Scaling Graph Convolutions for Mobile Vision

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

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

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

Abstract

To compete with existing mobile architectures, MobileViG introduces Sparse Vision Graph Attention (SVGA), a fast token-mixing operator based on the principles of GNNs. However, MobileViG scales poorly with model size, falling at most 1% behind models with similar latency. This paper introduces Mobile Graph Convolution (MGC), a new vision graph neural network (ViG) module that solves this scaling problem. Our proposed mobile vision architecture, MobileViGv2, uses MGC to demonstrate the effectiveness of our approach. MGC improves on SVGA by increasing graph sparsity and introducing conditional positional encodings to the graph operation. Our smallest model, MobileViGv2-Ti, achieves a 77.7% top-1 accuracy on ImageNet-1K, 2% higher than MobileViG-Ti, with 0.9 ms inference latency on the iPhone 13 Mini NPU. Our largest model, MobileViGv2-B, achieves an 83.4% top-1 accuracy, 0.8% higher than MobileViG-B, with 2.7 ms inference latency. Besides image classification, we show that MobileViGv2 generalizes well to other tasks. For object detection and instance segmentation on MS COCO 2017, MobileViGv2-M outperforms MobileViG-M by 1.2 AP^{box} and 0.7 AP^{mask}, and MobileViGv2-B outperforms MobileViG-B by 1.0 AP^{box} and 0.7 AP^{mask}. For semantic segmentation on ADE20K, MobileViGv2-M achieves 42.9% $mIoU$ and MobileViGv2-B achieves 44.3% $mIoU$ [1].

1. Introduction

With the explosion of interest in large generative models, the demand for artificial intelligence (AI) applications has skyrocketed. An increasingly important sector of this demand is the mobile market. The goal is the ability to run powerful AI applications directly on user devices. These

[1]Code: https://github.com/SLDGroup/MobileViGv2

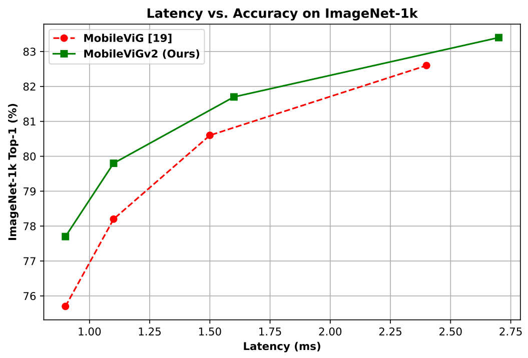

Figure 1. Latency versus top-1 % accuracy on ImageNet-1K of MobileViG [19] and MobileViGv2. From this graph, we can see that MobileViGv2 improves on MobileViG, shifting the accuracy-latency curve up for similar points of inference latency.

models must provide quick, personalized responses and, more importantly, keep users' data private and off the cloud. To achieve this, models must be small in size, fast, low power, and still maintain high performance on the target task.

Early efforts at targeting vision applications on mobile devices used convolutional neural networks (CNNs), such as with the MobileNet [10][23] and EfficientNet[24][25] family of architectures. While these models perform well, the introduction of vision transformers (ViTs) [5] brought in new hybrid CNN-ViT mobile architectures [1][17][18][14][13] that significantly outperformed their CNN counterparts. The success of CNN-ViT-based mobile architectures over CNN-based ones is mainly due to the global receptive field of the self-attention operation, which accounts for more complex relationships across tokens. However, this success comes at a cost. The self-attention module is much slower than a convolution layer, as it scales quadratically with the number of input tokens.

