Vision HGNN: An Image is More than a Graph of Nodes

Yan Han[1], Peihao Wang[1], Souvik Kundu[2], Ying Ding[1], Zhangyang Wang[1]
[1]University of Texas at Austin, [2]Intel Labs
{yh9442,peihaowang,atlaswang}@utexas.edu
ying.ding@ischool.utexas.edu souvikk.kundu@intel.com

Abstract

The realm of graph-based modeling has proven its adaptability across diverse real-world data types. However, its applicability to general computer vision tasks had been limited until the introduction of the Vision Graph Neural Network (ViG). ViG divides input images into patches, conceptualized as nodes, constructing a graph through connections to nearest neighbors. Nonetheless, this method of graph construction confines itself to simple pairwise relationships, leading to surplus edges and unwarranted memory and computation expenses. In this paper, we enhance ViG by transcending conventional "pairwise" linkages and harnessing the power of the ***hypergraph*** *to encapsulate image information. Our objective is to encompass more intricate inter-patch associations. In both training and inference phases, we adeptly establish and update the hypergraph structure using the Fuzzy C-Means method, ensuring minimal computational burden. This augmentation yields the* ***V****ision* ***H****yper****G****raph* ***N****eural* ***N****etwork (****ViHGNN****). The model's efficacy is empirically substantiated through its state-of-the-art performance on both image classification and object detection tasks, courtesy of the hypergraph structure learning module that uncovers higher-order relationships. Our code is available at: https://github.com/VITA-Group/ViHGNN.*

1. Introduction

The rapid strides made in deep learning have ushered in remarkable successes across diverse computer vision models. These encompass Convolutional Neural Networks (CNNs)[35, 33, 22], Vision Transformers (ViTs)[11, 2], and MLP-based vision models [54, 55]. In these networks, input images find representation either as a regular grid of pixels in the Euclidean space or as sequences of patches.

Nonetheless, the potential for more versatile image processing through a graph structure remains untapped. A recent development, ViG [18], has ingeniously harnessed Graph Neural Networks (GNNs) for substantial advancements in large-scale visual tasks. To mitigate excessive node proliferation, ViG borrows the partitioning concept from ViT, segmenting the image into smaller patches and designating each patch as a node. Consequently, the ViG framework forges connections between nodes and their closest neighbors, constructing an adaptable graph. Furthermore, the graph itself emerges as a generalized data structure, encompassing grids and sequences as distinct instances within its broader graph context.

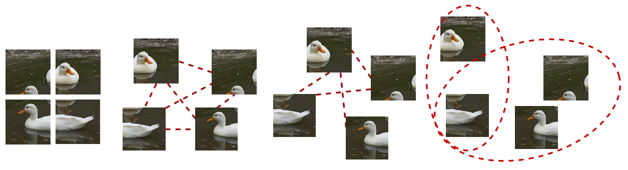

Figure 1. The illustration of image topologies modeled in difference visual backbones. (a) CNNs treat images as regular grids, (b) ViT parses images as full-connected graphs, (c) ViGs process images as sparse graphs with pairwise edges, while (d) our ViHGNN models images as a hypergraph, a "more universal" structure.

While the success of ViG has effectively showcased the advantages of treating an image as a graph in terms of enhancing flexibility and effectiveness in visual perception, there exist limitations to using a graph as the optimal data structure for image representation. These limitations are rooted in two primary reasons:

- **Complexity of Relationships**: A fundamental constraint of a simple graph lies in its ability to exclusively connect two nodes, thereby solely accommodating pairwise relationships. This inadequacy is evident when it comes to modeling high-order relations inherent in images. Consider the task of object recognition in computer vision, where an image is typically divided into patches, each representing a segment of the object. This division introduces intricate inter- and intra-object dependencies that a straightforward graph struggles to capture. This complexity arises from the interplay of patches belonging to the same or distinct objects. Consequently, a conventional graph structure finds it challenging to effectively model such multi-faceted relationships between patches.

- **Redundancy in Edge Generation**: Another drawback tied to simple graph representations pertains to the generation of redundant edges during image portrayal. ViG constructs its graph by identifying the nearest neighbors for each patch node and subsequently forming edges between these node pairs. During this process, all image patches are transformed into feature vectors, and feature distances are computed to determine nearest neighbors. This approach becomes problematic when considering that an object within an image is often an amalgamation of multiple patches, yielding similar feature vectors for patches belonging to the same object. Consequently, the graph construction method can inadvertently give rise to extraneous edges. In a worst-case scenario, an image with n patches could potentially generate n^2 edges, leading to quadratic complexity.

The aforementioned considerations underscore the limitations inherent in relying on a simple graph as the fundamental data structure for image representation. Consequently, it is imperative to explore alternative methodologies that effectively address these concerns, thus enhancing the refinement of visual perception models. In light of this, we propose a robust evolution of ViG, where the hypergraph assumes the role of image representation. This innovative framework, named **V**ision **H**yper**G**raph **N**eural **N**etwork (**ViHGNN**), introduces a dynamic approach to image representation. Specifically, a hypergraph serves as a generalized extension of a graph, characterized by a collection of nodes and hyperedges. In contrast to the pairwise connections in a simple graph, hyperedges within a hypergraph can link any number of nodes. Essentially, a graph can be viewed as a specialized form of a hypergraph, only accounting for pairwise connections between data points. Distinctively, hypergraphs exhibit superior aptitude in capturing the intricate correlations existing within images, transcending the limitations of pairwise relations. For tangible illustrations, please refer to Figure 1.

However, a fundamental challenge persists: *determining the optimal hypergraph structure for image representation*. This encapsulates a compelling "chicken-and-egg" quandary: the aspiration to utilize the hypergraph for image representation juxtaposed with the absence of a readily available hypergraph structure. Inspired by ViG's graph construction approach, we start by utilizing the patch features to construct the initial hypergraph. Subsequently, patch embeddings are generated by the initial hypergraph structure, and then these embeddings are harnessed for the construction of a renewed hypergraph structure. Crucially, both the patch embeddings and the hypergraph structure undergo dynamic updates, culminating in a self-reinforcing "feedback loop" of learning processes. In our ViHGNN framework, we choose the Fuzzy C-Means (see Sec. 3.3.) to construct and update the hypergraph structure, incurring negligible computational overhead. Our contributions are summarized as follows:

- We advance beyond the ViG framework by introducing a novel paradigm termed ViHGNN, which interprets an image as a dynamic hypergraph. Unlike ViG, ViHGNN not only captures higher-order relationships within images but also mitigates redundant memory and computational expenses linked to graph structures.
- To establish a robust hypergraph representation for images, we seamlessly integrate an adaptive hypergraph structure learning module within the framework, enhancing representation with little incurring overhead. This hypergraphical portrayal of images yields discernible advantages for downstream visual tasks.
- We execute comprehensive experiments to underscore the efficacy of the ViHGNN model across visual tasks, including image classification and object detection. Specifically, our ViHGNN model achieves a top-1 accuracy of 83.9% in the ImageNet classification task and an impressive 43.1% Average Precision (AP) in the COCO object detection task.

