GreedyViG: Dynamic Axial Graph Construction for Efficient Vision GNNs

Mustafa Munir, William Avery, Md Mostafijur Rahman, and Radu Marculescu
The University of Texas at Austin
Austin, Texas, USA
{mmunir, williamaavery, mostafijur.rahman, radum}@utexas.edu

Abstract

Vision graph neural networks (ViG) offer a new avenue for exploration in computer vision. A major bottleneck in ViGs is the inefficient k-nearest neighbor (KNN) operation used for graph construction. To solve this issue, we propose a new method for designing ViGs, Dynamic Axial Graph Construction (DAGC), which is more efficient than KNN as it limits the number of considered graph connections made within an image. Additionally, we propose a novel CNN-GNN architecture, GreedyViG, which uses DAGC. Extensive experiments show that GreedyViG beats existing ViG, CNN, and ViT architectures in terms of accuracy, GMACs, and parameters on image classification, object detection, instance segmentation, and semantic segmentation tasks. Our smallest model, GreedyViG-S, achieves 81.1% top-1 accuracy on ImageNet-1K, 2.9% higher than Vision GNN and 2.2% higher than Vision HyperGraph Neural Network (ViHGNN), with less GMACs and a similar number of parameters. Our largest model, GreedyViG-B obtains 83.9% top-1 accuracy, 0.2% higher than Vision GNN, with a 66.6% decrease in parameters and a 69% decrease in GMACs. GreedyViG-B also obtains the same accuracy as ViHGNN with a 67.3% decrease in parameters and a 71.3% decrease in GMACs. Our work shows that hybrid CNN-GNN architectures not only provide a new avenue for designing efficient models, but that they can also exceed the performance of current state-of-the-art models[1].

1. Introduction

Rapid growth in deep learning has lead to numerous successes across a diverse set of computer vision tasks including image classification [3], object detection [24], instance segmentation [24], and semantic segmentation [51]. Key drivers behind this growth include convolutional neural networks (CNNs) [9, 17, 19, 26], vision transformers (ViTs) [1, 4], and multi-layer perceptron (MLP)-based vision models [38, 40]. In CNNs and MLPs input images are represented as a grid of pixels, however in ViTs images are represented as a sequence of patches. By splitting an input image into a sequence of patch embeddings, the image is transformed into an input usable by the transformer modules often used in natural language processing [25, 42]. Unlike CNNs and MLPs, which have a local receptive field, ViTs have global receptive fields allowing them to learn from distant interactions within images [4].

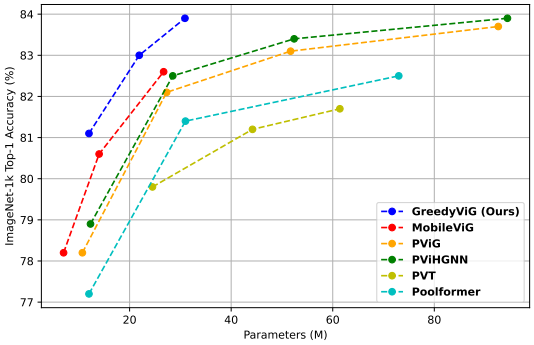

Figure 1. **Comparison of model size and performance (top-1 accuracy on ImageNet-1K)**. GreedyViG achieves the highest performance compared to other state-of-the-art models.

The recently proposed Vision GNN (ViG) [7] represents images in a more versatile manner through a graph structure rather than as a sequence of patches as in ViTs [4]. ViG constructs the graph through dividing an image into patches and then connecting the patches (i.e., nodes) through the K-nearest neighbors (KNN) algorithm [7]. Vision HyperGraph Neural Network (ViHGNN) [8] improves upon the original ViG by using the hypergraph structure to remove the constraint of exclusively connecting pairs of nodes. Like ViTs, ViG-based models can process global object interactions, but are also computationally expensive. To deal with the computationally expensive nature of ViG-based models, MobileViG [30] used a structured graph that does not

[1] Code: https://github.com/SLDGroup/GreedyViG.

change across input images and removes the need for KNN-based graph construction.

While the success of ViG [7], ViHGNN [8], and MobileViG [30] show the potential of treating an image as a graph for computer vision tasks, they also show some limitations. In general ViG-based models are computationally expensive, due to the expensive nature of graph construction. MobileViG alleviates this issue through static graph construction, but at the cost of a graph that does not change across input images, thus limiting the benefit of using a graph-based model. The limitations of current ViG-based models are as follows:

1. **Computational Cost of Graph Construction:** A fundamental issue facing ViG-based models is the cost of KNN-based graph construction. KNN-based graph construction requires comparing every single node within the ViG-based model to determine the K nearest nodes. This cost makes KNN-based ViG models inefficient.
2. **Inability of a Static Graph to Change Across Inputs:** The computational cost of KNN-based ViG models lead to static graph construction. The fundamental issue with static graph construction is it removes the benefit of using a ViG-based model as the graph constructed no longer changes across input images.