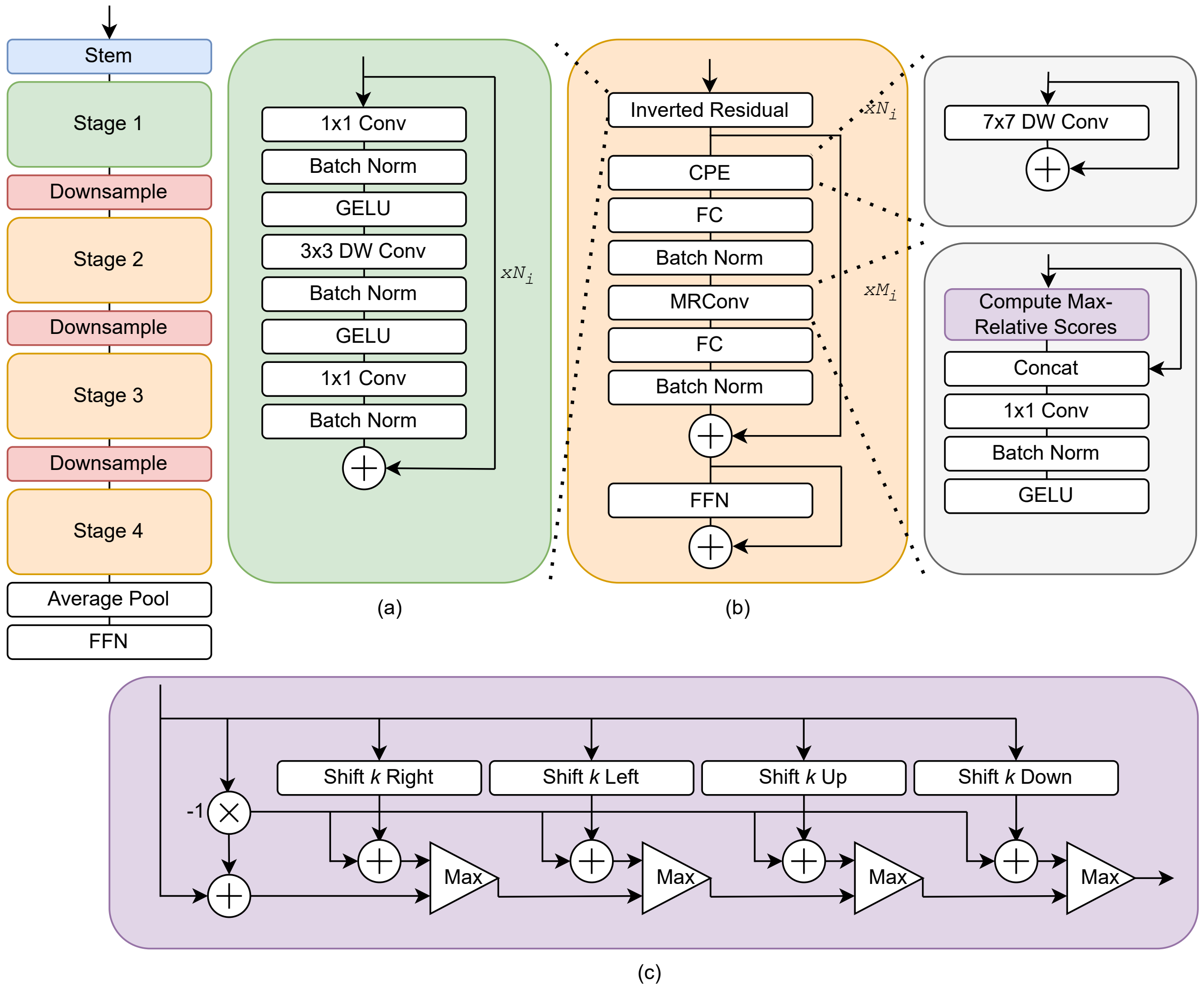

Figure 2. The new MobileViGv2 architecture. The full architecture is shown on the left. The stem is composed of two stride two convolutions that downsample the input image by $4\times$. Each downsampling block contains a single stride two convolution to downsample the input by $2\times$. (a) An inverted residual block using GELU activation. For Stage 1, only inverted residuals are used. The number of inverted residuals in this stage is controlled by N_1. (b) For stages 2-4, a combination of inverted residuals and MGCs are used. Each stage has N_i inverted residuals followed by M_i MGCs, where i is the stage number. The CPE block is a conditional positional encoding [2] implemented with a 7×7 depthwise convolution. The MRConv block contains graph construction and the max-relative message passing step. (c) Computing max-relative features using graph construction as outlined in MGC. Given an input image, this module computes the max-relative score against a fixed set of shifted inputs: shifting right, left, up, and down by k. The outputs of this stage are the max-relative scores, which are concatenated to the input and passed through a 1×1 convolution to complete message passing.

As such, most CNN-ViT-based mobile architectures only use self-attention in low-resolution stages.

More recently, vision graph neural networks (ViGs) [6] were introduced as an alternative to CNNs and ViTs. ViGs connect tokens based on a predefined algorithm, such as K-nearest neighbors (KNN), and then mix the tokens with a message-passing scheme. The first use of ViGs in mobile vision architectures came with MobileViG [19]. MobileViG uses Sparse Vision Graph Attention (SVGA), which replaces KNN with a static graph construction method, resulting in a fast CNN-ViG architecture. While MobileViG performs well for small model sizes, it scales poorly as the model size increases, falling nearly 1% behind CNN-ViT-based models with similar latency [13] [27].

