MobileViG: Graph-Based Sparse Attention for Mobile Vision Applications

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

William Avery*
The University of Texas at Austin
williamaavery@utexas.edu

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

Abstract

Traditionally, convolutional neural networks (CNN) and vision transformers (ViT) have dominated computer vision. However, recently proposed vision graph neural networks (ViG) provide a new avenue for exploration. Unfortunately, for mobile applications, ViGs are computationally expensive due to the overhead of representing images as graph structures. In this work, we propose a new graph-based sparse attention mechanism, Sparse Vision Graph Attention (SVGA), that is designed for ViGs running on mobile devices. Additionally, we propose the first hybrid CNN-GNN architecture for vision tasks on mobile devices, MobileViG, which uses SVGA. Extensive experiments show that MobileViG beats existing ViG models and existing mobile CNN and ViT architectures in terms of accuracy and/or speed on image classification, object detection, and instance segmentation tasks. Our fastest model, MobileViG-Ti, achieves 75.7% top-1 accuracy on ImageNet-1K with 0.78 ms inference latency on iPhone 13 Mini NPU (compiled with CoreML), which is faster than MobileNetV2x1.4 (1.02 ms, 74.7% top-1) and MobileNetV2x1.0 (0.81 ms, 71.8% top-1). Our largest model, MobileViG-B obtains 82.6% top-1 accuracy with only 2.30 ms latency, which is faster and more accurate than the similarly sized EfficientFormer-L3 model (2.77 ms, 82.4%). Our work proves that well designed hybrid CNN-GNN architectures can be a new avenue of exploration for designing models that are extremely fast and accurate on mobile devices. Our code is publicly available at https://github.com/SLDGroup/MobileViG.

1. Introduction

Artificial intelligence (AI) and machine learning (ML) have had explosive growth in the past decade. In computer vision, the key driver behind this growth has been the re-emergence of neural networks, especially convolutional neural networks (CNNs) and more recently vision transformers [4, 25]. Even though CNNs trained via back-propagation were invented in the 1980s [16, 25], they were used for more small-scale tasks such as character recognition [17]. The potential of CNNs to re-shape the field of artificial intelligence was not fully realized until AlexNet [15] was introduced in the ImageNet [32] competition. Further advancements to CNN architectures have been made improving their accuracy, efficiency, and speed [10, 12, 13, 33, 34]. Along with CNN architectures, pure multi-layer perceptron (MLP) architectures and MLP-like architectures have also shown promise as backbones for general-purpose vision tasks [2, 37, 38]

Though CNNs and MLPs had become widely used in computer vision, the field of natural language processing used recurrent neural networks (RNNs), specifically long-short term memory (LSTM), networks due to the disparity between the tasks of vision and language [11]. Though LSTMs are still used, they have largely been replaced with transformer architectures in NLP tasks [40]. With the introduction of Vision Transformer (ViT) [4] a network architecture applicable to both language and vision domains was introduced. By splitting an image into a sequence of patch embeddings an image can be transformed into an input usable by transformer modules [4]. One of the major advantages of the transformer architecture over CNNs or MLPs is its global receptive field, allowing it to learn from distant object interactions in images.

Graph neural networks (GNNs) have developed to operate on graph-based structures such as biological networks, social networks, or citation networks [7, 14, 43, 45]. GNNs have even been proposed for tasks such as node classification [14], drug discovery [5], fraud detection [23], and now computer vision tasks with the recently proposed Vision GNN (ViG) [8]. In short, ViG divides an image into

