AUTO-SCALING VISION TRANSFORMERS WITHOUT TRAINING

Wuyang Chen\textsuperscript{1}, Wei Huang\textsuperscript{2}, Xianzhi Du\textsuperscript{3}, Xiaodan Song\textsuperscript{3}, Zhangyang Wang\textsuperscript{1}, Denny Zhou\textsuperscript{3}
\textsuperscript{1}University of Texas, Austin \textsuperscript{2}University of Technology Sydney \textsuperscript{3}Google
\{wuyang.chen,atlaswang\}@utexas.edu weihuang.uts@gmail.com
\{xianzhi,xiaodansong,dennyzhou\}@google.com

ABSTRACT

This work targets automated designing and scaling of Vision Transformers (ViTs). The motivation comes from two pain spots: 1) the lack of efficient and principled methods for designing and scaling ViTs; 2) the tremendous computational cost of training ViT that is much heavier than its convolution counterpart. To tackle these issues, we propose \textbf{As-ViT}, an auto-scaling framework for ViTs without training, which automatically discovers and scales up ViTs in an efficient and principled manner. Specifically, we first design a “seed” ViT topology by leveraging a training-free search process. This extremely fast search is fulfilled by a comprehensive study of ViT’s network complexity, yielding a strong Kendall-tau correlation with ground-truth accuracies. Second, starting from the “seed” topology, we automate the scaling rule for ViTs by growing widths/depths to different ViT layers. This results in a series of architectures with different numbers of parameters in a single run. Finally, based on the observation that ViTs can tolerate coarse tokenization in early training stages, we propose a progressive tokenization strategy to train ViTs faster and cheaper. As a unified framework, \textbf{As-ViT} achieves strong performance on classification (83.5\% top1 on ImageNet-1k) and detection (52.7\% mAP on COCO) without any manual crafting nor scaling of ViT architectures: the end-to-end model design and scaling process costs only 12 hours on one V100 GPU. Our code is available at \url{https://github.com/VITA-Group/AsViT}.

1 INTRODUCTION

Transformer (Vaswani et al., 2017), a family of architectures based on the self-attention mechanism, is notable for modeling long-range dependencies in the data. The success of transformers has evolved from natural language processing to computer vision. Recently, Vision Transformer (ViT) (Dosovitskiy et al., 2020), a transformer architecture consisting of self-attention encoder blocks, has been proposed to achieve competitive performance to convolution neural networks (CNNs) (Simonyan & Zisserman, 2014; He et al., 2016) on ImageNet (Deng et al., 2009).

However, it remains elusive on how to effectively design, scale-up, and train ViTs, with three important gaps awaiting. First, Dosovitskiy et al. (2020) directly hard-split the 2D image into a series of local patches, and learn the representation with a pre-defined number of attention heads and channel expansion ratios. These ad-hoc “tokenization” and embedding mainly inherit from language tasks (Vaswani et al., 2017) but are not customized for vision, which calls for more flexible and principled designs. Second, the learning behaviors of ViT, including (loss of) feature diversity (Zhou et al., 2021), receptive fields (Raghu et al., 2021) and augmentations (Touvron et al., 2020; Jiang et al., 2021), differ vastly from CNNs. Benefiting from self-attention, ViT can capture global information even with shallow layers, yet its performance is quickly plateaued as going deeper. Strong augmentations are also vital to avoid ViTs from overfitting. These observations indicate that ViT architectures may require uniquely customized scaling-up laws to learn a more meaningful representation hierarchy. Third, training ViTs is both data and computation-consuming. To achieve state-of-the-art performance, ViT requires up to 300 million images and thousands of TPU-days. Although recent works attempt to enhance ViT’s data and resource efficiency (Touvron et al., 2020; Hassani et al., 2021; Pan et al., 2021; Chen et al., 2021d), the heavy computation cost (e.g., quadratic with respect to the number of tokens) is still overwhelming, compared with training CNNs.
We point out that the above gaps are inherently connected by the core architecture problem: how to design and scale-up ViTs? Different from the convolutional layer that directly digests raw pixels, ViTs embed coarse-level local patches as input tokens. Shall we divide an image into non-overlapping tokens of smaller size, or larger but overlapped tokens? The former could embed more visual details in each token but ignores spatial coherency, while the latter sacrifices the local details but may benefit more spatial correlations among tokens. A further question is on ViT’s depth/width trade-off: shall we prefer a wider and shallower ViT, or a narrower but deeper one? A similar dilemma also persists for ViT training: reducing the number of tokens would effectively speed up the ViT training, but meanwhile might sacrifice the training performance if sticking to coarse tokens from end to end.

In this work, we aim to reform the discovery of novel ViT architectures. Our framework, called \textbf{As-ViT} (Auto-scaling ViT), allows for extremely fast, efficient, and principled ViT design and scaling. In short, As-ViT first finds a promising “seed” topology for ViT of small depths and widths, then progressively “grow” it into different sizes (number of parameters) to meet different needs. Specifically, our “seed” ViT topology is discovered from a search space relaxed from recent manual ViT designs. To compare different topologies, we automate this process by a training-free architecture search approach and the measurement of ViT’s complexity, which are extremely fast and efficient. This training-free search is supported by our comprehensive study of various network complexity metrics, where we find the expected length distortion has the best trade-off between time costs and Kendall-tau correlations. Our “seed” ViT topology is then progressively scaled up from a small network to a large one, generating a series of ViT variants in a single run. Each step, the increases of depth and width are automatically and efficiently balanced by comparing network complexities. Furthermore, to address the data-hungry and heavy computation costs of ViTs, we make our ViT tokens elastic, and propose a progressive re-tokenization method for efficient ViT training. We summarize our contributions as below:

1. We for the first time automate both the backbone design and scaling of ViTs. A “seed” ViT topology is first discovered (in only seven V100 GPU-hours), and then its depths and widths are grown with a principled scaling rule in a single run (five more V100 GPU-hours).