To address this scaling problem, we propose a new ViG module called *Mobile Graph Convolution* (MGC), which improves on SVGA by increasing graph sparsity and introducing positional encodings to the graph operation. To demonstrate the effectiveness of MGC, we introduce MobileViGv2. This CNN-ViG-based architecture uses inverted residual blocks for processing using local receptive fields and MGC blocks for processing using long-range receptive fields. With higher graph sparsity, MobileViGv2 can use MGC at higher resolution stages without impacting latency compared to MobileViG. Unlike the original ViG [6] model, MobileViG does not use positional encodings, an improvement introduced in MGC that leads to a significant performance boost with a slight increase in parameters. We summarize our contributions below:

1. We propose Mobile Graph Convolution (MGC). This new mobile ViG module creates sparser graphs than Sparse Vision Graph Attention (SVGA) by fixing the number of possible connections per token regardless of input size. It uses conditional positional encodings to share the spatial relationships between tokens during message passing.
2. We propose MobileViGv2, as shown in Figure 2, a CNN-ViG-based mobile architecture that uses MGC to achieve similar performance to state-of-the-art CNN-ViT-based mobile architectures. Notably, MobileViGv2 can begin mixing tokens globally at much higher resolution stages than existing CNN-ViT-based architectures due to the high speed of MGC.
3. Our results show that a CNN-GNN-based mobile vision architecture can compete with state-of-the-art CNN-ViT-based image classification models and outperform them on downstream tasks. These results include latency and top-1 accuracy on ImageNet-1K [4], object detection and instance segmentation on MS COCO 2017 [15], and semantic segmentation on ADE20K [32].

The remainder of this paper is structured as follows. Section 2 covers recent works in the mobile architecture space. Section 3 provides background on Sparse Vision Graph Attention (SVGA), the ViG module used in MobileViG [19]. Section 4 describes the design of the MGC module and MobileViGv2 architecture. Section 5 describes the experimental setup and results for ImageNet-1K image classification, COCO object detection, COCO image segmentation, and ADE20K semantic segmentation. Additionally, Section 5 includes ablation studies further outlining the improvements of MGC and MobileViGv2 over SVGA and MobileViG. Section 6 summarizes our contributions.

2. Related Work

We break up previous works in the mobile vision space into three categories: CNN-based, CNN-ViT-based, and CNN-GNN-based. The most well known CNN-based methods are MobileNet [10] [23] [9] and EfficientNet [24] [25]. MobileNet introduced depthwise separable convolutions, which separate the convolution operation into a depthwise convolution followed by a pointwise convolution. This approach achieves performance similar to a normal convolution op with significantly lower computational cost. MobileNetv2 [23] built on depthwise separable convolutions with the new inverted residual block. Inverted residuals make residual links less memory-intensive on mobile devices by adding skip links around points where the channel layer is expanded. Hence, these large layers do not have to be saved in memory for future additions. Lastly, MobileNetv3 [9] uses neural architecture search and squeeze and excitation blocks to further improve on MobileNetv2. Like MobileNetv3, EfficientNet [24] and EfficientNetv2 [25] use neural architecture search to produce fast, highly accurate models.

There are many CNN-ViT-based models, but two recent works have achieved remarkable performance with very low mobile latency: EfficientFormerV2 [13] and FastViT [27]. EfficientFormerV2 combines inverted residual blocks with a modified transformer operation and SuperNet architecture search. The modified transformer block uses talking heads and a depthwise convolution on the value matrix to inject local information. FastViT [27] uses a new RepMixer block along with transformers to achieve similar results to that of EfficientFormerV2. The RepMixer block combines a depthwise convolution and convolution-based feed-forward network. Additionally, FastViT makes extensive use of reparamaterization.

To our knowledge, there is currently only one CNN-GNN-based mobile architecture: MobileViG [19]. MobileViG introduced Sparse Vision Graph Attention (SVGA), a vision graph neural network module for statically constructing graphs and performing message passing. For small model sizes, MobileViG appears to compete with state-of-the-art CNN-ViT-based models, but as the model size grows, MobileViG accuracy fails to scale as well as that of CNN-ViT-based counterparts. To address this limitation, we introduce a new CNN-GNN-based mobile architecture, MobileViGv2, which fixes this scaling problem.

