Multi-Scale High-Resolution Logarithmic Grapher Module for Efficient Vision GNNs

Mustafa Munir
The University of Texas at Austin
mmunir@utexas.edu

Alex Zhang
The University of Texas at Austin
alex.zhang@utexas.edu

Radu Marculescu
The University of Texas at Austin
radum@utexas.edu

Abstract

Vision graph neural networks (ViG) have demonstrated promise in vision tasks as a competitive alternative to conventional convolutional neural nets (CNN) and transformers (ViTs); however, common graph construction methods, such as k-nearest neighbor (KNN), can be expensive on larger images. While methods such as Sparse Vision Graph Attention (SVGA) have shown promise, SVGA's fixed step scale can lead to over-squashing and missing multiple connections to gain the same information that could be gained from a long-range link. Through this observation, we propose a new graph construction method, Logarithmic Scalable Graph Construction (LSGC) to enhance performance by limiting the number of long-range links. To this end, we propose LogViG, a novel hybrid CNN-GNN model that utilizes LSGC. Furthermore, inspired by the successes of multi-scale and high-resolution architectures, we introduce and apply a high-resolution branch and fuse features between our high-resolution and low-resolution branches for a multi-scale high-resolution Vision GNN network. Extensive experiments show that LogViG beats existing ViG, CNN, and ViT architectures in terms of accuracy, GMACs, and parameters on image classification and semantic segmentation tasks. Our smallest model, Ti-LogViG, achieves an average top-1 accuracy on ImageNet-1K of 79.9% with a standard deviation of $\pm$ 0.2%, 1.7% higher average accuracy than Vision GNN with a 24.3% reduction in parameters and 35.3% reduction in GMACs. Our work shows that leveraging long-range links in graph construction for ViGs through our proposed LSGC can exceed the performance of current state-of-the-art ViGs.

1 Introduction

The meteoric rise of deep learning over the past decade has resulted in numerous successes in computer vision. From the development of AlexNet [1] and convolutional neural networks (CNNs) [2–5] to a new generation of vision Transformers (ViTs) [6–8], neural networks have demonstrated their effectiveness in computer vision across the board. In the same vein, CNNs and ViTs have also shown competitive performance toward dense prediction tasks, such as semantic segmentation [9, 10]. More recently, Graph neural networks (GNNs), specifically Vision GNNs (ViGs) have emerged as competitive alternatives to current CNN and ViG models in computer vision [11–13]. Instead of a sliding window over a grid of pixels as in CNNs or a sequence of patches in ViTs, ViGs represent an image as a network of patches linked by content rather than spatial position [11]. Through these patches, a ViG can identify objects within an image by relating each patch with its neighbor. For example, if one patch contains features that are associated with a tire and another patch, connected with the first, contains features that represent handlebars, then a ViG is capable of generalizing the

M. Munir et al., Multi-Scale High-Resolution Logarithmic Grapher Module for Efficient Vision GNNs. *Proceedings of the Third Learning on Graphs Conference (LoG 2024)*, PMLR 269, Virtual Event, November 26–29, 2024.

object as a bicycle. Even if two patches are spatially distant, ViGs can exploit long-range links to form the shortest path between spatially distant patches.

Despite promising results, ViGs are computationally complex, especially during network generation [12]. A common method is to use K nearest neighbors (KNN) and generate a network by selecting a patch and connecting each patch to the K most similar patches [11]. Yet, as mentioned earlier, this method is computationally inefficient, especially for high-resolution images as the time needed to generate a graph can grow exponentially depending on the size of the image. An alternative is a static, structured grapher (SVGA) [12] which grows linearly horizontally and vertically across an image. While this significantly simplifies the process, this method suffers from over-squashing where too much information is being assimilated into a single vector. This becomes more apparent as the resolution increases, causing the number of connected patches to grow quadratically. To address this [14] proposed a fixed number of connections regardless of input resolution, but this can cause ViGs to lose their global context in high-resolution images.

To address these issues, we propose Logarithmic Scalable Graph Construction (LSGC) as an efficient alternative to KNN and SVGA-style ViGs. LSGC exploits logarithmic growth to create networks that scale with image size, simultaneously avoiding over-squashing and reducing computational complexity. We also deploy LSGC in a hybrid ViG-based CNN-GNN architecture, LogViG, and utilize a high-resolution shortcut [15] to inject high-resolution features at later stages into the model. We summarize our contributions as follows:

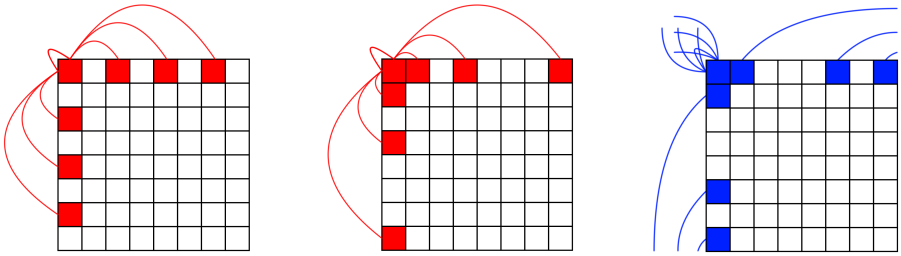
- We propose a novel approach to graph construction designed for efficient Vision GNNs. Instead of scaling the graph linearly as in SVGA, we scale the graph logarithmically. Logarithmic Scalable Graph Construction (LSGC) improves on SVGA in several ways. First, it generates fewer connections for high-resolution images to help mitigate over-squashing. Second, it helps preserve locality by prioritizing connections closer to a patch without sacrificing the ability to make long-range connections.
- We introduce a novel CNN-GNN architecture, LogViG, for image classification and semantic segmentation that uses LSGC and a high-resolution skip connection. We utilize convolutional and grapher layers in all four stages, performing local and global processing at each stage.
- We demonstrate that LoGViG outperforms traditional CNN, ViT, and ViG architectures on vision tasks and that LSGC broadly outperforms SVGA in image classification and semantic segmentation.

The paper is arranged as follows. Section 2 covers related work in the ViG and efficient computer vision architecture space. Section 3 describes the design methodology behind LSGC and the LogViG architecture. Section 4 describes the experimental setup and results for ImageNet-1k [16] image classification and ADE20K [17] 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

Current network architectures for computer vision commonly utilize convolution neural networks (CNNs) [2–5] and vision Transformers (ViTs) [6–8]. On dense prediction tasks, ViTs tend to outperform CNNs [18]; however, are computationally complex compared to CNNs, resulting in higher compute times and latency [19, 20]. In addition, ViT performance degrades on high-resolution images [18]. Conversely, CNNs lack a global receptive field and cannot capture global features when compared with ViTs [18]. In both cases, CNNs and ViTs are limited in their ability to represent image data, restricted to a grid of pixels or a sequence of patches respectively [11]. While CNNs tend to be more efficient than ViTs [20], some headway has been made to make ViTs competitive [21]. Still, pure CNN and ViT models suffer from the drawbacks discussed, opening the door for alternative architectures.

Vision GNNS are a proposed alternative to CNN and ViT-based architectures. Traditional ViG networks represent images by computing the k nearest-neighbors (KNN) for every patch in an image [11], attending to similar patches across an entire image. The result is a network of short and long-range connections across the image. By representing an image as a network of patches, it bypasses the inflexibility issues of CNNs and ViTs. Additionally, the mixture of short and long-range connections between patches allows ViGs to capture both local and global features, a trait lacking in CNNs.



(a) SVGA Grapher **(b)** LSGC Forward Pass **(c)** LSGC Backward Pass

Figure 1: a) SVGA graph attention for the top left pixel of an 8×8 image b) LSGC forward graph attention pass for the top left pixel of an 8×8 image and an expansion rate $K = 2$. As depicted, LSGC grows logarithmically rather than linearly. c) LSGC backward graph attention pass for the top left pixel of an 8×8 image and expansion rate $K = 2$. As shown, the LSGC wraps around the image to create connections

Historically, graph neural networks (GNNs) have been used for biological, language, and social data [22–25]. GNNs have also been used for vision tasks such as object localization, detection, and classification [26–28]. After the introduction of Vision GNN [11] (ViGs), GNN-based networks have grown dramatically [12, 29, 30]. Yet, despite significant progress, graph construction remains computationally expensive [12]. While static grapher methods such as Sparse Vision Graph Attention (SVGA) help mitigate the computational complexity issue of ViGs, they fail to scale with high-resolution images and introduce over-squashing. To tackle this issue, we introduce Logarthimic Scalable Graph Construction (LSGC), an efficient graph construction method capable of scaling with high-resolution images, and LogViG, a novel CNN-GNN hybrid network utilizing LSGC that outperforms competing state-of-the-art (SOTA) architectures.

3 Methodology

3.1 Logarthimic Scalable Graph Construction

We propose Logarthmic Scalable Graph Construction (LSGC) as an alternative efficient graph construction method to KNN graph attention from Vision GNN (ViG) [11]. We expand upon SVGA [12] by scaling the graph logarithmically based on the bit-length of an image rather than statically. In this way, the grapher is efficient on large images by constructing fewer links than SVGA while still creating global links over the entire image. In addition, these fewer, but global, connections avoid over-squashing [31] and redundancy by limiting the number of links per image patch.

For ViG architectures, the k-Nearest-Neighbors (KNN) computation is required for every input image, since one cannot know the nearest neighbors of every pixel on an image. This results in a graph with connections scattered over the image. Due to the nature of KNN, ViG contains two reshaping operations [11]. The first reshapes the input image from a 4D tensor to a 3D tensor for graph convolution and the second reshapes the input from 3D back to 4D for convolutional layers. SVGA [12] eliminates these reshaping operations utilizing a static graph where each patch is connected to every K^{th} patch in its row and column as in Figure 1a. A follow-up work to MobileViG [12] proposes MGC [14] which utilizes a fixed number of possible connections per token regardless of input size.