2. To estimate ViT’s performance at initialization without any training, we conduct the first comprehensive study of ViT’s network complexity measurements. We empirically find the expected length distortion has the best trade-off between the computation costs and its Kendall-tau correlations with ViT’s ground-truth accuracy.

3. During training, we propose a progressive re-tokenization scheme via the change of dilation and stride, which demonstrates to be a highly efficient ViT training strategy that saves up to 56.2% training FLOPs and 41.1% training time, while preserving a competitive accuracy.

4. Our As-ViT achieves strong performance on classification (83.5% top-1 on ImageNet-1k) and detection (52.7% mAP on COCO).

2 WHY WE NEED AUTOMATED DESIGN AND SCALING PRINCIPLE FOR ViT?

\textbf{Background and recent development of ViT}[^1]. To transform a 2D image into a sequence, ViT (Dosovitskiy et al., 2020) splits each image into $14 \times 14$ or $16 \times 16$ patches and embeds them into a fixed number of tokens; then following practice of the transformer for language modeling, ViT applies self-attention to learn reweighting masks as relationship modeling for tokens, and leverages FFN (Feed-Forward Network) layers to learn feature embeddings. To better facilitate the visual representation learning, recently works try to train deeper ViTs (Touvron et al., 2021; Zhou et al., 2021), incorporate convolutions (Wu et al., 2021; d’Ascoli et al., 2021; Yuan et al., 2021a), and design multi-scale feature extractions (Chen et al., 2021b; Zhang et al., 2021; Wang et al., 2021).

\textbf{Why manual design and scaling may be suboptimal?} As the ViT architecture is still in its infant stage, there is no principle in its design and scaling. Early designs incorporate large token sizes, constant sequence length, and hidden size (Dosovitskiy et al., 2020; Touvron et al., 2020), and recent trends include small patches, spatial reduction, and channel doubling (Zhou et al., 2021; Liu et al., 2021). They all achieve comparably good performance, leaving the optimal choices unclear. Moreover, different learning behaviors of transformers from CNNs make the scaling law of ViTs

[^1]: We generally use the term “ViT” to indicate deep networks of self-attention blocks for vision problems. We always include a clear citation when we specifically discuss the ViTs proposed by Dosovitskiy et al. (2020).
highly unclear. Recent works [Zhou et al., 2021] demonstrated that attention maps of ViTs gradually become similar in deeper layers, leading to identical feature maps and saturated performance. ViT also generates more uniform representations across layers, enabling early aggregation of global context (Raghu et al., 2021). This is contradictory to CNNs as deeper layers help the learning of visual global information (Chen et al., 2018). These observations all indicate that previously studied scaling laws (depth/width allocations) for CNNs (Tan & Le, 2019) may not be appropriate to ViTs.

What principle do we want? We aim to automatically design and scale-up ViTs, being principled and avoiding manual efforts and potential biases. We also want to answer two questions: 1) Does ViT have any preference in its topology (patch sizes, expansion ratios, number of attention heads, etc.)? 2) Does ViT necessarily follow the same scaling rule of CNNs?

3 Auto-design & Scaling of ViTs with Network Complexity

To accelerate in ViT designing and avoid tedious manual efforts, we target efficient, automated, and principled search and scaling of ViTs. Specifically, we have two problems to solve: 1) with zero training cost (Section 3.2), how to efficiently find the optimal ViT architecture topology (Section 3.3)? 2) how to scale-up depths and widths of the ViT topology to meet different needs of model sizes (Section 3.4)?

3.1 Expanded Topology Space for ViTs

Before designing and scaling, we first briefly introduce our expanded topology search space for our As-ViT (blue italics in Figure 1). We first embed the input image into patches of a \(\frac{1}{4}\)-scale resolution, and adopt a stage-wise spatial reduction and channel doubling strategy. This is for the convenience of dense prediction tasks like detection that require multi-scale features. Table 1 summarizes details of our topology space, and will be explained below.

Elastic kernels. Instead of generating non-overlapped image patches, we propose to search for the overlapped with their neighbors, introducing more spatial correlations among tokens. Moreover, each time we downsample the spatial resolution, we also introduce overlaps when re-embedding local tokens (implemented by either a linear or a convolutional layer).

Elastic attention splits. Splitting the attention into local windows is an important design to reduce the computation cost of self-attention without sacrificing much performance (Zaheer et al. [2020], Liu et al. [2021]). Instead of using a fixed number of splits, we propose to search for elastic attention splits for each stage. Note that we try to make our design general and do not use shifted windows (Liu et al. [2021]).

More search dimensions. ViT (Dosovitskiy et al., 2020) by default leveraged an FFN layer with \(4\times\) expanded hidden dimension for each attention block. To enable a more flexible design of ViT architectures, for each stage we further search over the FFN expansion ratio. We also search for the final number of heads for the self-attention module.