In this work, we propose Dynamic Axial Graph Construction (DAGC) to address the current limitations of ViG-based models. We also introduce GreedyViG, an efficient ViG-based architecture using a hybrid CNN-GNN approach. DAGC is more computationally efficient compared to KNN-based graph construction while maintaining a dynamic set of connections that changes across input images. In Figure 1, we show that our proposed GreedyViG architecture outperforms competing state-of-the-art (SOTA) models across all model sizes in terms of parameters. We summarize our contributions as follows:

1. We propose a new method for designing efficient vision GNNs, Dynamic Axial Graph Construction (DAGC). DAGC is more efficient compared to KNN-based ViGs as DAGC limits the graph connections made within an image to only the most significant ones. Our method is lightweight compared to KNN-based ViGs and more representative than SOTA static graph construction based ViGs.
2. We propose a novel efficient CNN-GNN architecture, GreedyViG, which uses DAGC, conditional positional encoding (CPE) [2], and max-relative graph convolution [20]. We use convolutional layers and grapher layers in all four stages of the proposed architecture to perform local and global processing for each resolution.
3. We conduct comprehensive experiments to underscore the efficacy of the GreedyViG architecture, which beats existing ViG architectures, efficient CNN architectures, and efficient ViT architectures in terms of accuracy and/or parameters and GMACs (number of MACs in billions) on four representative vision tasks: ImageNet image classification [3], COCO object detection [24], COCO instance segmentation [24], and ADE20K semantic segmentation [51]. Specifically our GreedyViG-B model achieves a top-1 accuracy of 83.9% on the ImageNet classification task, 46.3% Average Precision (AP) on the COCO object detection task, and 47.4% mean Intersection over Union ($mIoU$) on the ADE20K semantic segmentation task.

The paper is organized as follows. Section 2 covers related work in the ViG and efficient computer vision architecture space. Section 3 describes the design methodology behind DAGC and the GreedyViG architecture. Section 4 describes experimental setup and results for ImageNet-1k image classification, COCO object detection, COCO instance segmentation, and ADE20K semantic segmentation. Section 5 covers ablation studies on how different design decisions impact performance on ImageNet-1k. Lastly, Section 6 summarizes our main contributions.

2. Related Work

The mainstream network architecture for computer vision has historically been convolutional neural networks (CNN) [9, 12, 17, 19, 35]. In the efficient computer vision space, CNN-based architectures [11, 34, 36, 37] have been even more dominant due to the computationally expensive nature of ViTs [4]. Many works have attempted to address the computational costs associated with self-attention layers [31, 43] and recently hybrid architectures that combine CNNs and ViTs to effectively capture local and global information have been proposed [22, 23, 28, 29, 41].

Traditionally graph neural networks (GNNs) have operated on biological, social, or citation networks [6, 15, 48, 52]. GNNs have also been used for tasks in computer vision such as, point cloud classification and segmentation [18, 45], as well as human action recognition [49]. But, with the introduction of Vision GNN [7], the adoption of GNNs as a general purpose vision backbone has grown with works like [8, 30, 47]. MobileViG [30] introduces a hybrid CNN-GNN architecture to design an efficient computer vision backbone to compete with CNN, ViT, and hybrid architectures. MobileViG accomplishes this through introducing a static graph construction method called Sparse Vision Graph Attention (SVGA) to avoid the computationally expensive nature of ViGs. Despite the efficiencies of MobileViG [30], it does not take full advantage of the global processing possible with GNNs since it only uses graph convolution at the lowest resolution stage of its design. MobileViG [30] also loses representation ability because all images construct the same graph in their proposed static method, decreasing the benefits of using a GNN-based architecture. Thus, to address these limitations, we introduce

a new CNN-GNN architecture, GreedyViG, that takes advantage of graph convolution at higher resolution stages and constructs a graph that changes across input images.

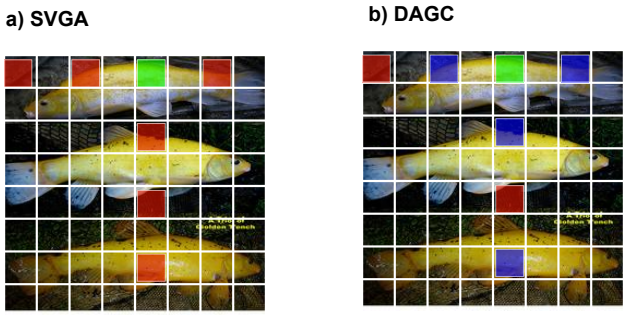

Figure 2. **DAGC and SVGA graph construction.** a) SVGA graph construction for the green patch of an 8×8 image. All red patches will be connected to the green patch regardless of similarity. b) DAGC for the green patch of an 8×8 image. DAGC dynamically constructs a graph along the axes, through applying a mask (the blue patches) to only connect similar patches in terms of Euclidean distance. The red patches will not be connected to the green patch as they are not a part of the mask.

3. Methodology

In this section, we describe the DAGC algorithm and provide details on the GreedyViG architecture design. More precisely, Section 3.1 describes the DAGC algorithm. Section 3.2 explains how we adapt the Grapher module from ViG [7] to create the DAGC block. Section 3.3 describes how we combine the DAGC blocks along with inverted residual blocks [34] to create the GreedyViG architecture.

3.1. Dynamic Axial Graph Construction

We propose Dynamic Axial Graph Construction (DAGC) as an efficient alternative dynamic graph construction method to the computationally expensive KNN graph construction method from Vision GNN [7]. DAGC builds upon SVGA [30], but instead of statically constructing a graph, DAGC constructs a graph that changes across input images. DAGC retains the efficiencies of SVGA through the removal of the KNN computation and input reshaping. It also introduces an efficient graph construction method based on the mean (μ) and standard deviation (σ) of the Euclidean distance between patches in the input image.