LSGC replaces the static graph structure of SVGA or the fixed number of connections of MGC and improves upon them with a logarithmic structure that scales based on the input image size. To do this, we first obtain the bit-depth from H and W, the height and width of the input image in pixels. We define bit-depth as the number of bits required to represent the dimensions in binary. For example, an 8×8 input results in a bit-depth of four for both width and height. We reason that this will ensure a global connection can be established for larger images because any pixel will be able to at least reach across an image to establish a long-distance connection. After calculating the bit-depth, we store these values as h and w as denoted in Algorithm 1.

Table 1: Shortest Average Path on Various Resolutions and Network Structures.

Resolution	Lattice	LSGC	SVGA
56 X 56	37.333	4.359	2.895
28 X 28	18.667	3.719	2.794
14 X 14	9.333	3.303	2.605
7 X 7	4.667	2.334	1.750

Next, we implement graph construction using a series of $expand$ operations which we denote as $expand_{forward}$ and $expand_{backward}$ as described in Algorithm 1. Using the bit-depth h and w as our scaling limit, and setting the expansion rate K, we first $expand_{forward}$ in the downwards and right directions (denoted as $expand_{forwardH}$ and $expand_{forwardW}$) from a pixel and create connections every $2^n - 1 \; \forall \; 1 \leq n \leq h, w$ pixels as shown for an 8×8 image in Figure 1b. Similarly, we then $expand_{backward}$ in the upwards and left directions (denoted as $expand_{backwardH}$ and $expand_{backwardW}$) opposite of $expand_{forward}$. If there is no room to expand, as our example in Figure 1c demonstrates, we simply wrap the image around such that we expand to the other side of the image. Then, after every directional expansion, we perform max-relative graph convolution (MRConv). To do this, we compute the element-wise max operation over the difference between the original image X and the expanded version computed by the $expand$ operation and store the result in X_j as shown in Algorithm 1. Finally, we concatenate X_j with the original image and apply a $Conv2d$ over the entire matrix. In this way, we achieve reduced computational complexity for graph construction compared to SVGA and KNN whilst still establishing global connections throughout the image. The effect of expansion is demonstrated in Table 1. While a normal square lattice has a significantly larger shortest path length across the image, LSGC allows for links to be established across an entire image with similar effectiveness to SVGA. For example, at the highest resolution of 56×56, a regular square lattice has an average shortest path length of 37.333. After applying LSGC or SVGA, the shortest path length is reduced to 4.359 and 2.895 respectively as shown in Table 1. The reduction of shortest path length indicates that longer-range links are being established, permitting global data over the entire image to be seen and processed. We note that SVGA has a shorter average path length than LSGC due to making significantly more connections, which leads to the over-squashing [31] effect SVGA suffers from.

Algorithm 1 LSGC

Require: H, W, the image resolution; X, the input image; K the expansion rate
  $h \leftarrow bitdepth(H)$ ▷ Calculate bit depth
  $w \leftarrow bitdepth(W)$
  $X_j \leftarrow 0$
  for $i = 1, 2, \dots \; h$ **do**
    $X_c \leftarrow X - expand_{forwardH}(X, K^i - 1)$ ▷ Get downwards relative features
    $X_j \leftarrow max(X_c, X_j)$ ▷ Keep max relative features
  end for
  for $i = 1, 2, \dots \; w$ **do**
    $X_c \leftarrow X - expand_{forwardW}(X, K^i - 1)$
    $X_j \leftarrow max(X_c, X_j)$
  end for
  for $i = 1, 2, \dots \; h$ **do**
    $X_c \leftarrow X - expand_{backwardH}(X, K^i - 1)$
    $X_j \leftarrow max(X_c, X_j)$
  end for
  for $i = 1, 2, \dots \; w$ **do**
    $X_c \leftarrow X - expand_{backwardW}(X, K^i - 1)$
    $X_j \leftarrow max(X_c, X_j)$
  end for
  return $Conv2d(Concat(X, X_j)$