![Figure 1: Overall architecture of our As-ViT. Blue italics indicates topology configurations to be searched (Table 1). Red indicates depth/width to be scaled-up.](image)

![Figure 1: Overall architecture of our As-ViT. Blue italics indicates topology configurations to be searched (Table 1). Red indicates depth/width to be scaled-up.](image)

<table>
<thead>
<tr>
<th>Stage</th>
<th>Sub-space</th>
<th>Choices</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>Kernel (K_1)</td>
<td>4, 5, 6, 7, 8</td>
</tr>
<tr>
<td></td>
<td>Attention Splits (S_1)</td>
<td>2, 4, 8</td>
</tr>
<tr>
<td></td>
<td>FFN Expansion (E_1)</td>
<td>2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>#2</td>
<td>Kernel (K_2)</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td></td>
<td>Attention Splits (S_2)</td>
<td>1, 2, 4</td>
</tr>
<tr>
<td></td>
<td>FFN Expansion (E_2)</td>
<td>2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>#3</td>
<td>Kernel (K_3)</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td></td>
<td>Attention Splits (S_3)</td>
<td>1, 2</td>
</tr>
<tr>
<td></td>
<td>FFN Expansion (E_3)</td>
<td>2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>#4</td>
<td>Kernel (K_4)</td>
<td>2, 3, 4</td>
</tr>
<tr>
<td></td>
<td>FFN Expansion (E_4)</td>
<td>2, 3, 4, 5, 6</td>
</tr>
<tr>
<td>-</td>
<td>Num. Heads</td>
<td>16, 32, 64</td>
</tr>
</tbody>
</table>

\(^{2}\)Due to spatial reduction, the 4\(^{th}\) stage may already reach a resolution at \(7 \times 7\) on ImageNet, and we set its splitting as 1.
3.2 Assessing ViT Complexity at Initialization via Manifold Propagation

Training ViTs is slow: hence an architecture search guided by evaluating trained models’ accuracies will be dauntingly expensive. We note a recent surge of training-free neural architecture search methods for ReLU-based CNNs, leveraging local linear maps (Mellor et al. 2020), gradient sensitivity (Abdelfattah et al. 2021), number of linear regions (Chen et al. 2021), or network topology (Rhardwaj et al. 2021). However, ViTs are equipped with more complex non-linear functions: self-attention, softmax, and GeLU. Therefore, we need to measure their learning capacity in a more general way. In our work, we consider measuring the complexity of manifold propagation through ViT, to estimate how complex functions can be approximated by ViTs.

Intuitively, a complex network can propagate a simple input into a complex manifold at its output layer, thus likely to possess a strong learning capacity. In our work, we study the manifold complexity of mapping a simple circle input through the ViT: \( h(\theta) = \sqrt{N} (u^0 \cos(\theta) + u^1 \sin(\theta)) \). Here, \( N \) is the dimension of ViT’s input (e.g. \( N = 3 \times 224 \times 224 \) for ImageNet images), \( u^0 \) and \( u^1 \) form an orthonormal basis for a 2-dimensional subspace of \( \mathbb{R}^N \) in which the circle lives. We further define the ViT network as \( N \), its input-output Jacobian \( v(\theta) = \partial_\theta h(\theta) \) at the input \( \theta \), and \( a(\theta) = \partial_\theta v(\theta) \). We will calculate expected complexities over a certain number of \( \theta \)s uniformly sampled from \([0, 2\pi]\). In our work, we study three different types of manifold complexities:

1. **Curvature** can be defined as the reciprocal of the radius of the osculating circle on the ViT’s output manifold. Intuitively, a larger curvature indicates that \( N(\theta) \) changes fast at a certain \( \theta \). According to Riemannian geometry (Lee 2006, Poole et al. 2016), the curvature can be explicitly calculated as
   \[
   \kappa = \int (v(\theta) \cdot a(\theta))^{-3/2} \sqrt{(v(\theta) \cdot a(\theta))(a(\theta) \cdot a(\theta)) - (v(\theta) \cdot a(\theta))^2} \, d\theta.
   \]

2. **Length Distortion** in Euclidean space is defined as \( L^E = \frac{\text{length}(h(\theta))}{\text{length}(v(\theta))} = \int \sqrt{\|v(\theta)\|^2} \, d\theta \). It measures when the network takes a unit-length curve as input, what is the length of the output curve. Since the ground-truth function we want to estimate (using \( N \)) is usually very complex, one may also expect that networks with better performance should also generate longer outputs.

3. The problem of \( L^E \) is that, stretched outputs not necessarily translate to complex outputs. A simple example: even an appropriately initialized linear network could grow a straight line into a long output (i.e. a large norm of input-output Jacobian). Therefore, one could instead use **Length Distortion taking curvature into consideration** to measure how quickly the normalized Jacobian \( \hat{v}(\theta) = v(\theta)/\sqrt{v(\theta) \cdot v(\theta)} \) changes with respect to \( \theta \), defined as \( L^E_{\kappa} = \int \sqrt{\|\partial_\theta \hat{v}(\theta)\|^2} \, d\theta \).

In our study, we aim to compare the potential of using these three complexity metrics to guide the ViT architecture selection. As the core of neural architecture search is to rank the performance of different architectures, we measure the Kendall-tau correlations (\( \tau \)) between these metrics and models’ ground-truth accuracies. We randomly sampled 87 ViT topologies from Table I (with \( L_1 = L_2 = L_3 = L_4 = 1 \), \( C = 32 \)), fully train them on ImageNet-1k for 300 epochs (following the same training recipe of DeiT (Touvron et al. 2020)), and also measure their \( \kappa, L^E, L^E_{\kappa} \) at initialization. As shown in Figure 2, we can clearly see that both \( \kappa \) and \( L^E \) exhibit high Kendall-tau correlations. \( \kappa \) has a negative correlation, which may indicate that changes of output manifold on the tangent direction are more important to ViT training, instead of on the perpendicular direction. Meanwhile, \( \kappa \) costs too much computation time due to second derivatives. We decide to choose \( L^E \) as our complexity measure for highly fast ViT topology search and scaling.

