This ICCV paper is the Open Access version, provided by the Computer Vision Foundation.
Except for this watermark, it is identical to the accepted version;
the final published version of the proceedings is available on IEEE Xplore.

Visual Transformers: Where Do Transformers Really Belong in Vision Models?

Bichen Wu[1], Chenfeng Xu[2], Xiaoliang Dai[1], Alvin Wan[2], Peizhao Zhang[1], Zhicheng Yan[1], Masayoshi Tomizuka[2], Joseph Gonzalez[2], Kurt Keutzer[2], Peter Vajda[1]

[1] Facebook Inc, [2] University of California, Berkeley

{wbc, xiaoliangdai, stzpz, zyan3, vajdap}@fb.com,
{xuchenfeng, alvinwan, tomizuka, jegonzal, keutzer}@berkeley.edu

Abstract

A recent trend in computer vision is to replace convolutions with transformers. However, the performance gain of transformers is attained at a steep cost, requiring GPU years and hundreds of millions of samples for training. This excessive resource usage compensates for a misuse of transformers: Transformers densely model relationships between its inputs – ideal for late stages of a neural network, when concepts are sparse and spatially-distant, but extremely inefficient for early stages of a network, when patterns are redundant and localized. To address these issues, we leverage the respective strengths of both operations, building convolution-transformer hybrids. Critically, in sharp contrast to pixel-space transformers, our ***Visual Transformer*** *(VT) operates in a semantic token space, judiciously attending to different image parts based on context. Our VTs significantly outperforms baselines: On ImageNet, our VT-ResNets outperform convolution-only ResNet by* ***4.6 to 7 points*** *and transformer-only ViT-B by* ***2.6 points*** *with 2.5× fewer FLOPs, 2.1× fewer 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 by adding model complexity.

To overcome the above challenges, we address the root cause and introduce the *Visual Transformer* (VT) (Figure 1) 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 a late-stage feature map. 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 self-attention module or *transformer* [39] 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 late-stage convolutions, our VT addresses the three challenges: 1) judiciously allocating computation by attending to important regions, instead of treating all pixels equally; 2) encoding semantic concepts in a few visual to-

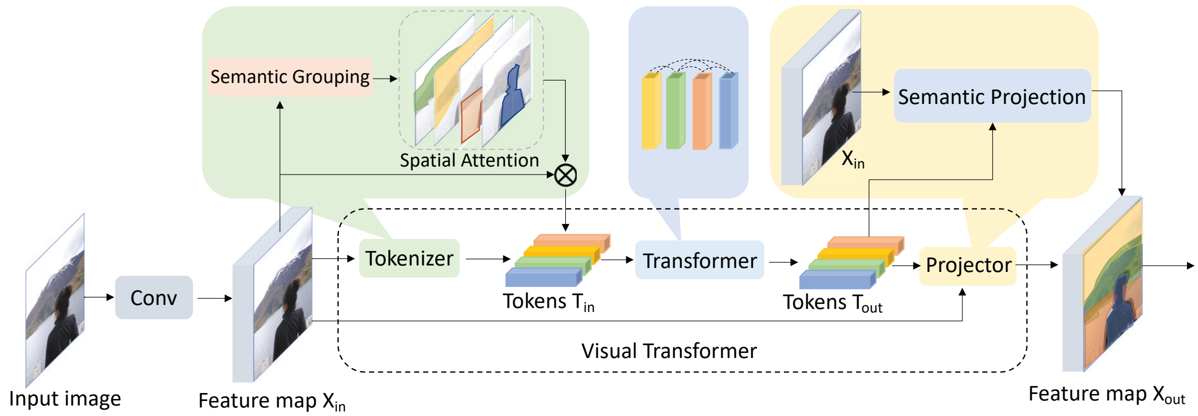

Figure 1: Diagram of a Visual Transformer (VT). For a given image, we first apply convolutional layers to extract low-level features. The output feature map is then fed to VT: First, apply a *tokenizer*, grouping pixels into a small number of *visual tokens*, each representing a semantic concept in the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

kens 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 16×16 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. Another relevant work, DETR[3], adopts transformers to simplify the hand-crafted anchor matching procedure in object detection training. This is an orthogonal use case that is not comparable to, but can be combined with, VT.