*Equal contribution

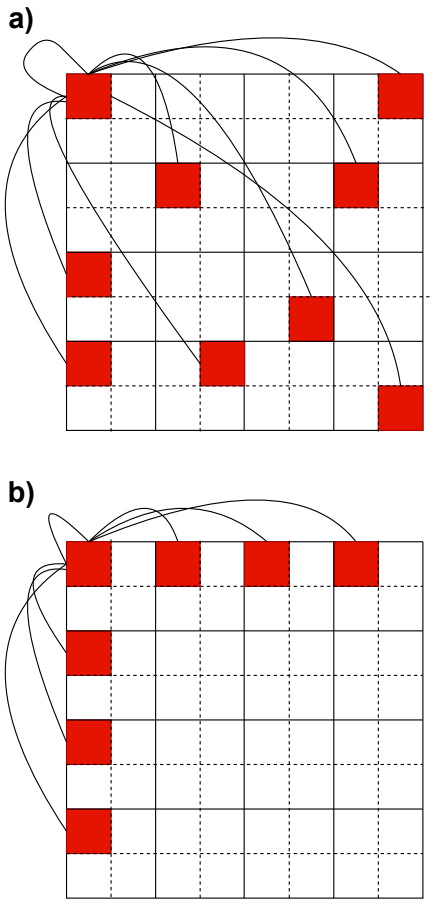

Figure 1. a) KNN graph attention for the top left pixel of an 8×8 image as used in Vision GNN. b) SVGA for the top left pixel of an 8×8 image. As shown, SVGA uses a structured graph, that does not change across input images. This structure removes the KNN and reshaping operations required in a) that are not mobile friendly.

patches and then connects the patches through the K-nearest neighbors (KNN) algorithm [8], thus providing the ability to process global object interactions similar to ViTs.

Research in computer vision for mobile applications has seen rapid growth, leading to hybrid architectures using CNNs for learning spatially local representations and vision transformers (ViT) for learning global representations [27]. Current ViG models are not suited for mobile tasks, as they are inefficient and slow when running on mobile devices. The concepts learned from the design of CNN and ViT models can be explored to determine whether CNN-GNN hybrid models can provide the speed of CNN-based models along with the accuracy of ViT-based models. In this work, we investigate hybrid CNN-GNN architectures for computer vision on mobile devices and develop a graph-based attention mechanism that can compete with existing efficient architectures. We summarize our contributions as follows:

1. We propose a new graph-based sparse attention method designed for mobile vision applications. We call our attention method Sparse Vision Graph Attention (SVGA). Our method is lightweight as it does not require reshaping and incurs little overhead in graph construction as compared to previous methods.

2. We propose a novel mobile CNN-GNN architecture for vision tasks using our proposed SVGA, max-relative graph convolution [18], and concepts from mobile CNN and mobile vision transformer architectures [12, 27] that we call MobileViG.

3. Our proposed model, MobileViG, matches or beats existing vision graph neural network (ViG), mobile convolutional neural network (CNN), and mobile vision transformer (ViT) architectures in terms of accuracy and/or speed on three representative vision tasks: ImageNet image classification, COCO object detection, and COCO instance segmentation.

To the best of our knowledge, we are the first to investigate hybrid CNN-GNN architectures for mobile vision applications. Our proposed SVGA attention method and MobileViG architecture open a new path of exploration for state-of-the-art mobile architectures and ViG architectures.

This paper is structured as follows. Section 2 covers related work in the ViG and mobile architecture space. Section 3 describes the design methodology behind SVGA and the MobileViG architecture. Section 4 describes experimental setup and results for ImageNet-1k image classification, COCO object detection, and COCO instance segmentation. Lastly, Section 5 concludes the paper and suggests future work with ViGs in mobile architecture design.

2. Related Work

ViG [8] is proposed as an alternative to CNNs and ViTs due to its capacity to represent image data in a more flexible format. ViG represents images through using the KNN algorithm [8], where each pixel in the image attends to similar pixels. ViG achieves comparable performance to popular ViT models, DeiT [39] and SwinTransformer [24], suggesting it is worth further investigations.

Despite the success of ViT-based models in vision tasks, they are still slower when compared to lightweight CNN-based models [21], in contrast CNN-based models lack the global receptive field of ViT-based models. Thus, ViG-based models may be a possible solution by providing speeds faster than ViT-based models and accuracies higher than CNN-based models. To the best of our knowledge, there are no works on mobile ViGs at this time; however,

Table 1. MobileViG architecture showing configuration of the stages, output size, downsample layers, and classification head.