2. Related Work

Network Architectures in Computer Vision. CNNs [35, 33, 22] used to be the de-facto backbone in computer vision, widely adopted by image classification [35, 33], object detection [47], semantic segmentation [43] and more. With the rapid development over the past decade, the representative CNNs include ResNet [22], MobileNet [25] and NAS-searched networks [73, 84]. Recently, inspired by the success of Transformer architectures in the NLP field, the Vision Transformer was introduced for vision tasks [17, 11, 3, 4]. The pioneering work of ViT [11] directly applies a Transformer architecture on non-overlapping medium-sized image patches for image classification. Since then, many variants of ViT have been proposed to improve the performance of vision tasks. The major improvements include pyramid architecture [62, 41], local attention [19, 41] and position encoding [67]. Besides, MLP architectures are also explored in computer vision [54, 55, 5, 37, 15, 53], so are the larger-kernel CNNs [42, 9, 40, 28].

Graph/Hypergraph for Computer Vision. While Graph Neural Networks (GNNs) traditionally find application in social networks [16], citation networks [49], and biochemical graphs [59], their adoption has extended to the realm of computer vision. Notably, GNNs have been explored for tasks like point cloud classification and segmentation [34, 63], as well as scene graph generation which

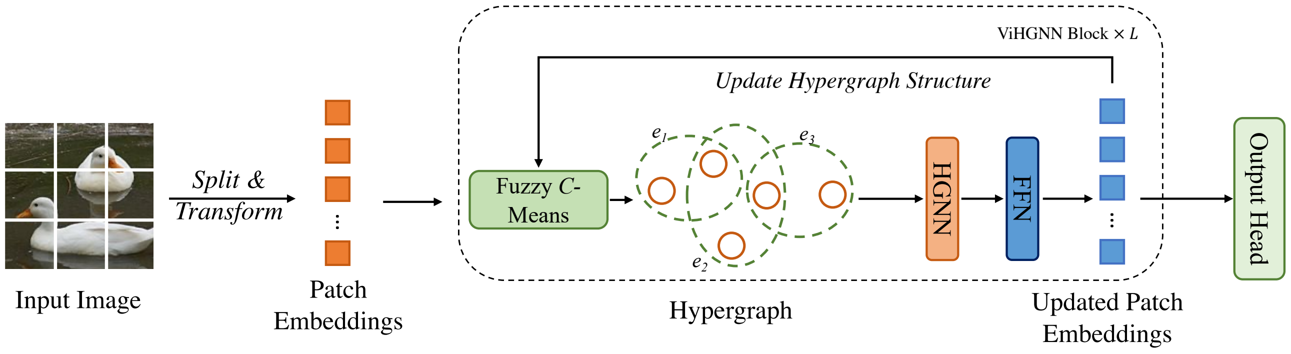

Figure 2. Overview of our ViHGNN model. HGNN and FNN represent hypergraph neural network and feed-forward network, respectively.

involves combining object detectors and GNNs [69, 72]. Moreover, GNNs prove useful in human action recognition through joint-connected graphs [31, 71].

Hypergraph neural networks (HGNNs) are a related paradigm, as hypergraphs encompass graphs [64, 20]. In computer vision, HGNNs find extensive utility. For instance, in image retrieval, vertices represent images and hyperedges are generated based on feature correlations [30]. Similarly, 3D object classification employs vertex representations for objects and uses view relationships to create hyperedges [29, 14]. Person re-identification leverages feature correlations for hypergraph construction [80]. Yet, while GNNs and HGNNs are prominent in various domains, ViG is among the few methods that directly process image data [18]. Nonetheless, the integration of hypergraphs for image representation remains an open challenge.

Graph/Hypergraph Structure Learning. GNNs and HGNNs are pivotal for analyzing graph and hypergraph data. Their effectiveness is yet intertwined with the quality of the given graph and hypergraph structures, respectively. Recent efforts have thus focused on graph/hypergraph structure learning [83]. For GNNs, methods like [6, 13] jointly and iteratively learn graph structure and graph node embedding, under certain assumptions on the graph adjacency matrix such as low-rank, sparsity, or smoothness. Other methods [45, 81] learn a metric or distribution to determine the existence of an edge in order to modify the graph structure.

In contrast, hypergraph structure learning is less explored. DHSL [79, 78] uses dual optimization to simultaneously learn the label projection matrix and the hypergraph structure. DHGNN [32] employs K-Means and k-NN to adaptively construct the hypergraph. HERALD [77] optimizes hypergraph Laplacian matrices. These methods optimize hypergraph structures but face drawbacks like non-convexity and high computational cost. Traditional non-convex optimization in dynamic hypergraph structures causes convergence issues, while high time complexity, particularly in DHSL [79, 78], raises concerns over time-efficient hypergraph updates during iterative optimization.

3. Method: An Image is Worth a Hypergraph

In this section, we first introduce the notations and definitions used throughout the paper. Then we present the Vision HyperGraph Neural Network (ViHGNN) framework. The overall pipeline is depicted in Figure 2.

3.1. Notations and Definitions

Notations. Let $\boldsymbol{I} \in \mathbb{R}^{H\times W\times 3}$ denote an image of a size $H \times W$, and we uniformly divide each image into N patches. After transforming each patch into a feature vector $\boldsymbol{x}_i \in \mathbb{R}^D$, we obtain a data matrix $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$ where each column corresponds to the feature vector of a patch, where D is the feature dimension and $i = 1, 2, \cdots, N$.

Hypergraph Definition. A hypergraph is defined as $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is a set of N unique vertices and $\mathcal{E}$ is a set of E hyperedges. Unlike ordinary graphs, each hyperedge $e \in \mathcal{E}$ can connect more than two vertices. The hypergraph can be represented by an incidence matrix $\boldsymbol{H} \in \{0,1\}^{N\times E}$, where $\boldsymbol{H}_{ie} = 1$ if the hyperedge $e \in \mathcal{E}$ contains a vertex $v_i \in \mathcal{V}$, otherwise $\boldsymbol{H}_{ie} = 0$. For each vertex and hyperedge, their degrees $\boldsymbol{D}_{ii}$ and $\boldsymbol{B}_{ee}$ are defined as $\boldsymbol{D}_{ii} = \sum_{e=1}^{E} \boldsymbol{H}_{ie}$ and $\boldsymbol{B}_{ee} = \sum_{i=1}^{N} = \boldsymbol{H}_{ie}$, respectively. Let diagonal matrices $\boldsymbol{D}$ and $\boldsymbol{B}$ be degree matrices. $\boldsymbol{D}_{ii}$ and $\boldsymbol{B}_{ee}$ are the i-th and e-th diagonal elements of $\boldsymbol{D}$ and $\boldsymbol{B}$, respectively. For an image, we view image patches as a set of unordered nodes and define the node set $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$. We convert image $\boldsymbol{I}$ to a hypergraph $\mathcal{G}(\boldsymbol{I})$ to denote the image's hypergraph structure. The hypergraph construction process will be introduced in detail in Sec. 3.3.