Graph convolutions in vision models: Our work is also related to previous efforts such as GloRe [6], Latent-GNN [51], and [26] that densely relate concepts in latent space using graph convolutions. To augment convolutions, [26, 6, 51] adopt a procedure similar to ours: (1) extracting latent variables as 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 [52]) to encode potential semantic concepts, our VT uses just 16 visual tokens and attains higher accuracy. Furthermore, while [26, 6, 51] augment convolutions in a pre-trained network, VTs replace convolutional layers to save FLOPs and parameters, and support training from scratch.

Attention in vision models: Attention is also widely used in different computer vision models [21, 20, 43, 46, 48, 42, 28, 18, 19, 1, 53, 30, 49]. Attention was first computed from the input and multiplied with the feature map [43, 21, 20, 46]. Later work [48, 34, 41] interprets this as a way to make convolutions spatially adaptive and content-aware. In [42], Wang *et al.* introduced non-local operators, equivalent to self-attention, to video understanding to capture long-range interactions. However, self-attention is expensive, so [1] use self-attention in convolutions with small channel sizes and [30, 28, 7, 53, 19] restrict the receptive field of self-attention. Starting from [30], self-attention is 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 works achieve better performance with lower computational cost. Early work in this direction includes [23, 31, 13, 17, 32, 16, 52, 27, 45]. Recent works use neural architecture search [44, 10, 40, 9, 37, 36] to automatically arrange existing convolution operators, which we show can be inefficient when used exclusively.

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. 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 [39] to these visual tokens. Third, project these visual tokens back to pixel-space to obtain an augmented feature map. With only 16 visual tokens, our VT outperforms previous methods [6, 51, 26] which use hundreds of semantic concepts ("nodes").

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 [51, 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,

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

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.

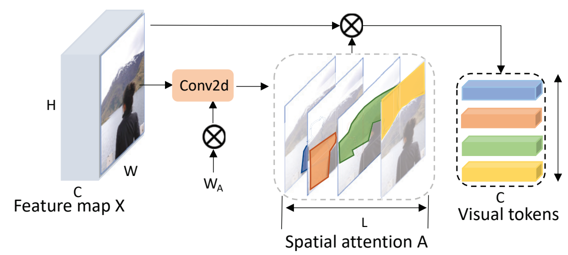

Figure 2: Filter-based tokenizer that use convolution to group pixels using a fixed convolution filter.

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

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

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

3.2. Transformer

After tokenization, we then need to model interactions between these visual tokens. Previous works [6, 51, 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

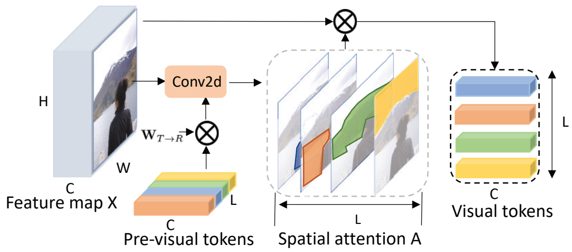

Figure 3: Recurrent tokenizer that uses previous tokens to guide the token extraction in the current VT module.

high-level concepts, even those that only appear in few images. To address this, we adopt transformers [39], 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:

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

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

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, 51, 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

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

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 VT 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: Following convention in image classification, we use ResNet backbones [14] to build visual-transformer-ResNets (VT-ResNets) by replacing the last stage of convolutions with VTs. First, we replace ResNet-{18, 34, 50, 101}'s 2 basic blocks, 3 basic blocks, 3 bottleneck blocks, and 3 bottleneck blocks, respectively, with the same number of VT modules. Second, since ResNet-$\{18, 34, 50, 101\}$ outputs $14^2 \times 256$, $14^2 \times 256$, $14^2 \times 1024$, $14^2 \times 1024$ feature maps after stage-4 (before stage-5 max pooling), we set VT's channel size to 256, 256, 1024, 1024. We use 16 visual tokens for all modules. The tokens are directly fed to the classification head – a standard average pool and fully-connected layer. Each model is exhaustively described in Appendix A. We reduce the last stage's FLOPs by up to 6.9x (Table 1).

