTs, on the other hand, operate on a small number of visual tokens regardless of the image resolution, and since it models concept interactions in the token-space, it bypasses the “long-range” challenge with pixel-arrays.

To validate our hypothesis, we use panoptic feature pyramid networks (FPN) [24] as a baseline and use VTs to improve the network. Panoptic FPNs use ResNet as backbone to extract feature maps from different stages with various resolutions. These feature maps are then fused by a feature pyramid network in a top-down manner to generate a multi-scale and detail preserving feature map with rich semantics for segmentation (Figure 4 left). FPN is computationally expensive since it heavily relies on spatial convolutions operating on high resolution feature maps with large channel sizes. We use VTs to replace convolutions in FPN. We name the new module as VT-FPN (Figure 4 right). From each resolution’s feature map, VT-FPN extract 8 visual tokens with a channel size of 1024. The visual tokens are combined and fed into one transformer to compute interactions between visual tokens across resolutions. The output tokens are then projected back to the original feature maps, which are then used to perform pixel-level prediction. Compared with the original FPN, the computational cost for VT-FPN is much smaller since we only operate on a very small number of visual tokens rather than all the pixels. Our experiment shows VT-FPN uses 6.4x fewer FLOPs than FPN while preserving or surpassing its performance (Table 9 & 10).

## 5. Experiments

We conduct experiments with VTs on image classification and semantic segmentation to (a) understand the key components of VTs and (b) validate their effectiveness.

### 5.1. Visual Transformer for Classification

We conduct experiments on the ImageNet dataset [11] with around 1.3 million images in the training set and 50 thousand images in the validation set. We implement VT models in PyTorch [29]. We use stochastic gradient descent (SGD) optimizer with Nesterov momentum [34]. We use an initial learning rate of 0.1, a momentum of 0.9, and a weight decay of 4e-5. We train the model for 90 epochs, and decay the learning rate by 10x every 30 epochs. We use a batch size of 256 and 8 V100 GPUs for training.

**VT vs. ResNet with default training recipe:** In Table 2, we first compare VT-ResNets and vanilla ResNets under the same training recipe. VT-ResNets in this experiment use a filter-based tokenizer for the first VT module and recurrent tokenizers in later modules. We can see that after replacing the last stage of convolutions in ResNet18 and ResNet34, VT-based ResNets use many fewer FLOPs: 244M fewer FLOPs for ResNet18 and 384M fewer for ResNet34. Meanwhile, VT-ResNets achieve much higher top-1 validation accuracy than the corresponding ResNets by up to 2.2 points. This confirms effectiveness of VTs. Also note that the training accuracy achieved by VT-ResNets are much higher than that of baseline ResNets: VT-R18 is 7.9 points higher and VT-R34 is 6.9 points higher. This indicates that VT-ResNets are overfitting more heavily than regular ResNets. We hypothesize this is because VT-ResNets have much larger capacity and we need stronger regularization (e.g., data augmentation) to fully utilize the model capacity. We address this in Section 5.2 and Table 8.

**Tokenizer ablation studies:** In Table 3, we compare different types of tokenizers used by VTs. We consider a *pooling-based* tokenizer, a *clustering-based* tokenizer, and a filter-based tokenizer (Section 3.1.1). We use the candidate tokenizer in the first VT module and use recurrent tokenizers in later modules. As a baseline, we implement a *pooling-based* tokenizer, which spatially downsamples a feature map  $\mathbf{X}$  to reduce its spatial dimensions from  $HW = 196$  to  $L = 16$ , instead of grouping pixels by their semantics. As a more advanced baseline, we consider a clustering-based tokenizer, which is described in Appendix C. It applies K-Means clustering in the semantic space to group pixels to visual tokens. As can be seen from Ta-

	Top-1 Acc (%) (Val)	Top-1 Acc (%) (Train)	FLOPs (M)	Params (M)
R18	69.9	68.6	1814	11.7
VT-R18	<b>72.1</b>	76.5	1570	11.7
R34	73.3	73.9	3664	21.8
VT-R34	<b>75.0</b>	80.8	3280	21.9

Table 2: VT-ResNet vs. baseline ResNets on the ImageNet dataset. By replacing the last stage of ResNets, VT-ResNet uses 224M, 384M fewer FLOPs than the baseline ResNets while achieving 1.7 points and 2.2 points higher validation accuracy. Note the training accuracy of VT-ResNets are much higher. This indicates VT-ResNets have higher model capacity and require stronger regularization (e.g., data augmentation) to fully utilize the model. See Table 8.

	Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	Pooling-based	70.5	1549
	Clustering-based	71.8	1579
	Filter-based	<b>72.1</b>	1580
R34	Pooling-based	73.6	3246
	Clustering-based	<b>75.2</b>	3299
	Filter-based	74.9	3280

Table 3: VT-ResNets using different types of tokenizers. Pooling-based tokenizers spatially downsample a feature map to obtain visual tokens. Clustering-based tokenizer (Appendix C) groups pixels in the semantic space. Filter-based tokenizers (3.1.1) use convolution filters to group pixels. Both filter-based and cluster-based tokenizers work much better than pooling-based tokenizers, validating the importance of grouping pixels by their semantics.

ble 3, filter-based and clustering-based tokenizers perform significantly better than the pooling-based baseline, validating our hypothesis that feature maps contain redundancies, and this can be addressed by grouping pixels in semantic space. The difference between filter-based and clustering-based tokenizers is small and vary between R18 and R34. We hypothesize this is because both tokenizers have their own drawbacks. The filter-based tokenizer relies on fixed convolution filters to detect and assign pixels to semantic groups, and is limited by the capacity of the convolution filters to deal with diverse and sparse high-level semantic concepts. On the other hand, the clustering-based tokenizer extracts semantic concepts that exist in the image, but it is not designed to capture the essential semantic concepts.

In Table 4, we validate the recurrent tokenizer’s effectiveness. We use a filter-based tokenizer in the first VT module and use recurrent tokenizers in subsequent modules. Experiments show that using recurrent tokenizers leads to higher accuracy.

	Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	w/ RT	<b>72.0</b>	1569
	w/o RT	71.2	1586
R34	w/ RT	<b>74.9</b>	3246
	w/o RT	74.5	3335

Table 4: VT-ResNets that use recurrent tokenizers achieve better performance, since recurrent tokenizers are content-aware. RT denotes recurrent tokenizer.

	Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	None	68.7	1528
	GraphConv	69.3	1528
	Transformer	<b>71.5</b>	1580
R34	None	73.3	3222
	GraphConv	73.7	3223
	Transformer	<b>75.2</b>	3299

Table 5: VT-ResNets using different modules to model token relationships. Models using transformers perform better than graph-convolution or no token-space operations. This validates that it is important to model relationships between visual token (semantic concepts) and transformer work better than graph convolution in relating tokens.

**Modeling token relationships:** In Table 5, we compare different methods of capturing token relationships. As a baseline, we do not compute the interactions between tokens. This leads to the worst performance among all variations. This validates the necessities of capturing the relationship between different semantic concepts. Another alternative is to use graph-convolutions similar to [6, 26, 47], but its performance is worse than that of VTs. This is likely due to the fact that graph-convolutions bind each visual token to a fixed semantic concept. In comparison, using transformers allows each visual token to encode any semantic concepts as long as the concept appears in the image. This allows the models to fully utilize its capacity.

**Token efficiency ablation:** In Table 6, we test varying numbers of visual tokens, only to find negligible or no increase in accuracy. This agrees with our hypothesis that VTs are already capturing a wide variety of concepts with just a few handfuls of tokens—additional tokens are not needed, as the space of possible, high-level concepts is already covered.

**Projection ablation:** In Table 7, we study whether we need to project visual tokens back to feature maps. We hypothesize that projecting the visual tokens is an important step since in vision understanding, pixel-level semantics are very important, and visual tokens are representations in the semantic space that do not encode any spatial information. As validated by Table 7, projecting visual tokens back to the feature map leads to higher performance, even for image classification tasks.

	No. Tokens	Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	16	71.8	1579	11.6
	32	71.9	1711	11.6
	64	<b>72.1</b>	1979	11.6
R34	16	<b>75.1</b>	3299	21.8
	32	75.0	3514	21.8
	64	75.0	3952	21.8

Table 6: Using more visual tokens do not improve the accuracy of VT, which agrees with our hypothesis that images can be described by a compact set of visual tokens.

	Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	w/ projector	<b>72.0</b>	1569
	w/o projector	71.0	1498
R34	w/ projector	<b>74.8</b>	3280
	w/o projector	73.9	3159

Table 7: VTs that projects tokens back to feature maps perform better. This may be because feature maps still encode important spatial information.

