Nested Hierarchical Transformer: Towards Accurate, Data-Efficient and Interpretable Visual Understanding

Zizhao Zhang[1], Han Zhang[2], Long Zhao[2], Ting Chen[2], Sercan Ö. Arık[1], Tomas Pfister[1]

[1]Google Cloud AI
[2]Google Research
{zizhaoz,zhanghan,longzh,iamtingchen,soarik,tpfister}@google.com

Abstract

Hierarchical structures are popular in recent vision transformers, however, they require sophisticated designs and massive datasets to work well. In this paper, we explore the idea of nesting basic local transformers on non-overlapping image blocks and aggregating them in a hierarchical way. We find that the block aggregation function plays a critical role in enabling cross-block non-local information communication. This observation leads us to design a simplified architecture that requires minor code changes upon the original vision transformer. The benefits of the proposed judiciously-selected design are threefold: (1) NesT converges faster and requires much less training data to achieve good generalization on both ImageNet and small datasets like CIFAR; (2) when extending our key ideas to image generation, NesT leads to a strong decoder that is 8 times faster than previous transformer-based generators; and (3) we show that decoupling the feature learning and abstraction processes via this nested hierarchy in our design enables constructing a novel method (named GradCAT) for visually interpreting the learned model. Source code is available https://github.com/google-research/nested-transformer.

Introduction

Vision Transformer (ViT) (Dosovitskiy et al. 2021) model and its variants have received significant interests recently due to their superior performance on many core visual applications (Cordonnier, Loukas, and Jaggi 2020; Liu et al. 2021). ViT first splits an input image into patches, and then patches are treated in the same way as tokens in NLP applications. Following, several self-attention layers are used to conduct global information communication to extract features for classification. Recent work (Dosovitskiy et al. 2021; Cordonnier, Loukas, and Jaggi 2020) shows that ViT models can achieve better accuracy than state-of-the-art convnets (Tan and Le 2019; He et al. 2016) when trained on datasets with tens or hundreds of millions of labeled samples. However, when trained on smaller datasets, ViT usually underperforms its counterparts based on convolutional layers. Addressing this data inefficiency is important to make ViT applicable to other application scenarios, e.g. semi-supervised learning (Sohn et al. 2020) and generative modeling (Goodfellow et al. 2014; Zhang et al. 2019).

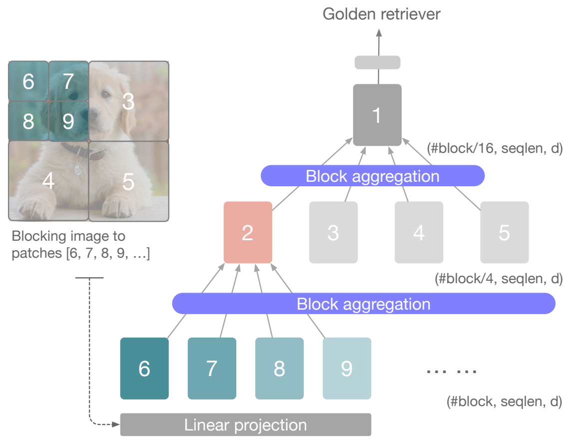
Copyright © 2022, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

Lack of inductive bias such as locality and translation equivariance, is one explanation for the data inefficiency of ViT models. Cordonnier, Loukas, and Jaggi (2020) discovered that transformer models learn locality behaviors in a deformable convolution manner (Dai et al. 2017): bottom layers attend locally to the surrounding pixels and top layers favor long-range dependency. On the other hand, global self-attention between pixel pairs in high-resolution images is computationally expensive. Reducing the self-attention range is one way to make the model training more computationally efficient (Beltagy, Peters, and Cohan 2020). These type of insights align with the recent structures with local self-attention and hierarchical transformer (Han et al. 2021; Vaswani et al. 2021; Liu et al. 2021). Instead of holistic global self-attention, these perform attention on local image patches. To promote information communication across patches, they propose specialized designs such as the "haloing operation" (Vaswani et al. 2021) and "shifted window" (Liu et al. 2021). These are based on modifying the self-attention mechanism and often yields in complex architectures. Our design goal on the other hand keeping the attention as is, and introducing the design of the aggregation function, to improve the accuracy and data efficiency, while bringing interpretability benefits.

The proposed NesT model stacks canonical transformer blocks to process non-overlapping image blocks individually. Cross-block self-attention is achieved by nesting these transformers hierarchically and connecting them with a proposed aggregation function. Fig. 1 illustrates the overall architecture and the simple pseudo code to generate it. Our contributions can be summarized as:

1. We demonstrate integrating hierarchically nested transformers with the proposed block aggregation function can outperform previous sophisticated (local) self-attention variants, leading to a substantially-simplified architecture and improved data efficiency. This provides a novel perspective for achieving effective cross-block communication.

2. NesT achieves impressive ImageNet classification accuracy with a significantly simplified architectural design. E.g., training a NesT with 38M/68M parameters obtains $83.3\%/83.8\%$ ImageNet accuracy.The favorable data efficiency of NesT is embodied by its fast



Pseudo code: NesT

```
# embed and block image to (#block,seqlen,d)
x = Block(PatchEmbed(input_image))

for i in range(num_hierarchy):
  # apply transformer layers T_i within each block
  # with positional encodings (PE)
  y = Stack([T_i(x[0] + PE_i[0]), ...])
  if i < num_hierarchy - 1:
    # aggregate blocks and reduce #block by 4
    x = Aggregate(y, i)

h = GlobalAvgPool(x) # (1,seqlen,d) to (1,1,d)
logits = Linear(h[0,0]) # (num_classes,)

def Aggregate(x, i):
  z = UnBlock(x) # unblock seqs to (h,w,d)
  z = ConvNormMaxPool_i(x) # (h/2,w/2,d)
  return Block(z) # block to seqs
```

Figure 1: (Left) Illustration of NesT with nested transformer hierarchy; (right) the simple pseudo code to generate the architecture. Each node T_i processes an image block. The block aggregation is performed between hierarchies (num_hierarchy$=3$ here) to achieve cross-block communication on the image (feature map) plane.

convergence, such as achieving 75.9%/82.3% training with 30/100 epochs. Moreover, NesT achieves matched accuracy on small datasets compared with popular convolutional architectures. E.g., training a NesT with 6M parameters using a single GPU results in 96% accuracy on CIFAR10 .