In ViG, the KNN computation is required for every input image, since the nearest neighbors of each patch cannot be known ahead of time. This results in a graph with connections throughout the image. Due to the unstructured nature of KNN, ViG [7] contains two reshaping operations. The first to reshape the input image from a 4D tensor to a 3D tensor for graph convolution and the second to reshape the input from 3D back to 4D for the convolutional layers. SVGA [30] eliminates these two reshaping operations and KNN computation through using a static graph where each patch is connected to every K^{th} patch in its row and column as seen in Figure 2a.

DAGC leverages the axial construction of SVGA to retain its efficiencies, while dynamically constructing a more representative graph. To do this, DAGC first obtains an estimate of the μ and σ of the Euclidean distance between nodes through using a subset of nodes. The subset of nodes is obtained by splitting the image into quadrants and comparing the quadrants diagonal to one another as shown in Figure 3 below. Then, the μ and σ can be calculated with those Euclidean distance values. This allows the estimated μ and σ to be computed between two images (the original and the one with its quadrants flipped across the diagonal). This is to decrease the number of comparisons for getting the μ and σ values. The reason we avoid calculating the true μ and σ is that computing them directly would require calculating the Euclidean distance between each individual node and all other nodes in the image. We then consider connections across the row and column as in SVGA to decrease computation, as MobileViG [30] demonstrated that not every patch needs to be considered. If the Euclidean distance between two nodes is less than the difference of the estimated μ and σ, then we connect the two nodes.

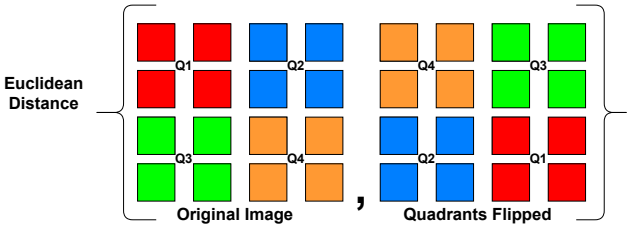

Figure 3. **Euclidean distance calculation** between the original image and the image with its quadrants flipped along the diagonal.

DAGC also enables a variable amount of connections across different images, unlike KNN's fixed K number of connections for all images. This is because in different images, different nodes will have a Euclidean distance between them be less than the difference of the estimated μ and σ. The intuition behind the use of μ and σ is that node pairs that are within one σ of μ are close to one another and should share information. These values are then used to make the connections generated by DAGC as shown in Figure 2b. In Figure 2 we can see that SVGA connects the fish to parts of the image that are not fish, while DAGC only connects the fish to other parts of the image that are fish.

Now that we have the estimated μ and σ within the image, we $roll$ the input image X, mK to the $right$ or $down$ while mK is less than H (height) and W (width) of the image as seen in Algorithm 1. The $roll$ operation is used to compare the patches that are N hops away. In Figure 2b,