3. Background

Before explaining our proposed solution, Mobile Graph Convolution (MGC), to the scaling problem experienced by MobileViG [19], we briefly explain SVGA to highlight the differences between the two approaches better. SVGA can be split into a token mixing step and a feed-forward network. Given an input $X \in \mathbb{R}^{N \times M}$,

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

where $Y \in \mathbb{R}^{N \times M}$ is the output of the token mixer, and W_{in} and W_{out} are fully connected layer weights.

For a single token x_i, the $MRConv$ operation [12] is described as follows,

$$MRConv(x_i) = max(\{x_j - x_i | x_j \in G(x_i)\}) \quad (2)$$

where x_j is a neighboring token to x_i, in the set of neighboring tokens $G(x_i)$.

The output of the token-mixing operation is then passed to a feed-forward network. The feed-forward network is a two-layer MLP expressed as

$$Z = \sigma(XW_1)W_2 + Y \quad (3)$$

where $Z \in \mathbb{R}^{N\times M}$, W_1 and W_2 are fully connected layer weights, and σ is a GeLU activation.

4. Methodology

In this section, we describe the design of Mobile Graph Convolution (MGC) and MobileViGv2. MobileViGv2 leverages the speed of MGC to use graph convolutions in higher resolution stages of the model architecture, resulting in significantly higher model accuracy on ImageNet-1K without a significant drop in latency.

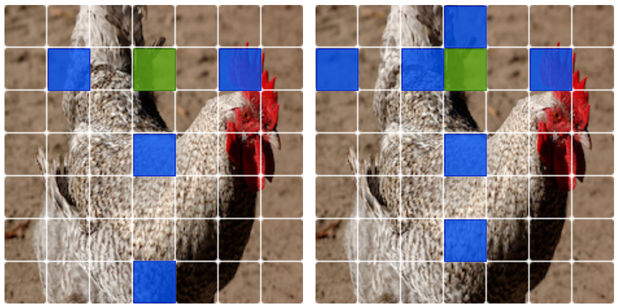
Figure 3. Mobile Graph Convolution (MGC) (left) versus Sparse Vision Graph Attention (SVGA) (right). Each grid is broken up such that the effective receptive field is equal to that of MobileViGv2 at Stage 4. **(left)** The connections made for the green token using MGC ($L = 2$) are shown in blue. **(right)** The connections made for the green token using SVGA ($K = 2$), as used in MobileViG, are shown in blue. The image above was obtained from the ImageNet-1K [4] dataset and has been modified for this paper.

4.1. Mobile Graph Convolution

We propose Mobile Graph Convolution (MGC) as a faster, highly scalable alternative to Sparse Vision Graph Attention (SVGA) [19]. MGC improves on SVGA by increasing graph sparsity and introducing conditional positional encodings to the graph operation.

The MGC algorithm alters equations 1 and 2, leaving 3 unchanged. Equation 1 is altered by adding a conditional positional encoding [2] (CPE), and in equation 2, the cardinality of $G(x_i)\forall x_i$ is reduced by using a different graph construction method. The updated equation for MGC is:

$$Y = \text{MRConv}((X + CPE(X))W_{in})W_{out} + X \quad (4)$$

where CPE is a depthwise convolution and stands for conditional positional encoding.

Unlike the original ViG [6], MobileViG [19] does not use positional encodings. As shown in ViT [5], without positional encodings, the model accuracy of ViTs drops significantly since the self-attention operation becomes permutation invariant. We find this performance drop also holds for graph operations in MobileViG [19] (see Table 3). As such, MGC uses conditional positional encodings (CPE) [2] to encode spatial information before message passing. MGC uses reparameterizable CPE as introduced in FastViT [27]. Before constructing the graph and message passing, a positional encoding is added to the feature map by taking a depthwise convolution of the feature map itself. During inference, the residual link can be merged with the depthwise convolution, saving a fraction of a millisecond. This simple change introduces spatial information in the message-passing step yet adds few parameters and substantially increases performance. For example, when CPE [2] is added to SVGA in MobileViG-B, the top-1 accuracy on ImageNet-1K improves by 0.3%, as seen in Table 3.

The differences in graph construction are straightforward. The SVGA algorithm uses a hyperparameter, K, to determine the distance and density of connections for each token. For token x_i, every K^{th} token to the right in its row and down in its column is connected to it. Thus, the number of connections to x_i grows according to $O(\frac{N+M}{K})$, where N and M are the dimensions of the input image. Thus, for a fixed value of K, the cardinality of $G(x_i)$ grows linearly with respect to the input resolution. As such, using SVGA for higher resolutions requires more computation during graph construction, making it difficult to scale to higher input resolutions while maintaining competitive inference latency. One could fix this by scaling K with input resolution, but a simple, more straightforward approach is to fix the number of possible connections per token regardless of input size.