3. We show that when extending this idea beyond classification to image generation, NesT can be repurposed into a strong decoder that achieves better performance than convolutional architectures meanwhile has comparable speed, demonstrated by 64×64 ImageNet generation, which is an important to be able to adopt transformers for efficient generative modeling.
4. Our proposed architectural design leads to decoupled feature learning and abstraction, which has significant interpretability benefits. To this end, we propose a novel method called GradCAT to interpret NesT reasoning process by traversing its tree-like structure. This providing a new type of visual interpretability that explains how aggregated local transformers selectively process local visual cues from semantic image patches.

Related Work

Vision transformer-based models (Cordonnier, Loukas, and Jaggi 2020; Dosovitskiy et al. 2021) and self-attention mechanisms (Vaswani et al. 2021; Ramachandran et al. 2019) have recently attracted significant interest in the research community, with explorations of more suitable architectural designs that can learn visual representation effectively, such as injecting convolutional layers (Li et al. 2021; Srinivas et al. 2021; Yuan et al. 2021) and building local or hierarchical structures (Zhang et al. 2021; Wang et al. 2021b). Existing methods focus on designing a variety of self-attention modifications. Hierarchical ViT structures becomes popular both in vision (Liu et al. 2021; Vaswani et al. 2021) and NLP (Zhang, Wei, and Zhou 2019; Santra, Anusha, and Goyal 2021; Liu and Lapata 2019; Pappagari et al. 2019). However, many methods often add significant architectural complexity in order to optimize accuracy.

One challenge for vision transformer-based models is data efficiency. Although the original ViT (Dosovitskiy et al. 2021) can perform better than convolutional networks with hundreds of millions images for pre-training, such a data requirement is not always practical. Data-efficient ViT (DeiT) (Touvron et al. 2021a,b) attempts to address this problem by introducing teacher distillation from a convolutional network. Although promising, this increases the supervised training complexity, and existing reported performance on data efficient benchmarks (Hassani et al. 2021; Chen et al. 2021) still significantly underperforms convolutional networks. Since ViT has shown to improve vision tasks beyond image classification, with prior work studying its applicability to generative modeling (Parmar et al. 2018; Child et al. 2019; Jiang, Chang, and Wang 2021; Hudson and Zitnick 2021), video understanding (Neimark et al. 2021; Akbari et al. 2021), segmentation and detection (Wang et al. 2021a; Liang et al. 2020; Kim et al. 2021), interpretability (Chefer, Gur, and Wolf 2021; Abnar and Zuidema 2020), a deeper understanding of the data efficiency and training difficulties from the architectural perspective is of significant impact.

Proposed Method

Main Architecture

According to Fig. 1, our overall design stacks canonical transformer layers to conduct local self-attention on every image block independently, and then nests them hierarchically. Coupling of processed information between spatially adjacent blocks is achieved through a proposed block aggregation between every two hierarchies. The overall hierarchical structure can be determined by two key hyper-parameters: patch size $S \times S$ and number of block hierarchies T_d. All blocks inside each hierarchy share one set of parameters.

Given an input of image with shape $H \times W \times 3$, each image patch with size $S \times S$ is linearly projected to an

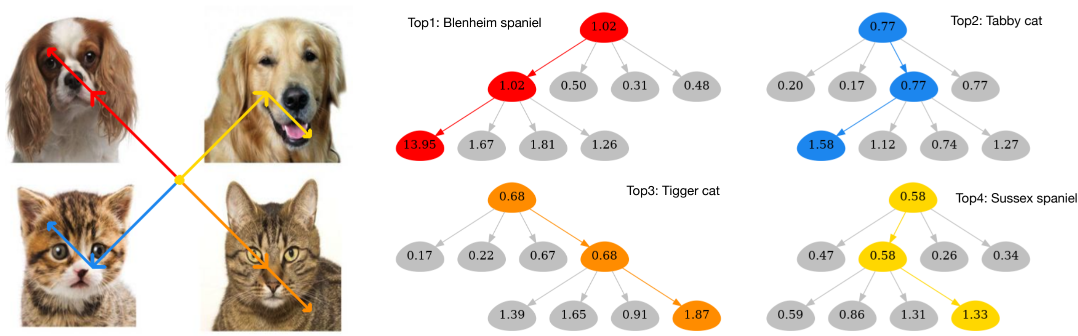


Figure 2: Example results of the proposed GradCAT. Given the left input image (containing four objects), the figure visualizes the top-4 class traversal results (4 colors) using an ImageNet-trained NesT (with three tree hierarchies). Each tree node denotes the averaged activation value ($\hat{h}_l$ defined in Algorithm 1). The traversals can correctly find the model decision path along the tree to locate an image patch belonging to the objects of given target classes.

embedding in $\mathbb{R}^d$. Then, all embeddings are partitioned to blocks and flattened to generate input $X \in \mathbb{R}^{b \times T_n \times n \times d}$, where b is the batch size, T_n is the total number of blocks at bottom of the NesT hierarchy, and n is the sequence length (the number of embeddings) at each block. Note that $T_n \times n = H \times W/S^2$.