![Figure 2: Correlations between \( \kappa, L^E, L^E_{\kappa} \) and trained accuracies of ViT topologies from our search space.](image)

<table>
<thead>
<tr>
<th>Complexity</th>
<th>( \tau )</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \kappa )</td>
<td>-0.49</td>
<td>38.3s</td>
</tr>
<tr>
<td>( L^E )</td>
<td>0.49</td>
<td>12.8s</td>
</tr>
<tr>
<td>( L^E_{\kappa} )</td>
<td>-0.01</td>
<td>48.2s</td>
</tr>
</tbody>
</table>

Table 2: Complexity Study. \( \tau \): Kendall-tau correlation. Time: per ViT topology on average on 1 V100 GPU.
3.3 $L^E$ AS REWARD FOR SEARCHING ViT TOPOLOGIES

We now propose our training-free search based on $L^E$ (Algorithm 1). Most NAS (neural architecture search) methods evaluate the accuracies or loss values of single-path or super networks as proxy inference. This training-based search will suffer from more computation costs when applied to ViTs. Instead of training ViTs, for each architecture we sample, we calculate $L^E$ and treat it as the reward to guide the search process. In addition to $L^E$, we also include the NTK condition number $\kappa_{\Theta}$ to indicate the trainableability of ViTs (Chen et al., 2021e, Xiao et al., 2019, Yang, 2020, Hron et al., 2020). $\lambda_{\text{max}}$ and $\lambda_{\text{min}}$ are the largest and smallest eigenvalue of NTK matrix $\Theta$.

Algorithm 1: Training-free ViT Topology Search.

1. **Input**: RL policy $\pi$, step $t = 0$, total steps $T$.
2. **while** $t < T$ **do**
   3. Sample topology $a_t$ from $\pi$.
   4. Calculate $L^E_t$ and $\kappa_{\Theta,t}$ for $a_t$.
   5. Normalization: $\hat{L}_t^E = \frac{L_t^E - L_{t-1}^E}{\kappa_{\Theta,t}}$, $\hat{\kappa}_{\Theta,t} = \frac{\kappa_{\Theta,t} - \kappa_{\Theta,t-1}}{\kappa_{\Theta,t-1}}$, $t' = 1, \ldots, t$.
   6. Update policy $\pi$ using reward $r_t = \hat{L}_t^E - \hat{\kappa}_{\Theta,t}$ by policy gradient (Williams, 1992).
   7. $t = t + 1$.
3. **return** Topology $a^*$ of highest probability from $\pi$.

Table 3: Statistics of topology search. *Standard deviation is normalized by mean due to different value ranges.

<table>
<thead>
<tr>
<th>Search Space</th>
<th>Mean</th>
<th>Std*</th>
</tr>
</thead>
<tbody>
<tr>
<td>$K_1$</td>
<td>7.3</td>
<td>0.1</td>
</tr>
<tr>
<td>$K_2$</td>
<td>4.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$K_3$</td>
<td>4.0</td>
<td>0.0</td>
</tr>
<tr>
<td>$E_1$</td>
<td>3.3</td>
<td>0.4</td>
</tr>
<tr>
<td>$E_2$</td>
<td>3.9</td>
<td>0.4</td>
</tr>
<tr>
<td>$E_3$</td>
<td>4.2</td>
<td>0.3</td>
</tr>
<tr>
<td>$E_4$</td>
<td>5.2</td>
<td>0.2</td>
</tr>
<tr>
<td>$S_1$</td>
<td>4.0</td>
<td>0.6</td>
</tr>
<tr>
<td>$S_2$</td>
<td>2.7</td>
<td>0.5</td>
</tr>
<tr>
<td>$S_3$</td>
<td>1.5</td>
<td>0.3</td>
</tr>
<tr>
<td>Head</td>
<td>42.7</td>
<td>0.5</td>
</tr>
</tbody>
</table>

We use reinforcement learning (RL) for search. The RL policy is formulated as a joint categorical distribution over the choices in Table 1 and is updated by policy gradient (Williams, 1992). We update our policy for 500 steps, which is observed enough for the policy to converge (entropy drops from 15.3 to 5.7). The search process is extremely fast: only seven GPU-hours (V100) on ImageNet-1k, thanks to the fast calculation of $L^E$ that bypasses the ViT training. To address the different magnitude of $L^E$ and $\kappa_{\Theta}$, we normalize them by their relative value ranges (line 5 in Algorithm 1). We summarize the ViT topology statistics from our search in Table 3. We can see that $L^E$ and $\kappa_{\Theta}$ highly prefer: (1) tokens with overlaps ($K_1 \sim K_4$ are all larger than strides), and (2) larger FFN expansion ratios in deeper layers ($E_1 < E_2 < E_3 < E_4$). No clear preference of $L^E$ and $\kappa_{\Theta}$ are found on attention splits and number of heads.

3.4 AUTOMATIC AND PRINCIPLED SCALING OF ViTs

After obtaining an optimal topology, another question is: how to balance the network depth and width? Currently, there is no such rule of thumb for ViT scaling. Recent works try to scale-up or grow convolutional networks of different sizes to meet various resource constraints (Liu et al., 2019a, Tan & Le, 2019). However, to automatically find a principled scaling rule, training ViTs will cost enormous computation costs. It is also possible to search different ViT variants (as in Section 3.3), but that requires multiple runs. Instead, “scaling-up” is a more natural way to generate multiple model variants in one experiment. We are therefore motivated to scale-up our searched basic “seed” ViT to a larger model in an efficient training-free and principled manner.

We depict our auto-scaling method in Algorithm 2. The starting-point architecture has one attention block for each stage, and an initial hidden dimension $C = 32$. In each iteration, we greedily find the optimal depth and width to scale-up next. For depth, we try to find out which stage to deepen (i.e., add one attention block to which stage); for width, we try to discover the best expansion ratio (i.e., widen the channel number to what extent). The rule to choose how to scale-up is by comparing the propagation complexity among a set of scaling choices. For example, in the case of four backbone stages (Table 1) and four expansion ratio choices ([0.05×, 0.1×, 0.15×, 0.2×]), we have $4 \times 4 = 16$ scaling choices in total for each step. We calculate $L^E$ and $\kappa_{\Theta}$ after applying each choice, and the one with the best $L^E / \kappa_{\Theta}$ trade-off (minimal sum of rankings by $L^E$ and $\kappa_{\Theta}$) will be selected to scale-up with. The scaling stops when a certain limit of parameter number is reached. In our work, we stop the scaling process once the number of parameters reaches 100 million, and the scaling only takes five GPU hours (V100) on ImageNet-1k.
During the auto-scaling, both the network’s complexity and trainability improve (numbers indicate scaling-up steps, \(L^E\) higher the better, \(\kappa_{\Theta}\) lower the better).