## 5.2. Training with Advanced Recipe

In Table 2, we show that under the regular training recipe, the VT-ResNets experience serious overfitting. Despite their accuracy improvement on the validation set, its training accuracy improves by a significantly larger margin. We hypothesize that this is because VT-based models have much higher model capacity. To fully unleash the potential of VT, we used a more advanced training recipe to train VT models. To prevent overfitting, we train with more training epochs, stronger data augmentation, stronger regularization, and distillation. Specifically, we train VT-ResNet models for 400 epochs with RMSProp. We use an initial learning rate of 0.01 and increase to 0.16 in 5 warmup epochs, then reduce the learning rate by a factor of 0.9875 per epoch. We use synchronized batch normalization and distributed training with a batch size of 2048. We use label smoothing and AutoAugment [8] and we set the stochastic depth survival probability [22] and dropout ratio to be 0.9 and 0.2, respectively. We use exponential moving average (EMA) on the model weights with 0.99985 decay. We use knowledge distillation [15] in the training recipe, where the teacher model is FBNetV3-G [9]. The total loss is a weighted sum of distillation loss ( $\times 0.8$ ) and cross entropy loss ( $\times 0.2$ ).

Our results are reported in Table 8. Compared with the baseline ResNet models, VT-ResNet models achieve 4.6 to 7 points higher accuracy and surpass all other related work that adopt attention of different forms based on ResNets [21, 41, 1, 5, 19, 30, 49, 6]. This validates that our advanced training recipe better utilizes the model capacity of VT-ResNet models to outperform all previous baselines.

Note that in addition to the architecture differences, previous works also used their own training recipes and it is

Models	Top-1 Acc (%)	FLOPs (G)	Params (M)
R18[14]	69.8	1.814	11.7
R18+SE[21, 41]	70.6	1.814	11.8
R18+CBAM[41]	70.7	1.815	11.8
LR-R18[19]	74.6	2.5	14.4
R18[14](ours)	73.8	1.814	11.7
VT-R18(ours)	<b>76.8</b>	<b>1.569</b>	<b>11.7</b>
R34[14]	73.3	3.664	21.8
R34+SE[21, 41]	73.9	3.664	22.0
R34+CBAM[41]	74.0	3.664	22.9
AA-R34[1]	74.7	3.55	20.7
R34[14](ours)	77.7	3.664	21.8
VT-R34(ours)	<b>79.9</b>	<b>3.236</b>	<b>19.2</b>
R50[14]	76.0	4.089	25.5
R50+SE[21, 41]	76.9	3.860*	28.1
R50+CBAM[41]	77.3	3.864*	28.1
LR-R50[19]	77.3	4.3	23.3
Stand-Alone[30]	77.6	3.6	<b>18.0</b>
AA-R50[1]	77.7	4.1	25.6
A <sup>2</sup> -R50[5]	77.0	-	-
SAN19[49]	78.2	<b>3.3</b>	20.5
GloRe-R50[6]	78.4	5.2	30.5
VT-R50(ours)	<b>80.6</b>	3.412	21.4
R101[14]	77.4	7.802	44.4
R101+SE [21, 41]	77.7	7.575*	49.3
R101+CBAM[41]	78.5	7.581*	49.3
LR-R101[19]	78.5	7.79	42.0
AA-R101[1]	78.7	8.05	45.4
GloRe-R200[6]	79.9	16.9	70.6
VT-R101(ours)	<b>82.3</b>	<b>7.129</b>	<b>41.5</b>

Table 8: Comparing VT-ResNets with other attention-augmented ResNets on ImageNet. \*The baseline ResNet FLOPs reported in [41] is lower than our baseline.

infeasible for us to try these recipes one by one. So to understand the source of the accuracy gain, we use the same training recipe to train baseline ResNet18 and ResNet34 and also observe significant accuracy improvement on baseline ResNets. But note that under the advanced training recipe, the accuracy gap between VT-ResNet and baselines increases from 1.7 and 2.2 points to 2.2 and 3.0 points. This further validates that a stronger training recipe can better utilize the model capacity of VTs. While achieving higher accuracy than previous work, VT-ResNets also use much fewer FLOPs and parameters, even we only replace the last stage of the baseline model. If we consider the reduction over the original stage, we observe FLOP reductions of up to 6.9x, as shown in Table 1.