		R18	R34	R50	R101
FLOPs	Total	1.14x	1.16x	1.20x	1.09x
	Stage-5	2.4x	5.0x	6.1x	6.9x
Params	Total	0.91x	1.21x	1.19x	1.19x
	Stage-5	0.9x	1.5x	1.26x	1.26x

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

Semantic segmentation: With convolutions, a) computational complexity grows with resolution and b) long-range spatial interactions are difficult to capture. However, VTs a) operate on a minimal set of visual tokens regardless of resolution and b) can capture long-range spatial interactions easily in latent space. We integrate VTs with the commonly-used panoptic feature pyramid networks (FPN) [24]. Panoptic FPNs extract ResNet feature maps at multiple stages and resolutions, which are then fused to produce multi-scale, detail-preserving feature maps (Figure 4 left). However, they rely on convolutions with large channel sizes operating on high resolution feature maps. We replace FPN convolutions with VT modules, producing VT-FPN (Figure 4 right). From each resolution, we extract 8 visual tokens with 1024 channels each, then relate tokens with a transformer before re-projecting back to the feature maps for pixel-level predictions. VT-FPN uses 6.4x fewer FLOPs than FPN with the same or better accuracy (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. VT for Classification and Ablations

We experiment on ImageNet [11], which features 1.3 million training images and 50 thousand validating images.

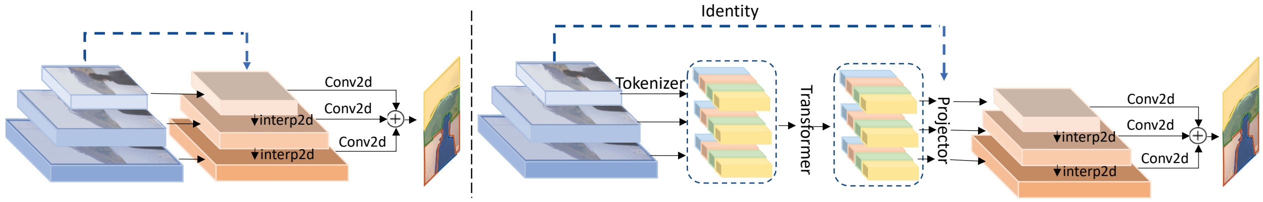

Figure 4: Feature Pyramid Networks (FPN) (left) vs visual-transformer-FPN (VT-FPN) (right) for semantic segmentation. FPN uses convolution and interpolation to merge feature maps with different resolutions. VT-FPN extraxt visual tokens from all feature maps, merge them with one transformer, and project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

	Top-1 Acc (%) (Val)	Top-1 Acc (%) (Train)	FLOPs (M)	Params (M)
R18	69.9	68.6	1814	11.7
VT-R18	**72.0**	76.5	1579	11.6
R34	73.3	73.9	3664	21.8
VT-R34	**74.8**	80.8	3299	21.8

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.

We implement VT models in PyTorch [29]. We use SGD with momentum [35]–an initial learning rate 0.1 decayed by 10x every 30 epochs, momentum 0.9, weight decay 4e-5, batch size 256, and 90 epochs. We use 8 V100 GPUs.

VT vs. ResNet with default training recipe: In Table 2, we compare VT-ResNets and vanilla ResNets under the same training recipe. VT-ResNets replace the last stage with a string of VT modules, using a filter-based tokenizer for the first module and recurrent tokenizers for subsequent modules. VT-ResNets outperform baselines by up to 2.1 points despite using fewer FLOPs–244M fewer (VT-R18) and 384M fewer (VT-R34). Furthermore, VT-ResNets overfit more heavily, with 7.9 (VT-R18) and 6.9 (VT-R34) points higher training accuracy. We hypothesize this is because VT-ResNets have much larger capacity and we need stronger regularization (e.g., data augmentation). We address this in Section 5.2 and Table 8.

Tokenizer ablation: We replace the first VT module's tokenizer with simpler baselines: First, we consider a naive *pooling-based* tokenizer, which simply bilinearly interpolates a feature map spatially, to reduce from $HW = 196$ to $L = 16$. Second, we consider a *clustering-based tokenizer* (Appendix C), which clusters pixels using k-means to form tokens. Per Table 3, the naive pooling-based tokenizer underperforms by a significant margin, validating the efficacy of "smarter" pixel grouping. However, filter-based

		Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	Pooling-based	70.6	1550	11.0
	Clustering-based	71.5	1580	11.6
	Filter-based	**72.0**	1579	11.6
R34	Pooling-based	73.5	3246	20.6
	Clustering-based	**75.1**	3230	21.8
	Filter-based	74.8	3299	21.8

Table 3: VT-ResNets using with 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.

		Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	w/ RT	**72.0**	1579	11.6
	w/o RT	71.4	1564	11.1
R34	w/ RT	**74.8**	3299	21.8
	w/o RT	74.5	3369	20.7

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

and clustering-based tokenizers perform similarly, with opposite rankings between VT-R18 and VT-R34. We hypothesize this is due to complementary drawbacks: Filter-based tokenizers are limited by fixed convolutional filters with non-content-aware weights, and clustering-based tokenizers extracts concepts that may not be critical for downstream classification performance. In Table 4, we validate the recurrent tokenizer's effectiveness.

Modeling token relationships: In Table 5, we compare different methods of capturing token relationships. Both (a) the baseline without computing token interactions and (b) graph convolutions graph convolutions [6, 26, 51] underperform VTs, validating the need for both token interaction and for content-aware token extraction.

Token efficiency ablation: In Table 6, we test varying

		Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	None	68.8	1528	8.5
	GraphConv	69.2	1528	8.5
	Transformer	**72.0**	1579	11.6
R34	None	73.4	3222	17.1
	GraphConv	73.6	3223	17.1
	Transformer	**74.8**	3299	21.8

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.

	No. Tokens	Top-1 Acc (%)	FLOPs (M)	Params (M)
R18	16	**72.0**	1579	11.6
	32	71.7	1711	11.6
	64	72.0	1979	11.6
R34	16	74.8	3299	21.8
	32	**75.1**	3514	21.8
	64	75.0	3951	21.8

Table 6: Using more visual tokens do not improve the accuracy of VT by significant margins, 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	**72.0**	1579	11.7
	w/o projector	70.9	1497	9.3
R34	w/ projector	**74.8**	3299	21.8
	w/o projector	74.1	3158	17.3

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

numbers of visual tokens, only to find negligible or no increase in accuracy. This suggests VTs are already cover the space of possible, high-level concepts.

Projection ablation: In Table 7, we show re-projecting visual tokens to the feature map is critical for performance. This is because pixel-level semantics are very important in vision understanding, which visual tokens lack entirely.

5.2. Training VT with Advanced Recipe

In Table 2, we show that under the regular training recipe, the VT-ResNets experience serious overfitting, with higher validation accuracy but even larger train-val accuracy gap than the baseline. We thus hypothesize VT-based models have much higher model capacity. To maximize this, we retrain with advanced training recipes, using more training epochs, stronger data augmentation, stronger regularization, and distillation. Specifically, we use 400 epochs, RMSProp, initial learning rate 0.01, 5 warmup epochs increasing learning rate to 0.16, then a learning rate reduction of 0.9875 per epoch, synchronized batch normalization, distributed training with batch size 2048, label smoothing, AutoAugment [8], stochastic depth survival probability [22] 0.9, dropout ratio 0.2, exponential moving average (EMA)with 0.99985 decay, and knowledge distillation [15] with FBNetV3-G [9] as teacher. The final loss weights the distillation term by 0.8 and cross entropy term by 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. Our VT-ResNets furthermore outperform other ResNet-based attention variants [21, 43, 1, 5, 19, 30, 53, 6]. This validates that our advanced training recipe better utilizes the VT-ResNet's model capacity. We also compare with concurrent work that adopt transformers in vision models [12, 38, 50, 33] though our work is earlier than these papers. Our models outperform competitors despite using far fewer FLOPs and parameters.

Each of the included baselines utilizes their own training recipes, in addition to their architectural changes; to understand the source of our accuracy gain, we train ResNet18 and ResNet34 with the same advanced training recipe. Despite this, the accuracy gap between VT-ResNet and ResNets increases from 1.7 and 2.2 points to 2.2 and 3.0 points, respectively, despite using fewer FLOPs and parameters. This further validates that a stronger training recipe can better utilize VT model capacity. For this last stage, we observe FLOP reductions of up to 6.9x (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 [2] and LIP [25] datasets. The COCO-stuff dataset contains annotations for 91 stuff classes with 118K training images and 5K validation images. LIP dataset is human image dataset 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 [47]. 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 train the model with SGD using 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.