Following this approach, each input token has only five connections in MGC: one self-connection, two long-range links to the left and right of the token on its row, and two long-range links up and down from the token on its column. This can be seen in greater detail in Figure 2c and Figure 3 for an input resolution of 7×7. For larger input resolutions, the distance of the long-range links can be increased to gather information from regions further away for each token. As implemented in MobileViG, SVGA uses 7 connections per token ($K = 2$), while MGC uses only 5 ($L = 2$), contributing to the speedup over SVGA. We also found that lowering the number of connections improves model performance, which can likely be attributed to reducing over-smoothing. For example, when swapping in SVGA for MGC in MobileViGv2-B, the resulting model is 0.2 milliseconds slower with 0.1% worse top-1 classification accuracy on ImageNet-1K.

Table 1. Results of MobileViGv2 and other mobile architectures on ImageNet-1K classification task roughly grouped by NPU latency of an iPhone13 Mini using the ModelBench [26] application. Type indicates whether the model is CNN-based, CNN-ViT-based, or CNN-GNN-based. Params lists the number of model parameters in millions. GMACs lists the number of MACs in billions. Gray highlights indicate the contributions of this paper. The Top-1 accuracy results for MobileViGv2 models are averaged over three experiments, and there is about a 0.1% fluctuation between training seeds. Missing entries could not be profiled on the iPhone 13 Mini (iOS 16). ↓ means the lower, the better. ↑ means the higher, the better.

Model	**Type**	**Params (M)**	**GMACs**	**iPhone 13 Latency ↓ (ms)**	**Top-1 (%) ↑**
MobileNetV2x1.0 [23]	CNN	3.5	0.3	0.8	71.8
EdgeViT-XXS [20]	CNN-ViT	4.1	0.6	-	74.4
FastViT-T8 [27]	CNN-ViT	3.6	0.7	0.8	76.7
EfficientFormerV2-S0 [13]	CNN-ViT	3.5	0.4	0.8	75.7
MobileViG-Ti [19]	CNN-GNN	5.2	0.7	0.9	75.7
MobileViGv2-Ti	CNN-GNN	5.6	0.6	0.9	77.7
MobileNetV2x1.4 [23]	CNN	6.1	0.6	1.1	74.7
EdgeViT-XS [20]	CNN-ViT	6.7	1.1	-	77.5
EfficientFormerV2-S1 [13]	CNN-ViT	6.1	0.7	1.0	79.0
MobileViG-S [19]	CNN-GNN	7.2	1.0	1.1	78.2
MobileViGv2-S	CNN-GNN	7.7	0.9	1.1	79.8
EfficientNet-B0 [24]	CNN	5.3	0.4	1.5	77.7
EdgeViT-S [20]	CNN-ViT	11.1	1.9	-	81.0
EfficientFormer-L1 [14]	CNN-ViT	12.3	1.3	1.3	79.2
FastViT-T12 [27]	CNN-ViT	6.8	1.4	1.2	80.3
FastViT-S12 [27]	CNN-ViT	8.8	1.8	1.4	80.9
FastViT-SA12 [27]	CNN-ViT	10.9	1.9	1.6	81.9
EfficientFormerV2-S2 [13]	CNN-ViT	12.6	1.3	1.5	81.6
MobileViG-M [19]	CNN-GNN	14.0	1.5	1.5	80.6
MobileViGv2-M	CNN-GNN	15.4	1.6	1.6	81.7
EfficientNet-B3 [24]	CNN	12.2	2.0	4.8	81.6
MobileViTv2-1.0 [18]	CNN-ViT	4.9	1.8	3.0	78.1
MobileViTv2-2.0 [18]	CNN-ViT	18.5	7.5	6.3	82.4
EfficientFormer-L3 [14]	CNN-ViT	31.3	3.9	2.6	82.4
EfficientFormer-L7 [14]	CNN-ViT	82.1	10.2	6.5	83.3
FastViT-SA24 [27]	CNN-ViT	20.6	3.8	2.7	83.4
EfficientFormerV2-L [14]	CNN-ViT	26.1	2.6	2.4	83.3
MobileViG-B [19]	CNN-GNN	26.7	2.8	2.4	82.6
MobileViGv2-B	CNN-GNN	27.7	3.6	2.7	83.4

4.2. MobileViGv2 Architecture

The MobileViGv2 architecture, as shown in Figure 2, can be broken into four main stages, where processing occurs at a single resolution in a given stage. Within each stage, a mixture of inverted residuals and Mobile Graph Convolutions (MGCs) are used.

