# Vision GNN: An Image is Worth Graph of Nodes

이스마일 ~ M2021765

Data Mining Lab

Kookmin University

# 목차

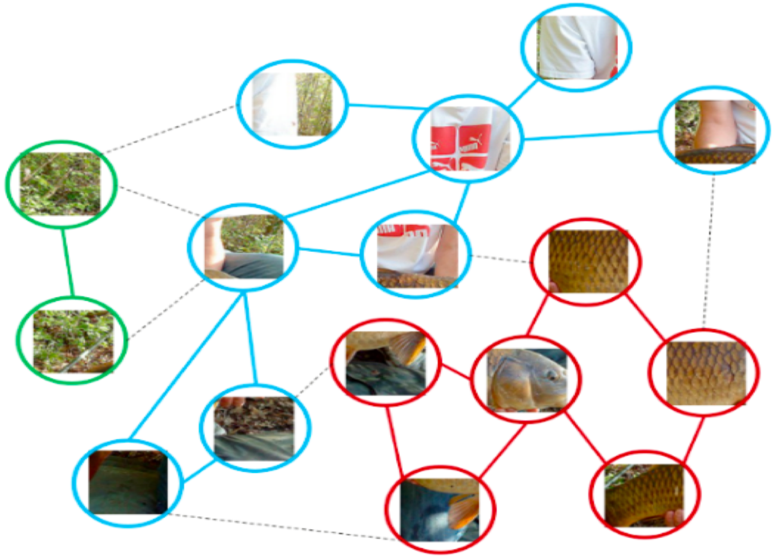
- Introduction
- VIG Block
- Experiments
- Conclusion

# Abstract

- In this paper Image is viewed as graph  structure and introduced a new *Vision GNN* (ViG) architecture to extract graph- level feature for visual tasks
- Divide the image into multiple patches that are treated like nodes
- Build graph by connecting the nearest neighbors
- Based on the graph representation of the image, the ViG model allows information to be converted and exchanged between all nodes

(a) Grid structure.

(b) Sequence structure.



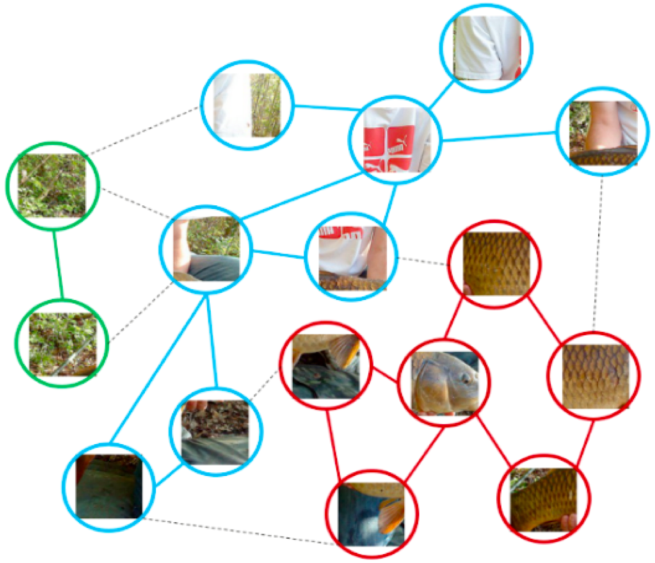
(b) Graph structure.

# Introduction

- Computer Vision 의 기본적인 task는 물체 인식이다. 대부분의 물체는 다양하고 불규칙한 모양이기에 기존의 방식은 낭비가 크고 유연하지 못하다.
- 물체는 부분의 집합으로 생각될 수 있다. 예를 들자면, 인간이 머리, 상체, 하체로 나눠지고, 이들의 관계가 자연스럽게 graph 구조로 표현될 수 있고, 그래프를 분석함으로 우리는 인간을 인식한다.

(a) Grid structure.

(b) Sequence structure.



(b) Graph structure.

# Introduction

- 각 픽셀을 노드로 만들면 너무 많은 노드가 생성되므로 이미지를 n개의 패치로 나누고, 이를 노드로 인식하여 ViG를 사용하여 이러한 노드 간에 정보를 변환하고 교환합니다.
- ViG의 기본 단위는 다음과 같습니다.

a) 그래퍼

b) FFN 모듈

(a) Grid structure. (b) Sequence structure. (b) Graph structure.