Inside each block, we stack a number of canonical transformer layers, where each is composed of a multi-head self-attention (MSA) layer followed by a feed-forward fully-connected network (FFN) with skip-connection (He et al. 2016) and Layer normalization (LN) (Ba, Kiros, and Hinton 2016). Trainable positional embedding vectors (Touvron et al. 2021a) are added to all sequence vectors in $\mathbb{R}^d$ to encode spatial information before feeding into the block function T:

$$\text{multiple } \times \begin{cases} y = x + \text{MSA}_{\text{NesT}}(x', x', x'), \; x' = \text{LN}(x) \\ x = y + \text{FFN}(\text{LN}(y)) \end{cases} \tag{1}$$

The FFN is composed of two layers: $\max(0, xW_1 + b)W_2 + b$. Given input $X \in \mathbb{R}^{b \times T_n \times n \times d}$, since all blocks at one NesT hierarchy share the same parameters, MSA_{NesT} basically MSA is applied (Vaswani et al. 2017) to all blocks in parallel:

$$\begin{aligned} \text{MSA}_{\text{NesT}}(Q, K, V) &= \text{Stack}(\text{block}_1, ..., \text{block}_{T_n}), \\ \text{where } \text{block}_i &= \text{MSA}(Q, K, V)W^O. \end{aligned} \tag{2}$$

block_i has shape $b \times n \times d$. Lastly, we build a nested hierarchy with block aggregation – every four spatially connected blocks are merged into one. The overall design makes NesT easy to implement, requiring minor code changes to the original ViT.

Block Aggregation

From a high-level view, NesT leads to hierarchical representations, which share similarity with several pyramid designs (Zhang et al. 2021; Wang et al. 2021b). However, most of these works use global self-attention throughout the layers, interleaved with (spatial) down-sampling. In contrast, we show that NesT, which leverages local attention, can lead to significantly improved data efficiency. In local self-attention, non-local communication is important to maintain translational equivariance (Vaswani et al. 2021). To this end, Halonet (Vaswani et al. 2021) allows the query to attend to slightly larger regions than the assigned block. Swin Transformer (Liu et al. 2021) achieves this by shifting the block partition windows between consecutive self-attention layers to connect adjacent blocks; applying special masked self-attention to guarantee spatial continuity. However, both add complexity to the self-attention layers and such sophisticated architectures are not desired from implementation perspective.

On the other hand, every block in NesT processes information independently via standard transformer layers, and only communicate and mix global information during the block aggregation step via simple spatial operations (e.g. convolution and pooling). One key ingredient of block aggregation is to perform it in the image plane so that information can be exchanged between nearby blocks. This procedure is summarized in Fig. 1. The output $X_l \in \mathbb{R}^{b \times \#\text{block} \times n \times d}$ at hierarchy l is unblocked to the full image plane $A_l \in \mathbb{R}^{b \times H' \times W' \times d'}$. A number of spatial operations are applied to down-sample feature maps $A'_l \in \mathbb{R}^{b \times H'/2 \times W'/2 \times d}$. Finally, the feature maps are blocked back to $X_{l+1} \in \mathbb{R}^{b \times \#\text{block}/4 \times n \times d'}$ for hierarchy $l+1$. The sequence length n always remains the same and the total number of blocks is reduced by a factor of 4, until reduced to 1 at the top (i.e. $\#\text{block}/4^{(T_d - 1)} = 1$). Therefore, this process naturally creates hierarchically nested structure where the "receptive field" expands gradually. $d' \geq d$ depends on the specific model configuration.

Our block aggregation is specially instantiated as a 3×3 convolution followed by LN and a 3×3 max pooling. Figure A2 in Appendix explains the core design and the importance of applying it on the image plane (i.e. full image feature maps) versus the block plane (i.e. partial feature maps corresponding to 2×2 blocks that will be merged). The small information exchange through the small convolution and max. pooling kernels across block boundaries are particularly important. We conduct comprehensive ablation studies to demonstrate the importance of each of the design components.

Note that the resulting design shares some similarities with recent works that combine transformer and convolutional networks (Wu et al. 2021; Yuan et al. 2021; Bello 2021) as specialized hybrid structures. However, unlike these, our proposed method aims to solve cross-block communications in local self-attention, and the resulting architecture is simple as a stacking of basic transformer layers.

Transposed NesT for Image Generation

The data efficiency and straightforward implementation of NesT makes it desirable for more complex learning tasks. With transpose the key ideas from NesT to propose a decoder for generative modeling, and show that it has better performance than convolutional decoders with comparable speed. Remarkably, it is nearly a magnitude faster than the transformer-based decoder TransGAN (Jiang, Chang, and Wang 2021).

Creating such a generator is straightforward by transposing NesT (see Table A6 of Appendix for architecture details). The input of the model becomes a noise vector and the output is a full-sized image. To support the gradually increased number of blocks, the only modification to NesT is replacing the block aggregation with appropriate block de-aggregation, i.e. up-sampling feature maps (we use pixel shuffle (Shi et al. 2016)). The feature dimensions in all hierarchies are $(b, nd) \rightarrow (b, 1, n, d) \rightarrow (b, 4, n, d'), ..., \rightarrow (b, \#\text{blocks}, n, 3)$. The number of blocks increases by a factor of 4. Lastly, we can unblock the output sequence tensor to an image with shape $H \times W \times 3$. The remaining adversarial training techniques are based on (Goodfellow et al. 2014; Zhang et al. 2019) as explained in experiments. Analogous to our results for image classification, we show the importance of careful block de-aggregation design, in making the model significantly faster while achieving better generation quality.