Hypergraph Neural Network (HNN). A general hypergraph convolutional layer [12, 70, 27, 7, 60] is defined as:

$$\boldsymbol{X}^{(l+1)} = \boldsymbol{D}^{-1/2}\boldsymbol{H}\sigma\left(\boldsymbol{W}\boldsymbol{B}^{-1}\boldsymbol{H}^{T}\sigma(\boldsymbol{D}^{-1/2}\boldsymbol{X}^{(l)}\boldsymbol{\Theta}_1)\boldsymbol{\Theta}_2\right), \quad (1)$$

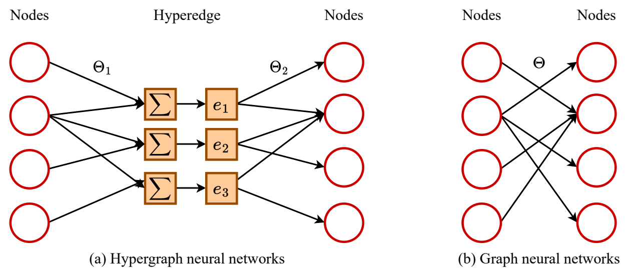

Figure 3. The comparison of message passing mechanisms in (a) HGNNs and (b) GNNs.

where $\boldsymbol{W}$ is the hyperedge weight matrix, $\boldsymbol{\Theta}_1$ and $\boldsymbol{\Theta}_2$ are the learnable parameter of the HGNN layer, σ is an activation function, $\boldsymbol{X}^l$ and $\boldsymbol{X}^{l+1}$ denote the input and output node embeddings, respectively. The hypergraph convolution can be viewed as a two-stage message passing, flowing information in a "node-hyperedge-node" manner. The multiplication of $\boldsymbol{H}^T$ achieves the information aggregation from the nodes to the hyperedges. Multiplying $\boldsymbol{H}$ can be viewed as information aggregation from hyperedges to nodes. In between, $\boldsymbol{D}$ and $\boldsymbol{B}$ conduct normalization.

The difference in message passing between vanilla GNNs and HGNNs is illustrated in Figure 3. At first glance, Equation 1 just runs a simple GNN on the clique expansion of the hypergraph. However, messages in HGNN will be received and transformed on both node and hyperedge sides, which cannot be implemented by clique expansion [7].

3.2. Overview: ViHGNN Framework

The overall framework of ViHGNN is shown in Figure 2. Note that, the components that are re-used from the original ViG are omitted. For details about other components, please refer to ViG [18]. Here we focus on describing the two essential improvements of ViHGNN compared with the original ViG: 1) hypergraph structure representation of an image, 2) hypergraph structure learning.

During one iteration, the input image $\boldsymbol{I}$ is first divided into N patches, and these patches are transformed into a set of feature vectors, shown as patch embeddings in the Figure 2. Then these patches are clustered via Fuzzy C-Means [1] to construct the hypergraph $G(\boldsymbol{I})$. Then a hypergraph convolutional layer can exchange information between nodes via a two-stage node-hyperedge-node message passing scheme. We apply a feed-forward network (FFN) after the hypergraph convolutional layer to project the node features into the same domain and increase the feature diversity. The FFN layers further encourage the feature transformation capacity and relieve the over-smoothing phenomenon [10, 61, 50]. The output updated patch embeddings are fed into the Fuzzy C-Means in turn, to update the hypergraph $G(\boldsymbol{I})$. Sec. 3.4 details the rationale behind this proposed back-and-forth process. The ViHGNN iterates the above task by progressively updating the hypergraph and evaluating hypergraph structured feature representation. We append a pooling and MLP block at the end to regress the output logits following [18]. We provide various architecture configurations of ViHGNN, listed in Table 2.

3.3. Hypergraph Structure of Image

We next explain how to construct the hypergraph $G(\boldsymbol{I})$ for an image $\boldsymbol{I}$. From the aforementioned, an image can be regarded as a set of unordered nodes $\mathcal{V} = \{v_1, v_2, \cdots, v_N\}$, and associated features $\boldsymbol{X} = [\boldsymbol{x}_1, \boldsymbol{x}_2, \cdots, \boldsymbol{x}_N]$. We construct a hypergraph to infer patch topology, drawing inspiration from ViG's graph construction approach. Similar to ViG's distance-based strategy, we utilize node feature distances for hypergraph construction. However, we deviate by employing the Fuzzy C-Means method to generate node clusters. These overlapping clusters represent image patch sets, akin to hyperedges. This configuration transforms the hypergraph into an ensemble of distinct node clusters. Importantly, in contrast to K-Means, Fuzzy C-Means allows clusters to have non-empty intersections. Consequently, nodes may feature in multiple hyperedges, facilitating intricate and varied inter-object relationships among image patches.

We restate the advantages of graph representation of the image including: 1) hypergraph is a generalized data structure where a grid, sequence and even graph can be viewed as a special case of hypergraph; 2) hypergraph is more flexible than graph to model the complex high-order relations in the image; 3) the advanced research on HGNN may be leveraged in efficient graph representation of visual tasks.

3.4. Adaptive Hypergraph Structure Learning

The hypergraph structure, while powerful for modeling intricate correlations among image patches, relies on the assumption that nodes connected by the same hyperedge possess similar representations. However, constructing this structure is challenging as it hinges on meaningful and robust patch embeddings, which aren't available before the learning process. Furthermore, noisy information in the hypergraph could integrate into learned node representations, potentially degrading the ViHGNN model's performance. This could lead to shared hyperedges among different objects, reducing image discrimination and impairing classification and object detection tasks.

To address this, optimizing the hypergraph structure becomes crucial. Eliminating or reducing task-irrelevant connections while enhancing influential ones is essential. Achieving an accurate and complete representation of high-order correlations in patches is the goal. However, this poses a "chicken-and-egg" problem: generating a robust hypergraph depends on patch embeddings, while meaningful patch embeddings depend on the hypergraph structure. To overcome this, we establish a "feedback loop"

Table 1. Detailed settings of Isotropic ViHGNN series. D: feature dimension, h: hidden dimension ratio in FFN, E: number of hyperedges in HGNN, $H \times W$: input image size. 'Ti' denotes tiny, 'S' denotes small, 'M' denotes medium, and 'B' denotes base.

Model	Depth	Dimension D	Hyperedges (E)	Parameters (M)	FLOPs (B)
Isotropic ViHGNN-Ti	12	192	50	8.2	1.8
Isotropic ViHGNN-S	16	320	50	23.2	5.6
Isotropic ViHGNN-B	16	640	50	88.1	19.4

Table 2. Detailed settings of Pyramid ViHGNN series. D: feature dimension, h: hidden dimension ratio in FFN, E: number of hyperedges in HGNN, $H \times W$: input image size. 'Ti' denotes tiny, 'S' denotes small, 'M' denotes medium, and 'B' denotes base.