# Proposed Method (VIG Framework)

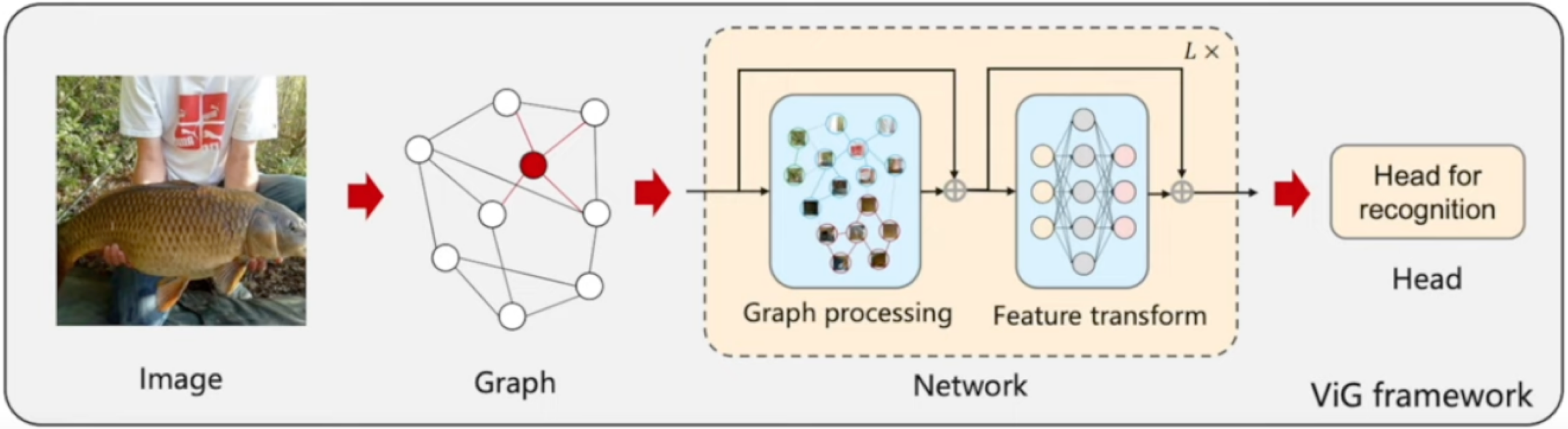
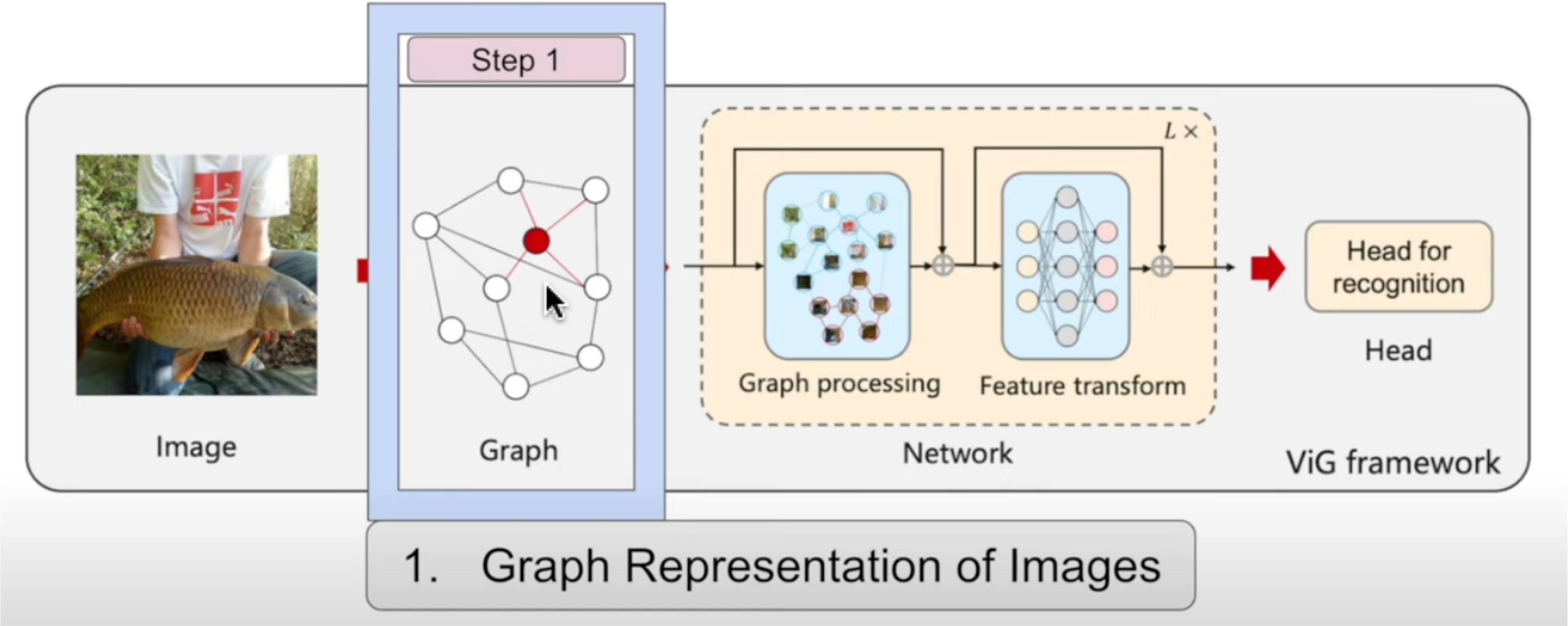