Stage	Output Size	MobileViG-Ti	MobileViG-S	MobileViG-M	MobileViG-B
Stem	$\frac{H}{4} \times \frac{W}{4}$	Conv $\times 2$	Conv $\times 2$	Conv $\times 2$	Conv $\times 2$
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} MBConv \\ C = 42 \end{bmatrix} \times 2$	$\begin{bmatrix} MBConv \\ C = 42 \end{bmatrix} \times 3$	$\begin{bmatrix} MBConv \\ C = 42 \end{bmatrix} \times 3$	$\begin{bmatrix} MBConv \\ C = 42 \end{bmatrix} \times 5$
$\downarrow$	$\frac{H}{8} \times \frac{W}{8}$	Conv	Conv	Conv	Conv
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	$\begin{bmatrix} MBConv \\ C = 84 \end{bmatrix} \times 2$	$\begin{bmatrix} MBConv \\ C = 84 \end{bmatrix} \times 3$	$\begin{bmatrix} MBConv \\ C = 84 \end{bmatrix} \times 3$	$\begin{bmatrix} MBConv \\ C = 84 \end{bmatrix} \times 5$
$\downarrow$	$\frac{H}{16} \times \frac{W}{16}$	Conv	Conv	Conv	Conv
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} MBConv \\ C = 168 \end{bmatrix} \times 6$	$\begin{bmatrix} MBConv \\ C = 176 \end{bmatrix} \times 9$	$\begin{bmatrix} MBConv \\ C = 224 \end{bmatrix} \times 9$	$\begin{bmatrix} MBConv \\ C = 240 \end{bmatrix} \times 15$
$\downarrow$	$\frac{H}{32} \times \frac{W}{32}$	Conv	Conv	Conv	Conv
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	$\begin{bmatrix} SVGA \\ K = 2 \\ C = 256 \end{bmatrix} \times 2$	$\begin{bmatrix} SVGA \\ K = 2 \\ C = 256 \end{bmatrix} \times 3$	$\begin{bmatrix} SVGA \\ K = 2 \\ C = 400 \end{bmatrix} \times 3$	$\begin{bmatrix} SVGA \\ K = 2 \\ C = 464 \end{bmatrix} \times 5$
Head	1×1	Pooling & MLP	Pooling & MLP	Pooling & MLP	Pooling & MLP

there are many existing works in the mobile CNN and hybrid model space. We classify mobile architecture designs into two primary categories: convolutional neural network (CNN) models and hybrid CNN-ViT models, which blend elements of CNNs and ViTs.

The MobileNetv2 [33] and EfficientNet [35, 36] families of CNN-based architectures are some of the first mobile models to see success in common image tasks. These models are lightweight with fast inference speeds. However, purely CNN-based models have steadily been replaced by hybrid competitors.

There are a vast number of hybrid mobile models, including MobileViTv2 [28], EdgeViT [29] LeViT [6], and EfficientFormerv2 [20]. These hybrid models consistently beat MobileNetv2 in image classification, object detection, and instance segmentation tasks, but some of these models do not always perform as well in terms of latency. The latency difference can be tied to the inclusion of ViT blocks, which have traditionally been slower on mobile hardware. To improve this state of affairs we propose MobileViG, which provides speeds comparable to MobileNetv2 [33] and accuracies comparable to EfficientFormer [21].

3. Methodology

In this section, we describe the SVGA algorithm and provide details on the MobileViG architecture design. More precisely, Section 3.1 describes the SVGA algorithm. Section 3.2 explains how we adapt the Grapher module from ViG [8] to create the SVGA block. Section 3.3 describes how we combine the SVGA blocks along with inverted residual blocks for local processing to create MobileViG-Ti, MobileViG-S, MobileViG-M, and MobileViG-B.

3.1. Sparse Vision Graph Attention

We propose Sparse Vision Graph Attention (SVGA) as a mobile-friendly alternative to KNN graph attention from Vision GNN [8]. The KNN-based graph attention introduces two non-mobile-friendly components, KNN computation and input reshaping, that we remove with SVGA.

In greater detail, the KNN computation is required for every input image, since the nearest neighbors of each pixel cannot be known ahead of time. This results in a graph with seemingly random connections as seen in Figure 1a. Due to the unstructured nature of KNN, the authors of [8] reshape the input image from a 4D to 3D tensor, allowing them to properly align the features of connected pixels for graph convolution. Following the graph convolution, the input must be reshaped from 3D back to 4D for subsequent convolutional layers. Thus, KNN-based attention requires the KNN computation and two reshaping operations, both of which are costly on mobile devices.