GradCAT: Interpretability via Tree Traversal

Different from previous work, the nested hierarchy with the independent block process in NesT resembles a decision tree in which each block is encouraged to learn non-overlapping features and be selected by the block aggregation. This unique behavior motivates us to explore a new method to explain the model reasoning, which is an important topic with significant real world impact in convnets (Selvaraju et al. 2017; Sundararajan, Taly, and Yan 2017).

We present a gradient-based class-aware tree-traversal (GradCAT) method (Algorithm 1). The main idea is to find the most valuable traversal from a child node to the root node that contributes to the classification logits the most. Intuitively, at the top hierarchy, each of four child nodes processes one of 2×2 non-overlapping partitions of feature maps A_{T_d}. We can use corresponding activation and class-specific gradient features to trace the high-value information flow recursively from the root to a leaf node. The negative gradient $-\frac{\partial Y_c}{A_l}$ provides the gradient ascent direction to maximize the class c logit, i.e., a higher positive value means higher importance. Fig. 2 illustrates a sample result.

Algorithm 1: GradGAT

Define: A_l denotes the feature maps at hierarchy l. Y_c is the logit of predicted class c. $[\cdot]_{2\times2}$ indexes one of 2×2 partitions of input maps.
Input: $\{A_l | l = 2, ..., T_d\}, \alpha_{T_d} = A_{T_d}, P = []$
Output: The traversal path P from top to bottom
for $l = [T_d, ..., 2]$ **do**
$h_l = \alpha_l \cdot (-\frac{\partial Y_c}{\alpha_l})$ *# obtain target activation maps*
$\hat{h}_l = \text{AvgPool}_{2\times2}(h_l) \in \mathbb{R}^{2\times2}$
$n_l^* = \arg\max \hat{h}_l, \quad P = P + [n_l^*]$ *# pick the maximum index*
$\alpha_l = A_l[n_l^*]_{2\times2}$ *# obtain the partition for the index*
end for

Arch. base	Method	C10 (%)	C100 (%)
Convolutional	Pyramid-164-48	95.97	80.70
	WRN28-10	95.83	80.75
Transformer full-attention	DeiT-T	88.39	67.52
	DeiT-S	92.44	69.78
	DeiT-B	92.41	70.49
	PVT-T	90.51	69.62
	PVT-S	92.34	69.79
	PVT-B	85.05*	43.78*
	CCT-7/3×1	94.72	76.67
Transformer local-attention	Swin-T	94.46	78.07
	Swin-S	94.17	77.01
	Swin-B	94.55	78.45
	NesT-T	96.04	78.69
	NesT-S	96.97	81.70
	NesT-B	**97.20**	**82.56**

Table 1: Test accuracy on CIFAR with input size 32×32. The compared convolutional architectures are optimized models for CIFAR. All transformer-based architectures are trained from random initialization with the same data augmentation. DeiT uses $S = 2$. Swin and our NesT uses $S = 1$. * means model tends to diverge.

Experiments

We first show the benefit of NesT for data efficient learning and then demonstrate benefits for interpretability and generative modeling. Finally, we present ablation studies to analyze the major constituents of the methods.

Experimental setup. We follow previous work (Dosovitskiy et al. 2021) to generate three architectures that have comparable capacity (in number of parameters and FLOPS), noted as tiny (NesT-T), small (NesT-S), and base (NesT-B). Most recent ViT-based methods follow the training techniques of DeiT (Touvron et al. 2021a). We follow the settings with minor modifications that we find useful for local self-attention (see Appendix for all architecture and training details). We do not explore the specific per-block configurations (e.g. number of heads and hidden dimensions), which we believe can be optimized through architecture search (Tan and Le 2019).

Arch. base	Method	#Params	Top-1 acc. (%)
Convolutional	ResNet-50	25M	76.2
	RegNetY-4G	21M	80.0
	RegNetY-16G	84M	82.9
Transformer full-attention	ViT-B/16	86M	77.9
	DeiT-S	22M	79.8
	DeiT-B	86M	81.8
Transformer local-attention	Swin-T	29M	81.3
	Swin-S	50M	83.0
	Swin-B	88M	83.3
	NesT-T	17M	81.5
	NesT-S	38M	83.3
	NesT-B	68M	**83.8**

Table 2: Comparison on the ImageNet dataset. All models are trained from random initialization. ViT-B/16 uses an image size 384 and others use 224.

	ViT-B/16	Swin-B	Nest-B
ImageNet Acc. (%)	84.0	86.0	**86.2**

Table 3: Comparison on ImageNet benchmark with ImageNet-22K pre-training.

Comparisons to Previous Work

CIFAR. We compare NesT to recent methods on CIFAR datasets (Krizhevsky, Hinton et al. 2009) in Table 1, to investigate the data efficiency. It is known that transformer-based methods usually perform poorly on such tasks as they typically require large datasets to be trained on. The models that perform well on large-scale ImageNet do not necessary work perform on small-scale CIFAR, as the full self-attention based models require larger training datasets. DeiT (Touvron et al. 2021a) performs poorly and does not improve given bigger model size. PVT (Wang et al. 2021b) has also a full self-attention based design, though with a pyramid structure. PVT-T seems to perform better than DeiT-T when model size is small, however, the performance largely drops and becomes unstable when scaling up, further suggesting that full self-attention at bottom layers is not desirable for data efficiency. Other transformer-based methods improve slowly with increasing model size, suggesting that bigger models are more challenging to train with less data. We attribute this to to their complex design (i.e. shifted windows with masked MSA) requiring larger training datasets, while NesT benefiting from a judiciously-designed block aggregation. We also include comparisons with convolutional architectures that are specifically optimized for small CIFAR images and show that NesT can give better accuracy without any small dataset specific architecture optimizations (while still being larger and slower, as they do not incorporate convolutional inductive biases). The learning capacity and performance of NesT get better with increased model size. Most variants of NesT in Fig. A1 of Appendix outperform compared methods with far better throughput. E.g., NesT_3-T ($S = 2$) leads to 94.5% CIFAR10 accuracy with 5384 images/s throughout, $10\times$ faster than the best compared result 94.6% accuracy. More details can be found in Appendix.