Models	Top-1 Acc (%)	FLOPs (G)	Params (M)
R18[14]	69.8	1.814	11.7
R18+SE[21, 43]	70.6	1.814	11.8
R18+CBAM[43]	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)	**76.8**	**1.569**	**11.7**
R34[14]	73.3	3.664	21.8
R34+SE[21, 43]	73.9	3.664	22.0
R34+CBAM[43]	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)	**79.9**	**3.236**	**19.2**
R50[14]	76.0	4.089	25.5
R50+SE[21, 43]	76.9	3.860*	28.1
R50+CBAM[43]	77.3	3.864*	28.1
LR-R50[19]	77.3	4.3	23.3
Stand-Alone[30]	77.6	3.6	**18.0**
AA-R50[1]	77.7	4.1	25.6
A^2-R50[5]	77.0	-	-
SAN19[53]	78.2	**3.3**	20.5
GloRe-R50[6]	78.4	5.2	30.5
VT-R50(ours)	**80.6**	3.412	21.4
R101[14]	77.4	7.802	44.4
R101+SE [21, 43]	77.7	7.575*	49.3
R101+CBAM[43]	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)	**82.3**	**7.129**	**41.5**
ViT-B/16-224 [12] †	79.7	17.6‡	86.4
DeiT-B/16-224 [38]	81.8	17.6‡	86
T2T-ViT$_t$-224 [50]	82.2	13.2	64.1
BoTNet-S1-59 [33]	81.7	7.3	33.5

Table 8: Comparing VT-ResNets with other attention-augmented ResNets on ImageNet. *The baseline ResNet FLOPs reported in [43] is lower than our baseline. † We are citing the accuracy of training from scratch at a resolution of 224 from [50]. ‡ FLOP estimation is cited from [50]. Figure 9 in the Appendix shows a plot of accuracy vs. parameters and FLOPs for models above.

		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. FLOPs are calculated with 800×1216 input.

6. Analyses

Why not use transformers at early stages? One prominent concurrent work, ViT [12] replaces convolutions at all stages of the network with transformers. However, we find using transformers early in a network is extremely inefficient, suffering from accuracy loss when compared with

		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 473×473.

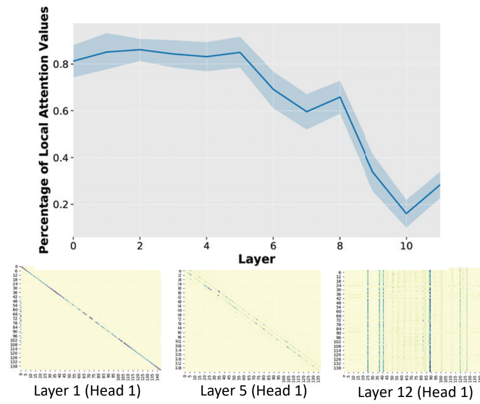

Figure 5: The upper image presents the percentage of local attention over the all attention values at different layers. The solid line is the mean of all the heads and the boundaries denote the standard deviations. The bottom row are self-attention patterns in a pretrained ViT-B/16 model.

baselines with similar resource constraints (Table 8). To study why, we analyze a pre-trained ViT-B/16 and find its self-attention patterns in early layers are highly-localized (Fig. 5, Row 2), with each token focusing on only neighboring patches (as shown by the diagonal lines). Self-attention spreads to non-local regions only in later layers. To quantify this observation, we compute the how "localized" each attention map is, as the ratio between (a) attention given to local pixels and (b) attention given to all pixels. Formally, given layer ℓ, its attention map $\mathbf{A}_\ell$ and pixel (i, j), we sum attention weights in a 3×3 patch centered on (i, j): $P_{\ell,ij} = \sum_{xy} \mathbf{A}_{\ell,xy} : x \in \{i-1, , i, i+1\}, y \in \{j-1, j, j+1\}$. We also compute the sum of all attention values $T_{\ell,ij} = \sum_{xy} \mathbf{A}_{\ell,xy}$. We average over all pixels and images, $R_\ell = \frac{1}{n_\ell n_i n_j} \sum_{\ell,ij} P_{\ell,ij}/T_{\ell,ij}$ and plot this ratio R_ℓ (Fig. 5). This confirms that ViT-B/16 transformers early in a network are highly-localized, only using global attention at later layers. This is reminiscent of convolutions: transformers early in the network mimic a convolution's highly-localized and sparse attention, at a far greater computational cost. This thus motivates our design pattern: use convolutions when localized attention is needed (early in a network), and use transformers when global attention is