To remove the overhead of the KNN computation and

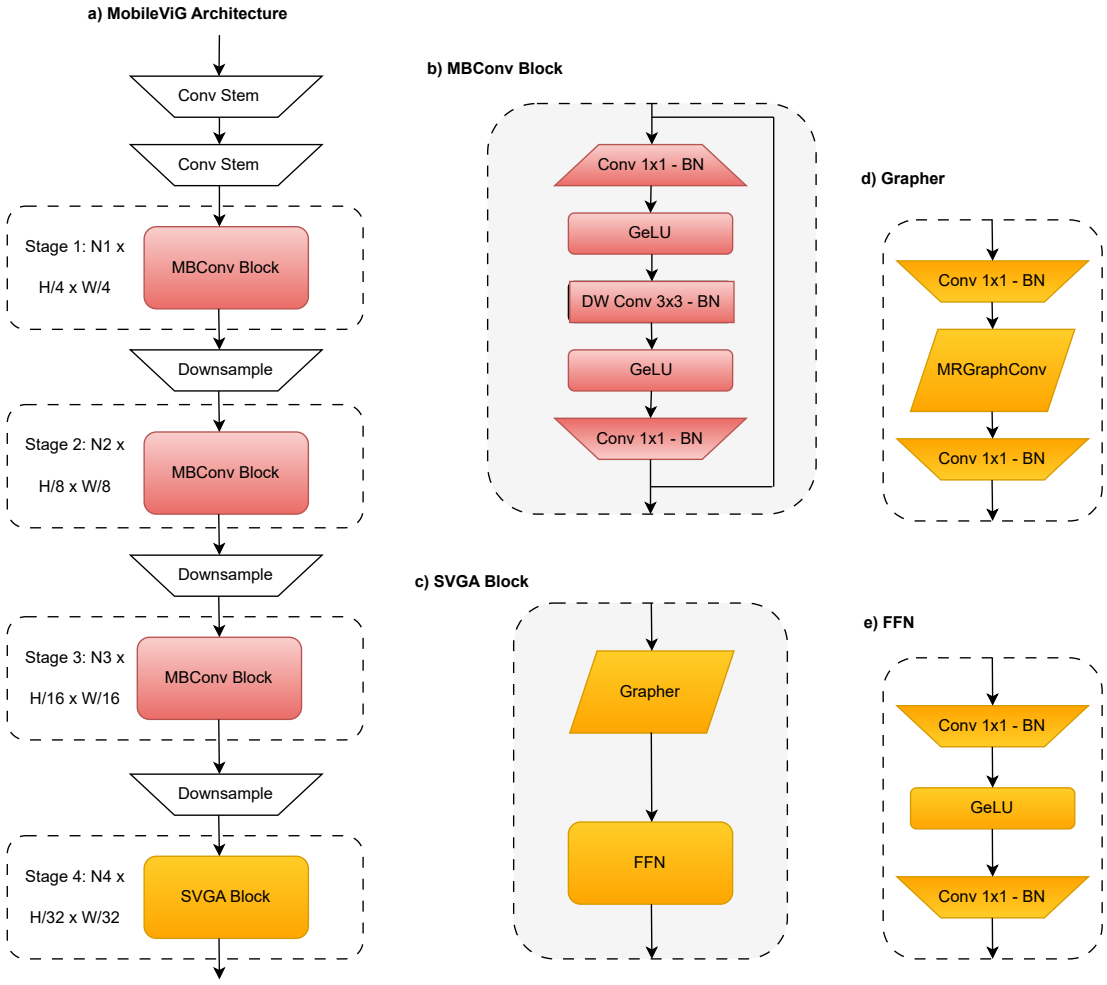

Figure 2. MobileViG architecture. (a) network architecture showing the stages and layers, where N1, N2, N3, and N4 represent the number of those blocks in the MobileViG-Ti, S, M, and B configurations (Section 3.3). (b) MBConv Block (Section 3.3). (c) SVGA Block (Section 3.2). (d) & (e) Grapher and FFN (Section 3.2).

reshaping operations, SVGA assumes a fixed graph, where each pixel is connected to every K^{th} pixel in its row and column. For example, given an 8×8 image and $K = 2$, the top left pixel would be connected to every second pixel across its row and every second pixel down its column as seen in Figure 1b. This same pattern is repeated for every pixel in the input image. Since the graph has a fixed structure (i.e., each pixel will have the same connections for all 8×8 input images), the input image does not have to be reshaped to perform the graph convolution.