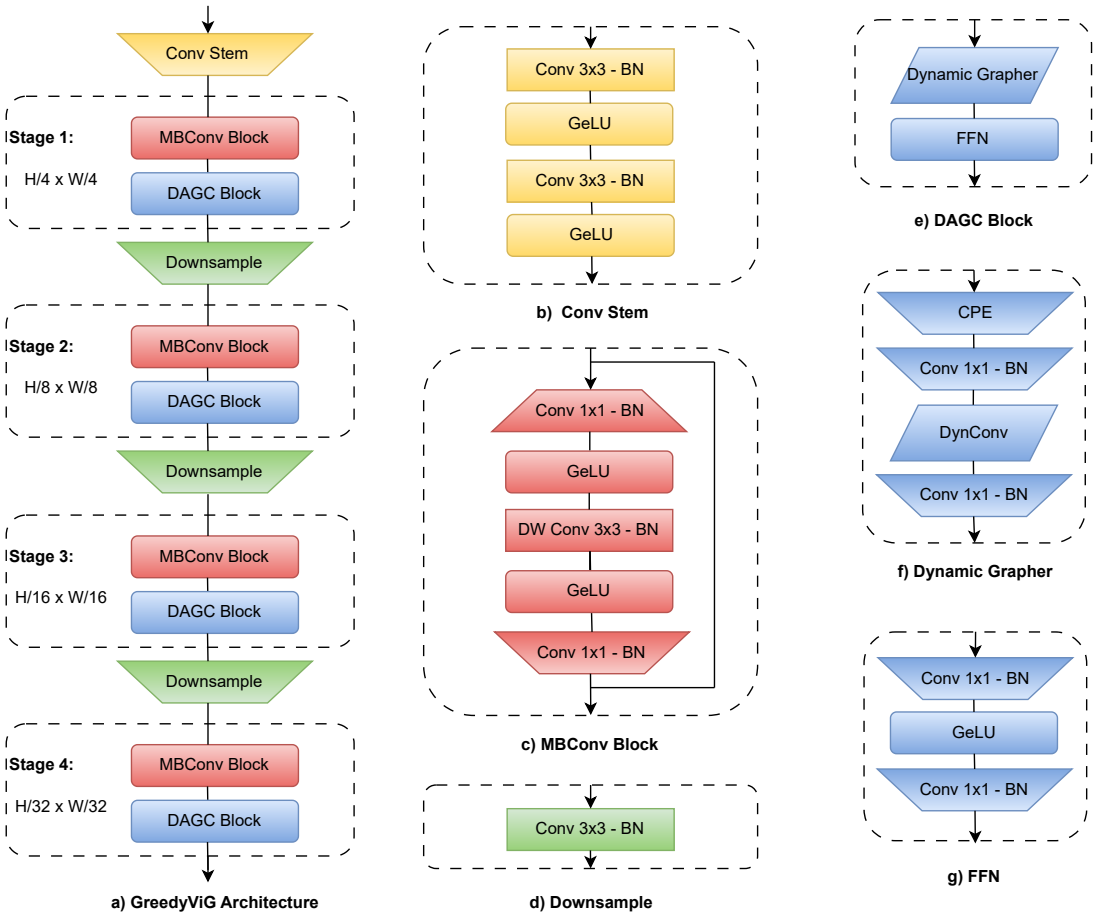

Figure 4. **GreedyViG architecture.** (a) Network architecture showing the stages and blocks. (b) The Conv Stem. (c) MBConv Block. (d) Downsample. (e) DAGC Block. (f) Dynamic Grapher. (g) FFN.

node (5,1) in *x, y* coordinates is compared to nodes (1,1), (3,1), (7,1), (5,3), (5,5), and (5,7) through "rolling" to the next node. After rolling, we compute the Euclidean distance between the input X and the rolled version (X_{rolled}) to determine whether to connect the two points. If the distance is less than $\mu - \sigma$, then the mask is assigned a value of 1, else it is assigned a value of 0. This mask is then multiplied with $X_{rolled} - X$, to mask out max-relative scores between patches not considered connections. This value is denoted as X_{down} and X_{right} in Algorithm 1. Next, the max operation is taken and the result is stored in X_{final}. Lastly, after the rolling, masking, and max-relative operations a final $Conv2d$ is performed.

Through this methodology, DAGC provides a more representative graph construction compared to SVGA [30], as dissimilar patches (i.e., nodes) are not connected. DAGC is also less computationally expensive compared to KNN due to less comparisons being needed for constructing the graph (KNN must compute the nearest neighbors for every patch). DAGC also does not require the reshaping needed for performing graph convolution in KNN-based methods [7]. Thus, DAGC provides representation flexibility like KNN and decreased computational complexity like SVGA.

3.2. DAGC Block

The DAGC block consists of a Dynamic Grapher module followed by a feed-forward network (FFN). The Dynamic Grapher module differs from the Grapher module used in [30] through the use of an updated max-relative graph convolution step called DynConv using Algorithm 1 and conditional positional encoding (CPE) [2]. DynConv dynamically creates a graph that changes across input images, unlike the graph construction used in the graph convolution of SVGA [30]. Given an input feature $X \in \mathbb{R}^{N \times N}$, the updated Dynamic Grapher is expressed as:

$$Y = \sigma(DynConv((X + CPE(X))W_{in}))W_{out} + X \quad (1)$$

where $Y \in \mathbb{R}^{N \times N}$, W_{in} and W_{out} are fully connected

Algorithm 1 DAGC

Given: K, the distance between connections; H, W, the image resolution; X, the input image; $X_{quadrants}$, the quadrants of the input flipped across the diagonals; m, the distance of each roll.

$m \leftarrow 0$
$norm \leftarrow norm(X, X_{quadrants})$ ▷ matrix norm of tensors
$\mu \leftarrow mean(norm)$
$\sigma \leftarrow std(norm)$
while $mK < H$ **do**
  $X_{rolled} \leftarrow roll_{down}(X, mK))$
  $dist \leftarrow norm(X, X_{rolled})$ ▷ get distance value
  if $dist < \mu - \sigma$ **then** ▷ generate mask
    $mask \leftarrow 1$
  else
    $mask \leftarrow 0$
  end if
  $X_{down} \leftarrow mask * (X_{rolled} - X)$ ▷ get features
  $X_{final} \leftarrow max(X_{down}, X_{final})$ ▷ keep max
  $m \leftarrow m + 1$
end while
$m \leftarrow 0$
while $mK < W$ **do**
  $X_{rolled} \leftarrow roll_{right}(X, mK)$
  $dist \leftarrow norm(X, X_{rolled})$
  if $dist < \mu - \sigma$ **then**
    $mask \leftarrow 1$
  else
    $mask \leftarrow 0$
  end if
  $X_{right} \leftarrow mask * (X_{rolled} - X)$
  $X_{final} \leftarrow max(X_{right}, X_{final})$
  $m \leftarrow m + 1$
end while
return $Conv2d(Concat(X, X_{final}))$

layer weights, CPE is a depthwise convolution, and σ is a GeLU activation. The updated Dynamic Grapher module is visually depicted in Figure 4f.

Following the updated Dynamic Grapher, we use the feed-forward network (FFN) module as used in Vision GNN [7] and MobileViG [30], which can be seen in Figure 4g. The FFN module is a two layer MLP expressed as:

$$Z = \sigma(XW_1)W_2 + Y \tag{2}$$

where $Z \in \mathbb{R}^{N \times N}$, W_1 and W_2 are fully connected layer weights, and σ is once again GeLU. We call this combination of the Dynamic Grapher module and FFN the DAGC block, as shown in Figure 4e.

CPE is introduced into the DAGC block to provide the position of the node within the image as this is important to performance [2, 13]. The CPE used in DAGC follows the method of [2], i.e., a depthwise convolution computes the encodings, and then the encodings are added to the input. The addition of CPE adds spatial information into the message passing step of DynConv improving performance.