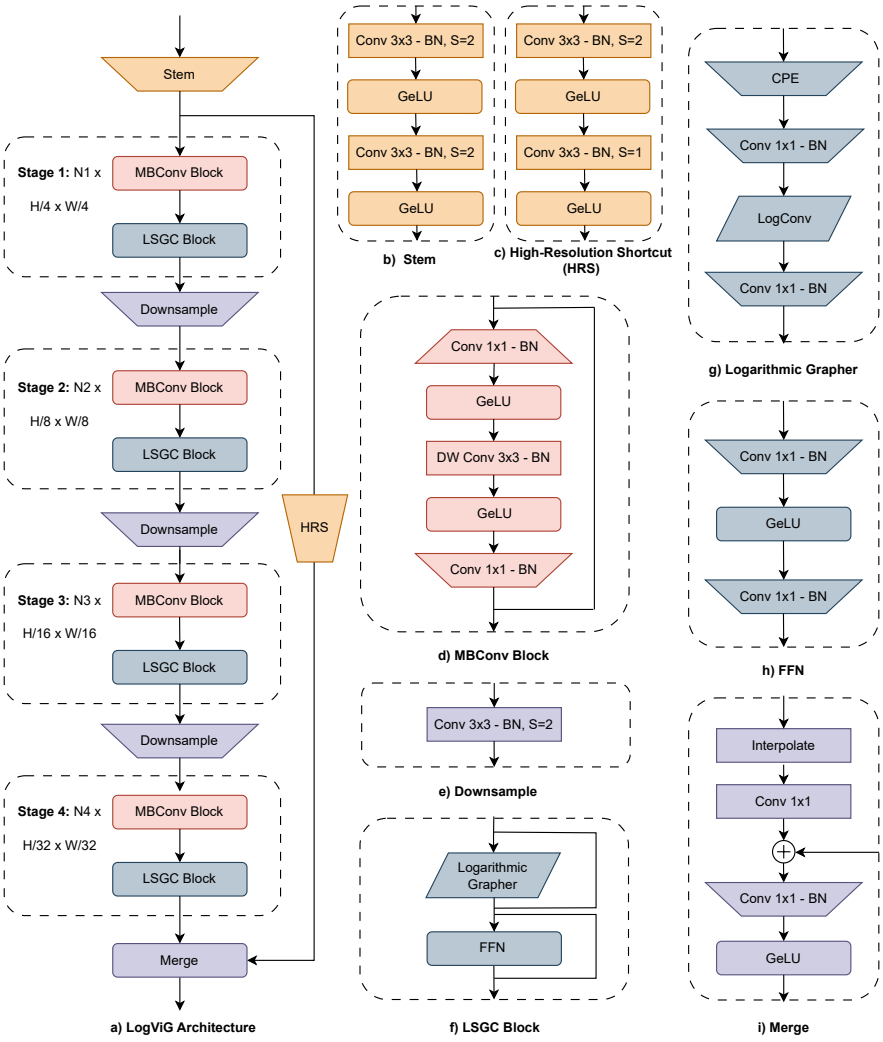


Figure 2: LogViG Architecture. (a) Network architecture showing the stages and blocks. (b) Stem. (c) High-Resolution Shortcut (HRS). (d) MBConv Block. (e)Downsample. (f) LSGC Block. (g) Logarithmic Grapher. (h) FFN. (i) Merge.

3.2 High-Resolution Shortcut

Networks typically downsample the feature maps to operate on lower-resolution representations due to the associated computational complexity (increased GMACs) with operating on high-resolution images [15, 32]. The issue with such networks is that they lack cross-resolution interaction, which can lead to worse performance. Inspired by prior work such as HRNet [32], Lite-HRNet [33], and HRViT [15] we propose to bring multi-scale high-resolution interactions to Vision GNNs.

To enable multi-scale feature interaction we use a High-Resolution Shortcut (HRS), which allows us to merge our higher resolution features with our lower resolution features. Our HRS consists of two 3×3 convolutions, one with a $stride = 2$ and the other with a $stride = 1$, each followed by batch normalization (BN) and the GeLU activation function as shown in Figure 2c. We then merge the features of our high-resolution branch with our low-resolution branch. To merge the features, first we upsample our low-resolution features through bilinear interpolation. Next, we perform a pointwise convolution to match the channel dimensions. Lastly, we sum our features, pass them into another pointwise convolution, followed by BN and GeLU.

3.3 Going Deeper Versus Going Wider

Deeper networks have been shown to better learn hierarchical features, which can improve accuracy on complex and large datasets [34]. To optimize our proposed LogViG architecture we studied whether our network performance is improved with a deeper and narrower network or a shallower and wider network. Namely, we compared our Ti-LogViG architecture shown in Table 2 with the Wide Ti-LogViG architecture. For our wider variation we cut the depth by 33% and increased the channel width by 50% for stages 1 to 3. For stage 4 we increased the channel width to 384 from 224 to get a similar number of parameters to our Ti-LogViG so we could fairly compare the deeper and wider network. The results in Table 5 show our comparison of both networks and we find the deeper LogViG-Ti achieves a 0.4% accuracy increase with only a 0.1 M increase in parameters.

3.4 Network Architecture

The overall LogViG architecture is shown in Figure 2a. First, we pass the image through the convolutional stem. The stem takes the input images and downsamples it $4\times$ using convolutions of $stride = 2$. The output of the stem is fed into two branches: The low-resolution branch and the high-resolution branch. In the low-resolution branch, the stem output is passed through four stages. Each stage is made up of two blocks: the MBConv Block and LSGC Block as described in Figure 2a. Between each stage are additional convolution-based downsampling steps. The high-resolution branch is made up of a single High-Resolution Shortcut block as shown in 2a. The output of the low-resolution branch is interpolated and channel adjusted before being added to the high-resolution branch as demonstrated in Figure 2i. Lastly, the merge block output is put through an average pooling step followed by a feed-forward network, producing the predicted class of the input image.

Table 2: Architecture details of LogViG showing configuration of the stem, stages, and classification head. C represents the channel dimensions.