ImageNet. We test NesT on standard ImageNet 2012 benchmarks (Deng et al. 2009) with commonly used 300 epoch training on TPUs in Table 2. The input size is 224×224 and no extra pre-training data is used. DeiT does not use teacher distillation, so it can be viewed as ViT (Dosovitskiy et al. 2021) with better data augmentation and regularization. NesT matches the performance of prior work with a significantly more straightforward design (e.g. NesT-S matches the accuracy of Swin-B, 83.3%). The results of NesT suggest that correctly aggregating the local transformer can improve the performance of local self-attention.

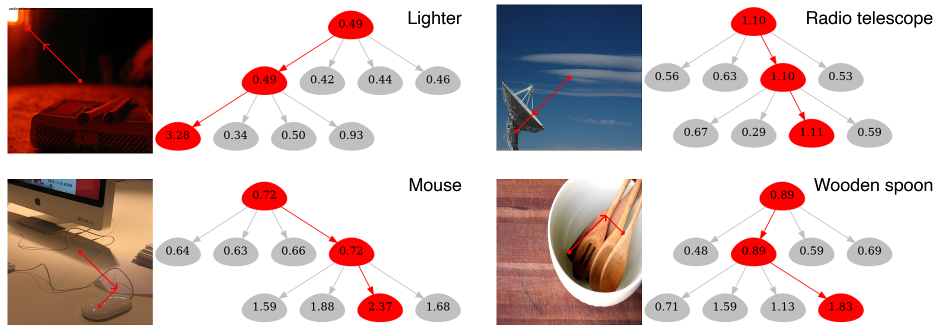
ImageNet-22K. We scale up NesT to ImageNet-22K following the exact training schedules in (Liu et al. 2021; Dosovitskiy et al. 2021). The pre-training is 90 epoch on 224×224 ImageNet21K images and finetuning is 30 epoch on 384×384 ImageNet images. Table 3 compares the results. NesT again achieves competitive results, with a significantly more straightforward design.

Visual Interpretability

GradGAT results. Fig. 3 (left) shows the explanations obtained with the proposed GradGAT. For GradGAT, each tree node corresponds to a value that reflects the mean activation strength. Visualizing the tree traversal through image blocks, we can get insights about the decision making process of NesT. The traversal passes through the path with the highest values. As can be seen, the decision path can correctly locate the object corresponding to the model prediction. The Lighter example is particularly interesting because the ground truth class – lighter/matchstick – actually defines the bottom-right matchstick object, while the most salient visual features (with the highest node values) are actually from the upper-left red light, which conceptually shares visual cues with a lighter. Thus, although the visual cue is a mistake, the output prediction is correct. This example reveals the potential of using GradGAT to conduct model diagnosis at different tree hierarchies. Fig. A5 of Appendix shows more examples.

Class attention map (CAM) results. In contrast to ViT (Dosovitskiy et al. 2021) which uses class tokens, NesT uses global average pooling before softmax. This enables conveniently applying CAM-like (Zhou et al. 2016) methods to interpret how well learned representations measure object features, as the activation coefficients can be directly without approximate algorithms. Fig. 3(right) shows quantitative evaluation of weakly-supervised object localization, which is a common evaluation metric for CAM-based methods (Zhou et al. 2016), including GradCAM++ (Chattopadhay et al. 2018) with ResNet50 (He et al. 2016), DeiT with Rollout attention (Abnar and Zuidema 2020), and our NesT CAM (Zhou et al. 2016). We follow this toolkit[1] to use an improved version of Rollout. NesT with standard CAM, outperforms others that are specifically designed for this task. Fig. 4 shows a qualitative comparison (details in the Appendix), exemplifying that NesT can generate clearer attention maps which converge better on objects.

[1] https://github.com/jacobgil/vit-explain



Method	Top-1 loc. err (%)
DeiT RollOut§	67.6
ADL (Choe and Shim 2019)	55.1
ACoL (Zhang et al. 2018)	54.2
R50 GradCAM++§	53.5
NesT CAM	**51.8**

Figure 3: Left: Output visualization of the proposed GradGAT. Tree nodes annotate the averaged responses to the predicted class. We use a NesT-S with three tree hierarchies. Right: CAM-based weakly supervised localization comparison on the ImageNet validation set. § indicates results obtained by us.