Stage	Output size	Pyramid ViHGNN-Ti	Pyramid ViHGNN-S	Pyramid ViHGNN-M	Pyramid ViHGNN-B
Stem	$\frac{H}{4} \times \frac{W}{4}$	Conv×3	Conv×3	Conv×3	Conv×3
Stage 1	$\frac{H}{4} \times \frac{W}{4}$	$\begin{bmatrix} D=48 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=80 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=96 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=128 \\ h=4 \\ E=50 \end{bmatrix} \times 2$
Downsample	$\frac{H}{8} \times \frac{W}{8}$	Conv	Conv	Conv	Conv
Stage 2	$\frac{H}{8} \times \frac{W}{8}$	$\begin{bmatrix} D=96 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=160 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=192 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=256 \\ h=4 \\ E=50 \end{bmatrix} \times 2$
Downsample	$\frac{H}{16} \times \frac{W}{16}$	Conv	Conv	Conv	Conv
Stage 3	$\frac{H}{16} \times \frac{W}{16}$	$\begin{bmatrix} D=240 \\ h=4 \\ E=50 \end{bmatrix} \times 6$	$\begin{bmatrix} D=400 \\ h=4 \\ E=50 \end{bmatrix} \times 6$	$\begin{bmatrix} D=384 \\ h=4 \\ E=50 \end{bmatrix} \times 16$	$\begin{bmatrix} D=512 \\ h=4 \\ E=50 \end{bmatrix} \times 18$
Downsample	$\frac{H}{32} \times \frac{W}{32}$	Conv	Conv	Conv	Conv
Stage 4	$\frac{H}{32} \times \frac{W}{32}$	$\begin{bmatrix} D=384 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=640 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=768 \\ h=4 \\ E=50 \end{bmatrix} \times 2$	$\begin{bmatrix} D=1024 \\ h=4 \\ E=50 \end{bmatrix} \times 2$
Head	1×1	Pooling & MLP	Pooling & MLP	Pooling & MLP	Pooling & MLP
Parameters (M)		12.3	28.5	52.4	94.4
FLOPs (B)		2.3	6.3	10.7	18.1

where patch embeddings and the hypergraph reinforce each other. Dynamic updates alternate between the two, enhancing structure-aware image representations. Introducing a fixed number of hyperedges as a regularizer further prevents trivial structures in the hypergraph, enhancing its effectiveness. This iterative process ensures mutual improvement of patch embeddings and hypergraph structure, despite their interdependence.

3.5. End-to-End Optimization

In ViHGNN, the adaptive hypergraph structure updates are dependent on learned patch embeddings. During training and inference in each ViHGNN block, these embeddings shape the hypergraph structure. At inference, fixed model weights lead to a stable hypergraph structure. This process operates independently of gradient calculation and does not impact backpropagation.

We perform an end-to-end optimization process for ViHGNN, with task-specific objectives like cross-entropy loss for image classification and localization loss for object detection. The computational bottleneck, the Fuzzy C-Means, carries a time complexity of $O(|\mathcal{V}|DE^2T)$. This complexity is largely determined by the number of hyperedges. We pragmatically set this number to a small value of 50 to manage computation (Table 2).

4. Experiments

4.1. Experimental Settings

Datasets. In the image classification task, the widely used benchmark ImageNet ILSVRC 2012 [48] is used in the following experiments. ImageNet has 120M training images and 50K validation images, which belong to 1000 categories. For the license of ImageNet dataset, please refer to `http://www.image-net.org/download`. For object detection, we use COCO 2017 [39] dataset with 80 object categories. COCO 2017 contains 118K training images and 5K validation images. For the licenses of these datasets, please refer to `https://cocodataset.org/#home`.

Baselines. To establish our baselines, we have opted to follow the choices made in the original ViG [18]. In the field of computer vision, there are generally two types of network architecture: isotropic and pyramid. The isotropic

Table 3. Training hyper-parameters for ImageNet.

ViHGNN	Ti	S	M	B
Epochs	300			
Optimizer	AdamW [44]			
Batch size	1024			
Start learning rate (LR)	2e-3			
Learning rate schedule	Cosine			
Warmup epochs	20			
Weight decay	0.05			
Label smoothing [52]	0.1			
Stochastic path [26]	0.1	0.1	0.1	0.3
Repeated augment [24]	✓			
RandAugment [8]	✓			
Mixup prob. [76]	0.8			
Cutmix prob. [75]	1.0			
Random erasing prob. [82]	0.25			
Exponential moving average	0.99996			

architecture maintains the feature size throughout the network's computational core, which allows for easy scaling and hardware acceleration. This architecture is widely used in transformer models for natural language processing (NLP) [58], as well as recent neural networks in vision, such as ConvMixer[54], ViT [11], DeiT [56], and ResMLP [55]. Therefore, we compare our isotropic ViHGNN against the aforementioned isotropic models. On the other hand, pyramid neural networks gradually reduce the spatial size of feature maps as the network deepens, leveraging the scale-invariant property of images to produce multi-scale features. There are numerous advanced networks with pyramid architecture, including ResNet [22, 65], BoTNet [51], PVT [62], CVT [66], Swin-Transformer [41], CycleMLP [5], and Poolformer [74]. In this work, we compare our pyramid ViHGNN against these models. Table 1 and Table 2 provide detailed settings for the isotropic and pyramid ViHGNN architectures used in our study, respectively.

Hyper-parameters Settings. For all the ViHGNN models, we utilize dilated aggregation [36] in HyperGrapher module and set the dilated rate as $\lceil l/4 \rceil$ for the l-th layer. GELU [23] is used as the nonlinear activation function. For ImageNet classification, we use the commonly-used training strategy proposed in DeiT [56] for fair comparison. The data augmentation includes RandAugment [8], Mixup [76], Cutmix [75], random erasing [82] and repeated augment [24]. The details are shown in Table 3. For COCO detection task, we take RetinaNet [38] and Mask R-CNN [21] as the detection frameworks and use our Pyramid ViHGNN as our backbone. All the models are trained on COCO 2017 training set on a "1×" schedule and evaluated on validation set. We implement the networks using PyTroch [46] and train all our models on 8 NVIDIA V100 GPUs of a single AWS EC2 instance.

Table 4. Results of ViHGNN and other isotropic networks on ImageNet. ♠ CNN, ♥ Transformer, ♣ MLP, ♦ GNN, 🟧 HGNN.

Model	Resolution	Params (M)	FLOPs (B)	Top-1	Top-5
♠ ResMLP-S12 conv3x3 [55]	224×224	16.7	3.2	77.0	-
♠ ConvMixer-768/32 [57]	224×224	21.1	20.9	80.2	-
♠ ConvMixer-1536/20 [57]	224×224	51.6	51.4	81.4	-
♥ ViT-B/16 [11]	384×384	86.4	55.5	77.9	-
♥ DeiT-Ti [56]	224×224	5.7	1.3	72.2	91.1
♥ DeiT-S [56]	224×224	22.1	4.6	79.8	95.0
♥ DeiT-B [56]	224×224	86.4	17.6	81.8	95.7
♣ ResMLP-S24 [55]	224×224	30	6.0	79.4	94.5
♣ ResMLP-B24 [55]	224×224	116	23.0	81.0	95.0
♣ Mixer-B/16 [54]	224×224	59	11.7	76.4	-
♦ Isotropic ViG-Ti	224×224	7.1	1.3	73.9	92.0
♦ Isotropic ViG-S	224×224	22.7	4.5	80.4	95.2
♦ Isotropic ViG-B	224×224	86.8	17.7	82.3	95.9
🟧 ViHGNN-Ti (ours)	224×224	8.2	1.8	**74.3**	**92.5**
🟧 ViHGNN-S (ours)	224×224	23.2	5.6	**81.5**	**95.7**
🟧 ViHGNN-B (ours)	224×224	88.1	19.4	**82.9**	**96.2**