Instead, it can be implemented using rolling operations across the two image dimensions, denoted as $roll_{right}$ and $roll_{down}$ in Algorithm 1. The first parameter to the $roll$ operation is the input to roll, and the second is the distance to roll in the $right$ or $down$ direction. Using the example from Figure 1b where $K = 2$, the top left pixel can be aligned with every second pixel in its row by rolling the image twice to the right, four times to the right, and six times to the right. The same can be done for every second pixel in its column, except by rolling down. Note that since every pixel is connected in the same way, the rolling operations used to align the top left pixel with its connections simultaneously align every other pixel in the image with its connections. In MobileViG, graph convolution is performed using max-relative graph convolution (MRConv). Therefore, after every $roll_{right}$ and $roll_{down}$ operation, the difference between the original input image and the rolled version is computed, denoted as X_r and X_c in Algorithm 1, and the max operation is taken element wise and stored in X_j, also denoted in Algorithm 1. After completing the rolling and max-relative operations, a final $Conv2d$ is performed. Through this approach, SVGA trades the KNN computation

for cheaper rolling operations, consequently not requiring reshaping to perform the graph convolution.

We note that SVGA eschews the representation flexibility of KNN in favor of being mobile friendly.

Algorithm 1 SVGA with MRConv

Require: K, the distance between connections; H, W, the image resolution; X, the input image; m, controls the distance of each roll
$m \leftarrow 0$
while $mK < H$ **do**
  $X_c \leftarrow X - roll_{down}(X, mK)$ ▷ get relative features
  $X_j \leftarrow max(X_c, X_j)$ ▷ keep max relative features
  $m \leftarrow m + 1$
end while
$m \leftarrow 0$
while $mK < W$ **do**
  $X_r \leftarrow X - roll_{right}(X, mK)$
  $X_j \leftarrow max(X_r, X_j)$
  $m \leftarrow m + 1$
end while
return $Conv2d(Concat(X, X_j))$

3.2. SVGA Block

We insert SVGA and the updated MRConv layer into the Grapher block proposed in Vision GNN [8]. Given an input feature $X \in \mathbb{R}^{N \times N}$, the updated Grapher is expressed as

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

where $Y \in \mathbb{R}^{N \times N}$, W_{in} and W_{out} are fully connected layer weights, and σ is a GeLU activation. We also change the number of filter groups from 4 (the value used in Vision GNN [8]) to 1 in the MRConv step to increase the expressive potential of the MRConv layer without a noticeable increase in latency. The updated Grapher module is visually depicted in Figure 2d

Following the updated Grapher, we use the feed-forward network (FFN) module as proposed in Vision GNN [8] and shown in Figure 2e The FFN module is a two layer MLP expressed as

$$Z = \sigma(XW_1)W_2 + Y \quad (2)$$

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 updated Grapher and FFN an SVGA block, as shown in Figure 2c.

3.3. MobileViG Architecture

The MobileViG architecture shown in Figure 2a is composed of a convolutional stem, followed by three stages of inverted residual blocks (MBConv) with an expansion ratio of four for local processing as proposed in MobileNetv2 [33]. Within the MBConv blocks, we swap ReLU6 for GeLU as it has been shown to improve performance in computer vision tasks [4, 20]. The MBConv blocks consist of a 1×1 convolution plus batch normalization (BN) and GeLU, a depth-wise 3×3 convolution plus BN and GeLU, and lastly a 1×1 convolution plus BN and a residual connection as seen in Figure 2b. Following the MBConv blocks we have one stage of SVGA blocks to capture global information as seen in Figure 2a. We also have a convolutional head after the SVGA blocks for classification. After each MBConv stage, a downsampling step halves the input resolution and expands the channel dimension. Each stage is composed of multiple MBConv or SVGA blocks, where the number of repetitions is changed depending on model size. The channel dimensions and number of blocks repeated per stage for MobileViG-Ti, MobileViG-S, MobileViG-M, and MobileViG-B can be seen in Table 1.

Table 2. Top-1 accuracy on ImageNet-1k classification and iPhone13 Mini GPU latency for PyramidViG and MobileViG. Bold entries indicate results obtained using MobileViG and SVGA in this paper.