## 5.3. Visual Transformer for Semantic Segmentation

We conduct experiments to test the effectiveness of VT for semantic segmentation on the COCO-stuff dataset [2] and the LIP dataset [25]. The COCO-stuff dataset contains annotations for 91 stuff classes with 118K training images and 5K validation images. LIP dataset is a collection of im-

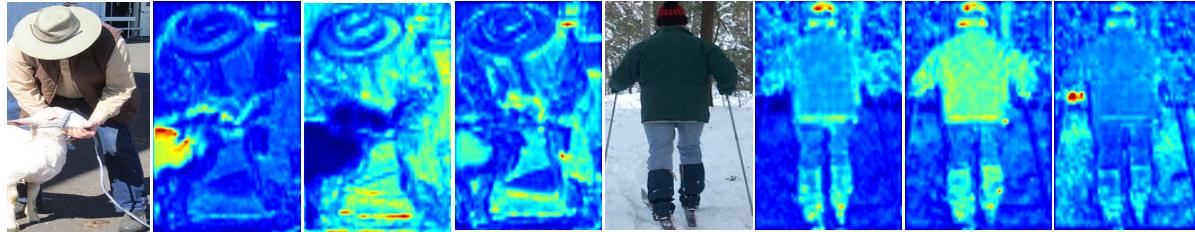


Figure 5: Visualization of the spatial attention generated by a filter-based tokenizer on images from the LIP dataset. Red denotes higher attention values and color blue denotes lower. Without any supervision, visual tokens automatically focus on different areas of the image that correspond to different semantic concepts, such as sheep, ground, clothes, woods.

		mIoU (%)	Total FLOPs (G)	FPN FLOPs (G)
R-50	FPN	40.78	159	55.1
	VT-FPN	41.00	113 (1.41x)	8.5 (6.48x)
R-101	FPN	41.51	231	55.1
	VT-FPN	41.50	185 (1.25x)	8.5 (6.48x)

Table 9: Semantic segmentation results on the COCO-stuff validation set. The FLOPs are calculated with a typical input resolution of  $800 \times 1216$ .

		mIoU (%)	Total FLOPs (G)	FPN FLOPs (G)
R50	FPN	47.04	37.1	12.8
	VT-FPN	47.39	26.4 (1.41x)	2.0 (6.40x)
R101	FPN	47.35	54.4	12.8
	VT-FPN	47.58	43.6 (1.25x)	2.0 (6.40x)

Table 10: Semantic segmentation results on the Look Into Person validation set. The FLOPs are calculated with a typical input resolution of  $473 \times 473$ .

ages containing humans with challenging poses and views. For the COCO-stuff dataset, we train a VT-FPN model with ResNet-{50, 101} backbones. Our implementation is based on Detectron2 [45]. Our training recipe is based on the semantic segmentation FPN recipe with 1x training steps, except that we use synchronized batch normalization in the VT-FPN, change the batch size to 32, and use a base learning rate of 0.04. For the LIP dataset, we also use synchronized batch normalization with a batch size of 96. We optimize the model via SGD with weight decay of 0.0005 and learning rate of 0.01.

As we can see in Table 9 and 10, after replacing FPN with VT-FPN, both ResNet-50 and ResNet-101 based models achieve slightly higher mIoU, but VT-FPN requires 6.5x fewer FLOPs than a FPN module.

#### 5.4. Visualizing Visual Tokens

We hypothesize that visual tokens extracted in the VT correspond to different high-level semantics in the image. To better understand this, we visualize the spatial attention

$\mathbf{A} \in \mathbb{R}^{HW \times L}$  generated by filter-based tokenizers, where each  $\mathbf{A}_{:,l} \in \mathbb{R}^{HW}$  is an attention map to show how each pixel of the image contributes to token- $l$ . We plot the attention map in Figure 5, and we can see that without any supervision, different visual tokens attend to different semantic concepts in the image, corresponding to parts of the background or foreground objects. More visualization results are provided in Appendix B.