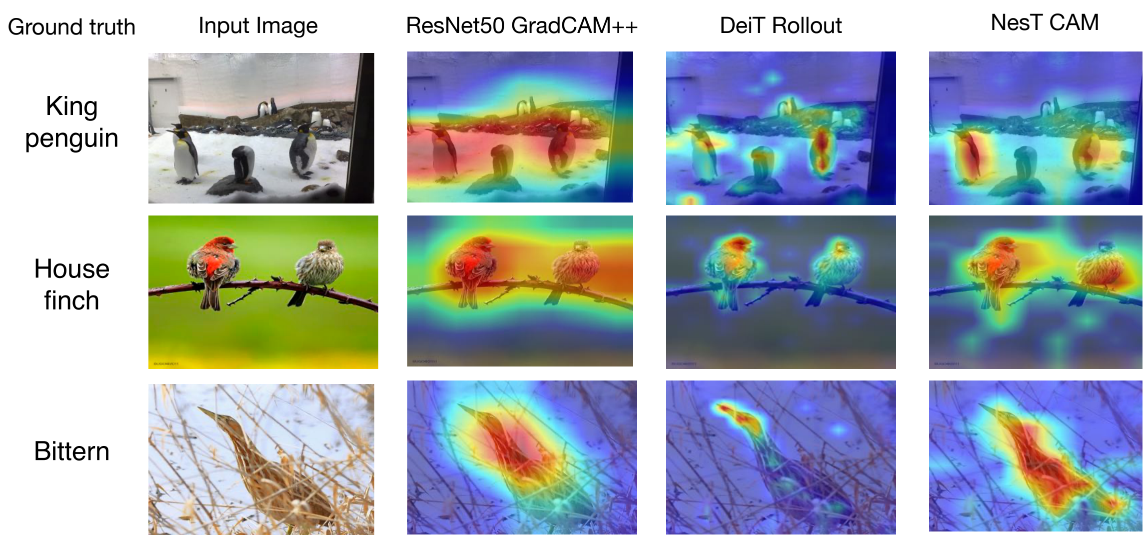


Figure 4: Visualization of CAM-based attention results. All models are trained on ImageNet. CAM (vanilla) with NesT achieves accurate attention patterns on object regions, yielding finer attention to objects than DeiT Rollout (Abnar and Zuidema 2020) and less noise than ResNet50 GradCAM++ (Chattopadhay et al. 2018).

Overall, decoupling local self-attention (transformer blocks) and global information selection (block aggregation), which is unique to our work, shows significant potential for making models easier to interpret.

Generative Modeling with Transposed NesT

We evaluate the generative ability of Transposed NesT on ImageNet (Russakovsky et al. 2015) where all images are resized to 64×64 resolution. We focus on the unconditional image generation setting to test the effectiveness of different decoders. We compare Transposed NesT to TransGAN (Jiang, Chang, and Wang 2021), that uses a full Transformer as the generator, as well as a convolutional baseline following the widely-used architecture from (Zhang et al. 2019) (its computationally expensive self-attention module is removed). Fig. 5 shows the results. Transposed NesT obtains significantly faster convergence and achieves the best FID and Inception score (see Fig. A6 of Appendix for results). Most importantly, it achieves $8\times$ throughput over TransGAN, showing its potential for significantly improving the efficiency of transformer-based generative modeling. More details are explained in the Appendix.

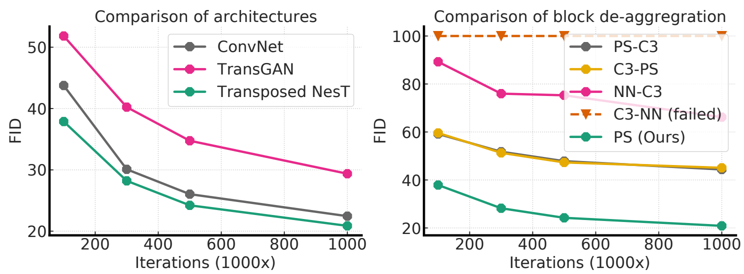
It is noticeable from Fig. 5 (middle) that appropriate unsampling (or block de-aggregation) impacts the generation quality. Pixel shuffle (Shi et al. 2016) works the best and the margin is considered surprisingly large compared to other alternatives widely-used in convnets. This aligns with our main findings in classification, suggesting that judiciously injecting spatial operations is important for nested local transformers to perform well.

Ablation Studies

We summarize key ablations below (more in Appendix).

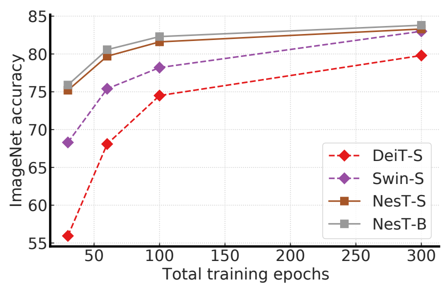
Fast convergence. NesT achieves fast convergence, as shown in Fig. 6 (top) for Imagenet training with $\{30, 60, 100, 300\}$ epochs. NesT-B merely loses 1.5% when reducing the training epoch from 300 to 100. The results suggest that NesT can learn effective visual features faster and more efficiently.

Less sensitivity to data augmentation. NesT uses several



Method	#Params (millions)	Throughput (images/s)
Convnet (Zhang et al. 2019)	77.8M	709.1
TransGAN (Jiang, Chang, and Wang 2021)	82.6M	67.7
Transposed NesT	74.4M	523.7

Figure 5: Left: FID comparison for 64 × 64 ImageNet generation at different training iterations. Middle: FID comparison of different popular un-sampling methods for block de-aggregation, including combinations of pixel shuffling (PS), Conv3x3 (C3), and nearest neighbor (NN). Right: The number of parameters and throughput of compared generators.



Augmentation Removed	ImageNet Accuracy (%)	
	DeiT-B	NesT-T
None	81.8	81.5
RandomErasing	4.3	81.4
RandAugment	79.6	81.2
CutMix&MixUp	75.8	79.8

Figure 6: Top: Training convergence. NesT achieves better performance than DeiT with the same total epoch of training (each point is a single run). Bottom: Data augmentation ablations. Results of DeiT-B (Touvron et al. 2021a) are reported by its paper. NesT shows less reliance on data augmentation.

kinds of data augmentation following (Touvron et al. 2021a). As shown in Fig. 6 (right) and Fig. A4, our method shows higher stability in data augmentation ablation studies compared to DeiT. Data augmentation is critical for global self-attention to generalize well, but reduced dependence on domain or task dependent data augmentation helps with generalization to other tasks.