3.3. GreedyViG Architecture

The GreedyViG architecture shown in Figure 4a is composed of a convolutional stem, and four stages of inverted residual blocks (MBConv) and DAGC blocks each followed by a downsample reducing the resolution to get to the next stage. The stem consists of 3×3 convolutions with the stride equal to 2, each followed by batch normalization (BN) and the GeLU activation function as seen in Figure 4b. The MBConv block is used for local processing at each stage, before the DAGC block performs global processing at each stage. Each MBConv block consists of pointwise convolutions, BN, GeLU, a depth-wise 3×3 convolution, and a residual connection as seen in Figure 4c. The DAGC block is used at each resolution to better learn global object interactions. Between each stage there is a downsample, which consists of a 3×3 convolution with a stride equal to 2 followed by BN as shown in Figure 4d to half the input resolution and expand the channel dimension. Each stage in the GreedyViG architecture is composed of multiple MBConv and DAGC blocks, where the number of repetitions and channel width is changed depending on model size. Within the DAGC blocks used in all GreedyViG model sizes, the distance between connections of nodes before masking is set to K = 8, 4, 2, 1 for stages 1 to 4, respectively. This allows the graph constructed to still be dense in lower resolution stages, as too sparse of a graph can negatively impact accuracy as seen Table 3 in our ablation studies. After the final DAGC block there is a classification head consisting of Average Pooling and an FFN.

4. Experimental Results

We compare GreedyViG to ViG [7], ViHGNN [8], MobileViG [30], and other efficient vision architectures to show that for each model size, GreedyViG has a superior performance on the tasks of image classification, object detection, instance segmentation, and semantic segmentation for similar or less parameters and GMACs.

4.1. Image Classification

We implement the model using PyTorch 1.12.1 [32] and Timm library [46]. We use 16 NVIDIA A100 GPUs to train our models, with an effective batch size of 2048. The models are trained from scratch for 300 epochs on ImageNet-1K [3] with AdamW optimizer [27]. Learning rate is set to $2e^{-3}$ with cosine annealing schedule. We use a standard image resolution, 224 × 224, for both training and testing.

Table 1. **Classification results on ImageNet-1k** for GreedyViG and other state-of-the-art models. Different training seeds result in about 0.1% variation in accuracy for GreedyViG over three runs. Bold entries indicate results obtained for GreedyViG proposed in this paper.

Model	Type	Parameters (M)	GMACs	Epochs	Top-1 Accuracy (%)
ResNet18 [9]	CNN	11.7	1.82	300	69.7
ResNet50 [9]	CNN	25.6	4.1	300	80.4
ConvNext-T [26]	CNN	28.6	7.4	300	82.7
EfficientFormer-L1 [23]	CNN-ViT	12.3	1.3	300	79.2
EfficientFormer-L3 [23]	CNN-ViT	31.3	3.9	300	82.4
EfficientFormer-L7 [23]	CNN-ViT	82.1	10.2	300	83.3
LeViT-192 [5]	CNN-ViT	10.9	0.7	1000	80.0
LeViT-384 [5]	CNN-ViT	39.1	2.4	1000	82.6
EfficientFormerV2-S2 [22]	CNN-ViT	12.6	1.3	300	81.6
EfficientFormerV2-L [22]	CNN-ViT	26.1	2.6	300	83.3
PVT-Small [44]	ViT	24.5	3.8	300	79.8
PVT-Large [44]	ViT	61.4	9.8	300	81.7
DeiT-S [39]	ViT	22.5	4.5	300	81.2
Swin-T [25]	ViT	29.0	4.5	300	81.4
PoolFormer-s12 [50]	Pool	12.0	2.0	300	77.2
PoolFormer-s24 [50]	Pool	21.0	3.6	300	80.3
PoolFormer-s36 [50]	Pool	31.0	5.2	300	81.4
PViHGNN-Ti [8]	GNN	12.3	2.3	300	78.9
PViHGNN-S [8]	GNN	28.5	6.3	300	82.5
PViHGNN-B [8]	GNN	94.4	18.1	300	83.9
PViG-Ti [7]	GNN	10.7	1.7	300	78.2
PViG-S [7]	GNN	27.3	4.6	300	82.1
PViG-B [7]	GNN	92.6	16.8	300	83.7
MobileViG-S [30]	CNN-GNN	7.2	1.0	300	78.2
MobileViG-M [30]	CNN-GNN	14.0	1.5	300	80.6
MobileViG-B [30]	CNN-GNN	26.7	2.8	300	82.6
GreedyViG-S (Ours)	**CNN-GNN**	**12.0**	**1.6**	**300**	**81.1**
GreedyViG-M (Ours)	**CNN-GNN**	**21.9**	**3.2**	**300**	**82.9**
GreedyViG-B (Ours)	**CNN-GNN**	**30.9**	**5.2**	**300**	**83.9**

Similar to DeiT [39], we perform knowledge distillation using RegNetY-16GF [33] with 82.9% top-1 accuracy.

As seen in Table 1, for a similar number of parameters and GMACs, GreedyViG outperforms Pyramid ViG (PViG) [7], Pyramid ViHGNN (PViHGNN) [8], and MobileViG [30] significantly. For example, our smallest model, GreedyViG-S, achieves 81.1% top-1 accuracy on ImageNet-1K with 12.0 M parameters and 1.6 GMACs, which is 2.9% higher top-1 accuracy compared to PViG-Ti [7] and 2.2% higher than PViHGNN-Ti [8] with less GMACs and a similar number of parameters. Our largest model, GreedyViG-B obtains 83.9% top-1 accuracy with only 30.9 M parameters and 5.2 GMACs, which is a 0.2% higher top-1 accuracy compared to PViG-B [7] with a 66.6% decrease in parameters (61.7 M fewer parameters) and a 69% decrease in GMACs (11.6 fewer GMACs) and the same top-1 accuracy as PViHGNN-B [8] with a 67.3% decrease in parameters (63.5 M fewer parameters) and a 71.3% decrease in GMACs (12.9 fewer GMACs).