The entry point into the architecture is the stem. The stem takes the input image and downsamples it 4× using convolutions with stride equal to two. The output of the stem is fed to Stage 1, which consists of N_1 inverted residuals as described in Figure 2a. Between each stage is another convolution-based downsampling step. Stages 2, 3, and 4 each start with a sequence of N_i inverted residuals, where i is the stage number. The output of the inverted residual sequence is then fed through M_i MGCs, as shown in Fig-

Table 2. Results of MobileViGv2 and other mobile architectures on COCO object detection, COCO instance segmentation tasks, and ADE20K semantic segmentation. Parameters lists the number of backbone parameters in millions, not including Mask-RCNN or Semantic FPN. AP^{box} and AP^{mask} scores are for object detection and instance segmentation on MS COCO 2017 [15]. $mIoU$ scores are for semantic segmentation on ADE20K [32]. Shaded regions show the contributions of this paper. A (-) denotes a model that did not report these results.

Model	**Params (M)**	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}	$mIoU$
EfficientFormer-L1 [14]	12.3	37.9	59.0	40.1	34.6	55.8	36.9	38.9
FastViT-SA12 [27]	10.9	38.9	60.5	42.2	35.9	57.6	38.1	38.0
MobileViG-M [19]	14.0	41.3	62.8	45.1	38.1	60.1	40.8	-
MobileViGv2-M	15.4	**42.5**	**63.9**	**46.3**	**38.8**	**60.8**	**41.7**	**42.9**
EfficientFormer-L3 [14]	31.3	41.4	63.9	44.7	38.1	61.0	40.4	43.5
FastViT-SA24 [27]	20.8	42.0	63.5	45.8	38.0	60.5	40.5	41.0
MobileViG-B [19]	26.7	42.0	64.3	46.0	38.9	61.4	41.6	-
MobileViGv2-B	27.7	**43.0**	**64.9**	**47.1**	**39.6**	**62.2**	**42.7**	**44.3**

ure 2b. After Stage 4, an average pooling step followed by a feed-forward network produces the predicted class of the input image.

To achieve different model sizes, the channel width of each stage and values of N_i and M_i are changed. There are four different MobileViGv2 configurations, MobileViGv2-Ti, MobileViGv2-S, MobileViGv2-M, and MobileViGv2-B. Note that all inverted residual blocks use an expansion factor of 4. Additionally, all FFNs used in MGC, as shown in Figure 2b, use an expansion factor of 4. While a different mixture of widths and expansion factors may produce better results, as could be found using a neural-architecture search, our work aims to show the potential of using graph convolutions in mobile vision architectures, not finding the optimal model structure. We leave this task of using NAS for future work.

5. Experimental Results

In this section, we describe the setup and results for MobileViGv2 experiments on ImageNet-1K [4] classification, COCO object detection, COCO instance segmentation [15], and ADE20K [32] semantic segmentation tasks.

5.1. Image Classification

MobileViGv2 is implemented using PyTorch 1.12.1 [21] and the Timm library [28]. Each model is trained using 16 NVIDIA A100 GPUs with an effective batch size of 2048. The models are trained from scratch for 300 epochs on the ImageNet-1K dataset with a standard training and inference resolution of 224×224. We use the AdamW [16] optimizer and a learning rate of 2e-3 with a cosine annealing schedule. Like many CNN-ViT-based [13] [19] mobile architectures, we use RegNetY-16GF [22] for knowledge distillation. Our data augmentation pipeline includes RandAugment [3], Mixup [30], Cutmix [29], random erasing [31], and repeated augment [8]. To measure inference latency, all models are packaged as MLModels using CoreML and profiled on the same iPhone 13 Mini (iOS 16) using ModelBench [26]. We use the following ModelBench settings to profile each model: 50 inference rounds, 50 inferences per round, and a low/high trim of 10. Table 1 shows ImageNet-1K classification results for MobileViGv2 and similar mobile vision architectures. Models are roughly grouped by latency.

For models with an inference latency under 1 ms, MobileViGv2-Ti has the highest accuracy, with the next closest model, FastViT-T8 [27], being a full 1% behind. Additionally, MobileViGv2-Ti is 2% more accurate than MobileViG-Ti for the same inference latency. When compared to EfficientFormerV2-S1 [13], MobileViGv2-S has 0.8% higher accuracy for a similar inference latency. MobileViGv2-S is also 1.7% more accurate than MobileViG-S for the same inference latency.