The scaling trajectory is visualized in Figure 3. By comparing our automated scaling against random scaling, we find our scaling principle prefers to sacrifice the depths to win more widths, keeping a shallower but wider network. Our scaling is more similar to the rule developed by Zhai et al. (2021). In contrast, ResNet and Swin Transformer (Liu et al. 2021) choose to be narrower and deeper.

**Algorithm 2:** Training-free Auto-Scaling ViTs.

```
1 Input: seed As-ViT topology \(a_0\), stop criterion (#parameters) \(P, t = 0\), channel expansion ratio choices \(C = \{1.05 \times, 1.1 \times, 1.15 \times, 1.2 \times\}\) (to increase the width by 5%, 10%, 15%, or 20%), depth choices \(D = \{+1, 0, 0, 0, 0, +1, 0, 0, 0\}\) (to add one more layer to one of the four stages in Table 1).
2 while \(P > \) number of parameters of \(a_t\) do
3   for each scaling choice \(g_i \in C \times D\) do
4     Scale-up: \(a_{t,i} = a_t \leftarrow g_i\). \(\triangleright\) Grow both the channel width and depth.
5     Calculate \(L^{E,i}_t\) and \(\kappa_{\Theta,i}\) for \(a_{t,i}\).
6     Get ranking of each scaling choice \(r_{L^E,i}\) by descendingly sort \(L^{E,i}_t, i = 1, \cdots, |C \times D|\).
7     Get ranking of each scaling choice \(r_{\kappa_{\Theta,i}}\) by ascendingly sort \(\kappa_{\Theta,i}, i = 1, \cdots, |C \times D|\).
8     Ascendingly sort each scaling choice \(g_i\) by \(r_{L^E,i} + r_{\kappa_{\Theta,i}}\).
9     Select the scaling choice \(g_{i}^*\) with the top (smallest) ranking.
10    \(a_{t+1} = a_t \leftarrow g_{i}^*\).
11    \(t = t + 1\).
12 return Grewed ViT architectures \(a_1, a_2, \cdots, a_t\).
```

## 4 EFFICIENT ViT TRAINING VIA PROGRESSIVE ELASTIC RE-TOKENIZATION

Recent works (Jia et al. 2018, Zhou et al. 2019, Fu et al. 2020) show that one can use mixed or progressive precision to achieve an efficient training purpose. The rationale behind this strategy is that, there exist some “short-cuts” on the network’s loss landscape that can be manually created to bypass perhaps less important gradient descent steps, especially during early training phases. As in ViT, both self-attention and FFN have quadratic computation costs to the number of tokens. It is therefore natural to ask: do we need full-resolution tokens during the whole training process?

We provide an affirming answer by proposing a progressive elastic re-tokenization training strategy. To update the number of tokens during training without affecting the shape of weights in linear projections, we adopt different sampling granularities in the first linear projection layer. Taking the...
first projection kernel $K_1 = 4$ with stride $= 4$ as an example: during training we gradually change the (stride, dilation) pair of the first projection kernel to (16, 5), (8, 2), and (4, 1), keeping the shape of weights and the architecture unchanged.

This re-tokenization strategy emulates curriculum learning for ViTs: when the training begins, we introduce coarse sampling to significantly reduce the number of tokens. In other words, our As-ViT quickly learns coarse information from images in early training stages at extremely low computation cost (only 13.2% FLOPs of full-resolution training). Towards the late phase of training, we progressively switch to fine-grained sampling, restore the full token resolution, and maintain the competitive accuracy. As shown in Figure 4 when the ViT is trained with coarse sampling in early training phases, it can still obtain high accuracy while requiring extremely low computation cost. The transition between different sampling granularities introduces a jump in performance, and eventually the network restores its competitive final performance.

5 Experiments

5.1 As-ViT: Auto-Scaling ViT

We show our searched As-ViT topology in Table 4. This architecture facilitates strong overlaps among tokens during both the first projection ("tokenization") step and three re-embedding steps. FFN expansion ratios are first narrow then become wider in deeper layers. A small number of attention splits are leveraged for better aggregation of global information.

The seed topology is automatically scaled-up, and three As-ViT variants of comparable sizes with previous works will be benchmarked. Our scaling rule prefers shallower and wider networks, and layers are more balanced among different resolution stages.

5.2 Image Classification

Settings. We benchmark our As-ViT on ImageNet-1k (Deng et al. 2009). We use TensorFlow and Keras for training implementations and conduct all training on TPUs. We set the default image size as 224 × 224, and use AdamW (Loshchilov & Hutter 2017) as the optimizer with cosine learning rate decay. (Loshchilov & Hutter 2016). A batch size of 1024, an initial learning rate of 0.001, and a weight decay of 0.05 are adopted.

Table 5 demonstrates comparisons of our As-ViT to other models. Compared to the previous both Transformer-based and CNN-based architectures, As-ViT achieves state-of-the-art performance with a comparable number of parameters and FLOPs.