When compared to other efficient architectures in Table 1, GreedyViG beats SOTA models in accuracy for a similar number of parameters and GMACs. GreedyViG-

Table 2. **Object detection, instance segmentation, and semantic segmentation results** of GreedyViG and other backbones on MS COCO 2017 and ADE20K. (-) denotes unrevealed or unsupported models. Bold entries indicate results obtained using GreedyViG and DAGC proposed in this paper.

Backbone	Parameters (M)	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	$mIoU$
ResNet18 [9]	11.7	34.0	54.0	36.7	31.2	51.0	32.7	32.9
EfficientFormer-L1 [23]	12.3	37.9	60.3	41.0	35.4	57.3	37.3	38.9
EfficientFormerV2-S2 [22]	12.6	43.4	65.4	47.5	39.5	62.4	42.2	42.4
PoolFormer-S12 [50]	12.0	37.3	59.0	40.1	34.6	55.8	36.9	37.2
FastViT-SA12 [41]	10.9	38.9	60.5	42.2	35.9	57.6	38.1	38.0
MobileViG-M [30]	14.0	41.3	62.8	45.1	38.1	60.1	40.8	-
GreedyViG-S (Ours)	**12.0**	**43.2**	**65.2**	**47.3**	**39.8**	**62.2**	**43.2**	**43.2**
ResNet50 [9]	25.5	38.0	58.6	41.4	34.4	55.1	36.7	36.7
EfficientFormer-L3 [23]	31.3	41.4	63.9	44.7	38.1	61.0	40.4	43.5
EfficientFormer-L7 [23]	82.1	42.6	65.1	46.1	39.0	62.2	41.7	45.1
EfficientFormerV2-L [22]	26.1	44.7	66.3	48.8	40.4	63.5	43.2	45.2
PoolFormer-S24 [50]	21.0	40.1	62.2	43.4	37.0	59.1	39.6	40.3
FastViT-SA36 [41]	30.4	43.8	65.1	47.9	39.4	62.0	42.3	42.9
Pyramid ViG-S [7]	27.3	42.6	65.2	46.0	39.4	62.4	41.6	-
Pyramid ViHGNN-S [8]	28.5	43.1	66.0	46.5	39.6	63.0	42.3	-
PVT-Small [44]	24.5	40.4	62.9	43.8	37.8	60.1	40.3	39.8
MobileViG-B [30]	26.7	42.0	64.3	46.0	38.9	61.4	41.6	-
GreedyViG-B (Ours)	**30.9**	**46.3**	**68.4**	**51.3**	**42.1**	**65.5**	**45.4**	**47.4**

S beats PoolFormer-s12 [50] with 3.9% higher top-1 accuracy while having the same number of parameters and 0.4 fewer GMACs. GreedyViG-M achieves 82.9% top-1 accuracy beating ConvNext-T [26] with 0.2% higher top-1 accuracy while having 6.7 M fewer parameters and 4.2 fewer GMACs. Additionally, GreedyViG-B achieves 83.9% top-1 accuracy beating the EfficientFormer [22, 23] family of models for a similar number of parameters.

4.2. Object Detection and Instance Segmentation

We show that GreedyViG generalizes well to downstream tasks by using it as a backbone in the Mask-RCNN framework [10] for object detection and instance segmentation tasks on the MS COCO 2017 dataset [24]. The dataset contains training and validations sets of 118K and 5K images, respectively. We implement the backbone using PyTorch 1.12.1 [32] and Timm library [46]. The model is initialized with ImageNet-1k pretrained weights from 300 epochs of training. We use the AdamW [14, 27] optimizer with an initial learning rate of $2e^{-4}$ and train the model for 12 epochs with a standard resolution (1333×800) following the process of prior work [21–23, 30].

As seen in Table 2, with similar model size GreedyViG outperforms PoolFormer [50], EfficientFormer [23], EfficientFormerV2 [22], MobileViG [30], and PVT [44] in terms of either parameters or improved average precision (AP) on object detection and instance segmentation. The GreedyViG-S model gets 43.2 AP^{box} and 39.8 AP^{mask} on the object detection and instance segmentation tasks outperforming PoolFormer-s12 [50] by 5.9 AP^{box} and 5.2 AP^{mask}. Our GreedyViG-B model achieves 46.3 AP^{box} and 42.1 AP^{mask} outperforming MobileViG-B [30] by 4.3 AP^{box} and 3.2 AP^{mask} and FastViT-SA36 [41] by 2.5 AP^{box} and 2.7 AP^{mask}. The strong performance of GreedyViG on object detection and instance segmentation shows the capability of DAGC and GreedyViG to generalize well to different tasks in computer vision.