The wide performance gap shown in Figure 1 indicates that MGC and the new model configuration of MobileViGv2 successfully solve the scaling problem experienced by MobileViG. This, along with better or comparable performance to CNN-ViT-based models such as EfficientFormerV2 [13] and FastViT [27], show the potential of CNN-GNN-based architectures to compete in the mobile vision space.

5.2. Object Detection and Instance Segmentation

We show that MobileViGv2 generalizes well to downstream tasks by using it as a backbone for object detection and instance segmentation on the MS COCO 2017 [15] dataset, which contains training and validation sets of 118K and

Table 3. An ablation study of the effects of conditional positional encodings, higher resolution graphers, and different graph construction methods on MobileViG-B and MobileViGv2-B. A checkmark indicates this component was used in the experiment. A (-) indicates this component was not used. 1-Stage indicates that graph convolutions were only used in Stage 4 of the model, while 3-Stage indicates that graph convolutions were used in stages 2, 3, and 4. The * indicates that this model swapped the graph convolution for a fully connected layer.

Base Model	Params (M)	Latency (ms)	SVGA	MGC	CPE	1-Stage	3-Stage	Top-1 (%)
MobileViG-B	26.7	2.4	✓	-	-	✓	-	82.6
MobileViG-B	27.6	2.9	✓	-	-	-	✓	83.2
MobileViG-B	26.8	2.4	✓	-	✓	✓	-	82.9
MobileViG-B	27.7	2.9	✓	-	✓	-	✓	83.3
MobileViG-B*	28.6	2.5	-	-	-	-	✓	83.0
MobileViGv2-B	27.7	2.7	-	✓	✓	-	✓	**83.4**

5K images. We use pre-trained MobileViGv2 backbones with Mask-RCNN [7] for training. Each model is trained on 16 NVIDIA A100 GPUs for 12 epochs with an effective batch size of 16. We use AdamW [16] optimizer, an initial learning rate of 2e-4, and a standard image resolution of 1333×800.

As shown in Table 2, MobileViGv2-M hits an AP^{box} and AP^{mask} of 42.5 and 38.8, respectively. This is 1.2 and 0.7 points higher than MobileViG-M [19] and 3.6 and 2.9 points higher than FastViT-SA12 [27]. MobileViGv2-B achieves an AP^{box} and AP^{mask} of 43.0 and 39.6, respectively. This is 1.0 and 0.7 points higher than MobileViG-B and 1.0 and 1.6 points higher than the comparable FastViT backbone, FastViT-SA24.

These results show that MobileViGv2 generalizes well to downstream tasks. Compared to competitive CNN-ViT-based models like FastViT [27], MobileViGv2 performs significantly better on these downstream tasks, even though the performance in image classification is comparable.

5.3. Semantic Segmentation

We also show that MobileViGv2 generalizes well to semantic segmentation on the ADE20K dataset [32], which contains 20K training images and 2K validation images with 150 semantic categories. For training, we use 8 NVIDIA RTX 6000 Ada generation GPUs, the AdamW optimizer, and a learning rate of 2e-4 with polynomial decay. We use MobileViGv2 as a backbone with Semantic FPN [11] as the segmentation decoder. The backbone is initialized with pre-trained weights on ImageNet-1K, and the model is trained for 40K iterations.

As shown in Table 2, MobileViGv2-M outperforms FastViT-SA12 [27] by 4.9% $mIoU$ and outperforms EfficientFormer-L1 [14] by 4% $mIoU$. Additionally, MobileViGv2-B outperforms FastViT-SA24 by 3.3% $mIoU$ and outperforms EfficientFormer-L3 by 0.8% $mIoU$. Again, when compared to competitive CNN-ViT-based models like FastViT [27] and EfficientFormer [14], MobileViGv2 performs significantly better on this downstream task even though the performance in image classification is comparable.

5.4. Ablation Studies

We perform ablation studies to show the benefits of MGC over SVGA and to demonstrate that graph convolutions provide benefits over a simple feed-forward network solution. A summary of these results can be found in Table 3.

Starting with MobileViG-B as a base model, we try using SVGA-style graph convolutions in stages 2, 3, and 4 of the model while keeping the number of parameters the same. We adjust the number of blocks in each stage and the channel depth to keep the number of parameters similar. The resulting model achieves a top-1 accuracy on ImageNet-1K of 83.2%, 0.6% higher than MobileViG-B. However, this model has an inference latency of 2.9 milliseconds, significantly slower than the 2.3 milliseconds of MobileViG-B without catching up to the top-1 performance of FastViT [27] and EfficientFormerV2 [13].