Impact of block aggregation. Here we show that the design of block aggregation is critical for performance and data efficiency. We study this from four perspectives: (1) whether unblocking to the full image plane is necessary; (2) how to use convolution; (3) what kinds of pooling to use; and (4) whether to perform query down-sampling inside self-attention (Vaswani et al. 2021). Fig. 7 and Fig. A3 of Appendix compare the results of different plausible designs.

The results show that: (1) when performing these spatial operations, it is important to apply it on the holistic image plane versus the block plane although both can reasonably introduce spatial priors; (2) small kernel convolution is sufficient and has to be applied ahead of pooling; (3)

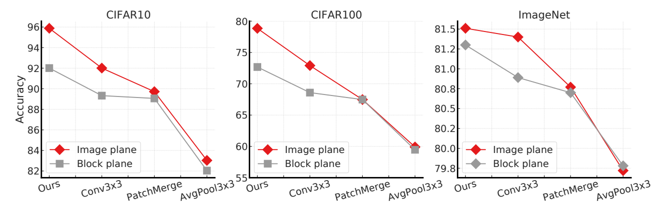


Figure 7: Demonstration of the impact of block aggregation on CIFAR and ImageNet. NesT-T is used. Conv3x3 has stride 2. AvgPool3x3 on ImageNet is followed by Conv1x1 to change hidden dimensions of self-attention. Four plausible block aggregation designs are shown in x-axis, and applied on the image plane and block plane both for comparison. Note that Ours in x-axis is Conv3x3 followed by LN and MaxPool3x3 (stride 2). More alternatives are validated in Fig. A3 of Appendix.

max. pooling is far better than other options, such as stride-2 sub-sampling and average pooling; (4) sub-sampling the query sequence length (similar to performing sub-sampling on the block plane as illustrated in Fig. A2), as used by Halonet (Vaswani et al. 2021), performs poorly on data efficient benchmarks. We also experiment PatchMerge from Swin Transformer (Liu et al. 2021) on both CIFAR and ImageNet. Our block aggregation closes the accuracy gap on ImageNet, suggesting that a conceptually negligible difference in aggregating nested transformers can lead to significant differences in model performance.

Conclusion

We have shown that aggregating nested transformers can match the accuracy of previous more complex methods with significantly improved data efficiency and convergence speed. In addition, we have shown that this idea can be extended to image generation, where it provides significant speed gains. Finally, we have shown that the decoupled feature learning and feature information extraction in this nested hierarchy design allows for better feature interpretability through a new gradient-based class-aware tree traversal method. In future work we plan to generalize this idea to non-image domains.

Acknowledgements

We thank Xiaohua Zhai, Jeremy Kubica, Kihyuk Sohn, and Madeleine Udell for their valuable feedback on this paper.

References

Abnar, S.; and Zuidema, W. 2020. Quantifying attention flow in transformers. In *ACL*.

Akbari, H.; Yuan, L.; Qian, R.; Chuang, W.-H.; Chang, S.-F.; Cui, Y.; and Gong, B. 2021. VATT: Transformers for Multimodal Self-Supervised Learning from Raw Video, Audio and Text. arXiv:2104.11178.

Ba, J. L.; Kiros, J. R.; and Hinton, G. E. 2016. Layer normalization. arXiv:1607.06450.

Bello, I. 2021. Lambdanetworks: Modeling long-range interactions without attention. arXiv:2102.08602.

Beltagy, I.; Peters, M. E.; and Cohan, A. 2020. Longformer: The long-document transformer. arXiv:2004.05150.

Chattopadhay, A.; Sarkar, A.; Howlader, P.; and Balasubramanian, V. N. 2018. Grad-cam++: Generalized gradient-based visual explanations for deep convolutional networks. In *WACV*.

Chefer, H.; Gur, S.; and Wolf, L. 2021. Generic Attention-model Explainability for Interpreting Bi-Modal and Encoder-Decoder Transformers. arXiv:2103.15679.

Chen, Z.; Xie, L.; Niu, J.; Liu, X.; Wei, L.; and Tian, Q. 2021. Visformer: The Vision-friendly Transformer. arXiv:2104.12533.

Child, R.; Gray, S.; Radford, A.; and Sutskever, I. 2019. Generating long sequences with sparse transformers. arXiv:1904.10509.

Choe, J.; and Shim, H. 2019. Attention-based dropout layer for weakly supervised object localization. In *CVPR*.

Cordonnier, J.-B.; Loukas, A.; and Jaggi, M. 2020. On the relationship between self-attention and convolutional layers. In *ICLR*.

Dai, J.; Qi, H.; Xiong, Y.; Li, Y.; Zhang, G.; Hu, H.; and Wei, Y. 2017. Deformable convolutional networks. In *ICCV*.

Deng, J.; Dong, W.; Socher, R.; Li, L.-J.; Li, K.; and Fei-Fei, L. 2009. Imagenet: A large-scale hierarchical image database. In *CVPR*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; Zhai, X.; Unterthiner, T.; Dehghani, M.; Minderer, M.; Heigold, G.; Gelly, S.; et al. 2021. An image is worth 16x16 words: Transformers for image recognition at scale. *ICLR*.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A.; and Bengio, Y. 2014. Generative adversarial networks. In *NeurIPS*.

Han, K.; Xiao, A.; Wu, E.; Guo, J.; Xu, C.; and Wang, Y. 2021. Transformer in transformer. In *NeurIPS*.

Hassani, A.; Walton, S.; Shah, N.; Abuduweili, A.; Li, J.; and Shi, H. 2021. Escaping the Big Data Paradigm with Compact Transformers. arXiv:2104.05704.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016. Deep residual learning for image recognition. In *CVPR*.