4.2. Image Classification

As shown in Table 4, ViHGNN surpasses other isotropic networks. Notably, our isotropic ViHGNN-S achieves an 81.5% top-1 accuracy, outperforming DeiT-Ti by 1.1% while maintaining comparable computational costs. Moreover, isotropic ViHGNN consistently outperforms ViG with similar model size and computational costs, highlighting the benefits of hypergraph image representation. Moving to pyramid networks (Table 5), our pyramid ViHGNN series matches or outperforms state-of-the-art models, particularly the pyramid ViG family. This underscores ViHGNN's superiority, attributed to hypergraphs capturing latent high-order patch relations. Overall, our results indicate hypergraph neural networks' effectiveness for visual tasks, positioning them as promising components in computer vision systems.

Table 5. Results of Pyramid ViHGNN and other pyramid networks on ImageNet. ♠ CNN, ♥ Transformer, ♣ MLP, ♦ GNN, 🟧 HGNN.

Model	Resolution	Params (M)	FLOPs (B)	Top-1	Top-5
♠ ResNet-18 [22, 65]	224×224	12	1.8	70.6	89.7
♠ ResNet-50 [22, 65]	224×224	25.6	4.1	79.8	95.0
♠ ResNet-152 [22, 65]	224×224	60.2	11.5	81.8	95.9
♠ BoTNet-T3 [51]	224×224	33.5	7.3	81.7	-
♠ BoTNet-T3 [51]	224×224	54.7	10.9	82.8	-
♠ BoTNet-T3 [51]	256×256	75.1	19.3	83.5	-
♥ PVT-Tiny [62]	224×224	13.2	1.9	75.1	-
♥ PVT-Small [62]	224×224	24.5	3.8	79.8	-
♥ PVT-Medium [62]	224×224	44.2	6.7	81.2	-
♥ PVT-Large [62]	224×224	61.4	9.8	81.7	-
♥ CvT-13 [66]	224×224	20	4.5	81.6	-
♥ CvT-21 [66]	224×224	32	7.1	82.5	-
♥ CvT-21 [66]	384×384	32	24.9	83.3	-
♥ Swin-T [41]	224×224	29	4.5	81.3	95.5
♥ Swin-S [41]	224×224	50	8.7	83.0	96.2
♥ Swin-B [41]	224×224	88	15.4	83.5	96.5
♣ CycleMLP-B2 [5]	224×224	27	3.9	81.6	-
♣ CycleMLP-B3 [5]	224×224	38	6.9	82.4	-
♣ CycleMLP-B4 [5]	224×224	52	10.1	83.0	-
♣ Poolformer-S12 [74]	224×224	12	2.0	77.2	93.5
♣ Poolformer-S36 [74]	224×224	31	5.2	81.4	95.5
♣ Poolformer-M48 [74]	224×224	73	11.9	82.5	96.0
♦ Pyramid ViG-Ti [18]	224×224	10.7	1.7	78.2	94.2
♦ Pyramid ViG-S [18]	224×224	27.3	4.6	82.1	96.0
♦ Pyramid ViG-M [18]	224×224	51.7	8.9	83.1	96.4
♦ Pyramid ViG-B [18]	224×224	92.6	16.8	83.7	96.5
🟧 Pyramid ViHGNN-Ti (ours)	224×224	12.3	2.3	**78.9**	**94.6**
🟧 Pyramid ViHGNN-S (ours)	224×224	28.5	6.3	**82.5**	**96.3**
🟧 Pyramid ViHGNN-M (ours)	224×224	52.4	10.7	**83.4**	**96.5**
🟧 Pyramid ViHGNN-B (ours)	224×224	94.4	18.1	**83.9**	**96.7**

4.3. Ablation Study

We conduct ablation study of the proposed method on ImageNet classification task and use the isotropic ViHGNN-Ti as the base architecture. And all the results are given in Table 6, Table 7, Table 8, Table 9 and Table 10.

Table 6. Ablation study on ImageNet for classification task.

Type of $G(\boldsymbol{I})$	Num. of E	HSL module	Params (M)	FLOPs (B)	Top-1	Top-5
k-NN	25	✗	7.9	1.5	72.4	89.7
k-NN	25	✓	8.2	1.9	72.8	90.2
k-NN	50	✗	8.9	2.4	72.5	89.6
k-NN	50	✓	9.4	2.9	73.0	90.1
K-Means	25	✗	7.3	1.4	72.6	90.1
K-Means	25	✓	8.0	2.1	73.1	90.3
K-Means	50	✗	8.4	2.5	73.5	91.1
K-Means	50	✓	9.2	2.8	73.7	92.3
Fuzzy C-Means	25	✗	8.2	2.2	72.9	90.4
Fuzzy C-Means	25	✓	8.8	2.9	73.5	90.9
Fuzzy C-Means	50	✗	9.1	3.2	74.1	92.2
Fuzzy C-Means	50	✓	9.7	3.8	74.9	92.9

Type of hypergraph construction methods. All the three methods, k-NN, K-Means and Fuzzy C-Means, can be adopted to construct the hypergraph. To investigate which method works the best, we evaluate the effects of these construction methods by ablation study. From the Table 6, we can find that the Fuzzy C-Means achieves the best performance with little overhead than the K-Means method. The results further demonstrate that the G_I generated by Fuzzy C-Means is the most robust.

Actually, there are alternative clustering methods available for updating the hypergraph structure. In addition, we explored *DBSCAN*, *Mean Shift Clustering*, and *Spectral Clustering* methods for comparison. As shown in Table 7, each method produced different results, with Fuzzy C-Means demonstrating the best performance. As demonstrated in the table, Fuzzy C-Means has the highest Top-1 and Top-5 scores compared to the other clustering methods. One possible reason for its superiority is that Fuzzy C-Means is more flexible and less sensitive to noise compared to other alternatives. This flexibility enables it to find better hyperedges, which, in turn, improves the alignment of the hypergraph structure with the patch-distribution. Consequently, Fuzzy C-Means can more effectively capture the complex relationships within patch embeddings and ultimately provide a more accurate representation of the images, leading to improved performance in our model.

Table 7. Comparison of various clustering methods with Istropic ViHGNN-Ti as the backbone model.

Clustering Methods	DBSCAN	Mean Shift Clustering	Spectral Clustering	Fuzzy C-Means
Top-1	74.2	74.5	74.7	**74.9**
Top-5	92.6	92.7	92.8	**92.9**

The number of update loops. In each ViHGNN block, the patch embeddings from the previous block are used to update the hypergraph structure, which is then employed to generate new patch embeddings for the next block. This results in L forward passes during the training process. Although it is possible to use multiple loops within a single block, the embeddings must be updated to ensure that the hypergraph update is meaningful. We conducted additional experiments with 2 and 3 updated loops for the Isotropic ViHGNN-Ti model, observing minimal accuracy differences (Table 8). Given the model depth and number of patches, we assume that one update loop may be sufficient for this particular scale, as using more loops would increase model parameters and FLOPs without significant benefits.