Stage	Ti-LogViG	S-LogViG	B-LogViG	Wide Ti-LogViG
Stem	Conv $\times 2$	Conv $\times 2$	Conv $\times 2$	Conv $\times 2$
Stage 1	$MBConv \times 3$ $LSGC \times 3$ $C = 32$	$MBConv \times 5$ $LSGC \times 5$ $C = 32$	$MBConv \times 5$ $LSGC \times 5$ $C = 48$	$MBConv \times 1$ $LSGC \times 1$ $C = 48$
Stage 2	$MBConv \times 3$ $LSGC \times 3$ $C = 64$	$MBConv \times 5$ $LSGC \times 5$ $C = 64$	$MBConv \times 5$ $LSGC \times 5$ $C = 96$	$MBConv \times 1$ $LSGC \times 1$ $C = 96$
Stage 3	$MBConv \times 9$ $LSGC \times 3$ $C = 128$	$MBConv \times 15$ $LSGC \times 5$ $C = 128$	$MBConv \times 15$ $LSGC \times 5$ $C = 192$	$MBConv \times 3$ $LSGC \times 1$ $C = 192$
Stage 4	$MBConv \times 3$ $LSGC \times 3$ $C = 224$	$MBConv \times 5$ $LSGC \times 5$ $C = 256$	$MBConv \times 5$ $LSGC \times 5$ $C = 384$	$MBConv \times 1$ $LSGC \times 1$ $C = 384$
Head	Pooling & MLP	Pooling & MLP	Pooling & MLP	Pooling & MLP

The network architecture details for Ti-LogViG, S-LogViG, and B-LogViG are provided in Table 2. We also provide the details of Wide Ti-LogViG, which we used to determine the improvement in performance we would gain by going deeper versus wider in our network. We report the configuration of the stem, stages, and classification head. For each stage we also report the channel dimensions and the number of MBConv and LSGC blocks used.

4 Experimental Results

We compare LogViG to ViG[11] and MobileViG [12] to show its superior performance in terms of image classification accuracy on ImageNet-1k[16] in Table 3 for all model sizes. We also demonstrate superior performance in terms of semantic segmentation on ADE20K [17] for all model sizes compared in Table 4. For each SOTA model we compare to **LogViG obtains superior performance for similar or less parameters and GMACs**.

Table 3: Classification results on ImageNet-1k for LogViG and other state-of-the-art models. Bold entries show best results for competing models for each column. Gray highlighted entries indicate results obtained for LogViG proposed in this paper. The Top-1 accuracy results for LogViG models are averaged over four experiments and show the mean ± standard deviation.

Model	Type	Params (M)	GMACs	Epochs	Acc (%)
ResNet18 [3]	CNN	11.7	1.82	300	69.7
ResNet50 [3]	CNN	25.6	4.1	300	80.4
ConvNext-T [35]	CNN	28.6	7.4	300	82.7
EfficientFormer-L1 [36]	CNN-ViT	12.3	1.3	300	79.2
EfficientFormer-L3 [36]	CNN-ViT	31.3	3.9	300	82.4
EfficientFormer-L7 [36]	CNN-ViT	82.1	10.2	300	83.3
LeViT-192 [37]	CNN-ViT	10.9	**0.7**	1000	80.0
LeViT-384 [37]	CNN-ViT	39.1	2.4	1000	82.6
HRViT-b1 [15]	ViT	19.7	2.7	300	80.5
HRViT-b2 [15]	ViT	32.5	5.1	300	82.3
HRViT-b3 [15]	ViT	37.9	5.7	300	82.8
PVT-Small [38]	ViT	24.5	3.8	300	79.8
PVT-Large [38]	ViT	61.4	9.8	300	81.7
DeiT-S [7]	ViT	22.5	4.5	300	81.2
Swin-T [39]	ViT	29.0	4.5	300	81.4
PoolFormer-S12 [40]	Pool	12.0	2.0	300	77.2
PoolFormer-S24 [40]	Pool	21.0	3.6	300	80.3
PoolFormer-S36 [40]	Pool	31.0	5.2	300	81.4
PViHGNN-Ti [13]	GNN	12.3	2.3	300	78.9
PViHGNN-S [13]	GNN	28.5	6.3	300	82.5
PViHGNN-B [13]	GNN	94.4	18.1	300	**83.9**
PViG-Ti [11]	GNN	10.7	1.7	300	78.2
PViG-S [11]	GNN	27.3	4.6	300	82.1
PViG-B [11]	GNN	92.6	16.8	300	83.7
MobileViG-S [12]	CNN-GNN	**7.2**	1.0	300	78.2
MobileViG-M [12]	CNN-GNN	14.0	1.5	300	80.6
MobileViG-B [12]	CNN-GNN	26.7	2.8	300	82.6
Ti-LogViG (Ours)	CNN-GNN	8.1	1.1	300	79.9 ± 0.2
S-LogViG (Ours)	CNN-GNN	13.9	1.9	300	81.5 ± 0.1
B-LogViG (Ours)	CNN-GNN	30.5	4.6	300	83.6 ± 0.1

4.1 Image Classification

We implement the model using PyTorch 1.12 [41] and Timm library [42]. The models are trained from scratch for 300 epochs on ImageNet1K [16] with AdamW optimizer [43]. We set the learning rate to $2e^{-3}$ with a cosine annealing schedule. We use a standard image resolution of 224 × 224, for both training and testing. Following prior work [7, 12, 14, 36, 44, 45], we perform knowledge distillation using RegNetY-16GF [46] with 82.9% top-1 accuracy. For data augmentation we use RandAugment [47], Mixup[48], Cutmix[49] random erasing [50], and repeated augment [51].