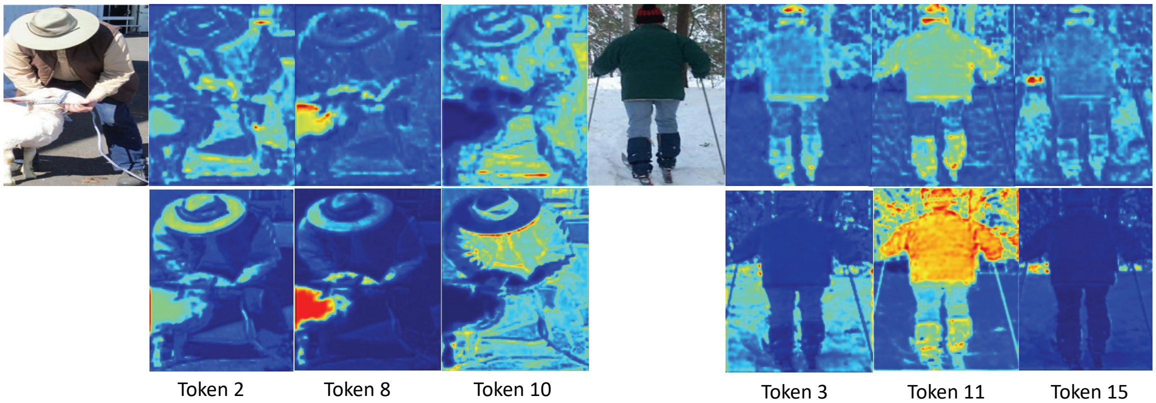

Figure 6: Visualization of the spatial attention generated by a filter-based tokenizer on images from the LIP dataset. Red denotes higher attention values and color blue denotes lower. Without any supervision, visual tokens automatically focus on different areas of the image that correspond to different semantic concepts, such as sheep, ground, clothes, woods. Row 1 shows pixels contributions to each token, and Row 2 shows how different pixels interact with the same token. Note in Row 2 that pixels may be from disparate, spatially-distant portions of the image, indicating VT can capture long-range interactions.

needed (later in a network).

What do the tokens learn? We show that extracted VT tokens correspond to different semantic image regions, by visualizing the spatial attention $\mathbf{A} \in \mathbb{R}^{HW \times L}$ for filter-based tokenizers in Figure 6, Row 1. Attention maps $\mathbf{A}_{:,l} \in \mathbb{R}^{HW}$ reflect each pixel's contribution to token-l, showing how each token represents different semantic parts of the scene. We also find VT trained on LIP assigns 28.3% higher attention to foreground pixels (annotated parts) than to background pixels. See more visual cases in Appendix B.

Does VT treat each pixel equally? We find, as hypothesized, VT assigns computation non-uniformly spatially. This is verified by the non-uniform attention distribution across the image, as shown in Figure 6, Row 1. We also quantify this by computing the visualized attention map A's entropy $E = -\sum_{i,j} \mathbf{A}_{i,j} \log(\mathbf{A}_{i,j})$. For a convolution, $\forall i, j, \mathbf{A}_{i,j} = 1/(HW)$. We use 473×473 for LIP images, making the baseline entropy $E_{conv} = 12.318$. For VT, the attention is $\mathbf{A} \in \mathbb{R}^{HW \times L}$ (Section 3.1.1), making VT entropy $E_{vt} = 0.941$. This is 13x smaller than E_{conv}, verifying VT does *not* treat each pixel equally.

Does VT capture long-range interactions? We design VT hoping it can capture long-range interactions and overcome the limitation of convolutions. We verify this by analyzing which pixels are interacting with each token. Formally, for token-l with attention map $\mathbf{A}_{i,j,l}$, its interaction with other tokens are captured by the self-attention weight $\mathbf{W}_{l,l'}$ computed in Equation (3). We analyze which pixels interacted with token-l by computing $\hat{\mathbf{A}}_{i,j,l} = \sum_{l'} \mathbf{W}_{l,l'} \times \mathbf{A}_{i,j,l'}$. We visualize $\hat{\mathbf{A}}_{i,j,l}$ in Fig. 6 Row 2. Same as $\mathbf{A}$, $\hat{\mathbf{A}}$ attends globally to the entire image. The focus regions of $\hat{\mathbf{A}}$ can be disparate, spatially distant portions of the image than $\mathbf{A}$, indicating VTs capture long-range interactions.