4.3. Semantic Segmentation

We further validate the performance of GreedyViG on semantic segmentation using the scene parsing dataset, ADE20k [51]. The dataset contains 20K training images and 2K validation images with 150 semantic categories. Following the methodologies of [22, 23, 41, 50], we build GreedyViG with Semantic FPN [16] as the segmentation decoder. The backbone is initialized with pretrained weights on ImageNet-1K and the model is trained for 40K iterations on 8 NVIDIA RTX 6000 Ada generation GPUs. We follow the process of existing works in segmentation, using the AdamW optimizer, set the learning rate as 2×10^{-4} with a poly decay by the power of 0.9, and set the training resolution to 512×512 [22, 23].

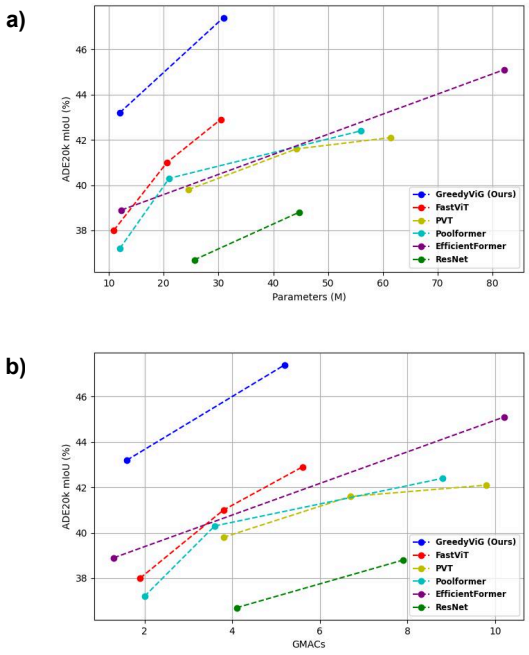

Figure 5. **Comparison of model size and performance (*mIoU* on ADE20K)**. GreedyViG achieves the highest performance on all model sizes compared to other state-of-the-art models. a) shows performance compared to parameters and b) shows performance compared to GMACs.

As shown in Table 2, GreedyViG-S outperforms PoolFormer-S12 [50], FastViT-SA12 [41], and EfficientFormer-L1 [23] by 6.0, 5.2, and 4.3 $mIoU$, respectively. Additionally, GreedyViG-B outperforms PoolFormer-S24 [50], FastViT-SA36 [41], and EfficientFormer-L3 [23] by 7.1, 4.5, and 3.9 $mIoU$, respectively. Figure 5 shows GreedyViG significantly outperforms FastViT [41], PVT [44], Poolformer [50], EfficientFormer [23], and ResNet [9] models with a similar number of parameters and GMACs.

5. Ablation Studies

The ablation studies are conducted on ImageNet-1K [3]. Table 3 reports the ablation study of GreedyViG-B on varying distances of considered graph connections (K) in the DAGC algorithm and how conditional positional encoding affects performance.

Distance between considered nodes for graph construction (K). The distance considered between possible node connections for graph construction can create a sparser graph, but can lead to decreased accuracy as the graph becomes too sparse and does not contain enough connections. We can see this in Table 3, which shows that for K = 16, 8, 4, 2 in stages 1, 2, 3, and 4 that the top-1 accuracy is 0.4% lower than for when K = 8, 4, 2, 1. We also find that using K = 9, 6, 3, 1 leads to a 0.2% decrease in top-1 accuracy compared to K = 8, 4, 2, 1 used in GreedyViG.

Table 3. **Ablation study for GreedyViG-B on ImageNet-1K benchmark** for varying distances between considered node connections (K) and the addition of conditional positional encoding.

K	Params (M)	CPE	Top-1 (%)
K = 16, 8, 4, 2	30.9	Yes	83.5
K = 9, 6, 3, 1	30.9	Yes	83.7
K = 8, 4, 2, 1	30.7	No	83.7
K = 8, 4, 2, 1	**30.9**	**Yes**	**83.9**

Conditional Positional Encoding (CPE). The encoding of positions within GreedyViG also boosts performance for relatively few parameters. When removing CPE from GreedyViG-B, we see a drop in accuracy of 0.2% with only a 0.2 M decrease in parameters, showing that CPE is beneficial in the GreedyViG architecture.

Further ablation studies on the effects of removing graph convolutions at higher resolution stages, static versus dynamic graph construction, and how graph construction impacts latency are included in the supplementary material.

6. Conclusion

In this work, we have proposed a new method for designing efficient vision GNNs, Dynamic Axial Graph Construction (DAGC). DAGC is more efficient compared to KNN-based ViGs and more representative compared to SVGA. This is because DAGC uses an axial graph construction method to limit graph connections, and it does not have a fixed number of graph connections allowing for a variable number of connections based on the input image. Compared to past axial graph construction methods, DAGC limits the graph connections made within an image to only the significant connections thereby constructing a more representative graph. Additionally, we have proposed a novel CNN-GNN architecture, GreedyViG, which uses DAGC. GreedyViG outperforms existing ViG, CNN, and ViT models on multiple representative vision tasks, namely image classification, object detection, instance segmentation, and semantic segmentation. GreedyViG shows that ViG-based models can be legitimate competitors to ViT-based models through DAGC and by performing local and global processing at each resolution through a hybrid CNN-GNN architecture.

7. Acknowledgements

This work is supported in part by the NSF grant CNS 2007284, and in part by the iMAGiNE Consortium (https://imagine.utexas.edu/).