## 6. Conclusion

The convention in computer vision is to represent images as pixel arrays and to apply the *de facto* deep learning operator – the convolution. In lieu of this, we propose *Visual Transformers* (VTs), as hallmarks of a new computer vision paradigm, learning and relating sparsely-distributed, high-level concepts far more efficiently: Instead of pixel arrays, VTs represent just the high-level concepts in an image using *visual tokens*. Instead of convolutions, VTs apply transformers to directly relate semantic concepts in token-space. To evaluate this idea, we replace convolutional modules with VTs, obtaining significant accuracy improvements across tasks and datasets. Using an advanced training recipe, our VT improves ResNet accuracy on ImageNet by 4.6 to 7 points. For semantic segmentation on LIP and COCO-stuff, VT-based feature pyramid networks (FPN) achieve 0.35 points higher mIoU despite 6.5x fewer FLOPs than convolutional FPN modules. This paradigm can furthermore be compounded with other contemporaneous tricks beyond the scope of this paper, including extra training data and neural architecture search. However, instead of presenting a mosh pit of deep learning tricks, our goal is to show that the pixel-convolution paradigm is fraught with redundancies. To compensate, modern methods add exceptional amounts of computational complexity. However, as model designers and practitioners, we can tackle the root cause instead of exacerbating compute demands, addressing redundancy in the pixel-convolution convention by adopting the newfound token-transformer paradigm moving forward.

## References

- [1] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V Le. Attention augmented convolutional networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3286–3295, 2019. 3, 7
- [2] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Coco-stuff: Thing and stuff classes in context. In *Computer vision and pattern recognition (CVPR), 2018 IEEE conference on*. IEEE, 2018. 2, 7
- [3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. *arXiv preprint arXiv:2005.12872*, 2020. 2
- [4] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L Yuille. Attention to scale: Scale-aware semantic image segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3640–3649, 2016. 2
- [5] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng.  $\Lambda^2$ -nets: Double attention networks. In *Advances in Neural Information Processing Systems*, pages 352–361, 2018. 7
- [6] Yunpeng Chen, Marcus Rohrbach, Zhicheng Yan, Yan Shuicheng, Jiashi Feng, and Yannis Kalantidis. Graph-based global reasoning networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 433–442, 2019. 2, 3, 4, 6, 7
- [7] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019. 3
- [8] Ekin D Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V Le. Autoaugment: Learning augmentation strategies from data. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 113–123, 2019. 7
- [9] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetsv3: Joint architecture-recipe search using neural acquisition function. *arXiv preprint arXiv:2006.02049*, 2020. 3, 7
- [10] Xiaoliang Dai, Peizhao Zhang, Bichen Wu, Hongxu Yin, Fei Sun, Yanghan Wang, Marat Dukhan, Yunqing Hu, Yiming Wu, Yangqing Jia, et al. Chamnet: Towards efficient network design through platform-aware model adaptation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 11398–11407, 2019. 3
- [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. 2, 5
- [12] 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. 2
- [13] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. Squeezennext: Hardware-aware neural network design. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1638–1647, 2018. 3
- [14] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2, 4, 7, 12
- [15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015. 7
- [16] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, et al. Searching for mobilenetv3. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1314–1324, 2019. 3
- [17] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 3
- [18] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3588–3597, 2018. 3
- [19] Han Hu, Zheng Zhang, Zhenda Xie, and Stephen Lin. Local relation networks for image recognition. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3464–3473, 2019. 3, 7
- [20] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Andrea Vedaldi. Gather-excite: Exploiting feature context in convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 9401–9411, 2018. 3
- [21] Jie Hu, Li Shen, and Gang Sun. Squeeze-and-excitation networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7132–7141, 2018. 3, 7
- [22] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *European conference on computer vision*, pages 646–661. Springer, 2016. 7
- [23] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezeenet: Alexnet-level accuracy with 50x fewer parameters and  $0.5$  mb model size. *arXiv preprint arXiv:1602.07360*, 2016. 3
- [24] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6399–6408, 2019. 5
- [25] Xiaodan Liang, Ke Gong, Xiaohui Shen, and Liang Lin. Look into person: Joint body parsing & pose estimation network and a new benchmark. *IEEE transactions on pattern analysis and machine intelligence*, 41(4):871–885, 2018. 2, 7