Table 8. Comparison of various update loops with Istropic ViHGNN-Ti as the backbone model.

# Update Loops	1	2	3
Top-1	74.9	**75.0**	74.4
Top-5	92.9	**93.0**	92.7

The number of hyperedges (clusters of Fuzzy C-Means). In Fuzzy C-Means, the cluster count directly influences the hyperedge quantity, subsequently dictating the hypergraph size. An optimal balance is essential; insufficient hyperedges diminish performance, whereas an excess elevates computational expenses. In our ViHGNN-Ti trials, segmenting images into 196 patches, we chose hyperedge counts of 25 or 50. While an uptick in hyperedges typically boosts performance, it introduces increased overhead.

Contrarily, the proposition "increased hyperedges elevate performance" predominantly aligns with isotropic ViHGNN-Ti. For pyramid ViHGNN-Ti, our detailed experimentation, presented in Table 9, assesses performance across varied hyperedge settings. Specifically, the results for Pyramid ViHGNN-Ti with uniform E values are juxtaposed against layer-specific E allocations. The $\downarrow$ symbol denotes a 50% hyperedge reduction at each stage, correlating with feature size. These outcomes debunk the generalized "more is better" hyperedge notion, suggesting hyperedges should correspond to patch node counts.

In summary, determining the ideal hyperedge count remains a nuanced challenge. However, two pivotal insights surface: (i) hyperedges can amplify performance, albeit with overhead; (ii) hyperedge distribution across layers should be commensurate with feature dimensions.

Table 9. Effects of E for Pyramid ViHGNN-Ti.

E	50	50 ($\downarrow$)	100	100 ($\downarrow$)
Top-1	78.9	79.4	77.6	78.9
Top-5	94.6	95.0	94.2	94.7

Effect of hypergraph structure learning. In order to verify the effect of the hypergraph structure learning (HSL) module, we further conduct additional experiments by removing the HSL module. We observe that the HSL can improve performance with only little overhead in terms of model size and flops. The results also demonstrate the effectiveness of the simple HSL module used in our framework.

Overhead of hypergraph structure learning. To analyze the potential overhead of multiple calls to Fuzzy *C*-Means, we measured the time costs of training ViHGNN and ViT. The results are presented in Table 10, where we observed that clustering had a minimal impact on forward time and did not affect backward time, which is consistent with our expectations.

Table 10. Comparison of the mean running time (*ms*) per sample.

Models	Forward	Backward	Total
ViT-Ti	10.74	19.03	29.77
Istropic ViHGNN-Ti	11.25	18.77	30.02

4.4. Object Detection

Table 11. Object detection and instance segmentation results on COCO val2017.

Backbone	RetinaNet 1×							
	Params (M)	FLOPs (B)	mAP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L
ResNet50 [22]	37.7	239.3	36.3	55.3	38.6	19.3	40.0	48.8
ResNeXt-101-32x4d [68]	56.4	319	39.9	59.6	42.7	22.3	44.2	52.5
PVT-Small [62]	34.2	226.5	40.4	61.3	44.2	25.0	42.9	55.7
CycleMLP-B2 [5]	36.5	230.9	40.6	61.4	43.2	22.9	44.4	54.5
Swin-T [41]	38.5	244.8	41.5	62.1	44.2	25.1	44.9	55.5
Pyramid ViG-S [18]	36.2	240.0	41.8	63.1	44.7	28.5	45.4	53.4
Pyramid ViHGNN-S (ours)	37.9	243.7	**42.2**	**63.8**	**45.1**	**29.3**	**45.9**	**55.7**
Backbone	Mask R-CNN 1×							
	Param	FLOPs	AP^b	AP^b_{50}	AP^b_{75}	AP^m	AP^m_{50}	AP^m_{75}
ResNet50 [22]	44.2	260.1	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Small [62]	44.1	245.1	40.4	62.9	43.8	37.8	60.1	40.3
CycleMLP-B2 [5]	46.5	249.5	42.1	64.0	45.7	38.9	61.2	41.8
PoolFormer-S24 [74]	41.0	-	40.1	62.2	43.4	37.0	59.1	39.6
Swin-T [41]	47.8	264.0	42.2	64.6	46.2	39.1	61.6	42.0
Pyramid ViG-S [18]	45.8	258.8	42.6	65.2	46.0	39.4	62.4	41.6
Pyramid ViHGNN-S (ours)	47.3	261.4	**43.1**	**66.0**	**46.5**	**39.6**	**63.0**	**42.3**

To evaluate the generalization ability of our ViHGNN model, we further apply ViHGNN to the object detection task. For a fair comparison, we utilize the ImageNet pretrained pyramid ViHGNN-S as the backbone of RetinaNet [38] and Mask R-CNN [21] detection frameworks. The models are trained on the commonly-used "1x" schedule and FLOPs are calculated at a 1280 × 800 input size. From the results in Table 11, we can see that our pyramid ViG-S performs better than the representative backbones of different types, including ResNet [22], CycleMLP [5] and Swin Transformer [41] on both RetinaNet and Mask R-CNN. The superior results demonstrate the generalization ability of the ViHGNN architecture.

4.5. Visualization

To gain insights into the workings of our ViHGNN model, we visualize the learned hypergraph structure in ViHGNN-S and compare it with the constructed graph structure in ViG-S. In Figure 4, the top row presents the input image, the middle row displays the graph structure of ViG-S, and the bottom row showcases the learned hypergraph structure of ViHGNN-S. Given the numerous edges in ViG-S's graph, we illustrate a central node and its first-order neighbors for clarity. Similarly, we show a subset of representative hyperedges for the hypergraph structure, avoiding clutter.

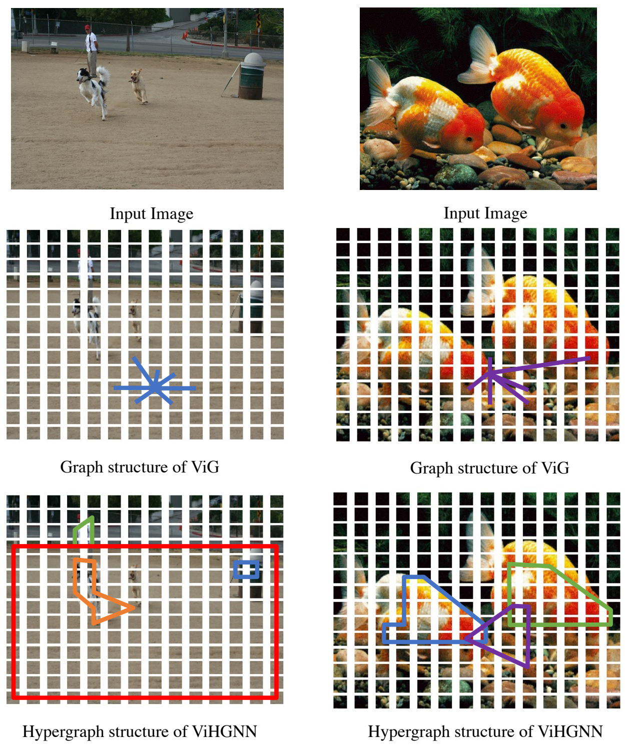

Figure 4. Visualization of the graph/hypergraph structure of ViG/ViHGNN. Each color line of the middle row denotes one edge. Each frame of the top row denotes one hyperedge.