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. 1

[2] Xiangxiang Chu, Zhi Tian, Bo Zhang, Xinlong Wang, Xiaolin Wei, Huaxia Xia, and Chunhua Shen. Conditional positional encodings for vision transformers. *arXiv preprint arXiv:2102.10882*, 2021. 2, 4, 5

[3] 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, 2009. 1, 2, 5, 8

[4] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 2

[5] Benjamin Graham, Alaaeldin El-Nouby, Hugo Touvron, Pierre Stock, Armand Joulin, Hervé Jégou, and Matthijs Douze. Levit: a vision transformer in convnet's clothing for faster inference. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 12259–12269, 2021. 6

[6] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 2

[7] Kai Han, Yunhe Wang, Jianyuan Guo, Yehui Tang, and Enhua Wu. Vision gnn: An image is worth graph of nodes. *arXiv preprint arXiv:2206.00272*, 2022. 1, 2, 3, 4, 5, 6, 7

[8] Yan Han, Peihao Wang, Souvik Kundu, Ying Ding, and Zhangyang Wang. Vision hgnn: An image is more than a graph of nodes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 19878–19888, 2023. 1, 2, 5, 6, 7

[9] 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. 1, 2, 6, 7, 8

[10] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017. 7

[11] 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. 2

[12] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017. 2

[13] Md Amirul Islam, Sen Jia, and Neil DB Bruce. How much position information do convolutional neural networks encode? In *International Conference on Learning Representations*, 2019. 5

[14] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 7

[15] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 2

[16] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6399–6408, 2019. 7

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *NeurIPS*, 2012. 1, 2

[18] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4558–4567, 2018. 2

[19] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1, 2

[20] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9267–9276, 2019. 2

[21] Jiashi Li, Xin Xia, Wei Li, Huixia Li, Xing Wang, Xuefeng Xiao, Rui Wang, Min Zheng, and Xin Pan. Next-vit: Next generation vision transformer for efficient deployment in realistic industrial scenarios. *arXiv preprint arXiv:2207.05501*, 2022. 7

[22] Yanyu Li, Ju Hu, Yang Wen, Georgios Evangelidis, Kamyar Salahi, Yanzhi Wang, Sergey Tulyakov, and Jian Ren. Rethinking vision transformers for mobilenet size and speed. *arXiv preprint arXiv:2212.08059*, 2022. 2, 6, 7

[23] Yanyu Li, Geng Yuan, Yang Wen, Eric Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *arXiv preprint arXiv:2206.01191*, 2022. 2, 6, 7, 8

[24] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014. 1, 2, 7

[25] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 10012–10022, 2021. 1, 6

[26] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11976–11986, 2022. 1, 6, 7

[27] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 5, 7

[28] Sachin Mehta and Mohammad Rastegari. Mobilevit: lightweight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 2

[29] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 2

[30] Mustafa Munir, William Avery, and Radu Marculescu. Mobilevig: Graph-based sparse attention for mobile vision applications. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2210–2218, 2023. 1, 2, 3, 4, 5, 6, 7

[31] Zizheng Pan, Jianfei Cai, and Bohan Zhuang. Fast vision transformers with hilo attention. In *NeurIPS*, 2022. 2

[32] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 5, 7

[33] Ilija Radosavovic, Raj Prateek Kosaraju, Ross Girshick, Kaiming He, and Piotr Dollár. Designing network design spaces. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10428–10436, 2020. 6

[34] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. 2, 3

[35] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[36] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 2

[37] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 2

[38] Ilya O Tolstikhin, Neil Houlsby, Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Thomas Unterthiner, Jessica Yung, Andreas Steiner, Daniel Keysers, Jakob Uszkoreit, et al. Mlp-mixer: An all-mlp architecture for vision. *Advances in neural information processing systems*, 34:24261–24272, 2021. 1

[39] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *International conference on machine learning*, pages 10347–10357. PMLR, 2021. 6

[40] Hugo Touvron, Piotr Bojanowski, Mathilde Caron, Matthieu Cord, Alaaeldin El-Nouby, Edouard Grave, Gautier Izacard, Armand Joulin, Gabriel Synnaeve, Jakob Verbeek, et al. Resmlp: Feedforward networks for image classification with data-efficient training. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 1

[41] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. Fastvit: A fast hybrid vision transformer using structural reparameterization. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2, 7, 8

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1

[43] Sinong Wang, Belinda Z. Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity, 2020. 2

[44] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 568–578, 2021. 6, 7, 8

[45] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *ACM Transactions on Graphics (tog)*, 38(5):1–12, 2019. 2

[46] Ross Wightman. PyTorch Image Models. `https://github.com/rwightman/pytorch-image-models`, 2019. 5, 7

[47] JiaFu Wu, Jian Li, Jiangning Zhang, Boshen Zhang, Mingmin Chi, Yabiao Wang, and Chengjie Wang. Pvg: Progressive vision graph for vision recognition. In *Proceedings of the 31st ACM International Conference on Multimedia*, page 2477–2486, New York, NY, USA, 2023. Association for Computing Machinery. 2

[48] Yongji Wu, Defu Lian, Yiheng Xu, Le Wu, and Enhong Chen. Graph convolutional networks with markov random field reasoning for social spammer detection. In *Proceedings of the AAAI conference on artificial intelligence*, pages 1054–1061, 2020. 2

[49] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI conference on artificial intelligence*, 2018. 2

[50] Weihao Yu, Mi Luo, Pan Zhou, Chenyang Si, Yichen Zhou, Xinchao Wang, Jiashi Feng, and Shuicheng Yan. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10819–10829, 2022. 6, 7, 8

[51] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 1, 2, 7

[52] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI open*, 1:57–81, 2020. 2