- [26] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolutions. In *Advances in Neural Information Processing Systems*, pages 1853–1863, 2018. [2](#), [3](#), [4](#), [6](#)
- [27] Ningning Ma, Xiangyu Zhang, Hai-Tao Zheng, and Jian Sun. Shufflenet v2: Practical guidelines for efficient cnn architecture design. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 116–131, 2018. [3](#)
- [28] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018. [3](#)
- [29] 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 H. Wallach, H. Larochelle, A. Beygelzimer, F. d’Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019. [5](#)
- [30] Prajit Ramachandran, Niki Parmar, Ashish Vaswani, Irwan Bello, Anselm Levskaya, and Jonathon Shlens. Stand-alone self-attention in vision models. *arXiv preprint arXiv:1906.05909*, 2019. [3](#), [7](#)
- [31] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016. [3](#)
- [32] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. [3](#)
- [33] Hang Su, Varun Jampani, Deqing Sun, Orazio Gallo, Erik Learned-Miller, and Jan Kautz. Pixel-adaptive convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. [3](#)
- [34] Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *International conference on machine learning*, pages 1139–1147, 2013. [5](#)
- [35] Mingxing Tan, Bo Chen, Ruoming Pang, Vijay Vasudevan, Mark Sandler, Andrew Howard, and Quoc V Le. Mnasnet: Platform-aware neural architecture search for mobile. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2820–2828, 2019. [3](#)
- [36] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019. [3](#)
- [37] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. [1](#), [3](#), [4](#)
- [38] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuan-dong Tian, Saining Xie, Bichen Wu, Matthew Yu, Tao Xu, Kan Chen, et al. Fbnetv2: Differentiable neural architecture search for spatial and channel dimensions. *arXiv preprint arXiv:2004.05565*, 2020. [3](#)
- [39] Weiyue Wang and Ulrich Neumann. Depth-aware cnn for rgb-d segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 135–150, 2018. [3](#)
- [40] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. [3](#)
- [41] Sanghyun Woo, Jongchan Park, Joon-Young Lee, and In So Kweon. Cbam: Convolutional block attention module. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 3–19, 2018. [3](#), [7](#)
- [42] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019. [3](#)
- [43] Bichen Wu, Alvin Wan, Xiangyu Yue, Peter Jin, Sicheng Zhao, Noah Golmant, Amir Gholaminejad, Joseph Gonzalez, and Kurt Keutzer. Shift: A zero flop, zero parameter alternative to spatial convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 9127–9135, 2018. [3](#)
- [44] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud. In *ICRA*, 2019. [3](#)
- [45] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019. [8](#)
- [46] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeeze-seg3: Spatially-adaptive convolution for efficient point-cloud segmentation. *arXiv preprint arXiv:2004.01803*, 2020. [3](#)
- [47] Songyang Zhang, Xuming He, and Shipeng Yan. Latent-gnn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*, pages 7374–7383, 2019. [2](#), [3](#), [4](#), [6](#)
- [48] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural network for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018. [2](#), [3](#)
- [49] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. *arXiv preprint arXiv:2004.13621*, 2020. [3](#), [7](#)

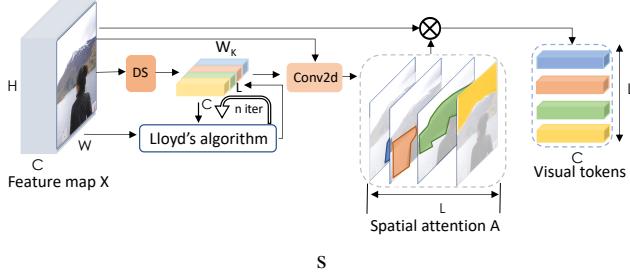


Figure 6: Cluster-based tokenizer that group pixels using the K-Means centroids of the pixels in the semantic space.

## 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 7.

## 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  $\mathbf{X}$ . 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:

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

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.

```

1 def kmeans(X_nchw, L, niter):
2     # Input:
3     #   X_nchw - feature map
4     #   L - num token
5     #   niter - num iters of Lloyd's
6     N, C, H, W = X_nchw.size()
7     # Initialization as down-sampled X
8     U_ncl = downsample(X).view(N, C, L)