Hudson, D. A.; and Zitnick, C. L. 2021. Generative Adversarial Transformers. arXiv:2103.01209.

Jiang, Y.; Chang, S.; and Wang, Z. 2021. Transgan: Two transformers can make one strong gan. In *NeruIPS*.

Kim, B.; Lee, J.; Kang, J.; Kim, E.-S.; and Kim, H. J. 2021. HOTR: End-to-End Human-Object Interaction Detection with Transformers. *CVPR*.

Krizhevsky, A.; Hinton, G.; et al. 2009. Learning multiple layers of features from tiny images. Technical report, University of Toronto.

Li, Y.; Zhang, K.; Cao, J.; Timofte, R.; and Van Gool, L. 2021. LocalViT: Bringing Locality to Vision Transformers. arXiv:2104.05707.

Liang, J.; Homayounfar, N.; Ma, W.-C.; Xiong, Y.; Hu, R.; and Urtasun, R. 2020. Polytransform: Deep polygon transformer for instance segmentation. In *CVPR*.

Liu, Y.; and Lapata, M. 2019. Hierarchical transformers for multi-document summarization. In *ACL*.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin transformer: Hierarchical vision transformer using shifted windows. In *ICCV*.

Neimark, D.; Bar, O.; Zohar, M.; and Asselmann, D. 2021. Video transformer network. *arXiv preprint arXiv:2102.00719*.

Pappagari, R.; Zelasko, P.; Villalba, J.; Carmiel, Y.; and Dehak, N. 2019. Hierarchical transformers for long document classification. In *Automatic Speech Recognition and Understanding Workshop (ASRU)*.

Parmar, N.; Vaswani, A.; Uszkoreit, J.; Kaiser, L.; Shazeer, N.; Ku, A.; and Tran, D. 2018. Image transformer. In *ICML*.

Ramachandran, P.; Parmar, N.; Vaswani, A.; Bello, I.; Levskaya, A.; and Shlens, J. 2019. Stand-alone self-attention in vision models. In *NeurIPS*.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. ImageNet large scale visual recognition challenge. *IJCV*.

Santra, B.; Anusha, P.; and Goyal, P. 2021. Hierarchical transformer for task oriented dialog systems. In *NAACL*.

Selvaraju, R. R.; Cogswell, M.; Das, A.; Vedantam, R.; Parikh, D.; and Batra, D. 2017. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *ICCV*.

Shi, W.; Caballero, J.; Huszár, F.; Totz, J.; Aitken, A. P.; Bishop, R.; Rueckert, D.; and Wang, Z. 2016. Real-time single image and video super-resolution using an efficient sub-pixel convolutional neural network. In *CVPR*.

Sohn, K.; Berthelot, D.; Li, C.-L.; Zhang, Z.; Carlini, N.; Cubuk, E. D.; Kurakin, A.; Zhang, H.; and Raffel, C. 2020. Fixmatch: Simplifying semi-supervised learning with consistency and confidence. In *NeurIPS*.

Srinivas, A.; Lin, T.-Y.; Parmar, N.; Shlens, J.; Abbeel, P.; and Vaswani, A. 2021. Bottleneck transformers for visual recognition. arXiv:2101.11605.

Sundararajan, M.; Taly, A.; and Yan, Q. 2017. Axiomatic attribution for deep networks. In *ICML*.

Tan, M.; and Le, Q. 2019. Efficientnet: Rethinking model scaling for convolutional neural networks. In *ICML*.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; Sablayrolles, A.; and Jégou, H. 2021a. Training data-efficient image transformers & distillation through attention. In *ICML*.

Touvron, H.; Cord, M.; Sablayrolles, A.; Synnaeve, G.; and Jégou, H. 2021b. Going deeper with image transformers. arXiv:2103.17239.

Vaswani, A.; Ramachandran, P.; Srinivas, A.; Parmar, N.; Hechtman, B.; and Shlens, J. 2021. Scaling Local Self-Attention For Parameter Efficient Visual Backbones. In *CVPR*.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, L.; and Polosukhin, I. 2017. Attention is all you need. *NeurIPS*.

Wang, H.; Zhu, Y.; Adam, H.; Yuille, A.; and Chen, L.-C. 2021a. MaX-DeepLab: End-to-End Panoptic Segmentation with Mask Transformers. *CVPR*.

Wang, W.; Xie, E.; Li, X.; Fan, D.-P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2021b. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *ICCV*.

Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. Cvt: Introducing convolutions to vision transformers. arXiv:2103.15808.

Yuan, L.; Chen, Y.; Wang, T.; Yu, W.; Shi, Y.; Jiang, Z.; Tay, F. E.; Feng, J.; and Yan, S. 2021. Tokens-to-token vit: Training vision transformers from scratch on imagenet. In *ICCV*.

Zhang, H.; Goodfellow, I.; Metaxas, D.; and Odena, A. 2019. Self-attention generative adversarial networks. In *ICML*.

Zhang, P.; Dai, X.; Yang, J.; Xiao, B.; Yuan, L.; Zhang, L.; and Gao, J. 2021. Multi-Scale Vision Longformer: A New Vision Transformer for High-Resolution Image Encoding. In *ICCV*.

Zhang, X.; Wei, F.; and Zhou, M. 2019. HIBERT: Document level pre-training of hierarchical bidirectional transformers for document summarization. arXiv:1905.06566.

Zhang, X.; Wei, Y.; Feng, J.; Yang, Y.; and Huang, T. S. 2018. Adversarial complementary learning for weakly supervised object localization. In *CVPR*.

Zhou, B.; Khosla, A.; Lapedriza, A.; Oliva, A.; and Torralba, A. 2016. Learning deep features for discriminative localization. In *CVPR*.