As seen in Table 3, for a similar number of parameters and GMACs, LogViG outperforms high-resolution architectures such as HRViT [15], as well as GNN architectures such as Pyramid ViG (PViG) [11], Pyramid ViHGNN (PViHGNN) [13], and MobileViG [12] by a significant margin. For example, our S-LogViG, achieves 81.5% top-1 accuracy on ImageNet-1K with 13.9 million (M) parameters and 1.9 GMACs compared to HRViT-b2 [15] with 80.5% top-1 accuracy at 19.7 M parameters and 2.7 GMACs. Our largest model B-LogViG obtains 83.6% top-1 accuracy with

only 30.5 M parameters and 4.6 GMACs. Meanwhile, HRViT-b3 [15] obtains only 82.8%, a 0.8% decrease in accuracy, at a higher cost of 37.9 M parameters and 5.7 GMACs.

Compared to similar GNN architectures, our smallest model, Ti-LogViG outperforms PViHGNN-Ti [13] and PViG-Ti[11] in accuracy with less parameters and GMACs. Ti-LogViG obtains 79.9% top-1 accuracy at 8.1 M parameters and 1.1 GMACs. Meanwhile, PViHGNN-Ti [13] only achieves 78.9% top-1 accuracy at 12.3 M parameters and 2.3 GMACs, a 1.0% difference in accuracy for 4.3 M more parameters and 1.1 more GMACs. Additionally, PViG-Ti [11] reaches 78.2% top-1 accuracy at 10.7 M parameters and 1.7 GMACs, a decrease of 1.7% for higher parameters and GMACs.

When juxtaposed to other architectures in Table 3, LogViG beats SOTA models in accuracy for a similar number of parameters and GMACs. Ti-LogViG beats PoolFormer-S12 [40] with 2.7% higher top-1 accuracy and 3.9 M fewer parameters and 0.9 fewer GMACs. S-LogViG beats DeiT [7] with 0.3% higher top-1 accuracy while having 8.6 M fewer parameters and 2.6 fewer GMACs. Additionally, B-LogViG beats the EfficientFormer [36] family of models with significantly fewer parameters.

4.2 Semantic Segmentation

We further compare the performance of LogViG on semantic segmentation using the scene parsing dataset, ADE20k [17]. The dataset contains 20K training images and 2K validation images with 150 semantic categories. We build LogViG with semantic FPN [52] as the segmentation decoder, following the methodologies of [36, 40, 44, 45, 53]. 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. Following the process of prior works in segmentation, we use 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 [12, 36, 44, 45].

As shown in Table 4, S-LogViG outperforms PoolFormer-S12 [40], FastViT-SA12 [53], EfficientFormer-L1 [36], and MobileViG-M by 6.9, 6.1, 5.2, and 2.3 $mIoU$, respectively. Additionally, B-LogViG outperforms FastViT-SA36 [53] by 3.9 $mIoU$ with only 0.1 M more parameters. B-LogViG also outperforms EfficientFormer-L3 [36] by 3.3 $mIoU$ with 0.8 M fewer parameters.

Table 4: Semantic segmentation results of LogViG and other backbones on ADE20K. Bold entries indicate results obtained using LogViG and LSGC proposed in this paper. The * on MobileViG-M indicates the authors do not report semantic segmentation results so we trained the model and report the mIoU we obtained. For all other models that are not our LogViG we report the mIoU's reported in their original papers. The mIoU results for LogViG models are averaged over four experiments and show the mean $\pm$ standard deviation.

Backbone	Parameters (M)	$mIoU$
ResNet18 [3]	11.7	32.9
PVT-Tiny [38]	13.2	35.7
EfficientFormer-L1 [36]	12.3	38.9
PoolFormer-S12 [40]	12.0	37.2
FastViT-SA12 [53]	10.9	38.0
MobileViG-M* [12]	14.0	41.8
S-LogViG (Ours)	**13.9**	**44.1 $\pm$ 0.6**
ResNet50 [3]	25.6	36.7
PVT-Small [38]	24.5	39.8
PVT-Large [38]	61.4	42.1
EfficientFormer-L3 [36]	31.3	43.5
EfficientFormer-L7 [36]	82.1	45.1
PoolFormer-S24 [40]	21.0	40.3
PoolFormer-S36 [40]	31.0	42.0
PoolFormer-M48 [40]	73.0	42.7
FastViT-SA36 [53]	30.4	42.9
B-LogViG (Ours)	**30.5**	**46.8 $\pm$ 0.4**

4.3 Ablation Studies

We perform ablation studies to show the benefits of LSGC, our deeper network configuration, and of our high-resolution shortcut (HRS). A summary of these results can be found in Tables 5 and 6.

Table 5: An ablation study of the effects of higher resolution graphers in our network, and using deeper and narrower networks. A checkmark indicates this component was used in the experiment. A (-) indicates this component was not used. 1-S indicates that graph convolutions were only used in Stage 4 of the model, 3-S indicates that graph convolutions were used in stages 2, 3, and 4, and 4-S indicates that graph convolutions were used in all stages of the network.