<table>
<thead>
<tr>
<th>Method</th>
<th>Params.</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegNetY-8GF</td>
<td>Radosavovic et al, 2020</td>
<td>21.0 M</td>
<td>4.0 B</td>
</tr>
<tr>
<td>ViT-S</td>
<td>Dosovitskiy et al, 2020</td>
<td>22.1 M</td>
<td>9.2 B</td>
</tr>
<tr>
<td>DeiT-S</td>
<td>Touvron et al, 2021</td>
<td>22.0 M</td>
<td>4.6 B</td>
</tr>
<tr>
<td>T2T-ViT</td>
<td>Yuan et al, 2021</td>
<td>21.5 M</td>
<td>6.1 B</td>
</tr>
<tr>
<td>TNT-S</td>
<td>Han et al, 2021</td>
<td>23.8 M</td>
<td>5.2 B</td>
</tr>
<tr>
<td>PVT-Small</td>
<td>Wang et al, 2021</td>
<td>24.5 M</td>
<td>3.8 B</td>
</tr>
<tr>
<td>CaIT XS-24</td>
<td>Touvron et al, 2021</td>
<td>26.6 M</td>
<td>5.4 B</td>
</tr>
<tr>
<td>DeepViT-S</td>
<td>Zhou et al, 2021</td>
<td>27 M</td>
<td>6.2 B</td>
</tr>
<tr>
<td>ConViT-S</td>
<td>Asilak et al, 2021</td>
<td>27 M</td>
<td>5.4 B</td>
</tr>
<tr>
<td>CvT-13</td>
<td>Wu et al, 2021</td>
<td>20 M</td>
<td>4.5 B</td>
</tr>
<tr>
<td>CvT-21</td>
<td>Wu et al, 2021</td>
<td>32 M</td>
<td>7.1 B</td>
</tr>
<tr>
<td>Swin-T</td>
<td>Lin et al, 2021</td>
<td>29.0 M</td>
<td>4.5 B</td>
</tr>
<tr>
<td>BossNet-TU</td>
<td>Li et al, 2021</td>
<td>-</td>
<td>3.4 B</td>
</tr>
<tr>
<td>AutoFormer-s</td>
<td>Chen et al, 2021c</td>
<td>22.9 M</td>
<td>5.1 B</td>
</tr>
<tr>
<td>GLiT-Small</td>
<td>Chen et al, 2021a</td>
<td>24.6 M</td>
<td>4.4 B</td>
</tr>
<tr>
<td>As-ViT Small (ours)</td>
<td>-</td>
<td>29.0 M</td>
<td>5.3 B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Params.</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegNetY-16GF</td>
<td>Radosavovic et al, 2020</td>
<td>39.0 M</td>
<td>8.0 B</td>
</tr>
<tr>
<td>T2T-ViT-F9</td>
<td>Yuan et al, 2021</td>
<td>39.2 M</td>
<td>9.8 B</td>
</tr>
<tr>
<td>CaIT S-24</td>
<td>Touvron et al, 2021</td>
<td>46.9 M</td>
<td>9.4 B</td>
</tr>
<tr>
<td>ConViT-S+</td>
<td>Asilak et al, 2021</td>
<td>48 M</td>
<td>10 B</td>
</tr>
<tr>
<td>ViT-S/16</td>
<td>Dosovitskiy et al, 2020</td>
<td>48.6 M</td>
<td>20.2 B</td>
</tr>
<tr>
<td>Swin-S</td>
<td>Lin et al, 2021</td>
<td>50.0 M</td>
<td>8.7 B</td>
</tr>
<tr>
<td>DeepViT-L</td>
<td>Zhou et al, 2021</td>
<td>55 M</td>
<td>12.5 B</td>
</tr>
<tr>
<td>PVT-Medium</td>
<td>Wang et al, 2021</td>
<td>44.2 M</td>
<td>6.7 B</td>
</tr>
<tr>
<td>PVT-Large</td>
<td>Wang et al, 2021</td>
<td>61.4 M</td>
<td>9.8 B</td>
</tr>
<tr>
<td>T2T-ViT-24</td>
<td>Yuan et al, 2021</td>
<td>64.1 M</td>
<td>15.0 B</td>
</tr>
<tr>
<td>TNT-B</td>
<td>Han et al, 2021</td>
<td>65.6 M</td>
<td>14.1 B</td>
</tr>
<tr>
<td>BossNet-TL</td>
<td>Li et al, 2021</td>
<td>-</td>
<td>7.9 B</td>
</tr>
<tr>
<td>AutoFormer-b</td>
<td>Chen et al, 2021c</td>
<td>54 M</td>
<td>11 B</td>
</tr>
<tr>
<td>ViT-ResNAS-f</td>
<td>Liao et al, 2021</td>
<td>41 M</td>
<td>1.8 B</td>
</tr>
<tr>
<td>ViT-ResNAS-s</td>
<td>Liao et al, 2021</td>
<td>65 M</td>
<td>2.8 B</td>
</tr>
<tr>
<td>As-ViT Base (ours)</td>
<td>-</td>
<td>52.6 M</td>
<td>8.9 B</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Method</th>
<th>Params.</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>RegNetY-16GF</td>
<td>Radosavovic et al, 2020</td>
<td>84.0 M</td>
<td>16.0 B</td>
</tr>
<tr>
<td>ViT-B/16</td>
<td>Dosovitskiy et al, 2020</td>
<td>86.0 M</td>
<td>55.4 B</td>
</tr>
<tr>
<td>DeiT-B</td>
<td>Touvron et al, 2020</td>
<td>86.0 M</td>
<td>17.3 B</td>
</tr>
<tr>
<td>ConViT-B</td>
<td>Asilak et al, 2021</td>
<td>86 M</td>
<td>17 B</td>
</tr>
<tr>
<td>Swin-B</td>
<td>Lin et al, 2021</td>
<td>88.0 M</td>
<td>15.1 B</td>
</tr>
<tr>
<td>GLiT-Base</td>
<td>Chen et al, 2021a</td>
<td>96.1 M</td>
<td>17.0 B</td>
</tr>
<tr>
<td>ViT-ResNAS-f</td>
<td>Liao et al, 2021</td>
<td>97 M</td>
<td>4.5 B</td>
</tr>
<tr>
<td>CaIT S-48</td>
<td>Touvron et al, 2021</td>
<td>89.5 M</td>
<td>18.6 B</td>
</tr>
<tr>
<td>As-ViT Large (ours)</td>
<td>-</td>
<td>88.1 M</td>
<td>22.6 B</td>
</tr>
</tbody>
</table>

† Under 384 × 384 resolution.