Observing the visualization, ViG tends to generate redundant edges among patches with similar local features like color and texture. For instance, regions with similar patches, such as sand in the left image, result in multiple edges, despite their limited relevance to downstream tasks. This redundancy represents an overhead waste, which ViHGNN addresses by using fewer hyperedges to model such relationships. Moreover, ViG may also introduce noisy edges connecting object patches to semantically different patches with similar attributes. In contrast, ViHGNN captures higher-order patch relations while maintaining robustness against noisy connections.

5. Conclusion & Limitation

In this paper, we present a novel advancement in image representation through the utilization of hypergraph data encoding and hypergraph neural networks tailored for visual tasks. Our proposed model, ViHGNN, innovatively dissects images into patches, treating them as nodes, and subsequently constructs a hypergraph based on these nodes. This strategic approach empowers ViHGNN to leverage the hypergraph's unique capacity to capture intricate high-order relationships among patches. Thorough experimentation across image classification and object detection tasks affirms ViHGNN's remarkable capabilities. An identifiable constraint of ViHGNN pertains to the manual determination of hypergraph size. Our future endeavors will be directed towards refining the adaptation of the hypergraph structure to optimize performance for diverse image categories.

References

[1] James C Bezdek, Robert Ehrlich, and William Full. Fcm: The fuzzy c-means clustering algorithm. *Computers & geosciences*, 10(2-3):191–203, 1984. 4

[2] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229, 2020. 2

[4] Hanting Chen, Yunhe Wang, Tianyu Guo, Chang Xu, Yiping Deng, Zhenhua Liu, Siwei Ma, Chunjing Xu, Chao Xu, and Wen Gao. Pre-trained image processing transformer. In *CVPR*, 2021. 2

[5] Shoufa Chen, Enze Xie, Chongjian Ge, Ding Liang, and Ping Luo. Cyclemlp: A mlp-like architecture for dense prediction. In *ICLR*, 2022. 2, 6, 8

[6] Yu Chen, Lingfei Wu, and Mohammed Zaki. Iterative deep graph learning for graph neural networks: Better and robust node embeddings. *Advances in neural information processing systems*, 33:19314–19326, 2020. 3

[7] Eli Chien, Chao Pan, Jianhao Peng, and Olgica Milenkovic. You are allset: A multiset function framework for hypergraph neural networks. In *International Conference on Learning Representations*, 2022. 3, 4

[8] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le. Randaugment: Practical automated data augmentation with a reduced search space. In *CVPR Workshops*, 2020. 6

[9] Xiaohan Ding, Xiangyu Zhang, Jungong Han, and Guiguang Ding. Scaling up your kernels to 31x31: Revisiting large kernel design in cnns. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 11963–11975, 2022. 2

[10] Yihe Dong, Jean-Baptiste Cordonnier, and Andreas Loukas. Attention is not all you need: Pure attention loses rank doubly exponentially with depth. In *ICML*, pages 2793–2803. PMLR, 2021. 4

[11] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021. 1, 2, 6

[12] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565, 2019. 3

[13] Luca Franceschi, Mathias Niepert, Massimiliano Pontil, and Xiao He. Learning discrete structures for graph neural networks. In *International conference on machine learning*, pages 1972–1982. PMLR, 2019. 3

[14] Yue Gao, Meng Wang, Dacheng Tao, Rongrong Ji, and Qionghai Dai. 3-d object retrieval and recognition with hypergraph analysis. *IEEE Transactions on Image Processing*, 21(9):4290–4303, 2012. 3

[15] Jianyuan Guo, Yehui Tang, Kai Han, Xinghao Chen, Han Wu, Chao Xu, Chang Xu, and Yunhe Wang. Hire-mlp: Vision mlp via hierarchical rearrangement. In *CVPR*, 2022. 2

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

[17] Kai Han, Yunhe Wang, Hanting Chen, Xinghao Chen, Jianyuan Guo, Zhenhua Liu, Yehui Tang, An Xiao, Chunjing Xu, Yixing Xu, et al. A survey on vision transformer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022. 2

[18] 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, 3, 4, 5, 6, 8

[19] Kai Han, An Xiao, Enhua Wu, Jianyuan Guo, Chunjing Xu, and Yunhe Wang. Transformer in transformer. In *NeurIPS*, 2021. 2

[20] Yan Han, Edward W Huang, Wenqing Zheng, Nikhil Rao, Zhangyang Wang, and Karthik Subbian. Search behavior prediction: A hypergraph perspective. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 697–705, 2023. 3

[21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, pages 2961–2969, 2017. 6, 8

[22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016. 1, 2, 6, 8

[23] Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016. 6

[24] Elad Hoffer, Tal Ben-Nun, Itay Hubara, Niv Giladi, Torsten Hoefler, and Daniel Soudry. Augment your batch: Improving generalization through instance repetition. In *CVPR*, 2020. 6

[25] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017. 2

[26] Gao Huang, Yu Sun, Zhuang Liu, Daniel Sedra, and Kilian Q Weinberger. Deep networks with stochastic depth. In *ECCV*, pages 646–661. Springer, 2016. 6

[27] Jing Huang and Jie Yang. Unignn: a unified framework for graph and hypergraph neural networks. *arXiv preprint arXiv:2105.00956*, 2021. 3

[28] Tianjin Huang, Lu Yin, Zhenyu Zhang, Li Shen, Meng Fang, Mykola Pechenizkiy, Zhangyang Wang, and Shiwei Liu. Are large kernels better teachers than transformers for convnets? In *ICML*, 2023. 2

[29] Yuchi Huang, Qingshan Liu, and Dimitris Metaxas.] video object segmentation by hypergraph cut. In *2009 IEEE conference on computer vision and pattern recognition*, pages 1738–1745. IEEE, 2009. 3

[30] Yuchi Huang, Qingshan Liu, Shaoting Zhang, and Dimitris N Metaxas. Image retrieval via probabilistic hypergraph ranking. In *2010 IEEE computer society conference on computer vision and pattern recognition*, pages 3376–3383. IEEE, 2010. 3

[31] Ashesh Jain, Amir R Zamir, Silvio Savarese, and Ashutosh Saxena. Structural-rnn: Deep learning on spatio-temporal graphs. In *CVPR*, pages 5308–5317, 2016. 3

[32] Jianwen Jiang, Yuxuan Wei, Yifan Feng, Jingxuan Cao, and Yue Gao. Dynamic hypergraph neural networks. In *IJCAI*, pages 2635–2641, 2019. 3

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *NeurIPS*, pages 1097–1105, 2012. 1, 2