```

```

9     X_ncp = X_nchw.view(N, C, H*W)    # p = h*w
10    # Normalize to unit vectors
11    U_ncl = U_ncl.normalize(dim=1)
12    X_ncp = X_ncp.normalize(dim=1)
13    for _ in range(niter):    # Lloyd's algorithm
14        dist_npl = (
15            X_ncp[..., None] - U_ncl[:, :, None, :]
16        ).norm(dim=1)
17        mask_npl = (dist_npl == dist_npl.min(dim=2))
18        U_ncl = X_ncp.MatMul(mask_npl)
19        U_ncl = U_ncl / mask_npl.sum(dim=1)
20        U_ncl = U_ncl.normalize(dim=1)
21    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.

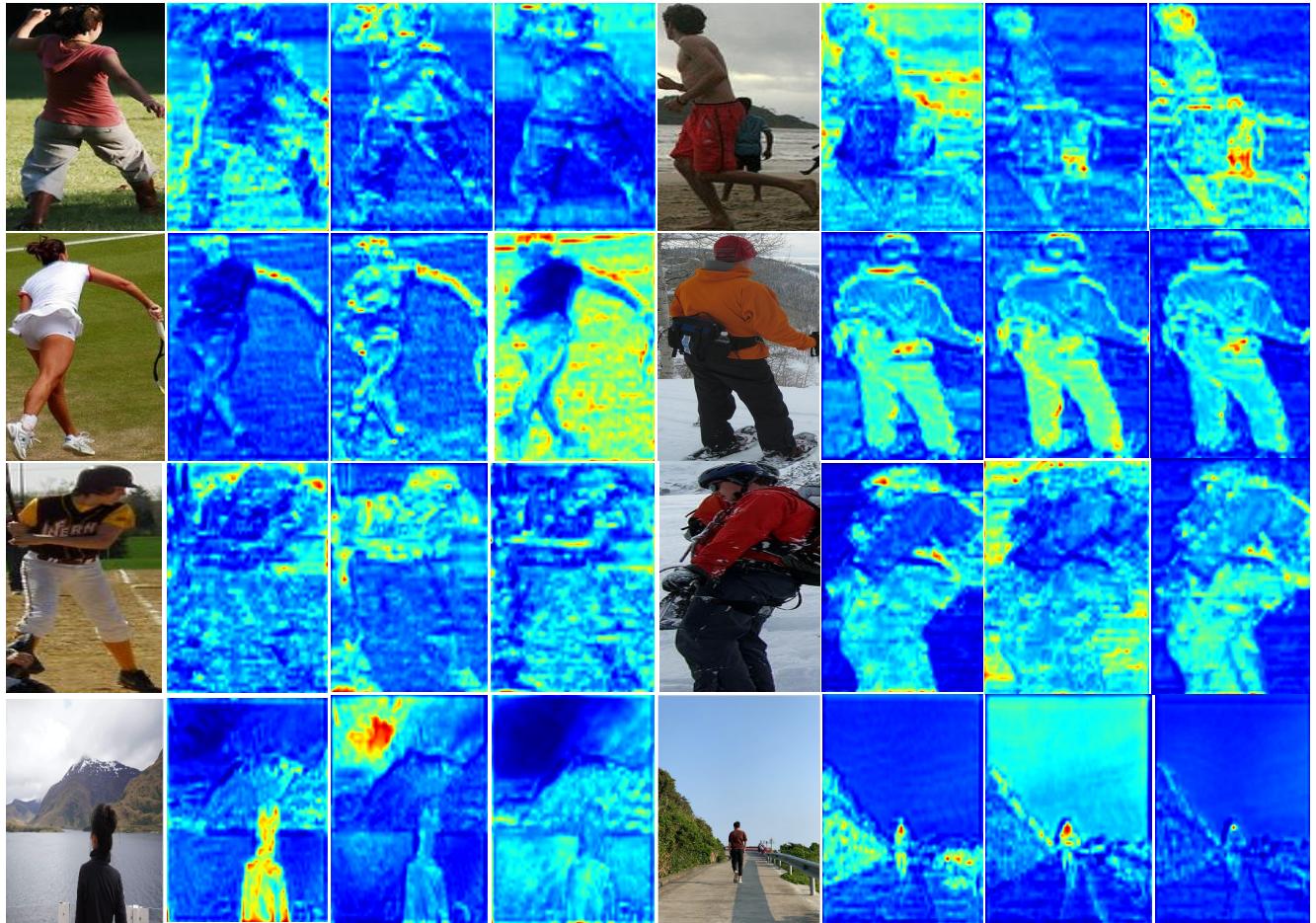


Figure 7: Visualization of the spatial attention generated by a filter-based tokenizer. Red denotes higher attention values and color blue denotes lower. Without any supervision, visual tokens automatically focus on different areas of the image that correspond to different semantic concepts.