Base Model	Params (M)	LSGC	1-S	3-S	4-S	Deep Network	Top-1 (%)
MobileViG-S	7.2	-	✓	-	-	-	78.2
Ti-LogViG	7.0	✓	✓	-	-	✓	78.8
Ti-LogViG	8.0	✓	-	✓	-	✓	79.8
Ti-LogViG	8.0	✓	-	-	✓	-	79.6
Ti-LogViG	8.1	✓	-	-	✓	✓	79.9

Starting with MobileViG-S as a base model for comparison, we first try using our LSGC-style graph construction and our deeper and narrower network architecture. We find our LogViT-Ti in this case outperforms MobileViG-S by 0.6% with 0.2 M fewer parameters. Next we use graph convolutions in stages 2, 3, and 4 of the model, the resulting model achieves a top-1 accuracy on ImageNet-1K of 79.8%, 1.0% higher than with only using graph convolutions in stage 4. Next when we use graph convolutions in all 4 stages of the network we achieve a top-1 accuracy on ImageNet-1K of 79.9% with 8.1 M parameters. When we modify the network to be wider and less deep as shown in Table 2 indicated by Wide Ti-LogViG, we see a decrease in accuracy of 0.3% with only 0.1 M fewer parameters.

Table 6: An ablation study of the effects of our LSGC graph construction versus SVGA graph construction and of our high-resolution branch using the high-resolution shortcut. A checkmark indicates this component was used in the experiment. A (-) indicates this component was not used.

Base Model	Params (M)	SVGA	LSGC	High-Resolution	Top-1 (%)
Ti-LogViG	8.0	✓	-	-	79.4
Ti-LogViG	8.0	-	✓	-	79.8
Ti-LogViG	8.1	-	✓	✓	**79.9**

Starting with Ti-LogViG with SVGA and without our high-resolution shortcut (HRS) we achieve a Top-1 accuracy on ImageNet-1k of 79.4%. When we add our LSGC we gain 0.4% in accuracy with no additional parameters. When we add HRS we gain another 0.1% in accuracy with only 0.1 M additional parameters as shown in Table 6.

5 Conclusion

In this work, we have introduced Logarithmic Scalable Graph Construction (LSGC), a novel method for constructing graphs in Vision GNNs that efficiently balances the inclusion of long-range links with reducing the computational cost typically associated with methods like KNN. LSGC overcomes the limitations seen in SVGA by scaling connections logarithmically, improving information flow. We have also proposed LogViG, a hybrid CNN-GNN architecture that integrates LSGC, uses multi-scale and high-resolution features, and leverages a deeper network architecture for enhanced performance. LogViG outperforms existing ViG, CNN, and ViT models on key vision tasks such as image classification and semantic segmentation. Our results demonstrate that LogViG is more efficient in terms of GMACs and parameters while achieving superior accuracy, proving that LSGC and our hybrid architecture offer a significant advancement in the design of Vision GNNs.

6 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] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 1

[2] 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. 1, 2

[3] 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. 7, 8

[4] Jing Xu, Yu Pan, Xinglin Pan, Steven Hoi, Zhang Yi, and Zenglin Xu. RegNet: Self-Regulated Network for Image Classification, 2021. URL https://arxiv.org/abs/2101.00590.

[5] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 1, 2

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale, 2021. URL https://arxiv.org/abs/2010.11929. 1, 2

[7] 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. 7, 8

[8] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEiT: BERT Pre-Training of Image Transformers, 2022. URL https://arxiv.org/abs/2106.08254. 1, 2

[9] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional Networks for Biomedical Image Segmentation, 2015. URL https://arxiv.org/abs/1505.04597. 1

[10] Bowen Zhang, Zhi Tian, Quan Tang, Xiangxiang Chu, Xiaolin Wei, Chunhua Shen, and Yifan Liu. SegViT: Semantic Segmentation with Plain Vision Transformers, 2022. URL https://arxiv.org/abs/2210.05844. 1

[11] 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, 6, 7, 8

[12] 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 (CVPR) Workshops*, pages 2211–2219, June 2023. 2, 3, 6, 7, 8

[13] 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, 7, 8

[14] William Avery, Mustafa Munir, and Radu Marculescu. Scaling graph convolutions for mobile vision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, pages 5857–5865, 2024. 2, 3, 7

[15] Jiaqi Gu, Hyoukjun Kwon, Dilin Wang, Wei Ye, Meng Li, Yu-Hsin Chen, Liangzhen Lai, Vikas Chandra, and David Z Pan. Multi-scale high-resolution vision transformer for semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 12094–12103, 2022. 2, 5, 7, 8

[16] 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. doi: 10.1109/CVPR.2009.5206848. 2, 6, 7

[17] 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. 2, 6, 8

[18] Kishaan Jeeveswaran, Senthilkumar Kathiresan, Arnav Varma, Omar Magdy, Bahram Zonooz, and Elahe Arani. A Comprehensive Study of Vision Transformers on Dense Prediction Tasks, 2022. URL https://arxiv.org/abs/2201.08683. 2