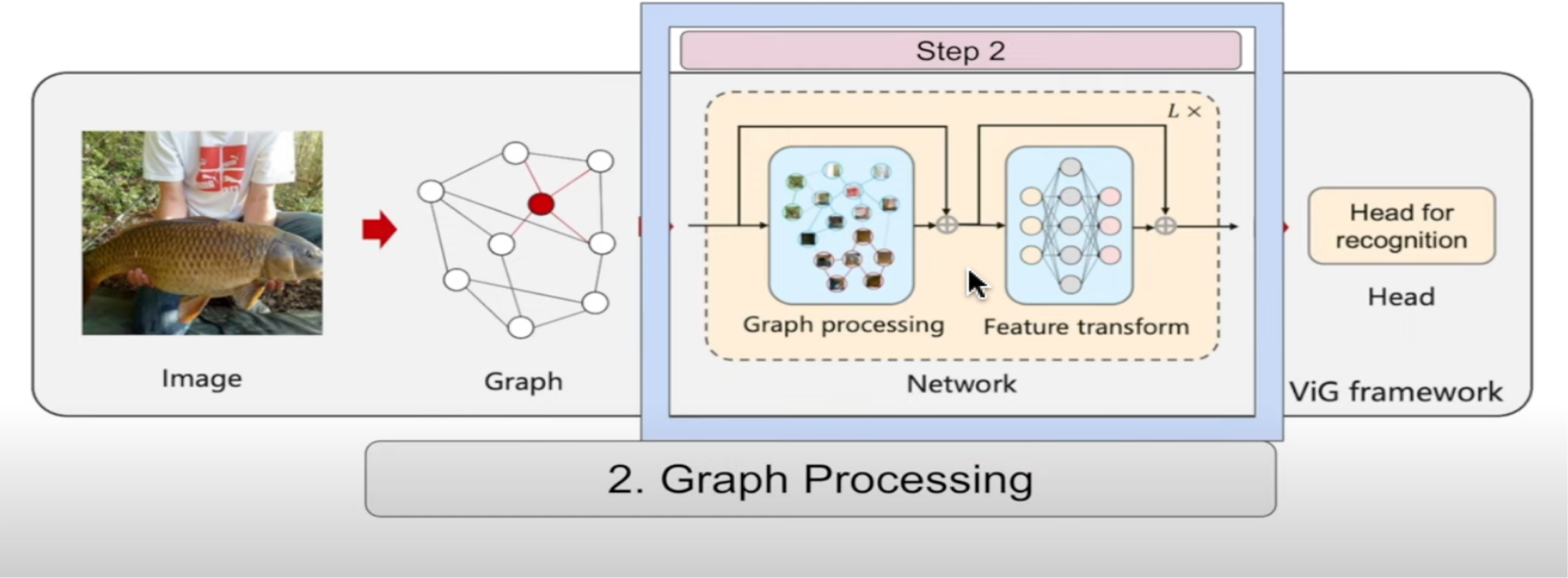

Figure 2: The framework of the proposed ViG model.

How the graph representation of the image is created ?