7. Conclusion

A recent trend in computer vision replaces convolutions with transformers. However, this ignores the motivation for a convolution: convolutions are efficient for processing highly-redundant, highly-localized patterns like edges and corners, which occur early in a network. In lieu of this, we design convolution-transformer hybrids that leverage the strengths of both operations. We propose *Visual Transformers* (VTs), 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, which can be mitigated by tackling the root cause – addressing redundancy in the pixel-convolution convention by adopting the token-transformer paradigm, instead of exacerbating compute demands.

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.

[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.

[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.

[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.

[5] Yunpeng Chen, Yannis Kalantidis, Jianshu Li, Shuicheng Yan, and Jiashi Feng. Aˆ 2-nets: Double attention networks. In *Advances in Neural Information Processing Systems*, pages 352–361, 2018.

[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.

[7] Jean-Baptiste Cordonnier, Andreas Loukas, and Martin Jaggi. On the relationship between self-attention and convolutional layers. *arXiv preprint arXiv:1911.03584*, 2019.

[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.

[9] Xiaoliang Dai, Alvin Wan, Peizhao Zhang, Bichen Wu, Zijian He, Zhen Wei, Kan Chen, Yuandong Tian, Matthew Yu, Peter Vajda, et al. Fbnetv3: Joint architecture-recipe search using neural acquisition function. *arXiv preprint arXiv:2006.02049*, 2020.

[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.

[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.

[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.

[13] Amir Gholami, Kiseok Kwon, Bichen Wu, Zizheng Tai, Xiangyu Yue, Peter Jin, Sicheng Zhao, and Kurt Keutzer. Squeezenext: Hardware-aware neural network design. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1638–1647, 2018.

[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.

[15] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[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.

[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.

[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.

[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.

[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.

[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.

[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.

[23] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. Squeezenet: Alexnet-level accuracy with 50x fewer parameters and¡ 0.5 mb model size. *arXiv preprint arXiv:1602.07360*, 2016.

[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.

[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.

[26] Xiaodan Liang, Zhiting Hu, Hao Zhang, Liang Lin, and Eric P Xing. Symbolic graph reasoning meets convolu-

tions. In *Advances in Neural Information Processing Systems*, pages 1853–1863, 2018.

[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.

[28] Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

[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.

[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.

[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.

[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.

[33] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. *arXiv preprint arXiv:2101.11605*, 2021.

[34] 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.

[35] 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.

[36] 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.

[37] Mingxing Tan and Quoc V Le. Efficientnet: Rethinking model scaling for convolutional neural networks. *arXiv preprint arXiv:1905.11946*, 2019.

[38] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. *arXiv preprint arXiv:2012.12877*, 2020.

[39] 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.

[40] Alvin Wan, Xiaoliang Dai, Peizhao Zhang, Zijian He, Yuandong 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2, 2019.

[48] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation. *arXiv preprint arXiv:2004.01803*, 2020.

[49] Chenfeng Xu, Bohan Zhai, Bichen Wu, Tian Li, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. You only group once: Efficient point-cloud processing with token representation and relation inference module. *arXiv preprint arXiv:2103.09975*, 2021.

[50] Li Yuan, Yunpeng Chen, Tao Wang, Weihao Yu, Yujun Shi, Francis EH Tay, Jiashi Feng, and Shuicheng Yan. Tokens-to-token vit: Training vision transformers from scratch on imagenet. *arXiv preprint arXiv:2101.11986*, 2021.

[51] Songyang Zhang, Xuming He, and Shipeng Yan. Latentgnn: Learning efficient non-local relations for visual recognition. In *International Conference on Machine Learning*, pages 7374–7383, 2019.

[52] Xiangyu Zhang, Xinyu Zhou, Mengxiao Lin, and Jian Sun. Shufflenet: An extremely efficient convolutional neural net-

work for mobile devices. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6848–6856, 2018.

[53] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. *arXiv preprint arXiv:2004.13621*, 2020.