[34] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *CVPR*, pages 4558–4567, 2018. 2

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

[36] Guohao Li, Matthias Muller, Ali Thabet, and Bernard Ghanem. Deepgcns: Can gcns go as deep as cnns? In *ICCV*, pages 9267–9276, 2019. 6

[37] Dongze Lian, Zehao Yu, Xing Sun, and Shenghua Gao. As-mlp: An axial shifted mlp architecture for vision. In *ICLR*, 2022. 2

[38] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 6, 8

[39] 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 *ECCV*, pages 740–755, 2014. 5

[40] Shiwei Liu, Tianlong Chen, Xiaohan Chen, Xuxi Chen, Qiao Xiao, Boqian Wu, Tommi Kärkkäinen, Mykola Pechenizkiy, Decebal Constantin Mocanu, and Zhangyang Wang. More convnets in the 2020s: Scaling up kernels beyond 51x51 using sparsity. In *The Eleventh International Conference on Learning Representations*, 2023. 2

[41] 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 *ICCV*, pages 10012–10022, 2021. 2, 6, 8

[42] 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. 2

[43] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 2

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

[45] Dongsheng Luo, Wei Cheng, Wenchao Yu, Bo Zong, Jingchao Ni, Haifeng Chen, and Xiang Zhang. Learning to drop: Robust graph neural network via topological denoising. In *Proceedings of the 14th ACM international conference on web search and data mining*, pages 779–787, 2021. 3

[46] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *NeurIPS*, 2019. 6

[47] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *NIPS*, pages 91–99, 2015. 2

[48] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 5

[49] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008. 2

[50] Han Shi, Jiahui Gao, Hang Xu, Xiaodan Liang, Zhenguo Li, Lingpeng Kong, Stephen Lee, and James T Kwok. Revisiting over-smoothing in bert from the perspective of graph. *arXiv preprint arXiv:2202.08625*, 2022. 4

[51] Aravind Srinivas, Tsung-Yi Lin, Niki Parmar, Jonathon Shlens, Pieter Abbeel, and Ashish Vaswani. Bottleneck transformers for visual recognition. In *CVPR*, pages 16519–16529, 2021. 6

[52] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *CVPR*, 2016. 6

[53] Yehui Tang, Kai Han, Jianyuan Guo, Chang Xu, Yanxi Li, Chao Xu, and Yunhe Wang. An image patch is a wave: Phase-aware vision mlp. In *CVPR*, 2022. 2

[54] 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. In *NeurIPS*, volume 34, 2021. 1, 2, 6

[55] 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. *arXiv preprint arXiv:2105.03404*, 2021. 1, 2, 6

[56] Hugo Touvron, Matthieu Cord, Matthijs Douze, Francisco Massa, Alexandre Sablayrolles, and Hervé Jégou. Training data-efficient image transformers & distillation through attention. In *ICML*, 2021. 6

[57] Asher Trockman and J Zico Kolter. Patches are all you need? *arXiv preprint arXiv:2201.09792*, 2022. 6

[58] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *NeurIPS*, 2017. 6

[59] Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval

and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008. 2

[60] Peihao Wang, Shenghao Yang, Yunyu Liu, Zhangyang Wang, and Pan Li. Equivariant hypergraph diffusion neural operators. *arXiv preprint arXiv:2207.06680*, 2022. 3

[61] Peihao Wang, Wenqing Zheng, Tianlong Chen, and Zhangyang Wang. Anti-oversmoothing in deep vision transformers via the fourier domain analysis: From theory to practice. *arXiv preprint arXiv:2203.05962*, 2022. 4

[62] 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 *ICCV*, 2021. 2, 6, 8

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

[64] Tianxin Wei, Yuning You, Tianlong Chen, Yang Shen, Jingrui He, and Zhangyang Wang. Augmentations in hypergraph contrastive learning: Fabricated and generative. *Advances in neural information processing systems*, 35:1909–1922, 2022. 3

[65] Ross Wightman, Hugo Touvron, and Hervé Jégou. Resnet strikes back: An improved training procedure in timm. *arXiv preprint arXiv:2110.00476*, 2021. 6

[66] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *ICCV*, pages 22–31, 2021. 6

[67] Kan Wu, Houwen Peng, Minghao Chen, Jianlong Fu, and Hongyang Chao. Rethinking and improving relative position encoding for vision transformer. In *ICCV*, pages 10033–10041, 2021. 2

[68] Saining Xie, Ross Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, pages 1492–1500, 2017. 8

[69] Danfei Xu, Yuke Zhu, Christopher B Choy, and Li Fei-Fei. Scene graph generation by iterative message passing. In *CVPR*, pages 5410–5419, 2017. 3

[70] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. Hypergcn: A new method for training graph convolutional networks on hypergraphs. *Advances in neural information processing systems*, 32, 2019. 3

[71] Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *AAAI*, 2018. 3

[72] Jianwei Yang, Jiasen Lu, Stefan Lee, Dhruv Batra, and Devi Parikh. Graph r-cnn for scene graph generation. In *ECCV*, pages 670–685, 2018. 3

[73] Zhaohui Yang, Yunhe Wang, Xinghao Chen, Boxin Shi, Chao Xu, Chunjing Xu, Qi Tian, and Chang Xu. Cars: Continuous evolution for efficient neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1829–1838, 2020. 2

[74] 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 *CVPR*, 2022. 6, 8

[75] 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 *ICCV*, 2019. 6

[76] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. In *ICLR*, 2018. 6

[77] Jiying Zhang, Yuzhao Chen, Xi Xiao, Runiu Lu, and Shu-Tao Xia. Learnable hypergraph laplacian for hypergraph learning. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4503–4507. IEEE, 2022. 3

[78] Zizhao Zhang, Yifan Feng, Shihui Ying, and Yue Gao. Deep hypergraph structure learning. *arXiv preprint arXiv:2208.12547*, 2022. 3

[79] Zizhao Zhang, Haojie Lin, Yue Gao, and KLISS BNRist. Dynamic hypergraph structure learning. In *IJCAI*, pages 3162–3169, 2018. 3

[80] Wei Zhao, Shulong Tan, Ziyu Guan, Boxuan Zhang, Maoguo Gong, Zhengwen Cao, and Quan Wang. Learning to map social network users by unified manifold alignment on hypergraph. *IEEE transactions on neural networks and learning systems*, 29(12):5834–5846, 2018. 3

[81] Cheng Zheng, Bo Zong, Wei Cheng, Dongjin Song, Jingchao Ni, Wenchao Yu, Haifeng Chen, and Wei Wang. Robust graph representation learning via neural sparsification. In *International Conference on Machine Learning*, pages 11458–11468. PMLR, 2020. 3

[82] Zhun Zhong, Liang Zheng, Guoliang Kang, Shaozi Li, and Yi Yang. Random erasing data augmentation. In *AAAI*, volume 34, pages 13001–13008, 2020. 6

[83] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. A survey on graph structure learning: Progress and opportunities. *arXiv e-prints*, pages arXiv–2103, 2021. 3

[84] Barret Zoph and Quoc V Le. Neural architecture search with reinforcement learning. In *ICLR*, 2017. 2