이미지의 그래프 표현은 어떻게 만들어지나요



# Graph Structure of an Image
# 하나 이미지를 그래프 구조로 다루는 방법



# Graph Structure of an Image
# 하나 이미지를 그래프 구조로 다루는 방법

## 1. Graph Representation of Images

H x W x 3 = N Patches

Transforming Each patch into the feature Vector with D as Feature Dimension.

$Z = [\ x_1, x_2, x_3, \ldots.. x_n\ ]$

Features are set of unordered nodes

$V = [\ v_1, v_2, v_3, \ldots.. v_n\ ]$

1. For each node $v_i$ its K nearest neighbors are found $N(v_i)$
2. Adding an edge $e_{ji}$ directed from $V_j$ to $V_i$ for all $Vj\ \varepsilon\ N(v_i)$

Finally the Graph G is constructed:

**$G = (\ V, E)$**

V all image patches

E - All edges

# Graph Structure of an Image
# 하나 이미지를 그래프 구조로 다루는 방법

## 1. Graph Representation of Images

H x W x 3 = N Patches

Transforming Each patch into the feature Vector with D as Feature Dimension.

$Z = [\, x_1, x_2, x_3, \ldots\ldots x_n \,]$

Features are set of unordered nodes

$V = [\, v_1, v_2, v_3, \ldots\ldots v_n \,]$

1. For each node $v_i$ its K nearest neighbors are found $N(v_i)$
2. Adding an edge $e_{ji}$ directed from $V_j$ to $V_i$ for all $V_j \in N(v_i)$

Finally the Graph G is constructed:

**G = ( V, E)**

V all image patches
E - All edges

# Graph Structure of an Image
# 하나 이미지를 그래프 구조로 다루는 방법

## 1. Graph Representation of Images

$H \times W \times 3 = N$ Patches

Transforming Each patch into the feature Vector with D as Feature Dimension.

$Z = [x_1, x_2, x_3, \ldots\ldots x_n]$

Features are set of unordered nodes

$V = [v_1, v_2, v_3, \ldots\ldots v_n]$

1. For each node $v_i$ its K nearest neighbors are found $N(v_i)$
2. Adding an edge $e_{ji}$ directed from $V_j$ to $V_i$ for all $V_j \in N(v_i)$

Finally the Graph G is constructed:

$$G = (V, E)$$

V all image patches

E - All edges

# Graph Structure of an Image

# 하나 이미지를 그래프 구조로 다루는 방법

## 1. Graph Representation of Images

H x W x 3 = N Patches

Transforming Each patch into the feature Vector with D as Feature Dimension.

$Z = [\, x_1, x_2, x_3, \ldots\ldots x_n \,]$

Features are set of unordered nodes

$V = [\, v_1, v_2, v_3, \ldots\ldots v_n \,]$

1. For each node $v_i$ its K nearest neighbors are found $N(v_i)$
2. Adding an edge $e_{ji}$ directed from $V_j$ to $V_i$ for all $Vj$ Ɛ $N(v_i)$