[19] Hans Thisanke, Chamli Deshan, Kavindu Chamith, Sachith Seneviratne, Rajith Vidanaarachchi, and Damayanthi Herath. Semantic Segmentation using Vision Transformers: A survey, 2023. URL https://arxiv.org/abs/2305.03273. 2

[20] Xudong Wang, Li Lyna Zhang, Yang Wang, and Mao Yang. Towards Efficient Vision Transformer Inference: A First Study of Transformers on Mobile Devices. In *Proceedings of the 23rd Annual International Workshop on Mobile Computing Systems and Applications*, HotMobile '22, page 1–7. Association for Computing Machinery, 2022. ISBN 9781450392181. 2

[21] Xinyu Liu, Houwen Peng, Ningxin Zheng, Yuqing Yang, Han Hu, and Yixuan Yuan. EfficientViT: Memory Efficient Vision Transformer with Cascaded Group Attention, 2023. URL https://arxiv.org/abs/2305.07027. 2

[22] Tianle Ma and Aidong Zhang. AffinityNet: Semi-supervisedFew-shot Learning for Disease Type Prediction, 2018. URL https://arxiv.org/abs/1805.08905. 3

[23] Diego Marcheggiani and Ivan Titov. Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv preprint arXiv:1703.04826*, 2017.

[24] Thien Nguyen and Ralph Grishman. Graph Convolutional Networks With Argument-Aware Pooling for Event Detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32 (1), Apr. 2018. doi: 10.1609/aaai.v32i1.12039. URL https://ojs.aaai.org/index.php/AAAI/article/view/12039.

[25] Jiezhong Qiu, Jian Tang, Hao Ma, Yuxiao Dong, Kuansan Wang, and Jie Tang. DeepInf: Social Influence Prediction with Deep Learning. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18. ACM, July 2018. doi: 10.1145/3219819.3220077. URL http://dx.doi.org/10.1145/3219819.3220077. 3

[26] Gabriele Monfardini, Vincenzo Di Massa, Franco Scarselli, Marco Gori, et al. Graph Neural Networks for Object Localization. *Frontiers in Artificial Intelligence and Applications*, 141: 665–669, 2006. 3

[27] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017.

[28] Asish Bera, Zachary Wharton, Yonghuai Liu, Nik Bessis, and Ardhendu Behera. SR-GNN: Spatial Relation-Aware Graph Neural Network for Fine-Grained Image Categorization. *IEEE Transactions on Image Processing*, 31:6017–6031, 2022. doi: 10.1109/TIP.2022.3205215. 3

[29] Juntao Jiang, Xiyu Chen, Guanzhong Tian, and Yong Liu. ViG-UNet: Vision Graph Neural Networks for Medical Image Segmentation. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5, 2023. doi: 10.1109/ISBI53787.2023.10230496. 3

[30] Yuntao Shou, Wei Ai, Tao Meng, and Nan Yin. Graph Information Bottleneck for Remote Sensing Segmentation, 2024. URL https://arxiv.org/abs/2312.02545. 3

[31] Uri Alon and Eran Yahav. On the bottleneck of graph neural networks and its practical implications. *arXiv preprint arXiv:2006.05205*, 2020. 3, 4

[32] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep high-resolution representation learning for visual recognition. *IEEE transactions on pattern analysis and machine intelligence*, 43(10): 3349–3364, 2020. 5

[33] Changqian Yu, Bin Xiao, Changxin Gao, Lu Yuan, Lei Zhang, Nong Sang, and Jingdong Wang. Lite-hrnet: A lightweight high-resolution network. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10440–10450, 2021. 5

[34] Thao Nguyen, Maithra Raghu, and Simon Kornblith. Do wide and deep networks learn the same things? uncovering how neural network representations vary with width and depth. *arXiv preprint arXiv:2010.15327*, 2020. 6

[35] 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. 7

[36] Yanyu Li, Geng Yuan, Yang Wen, Ju Hu, Georgios Evangelidis, Sergey Tulyakov, Yanzhi Wang, and Jian Ren. Efficientformer: Vision transformers at mobilenet speed. *Advances in Neural Information Processing Systems*, 35:12934–12949, 2022. 7, 8

[37] 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. 7

[38] 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. 7, 8

[39] 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. 7

[40] 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. 7, 8

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

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

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

[44] Mustafa Munir, William Avery, Md Mostafijur Rahman, and Radu Marculescu. Greedyvig: Dynamic axial graph construction for efficient vision gnns. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6118–6127, 2024. 7, 8

[45] 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. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 16889–16900, 2023. 7, 8

[46] 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. 7

[47] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 702–703, 2020. 7

[48] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. URL `https://openreview.net/forum?id=r1Ddp1-Rb`. 7

[49] Sangdoo Yun, Dongyoon Han, Seong Joon Oh, Sanghyuk Chun, Junsuk Choe, and Youngjoon Yoo. Cutmix: Regularization strategy to train strong classifiers with localizable features. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 6023–6032, 2019. 7

[50] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13001–13008, 2020. 7

[51] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8129–8138, 2020. 7

[52] 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. 8

[53] 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 (ICCV)*, pages 5785–5795, October 2023. 8