$dilation = round((stride/S_1 - 1) \times K_1/(K_1 - 1)) + 1, S_1 = 4$ is the stride at the full token resolution.
More importantly, our As-ViT framework achieves competitive or stronger performance than concurrent NAS works for ViTs with much more search efficiency. As-ViTs are designed with highly reduced human or NAS efforts. All our three As-ViT variants are generated in only 12 GPU hours (on a single V100 GPU). In contrast, BoneNAS (Li et al., 2021) requires 10 GPU days to search a single architecture. For each variant of ViT-ResNAS (Liao et al., 2021), the super-network training takes 16.7~21 hours, followed by another 5.5~6 hours of evolutionary search.

**Efficient Training.** We leverage the progressive elastic re-tokenization strategy proposed in Section 4 to reduce both FLOPs and training time for large ViT models. As illustrated in Figure 4, we progressively apply $4\times$ and $2\times$ reductions on the number of tokens during training by changing both the dilation and the stride of the first linear projection layer. We tune the epochs allocated to each token reduction stage and show the results in Table 6. Standard training takes 42.8 TPU days, whereas our efficient training could save up to 56.2% training FLOPs and 41.1% training TPU days, still achieving a strong accuracy.

**Decoupled Contributions from Topology and Scaling.** To better verify the contribution from our searched topology and scaling rule, we conduct more ablation studies (Table 7). First, we directly train the searched topology before scaling. Our searched seed topology is better than the best from 87 random topologies in Figure 2. Second, we compare our complexity-based scaling rule with “random scaling + As-ViT topology”. At different scales, our automated scaling is also better than random scaling.

### 5.3 Object Detection on COCO

**Settings** Beyond image classification, we further evaluate our designed As-ViT on the detection task. Object detection is conducted on COCO 2017 that contains 118,000 training and 5000 validation images. We adopt the popular Cascade Mask R-CNN as the object detection framework for our As-ViT. We use an input size of $1024 \times 1024$, AdamW optimizer (initial learning rate of 0.001), weight decay of 0.0001, and a batch size of 256. Efficiently pretrained ImageNet-1K checkpoint is leveraged as the initialization.

We compare our As-ViT to standard CNN (ResNet) and previous Transformer network (Swin (Liu et al., 2021)). The comparisons are conducted by changing only the backbones with other settings unchanged. In Table 8, we can see that our As-ViT can also capture multi-scale features and achieve state-of-the-art detection performance, although being designed on ImageNet and its complexity is measured for classification.

### Table 6: Efficient training on ImageNet-1k ($224 \times 224$) via progressive elastic re-tokenization strategy (Section 4). $4\times$ (resp. $2\times$) indicates we reduce the number of tokens by 4 (resp. 2) times, and "N/A" indicates no token reduction.

<table>
<thead>
<tr>
<th>Token Reduction (Epochs)</th>
<th>FLOPs Saving</th>
<th>Training Time (TPU days)</th>
<th>Top1 Acc.</th>
</tr>
</thead>
<tbody>
<tr>
<td>4\times</td>
<td>2\times</td>
<td>N/A</td>
<td></td>
</tr>
<tr>
<td>1~40</td>
<td>41~70</td>
<td>71~300</td>
<td>18.7%</td>
</tr>
<tr>
<td>1~80</td>
<td>81~140</td>
<td>141~300</td>
<td>37.4%</td>
</tr>
<tr>
<td>1~120</td>
<td>121~210</td>
<td>211~300</td>
<td>56.2%</td>
</tr>
<tr>
<td>Baseline</td>
<td>100%</td>
<td>42.8</td>
<td>83.5</td>
</tr>
</tbody>
</table>

### Table 7: Decoupling the contributions from the seed topology and the scaling, on ImageNet-1K.

<table>
<thead>
<tr>
<th>Model</th>
<th>Params.</th>
<th>FLOPs</th>
<th>Top-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>As-ViT Topology</td>
<td>2.4 M</td>
<td>0.5 B</td>
<td>61.7%</td>
</tr>
<tr>
<td>Random Topology</td>
<td>2.2 M</td>
<td>0.4 B</td>
<td>61.4%</td>
</tr>
<tr>
<td>As-ViT Small</td>
<td>29.0 M</td>
<td>5.3 B</td>
<td>81.2%</td>
</tr>
<tr>
<td>Random Scaling</td>
<td>24.2 M</td>
<td>8.7 B</td>
<td>80.5%</td>
</tr>
<tr>
<td>As-ViT Base</td>
<td>52.6 M</td>
<td>8.9 B</td>
<td>82.5%</td>
</tr>
<tr>
<td>Random Scaling</td>
<td>42.4 M</td>
<td>15.5 B</td>
<td>82.2%</td>
</tr>
<tr>
<td>As-ViT Large</td>
<td>88.1 M</td>
<td>22.6 B</td>
<td>83.5%</td>
</tr>
<tr>
<td>Random Scaling</td>
<td>81.1 M</td>
<td>28.7 B</td>
<td>83.2%</td>
</tr>
</tbody>
</table>

### Table 8: Two-stage object detection and instance segmentation results. We compare employing different backbones with Cascade Mask R-CNN on single model without test-time augmentation.