Model	Params (M)	GMACs	GPU Latency (ms)	Top-1 (%)
PViG-Ti [8]	10.7	1.7	81.2	78.2
PViG-S [8]	27.3	4.6	111	82.1
PViG-M [8]	51.7	8.9	171	83.1
PViG-B [8]	92.6	16.8	242	83.7
MViG-Ti (Ours)	**5.2**	**0.7**	**18.0**	**75.7**
MViG-S (Ours)	**7.2**	**1.0**	**28.7**	**78.2**
MViG-M (Ours)	**14.0**	**1.5**	**33.2**	**80.6**
MViG-B (Ours)	**26.7**	**2.8**	**53.4**	**82.6**

4. Experimental Results

We compare MobileViG to ViG [8] and show its superior performance in terms of latency, model size, and image classification accuracy on ImageNet-1k [3] in Table 2. We also compare MobileViG to several mobile models and show that, for each model, it has superior or comparable performance in terms of accuracy and latency in Table 3.

4.1. Image Classification

We implement the model using PyTorch 1.12 [30] and Timm library [42].We use 8 NVIDIA A100 GPUs to train each model, with an effective batch size of 1024. The models are trained from scratch for 300 epochs on ImageNet-1K [3] with AdamW optimizer [26]. Learning rate is set to 2e-3 with cosine annealing schedule. We use a standard image resolution, 224 × 224, for both training and testing. Similar to DeiT [39], we perform knowledge distillation using RegNetY-16GF [31] with 82.9% top-1 accuracy. For data augmentation we use RandAugment, Mixup, Cutmix, random erasing, and repeated augment.

Table 3. Results of MobileViG and other mobile architectures on ImageNet-1k. Latency is reported on the NPU and GPU of the iPhone13 Mini. Models are compiled with CoreML. Bold entries indicate results obtained using MobileViG and SVGA in this paper.

Model	Type	Params (M)	GMACs	Latency (ms)		Epochs	Top-1 (%)
				NPU	GPU		
MobileNetV2x1.0 [33]	CONV	3.5	0.3	0.81	13.0	300	71.8
MobileViTv2-1.0 [28]	Hybrid	4.9	1.8	3.13	40.2	300	78.1
EfficientFormerV2-S0 [20]	Hybrid	3.5	0.4	0.85	19.0	300	75.7
EdgeViT-XXS [29]	Hybrid	4.1	0.6	-	25.0	300	74.4
MobileViG-Ti (Ours)	**CNN-GNN**	**5.2**	**0.7**	**0.78**	**18.0**	**300**	**75.7**
MobileNetV2x1.4 [33]	CONV	6.1	0.6	1.02	14.8	300	74.7
EfficientNet-B0 [35]	CONV	5.3	0.4	1.89	17.9	300	77.7
DeiT-T [39]	Attention	5.9	1.2	8.60	27.3	300	74.5
EdgeViT-XS [29]	Hybrid	6.7	1.1	-	36.9	300	77.5
EfficientFormerV2-S1 [20]	Hybrid	6.1	0.7	0.93	31.7	300	79.0
LeViT-128S [6]	Hybrid	7.8	0.3	7.63	8.09	1000	76.6
MobileViG-S (Ours)	**CNN-GNN**	**7.2**	**1.0**	**0.99**	**28.7**	**300**	**78.2**
ResNet18	CONV	11.7	1.82	1.20	15.7	300	69.7
MobileViTv2-1.5 [28]	Hybrid	10.6	4.0	4.52	70.0	300	80.4
EfficientNet-B3 [35]	CONV	12.2	2.0	5.46	61.4	300	82.2
PoolFormer-s12 [44]	Pool	12.0	2.0	1.47	91.7	300	77.2
LeViT-192 [6]	Hybrid	10.9	0.7	41.8	13.0	1000	80.0
EdgeViT-S [29]	Hybrid	11.1	1.9	-	57.5	300	81.0
EfficientFormerV2-S2 [20]	Hybrid	12.6	1.3	1.42	60.0	300	81.6
EfficientFormer-L1 [21]	Hybrid	12.3	1.3	1.18	18.0	300	79.2
MobileViG-M (Ours)	**CNN-GNN**	**14.0**	**1.5**	**1.38**	**33.2**	**300**	**80.6**
ResNet50 [10]	CONV	25.6	4.1	2.29	38.2	300	80.4
ConvNext-T [25]	CONV	28.6	7.4	147	227	300	82.7
MobileViTv2-2.0 [28]	Hybrid	18.5	7.5	6.13	128	300	81.2
PoolFormer-s24 [44]	Pool	21.0	3.6	2.48	177	300	80.3
PoolFormer-s36 [44]	Pool	31.0	5.2	3.40	266	300	81.4
PoolFormer-m36 [44]	Pool	56.0	8.8	5.73	343	300	82.1
DeiT-S [39]	Attention	22.5	4.5	13.7	76.9	300	81.2
Swin-T [24]	Attention	29.0	4.5	-	-	300	81.4
LeViT-256 [6]	Hybrid	18.9	1.1	48.5	18.7	1000	81.6
LeViT-384 [6]	Hybrid	39.1	2.4	62.0	30.7	1000	82.6
EfficientFormerV2-L [21]	Hybrid	26.1	2.6	2.36	83.7	300	83.3
EfficientFormer-L3 [21]	Hybrid	31.3	3.9	2.77	38.1	300	82.4
EfficientFormer-L7 [21]	Hybrid	82.1	10.2	6.87	83.3	300	83.3
MobileViG-B (Ours)	**CNN-GNN**	**26.7**	**2.8**	**2.30**	**53.4**	**300**	**82.6**