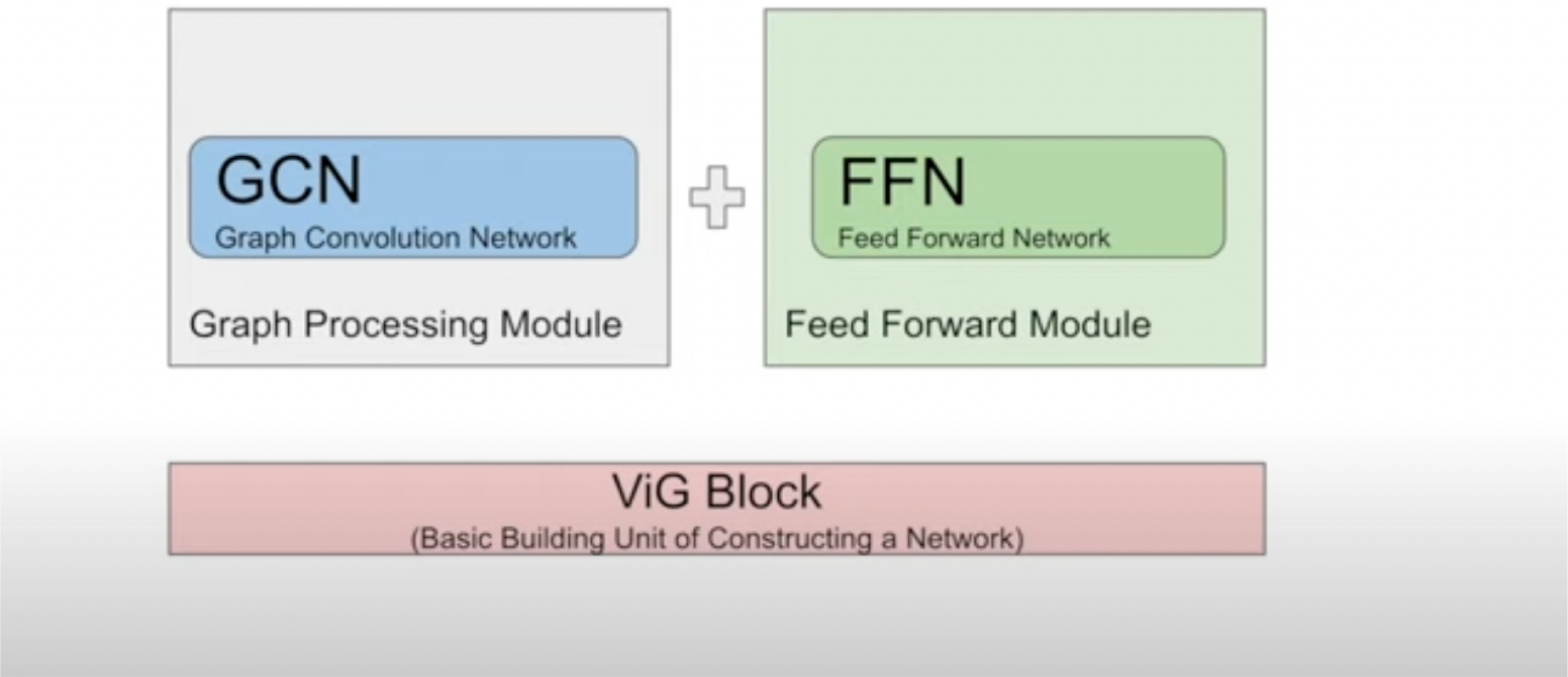
Finally the Graph G is constructed:

$$G = (\, V, E)$$

V all image patches

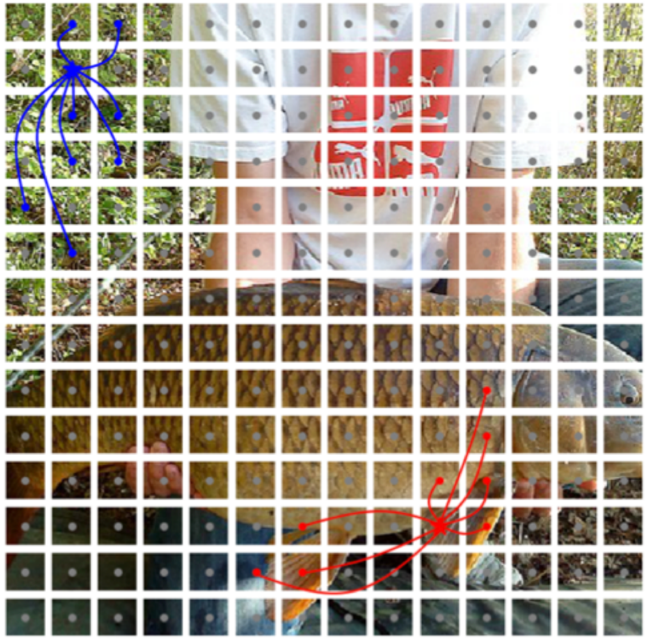
E - All edges

# Graph Structure of an Image
# 하나 이미지를 그래프 구조로 다루는 방법

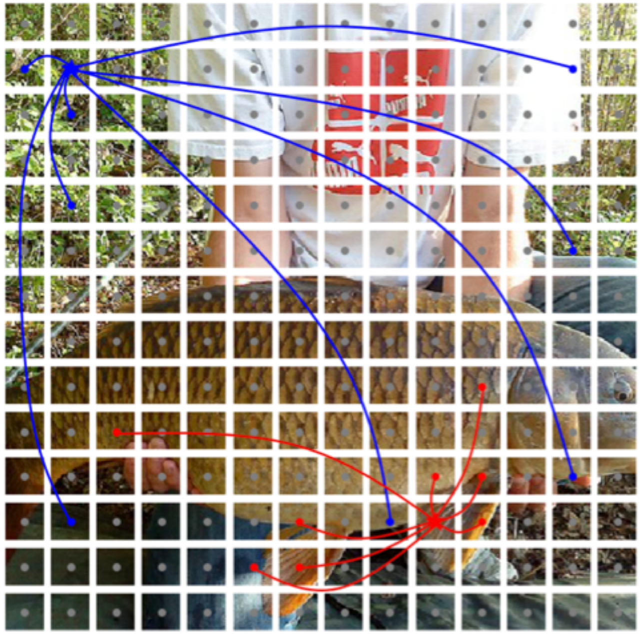


# Constructed Graph Structure

Input Image

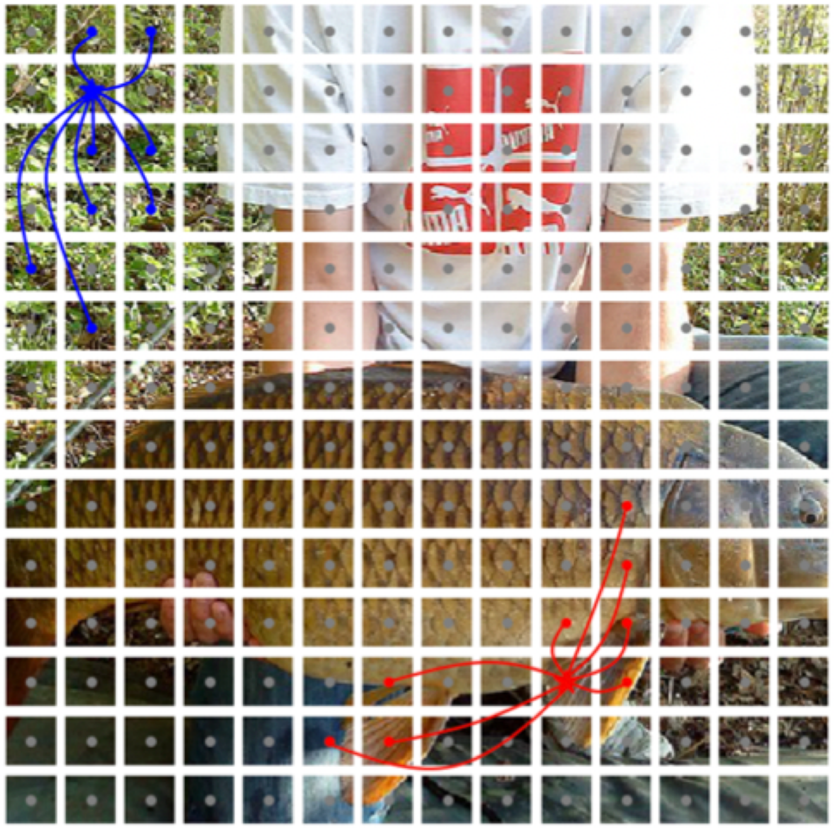


Graph Connection in 1st Block



Graph Connection in 12th Block

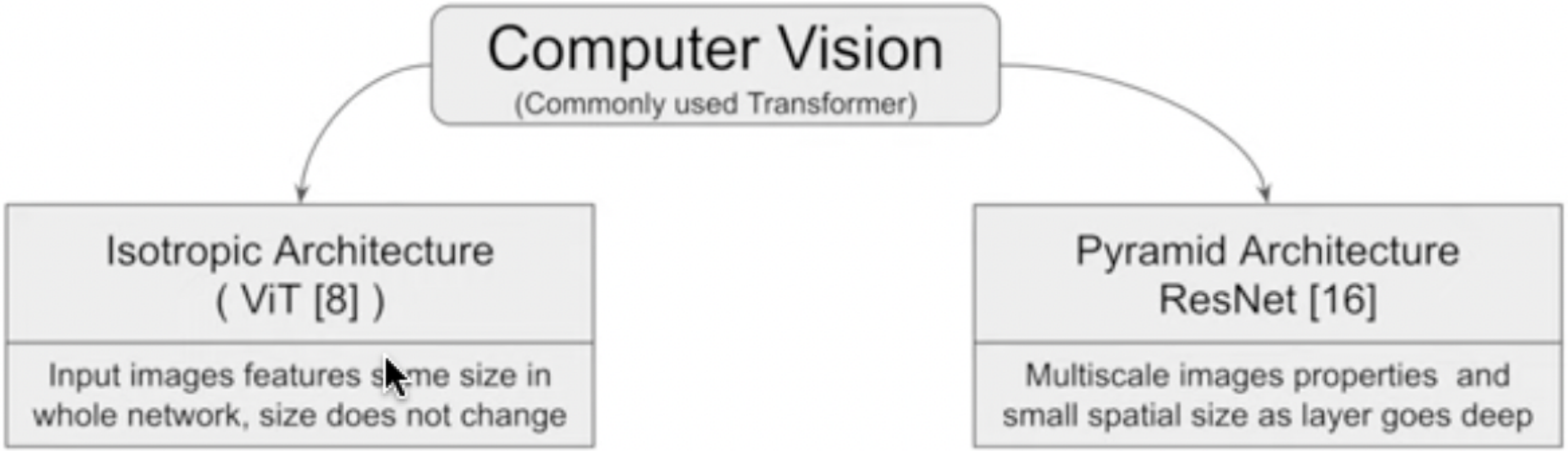
# Constructed Graph Structure



Input Image

Graph Connection in 12th Block

# Network Architecture




| Model | Resolution | Params (M) | FLOPs (B) | Top-1 | Top-5 |
|---|---|---|---|---|---|
| ♠ ResMLP-S12 conv3x3 [48] | 224×224 | 16.7 | 3.2 | 77.0 | - |
| ♠ ConvMixer-768/32 [50] | 224×224 | 21.1 | 20.9 | 80.2 | - |
| ♠ ConvMixer-1536/20 [50] | 224×224 | 51.6 | 51.4 | 81.4 | - |
| ♦ ViT-B/16 [8] | 384×384 | 86.4 | 55.5 | 77.9 | - |
| ♦ DeiT-Ti [49] | 224×224 | 5.7 | 1.3 | 72.2 | 91.1 |
| ♦ DeiT-S [49] | 224×224 | 22.1 | 4.6 | 79.8 | 95.0 |
| ♦ DeiT-B [49] | 224×224 | 86.4 | 17.6 | 81.8 | 95.7 |
| ■ ResMLP-S24 [48] | 224×224 | 30 | 6.0 | 79.4 | 94.5 |