We also try adding CPE to SVGA in MobileViG-B and find that it improves model performance by 0.3%. We then combine both CPE and 3-stage SVGA, which results in a top-1 accuracy of 83.3%. Even though SVGA makes more connections than MGC, it performs slightly worse than MGC when used with the same model settings. We expect that this occurs due to over-smoothing from the higher connection count.

To get the benefits of using more stages without a significant hit to latency, we use MGC, which uses sparser graphs and CPE, in MobileViGv2-B, to achieve the best performance of 83.4%. This final configuration has the same top-1 performance and mobile latency as FastViT-SA24 [27].

To verify that graph convolutions boost model performance, we swap each graph convolution with a fully connected layer of the same expansion size and use this in stages 2, 3, and 4 of the model. This experiment is marked as MobileViG-B* in Table 3. We find that this model achieves an accuracy of only 83.0%, which is 0.2% less than using SVGA in three stages and 0.4% less than using MGC in three stages. This shows that graph convolutions are improving model performance.

6. Conclusion

In this work, we have proposed Mobile Graph Convolution (MGC) and MobileViGv2, a model architecture that uses MGC and competes with state-of-the-art CNN-ViT-based mobile architectures. MGC uses a sparser, static graph construction method than SVGA, resulting in faster inference speeds. Additionally, MGC introduces conditional positional encodings to the graph operation, considerably boosting model accuracy with only a slight increase in the number of parameters. With these changes, MGC can be used in much higher resolution stages than SVGA without significantly impacting latency.

MobileViGv2 takes advantage of this by using MGC in the last three processing stages. Earlier global processing and the sharing of spatial information during message passing through CPE solves the scaling problem experienced by MobileViG, thus making MobileViGv2 a genuine competitor to state-of-the-art CNN-ViT-based mobile architectures and, consequently, MGC a competitor to self-attention in the mobile vision model space.

References

[1] Yinpeng Chen, Xiyang Dai, Dongdong Chen, Mengchen Liu, Xiaoyi Dong, Lu Yuan, and Zicheng Liu. Mobile-former: Bridging mobilenet and transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5270–5279, 2022. 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

[3] 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. 6

[4] 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. 3, 4, 6

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

[6] 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. 2, 3, 4

[7] 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

[8] 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. 6

[9] Andrew Howard, Mark Sandler, Grace Chu, Liang-Chieh Chen, Bo Chen, Mingxing Tan, Weijun Wang, Yukun Zhu, Ruoming Pang, Vijay Vasudevan, Quoc V. Le, and Hartwig Adam. Searching for mobilenetv3. *CoRR*, abs/1905.02244, 2019. 3

[10] 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, 3

[11] 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

[12] 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. 3

[13] 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. 1, 2, 3, 5, 6, 7

[14] 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. 1, 5, 6, 7

[15] 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. 3, 6

[16] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 6, 7

[17] Sachin Mehta and Mohammad Rastegari. Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*, 2021. 1

[18] Sachin Mehta and Mohammad Rastegari. Separable self-attention for mobile vision transformers. *arXiv preprint arXiv:2206.02680*, 2022. 1, 5

[19] 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, 2023. 1, 2, 3, 4, 5, 6, 7

[20] 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. 5

[21] Adam Paszke et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 6

[22] 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

[23] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520, 2018. 1, 3, 5

[24] Mingxing Tan and Quoc Le. Efficientnet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114. PMLR, 2019. 1, 3, 5

[25] Mingxing Tan and Quoc Le. Efficientnetv2: Smaller models and faster training. In *International conference on machine learning*, pages 10096–10106. PMLR, 2021. 1, 3

[26] Pavan Kumar Anasosalu Vasu, James Gabriel, Jeff Zhu, Oncel Tuzel, and Anurag Ranjan. An improved one millisecond mobile backbone. *arXiv preprint arXiv:2206.04040*, 2022. 5, 6

[27] 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, 2023. 2, 3, 4, 5, 6, 7

[28] Ross Wightman. PyTorch Image Models. `https://github.com/rwightman/pytorch-image-models`, 2019. 6

[29] 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. 6

[30] Hongyi Zhang, Moustapha Cisse, Yann N. Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *International Conference on Learning Representations*, 2018. 6

[31] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *Proceedings of the AAAI conference on artificial intelligence*, pages 13001–13008, 2020. 6

[32] 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. 3, 6, 7