<table>
<thead>
<tr>
<th>Backbone</th>
<th>Resolution</th>
<th>FLOPs</th>
<th>Params.</th>
<th>AP$_{val}$</th>
<th>AP$_{mask\ val}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>ResNet-152</td>
<td>480~800×1333</td>
<td>527.7 B</td>
<td>96.7 M</td>
<td>49.1</td>
<td>42.1</td>
</tr>
<tr>
<td>Swin-B (Liu et al., 2021)</td>
<td>480~800×1333</td>
<td>982 B</td>
<td>145 M</td>
<td>51.9</td>
<td>45</td>
</tr>
<tr>
<td>As-ViT Large (ours)</td>
<td>1024×1024</td>
<td>1094.2 B</td>
<td>138.8 M</td>
<td>52.7</td>
<td>45.2</td>
</tr>
</tbody>
</table>
6 RELATED WORKS

6.1 VISION TRANSFORMER

Transformers (Vaswani et al., 2017) leverage the self-attention to extract global correlation, and become the dominant models for natural language processing (NLP) (Devlin et al., 2018; Radford et al., 2018; Brown et al., 2020; Liu et al., 2019b). Recent works explored transformers to vision problems: image classification (Dosovitskiy et al., 2020), object detection (Carion et al., 2020; Zhu et al., 2020; Zheng et al., 2020; Dai et al., 2020; Sun et al., 2020), segmentation (Chen et al., 2020; Wang et al., 2020), etc. The Vision Transformer (ViT) (Dosovitskiy et al., 2020) designed a pure transformer architecture and achieved SOTA performance on image classification. However, ViT heavily relies on large-scale datasets (ImageNet-21k (Deng et al., 2009), JFT-300M (Sun et al., 2017)) for pretraining, requiring huge computation resources. DeiT (Touvron et al., 2020) proposed Knowledge Distillation (KD) (Hinton et al., 2015; Yuan et al., 2020) via a special KD token to improve both performance and training efficiency. In contrast, our proposed As-ViT introduces more flexible tokenization, attention splitting, and FFN expansion strategies, with automated discovery.

6.2 NEURAL ARCHITECTURE DESIGN AND SCALE

Manual design of network architectures heavily relies on human prior, which is difficult to scale-up. Recent works leverage AutoML to find optimal combinations of operators/topology in a given search space (Zoph & Le, 2016; Real et al., 2019; Liu et al., 2018; Dong & Yang, 2019). However, the searched model are small due to the fixed and hand-crafted search space, far from being scaled-up to modern networks. For example, models from NASNet space (Zoph et al., 2018) only have ~5M parameters, much smaller than real-world ones (20 to over 100M). One main reason for not being scalable is because NAS is a computation-consuming task, typically costing 1~2 GPU days to search even small architectures. Meanwhile, many works try to grow a “seed” architecture to different variants. EfficientNet (Tan & Le, 2019) manually designed a scaling rule for width and depth. Give a template backbone with fixed depth, Liu et al. (2019a) grow the width by gradient descent. For ViT, we for the first time bring both architecture design and scaling together in one framework. To overcome the computation-consuming problem in the training of transformers, we directly use the complexity of manifold propagation as a surrogate measure towards a training-free search and scale.

6.3 EFFICIENT TRAINING

A number of methods have been developed to accelerate the training of deep neural networks, including mixed precision (Jia et al., 2018), distributed optimization (Cho et al., 2017), large-batch training (Goyal et al., 2017; Akiba et al., 2017; You et al., 2018), etc. Jia et al. (2018) combined distributed training with a mixed-precision framework. Wang et al. (2019) proposed to save deep CNN training energy cost via stochastic mini-batch dropping and selective layer update. In our work, customized progressive tokenization via the changing of stride/dilation can effectively reduce the number of tokens during ViT training, thus largely saving the training cost.

7 CONCLUSIONS

To automate the principled design of vision transformers without tedious human efforts, we propose As-ViT, a unified framework that searches and scales ViTs without any training. Compared with hand-crafted ViT architecture, our As-ViT leverages more token overlaps, increased FFN expansion ratios, and is wider and shallower. Our As-ViT achieves state-of-the-art accuracies on both ImageNet-1K classification and COCO detection, which verifies the strong performance of our framework. Moreover, with progressive tokenization, we can train heavy ViT models with largely reduced training FLOPs and time. We hope our methodology could encourage the efficient design and training of ViTs for both the transformer and the NAS communities.

ACKNOWLEDGEMENT

Z.W. is in part supported by the NSF AI Institute for Foundations of Machine Learning (IFML) and a Google TensorFlow Model Garden Award.
REFERENCES


A IMPLEMENTATIONS

Training-free Topology Search and Scaling. We calculate \( \mathcal{L}_E \) by uniformly sampling 10 \( \theta \)s in \([0, 2\pi)\). For one architecture, the calculation of \( \mathcal{L}_E \) is repeated five times with different (random) network initializations, and \( \mathcal{L}_E \) is set as their mean.

Image Classification. We use 20 epochs of linear warm-up, a batch size of 1,024, an initial learning rate of 0.001, and a weight decay of 0.05. Augmentations including stochastic depth (Huang et al., 2016), Mixup (Zhang et al., 2017), Cutmix (Yun et al., 2019), RandAug (Cubuk et al., 2020), Exponential Moving Average (EMA) are also applied.

Object Detection. Our training adopts a batch size of 256 for 36 epochs, with also stochastic depth. We do not use any stronger techniques like HTC (Chen et al., 2019), multi-scale testing, soft-NMS (Bodla et al., 2017), etc.

B CONVERGENCE OF TRAINING-FREE SEARCH

To demonstrate the convergence of the policy learned by our RL search, we show the entropy during learning the policy in Figure 5. We can see that a training of 500 steps is enough for the policy to converge to low entropy (high confidence).

Figure 5: Entropy of policy during our search (Section 3.3).