Table 4. Object detection and instance segmentation results of MobileViG and other backbones on MS COCO 2017. Bold entries indicate results obtained using MobileViG and SVGA in this paper.

Backbone	Params (M)	APb	APb50	APb75	APm	APm50	APm75
ResNet18 [10]	11.7	34.0	54.0	36.7	31.2	51.0	32.7
EfficientFormer-L1 [21]	12.3	37.9	60.3	41.0	35.4	57.3	37.3
PoolFormer-S12 [44]	12.0	37.3	59.0	40.1	34.6	55.8	36.9
MobileViG-M (Ours)	**14.0**	**41.3**	**62.8**	**45.1**	**38.1**	**60.1**	**40.8**
ResNet50 [10]	25.5	38.0	58.6	41.4	34.4	55.1	36.7
EfficientFormer-L3 [21]	31.3	41.4	63.9	44.7	38.1	61.0	40.4
PoolFormer-S24 [44]	21.0	40.1	62.2	43.4	37.0	59.1	39.6
PVT-Small [41]	24.5	40.4	62.9	43.8	37.8	60.1	40.3
MobileViG-B (Ours)	**26.7**	**42.0**	**64.3**	**46.0**	**38.9**	**61.4**	**41.6**

We use an iPhone 13 Mini (iOS 16) to benchmark latency on NPU and GPU. The models are compiled with CoreML and latency is averaged over 1000 predictions [1].

As seen in Table 2, for a similar number of parameters, MobileViG outperforms Pyramid ViG [8] both in accuracy and GPU latency. For example, for 3.5 M fewer parameters, MobileViG-S matches Pyramid ViG-Ti in top-1 accuracy, while being 2.83× faster. Additionally, for 0.6 M fewer parameters, MobileViG-B beats Pyramid ViG-S by 0.5% in top-1 accuracy, while being 2.08× faster.

When compared to mobile models in Table 3, MobileViG consistently beats every model in at least NPU latency, GPU latency, or accuracy. MobileViG-Ti is faster than MobileNetv2 with 3.9% higher top-1 accuracy. It also matches EfficientFormerv2 [20] in top-1 while having a slight edge in NPU and GPU latency. MobileViG-S is nearly 2x faster than EfficientNet-B0 [35] in NPU latency and has 0.5% higher top-1 accuracy. Compared to MobileViTv2-1.5 [28], MobileViG-M is over 3x faster in NPU latency and 2x faster in GPU latency with 0.2% higher top-1 accuracy. Additionally, MobileViG-B is 6x faster than DeiT-S and is able to beat both DeiT-S and Swin-Tiny in top-1 accuracy.

4.2. Object Detection and Instance Segmentation

We evaluate MobileViG on object detection and instance segmentation tasks to further prove the potential of SVGA. We integrate MobileViG as a backbone in the Mask-RCNN framework [9] and experiment using the MS COCO 2017 dataset [22]. We implement the backbone using PyTorch 1.12 [30] and Timm library [42], and use 4 NVIDIA RTX A6000 GPUs to train our models. We initialize the model with pretrained ImageNet-1k weights from 300 epochs of training, use AdamW [26] optimizer with an initial learning rate of 2e-4 and train the model for 12 epochs with a standard resolution (1333 X 800) following the process of Next-ViT, EfficientFormer, and EfficientFormerV2 [19–21].

As seen in Table 4, with similar model size MobileViG outperforms ResNet, PoolFormer, EfficientFormer, and PVT in terms of either parameters or improved average precision (AP) on object detection and/or instance segmentation. The medium size MobileViG-M model gets 41.3 APbox, 62.8 APbox when 50 Intersection over Union (IoU), and 45.1 APbox when 75 IoU on the object detection task. MobileViG-M gets 38.1 APmask, 60.1 APmask when 50 IoU, and 40.8 APmask when 75 IoU for the instance segmentation task. The big size MobileViG-B model gets 42.0 APbox, 64.3 APbox when 50 IoU, and 46.0 APbox when 75 IoU on the object detection task. MobileViG-B gets 38.9 APmask, 61.4 APmask when 50 IoU, and 41.6 APmask when 75 IoU on the instance segmentation task. The strong performance of MobileViG on object detection and instance segmentation shows that MobileViG generalizes well as a backbone for different tasks in computer vision.

The design of MobileViG is partly inspired by the designs of Pyramid ViG [8], EfficientFormer [21], and the MetaFormer concept [44]. The results achieved in MobileViG demonstrate that hybrid CNN-GNN architectures are a viable alternative to CNN, ViT, and hybrid CNN-ViT designs. Hybrid CNN-GNN architectures can provide the speed of CNN-based models along with the accuracy of ViT models making them an ideal candidate for high accuracy mobile architecture designs. Further explorations of hybrid CNN-GNN architectures for mobile computer vision tasks can improve on the MobileViG concept and introduce new state-of-the-art architectures.

5. Conclusion

In this work, we have proposed a graph-based attention mechanism, Sparse Vision Graph Attention (SVGA), and MobileViG, a competitive mobile vision architecture that uses SVGA. SVGA does not require reshaping and allows for the graph structure to be known prior to inference, unlike

previous methods. We use inverted residual blocks, max-relative graph convolution, and feed-forward network layers to create MobileViG, a hybrid CNN-GNN architecture, that achieves competitive results on image classification, object detection, and instance segmentation tasks. MobileViG outperforms existing ViG models and many existing mobile models, including MobileNetv2, in terms of accuracy and latency. Future research on mobile architectures can further explore the potential of GNN-based models on resource-constrained devices for IoT applications.

References

[1] Apple Inc. Core ml. https://developer.apple.com/documentation/coreml, 2017. 7

[2] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. *arXiv preprint arXiv:2107.10224*, 2021. 1

[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. 5

[4] Alexey Dosovitskiy et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020. 1, 5

[5] Thomas Gaudelet, Ben Day, Arian R Jamasb, Jyothish Soman, Cristian Regep, Gertrude Liu, Jeremy BR Hayter, Richard Vickers, Charles Roberts, Jian Tang, et al. Utilizing graph machine learning within drug discovery and development. *Briefings in bioinformatics*, 22(6):bbab159, 2021. 1

[6] 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. 3, 6

[7] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. *Advances in neural information processing systems*, 30, 2017. 1

[8] 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, 5, 7

[9] 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

[10] 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, 6, 7

[11] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1

[12] 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

[13] 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. 1

[14] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 1

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

[16] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989. 1

[17] 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

[18] 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

[19] 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

[20] 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. 3, 5, 6, 7

[21] 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, 3, 6, 7

[22] 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. 7

[23] Yang Liu, Xiang Ao, Zidi Qin, Jianfeng Chi, Jinghua Feng, Hao Yang, and Qing He. Pick and choose: a gnn-based imbalanced learning approach for fraud detection. In *Proceedings of the Web Conference 2021*, pages 3168–3177, 2021. 1

[24] 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. 2, 6

[25] 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

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

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

[28] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 3, 6, 7

[29] Junting Pan, Adrian Bulat, Fuwen Tan, Xiatian Zhu, Lukasz Dudziak, Hongsheng Li, Georgios Tzimiropoulos, and Brais Martinez. Edgevits: Competing light-weight cnns on mobile devices with vision transformers. In *Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XI*, pages 294–311. Springer, 2022. 3, 6

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

[31] 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. 5

[32] Olga Russakovsky et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015. 1

[33] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. 2018. 1, 3, 5, 6

[34] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 1

[35] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 3, 6, 7

[36] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 3

[37] 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

[38] 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

[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. 2, 5, 6

[40] 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

[41] 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

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

[43] 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*, volume 34, pages 1054–1061, 2020. 1

[44] 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

[45] 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. 1