| ■ ResMLP-B24 [48] | 224×224 | 116 | 23.0 | 81.0 | 95.0 |
| ■ Mixer-B/16 [47] | 224×224 | 59 | 11.7 | 76.4 | - |
| ★ ViG-Ti (ours) | 224×224 | 7.1 | 1.3 | **73.9** | **92.0** |
| ★ ViG-S (ours) | 224×224 | 22.7 | 4.5 | **80.4** | **95.2** |
| ★ ViG-B (ours) | 224×224 | 86.8 | 17.7 | **82.3** | **95.9** |

| Model | Resolution | Params (M) | FLOPs (B) | Top-1 | Top-5 |
|---|---|---|---|---|---|
| ♠ ResNet-18 [16, 56] | 224×224 | 12 | 1.8 | 70.6 | 89.7 |
| ♠ ResNet-50 [16, 56] | 224×224 | 25.6 | 4.1 | 79.8 | 95.0 |
| ♠ ResNet-152 [16, 56] | 224×224 | 60.2 | 11.5 | 81.8 | 95.9 |
| ♠ BoTNet-T3 [44] | 224×224 | 33.5 | 7.3 | 81.7 | - |
| ♠ BoTNet-T3 [44] | 224×224 | 54.7 | 10.9 | 82.8 | - |
| ♠ BoTNet-T3 [44] | 256×256 | 75.1 | 19.3 | 83.5 | - |
| ♦ PVT-Tiny [54] | 224×224 | 13.2 | 1.9 | 75.1 | - |
| ♦ PVT-Small [54] | 224×224 | 24.5 | 3.8 | 79.8 | - |
| ♦ PVT-Medium [54] | 224×224 | 44.2 | 6.7 | 81.2 | - |
| ♦ PVT-Large [54] | 224×224 | 61.4 | 9.8 | 81.7 | - |
| ♦ CvT-13 [57] | 224×224 | 20 | 4.5 | 81.6 | - |
| ♦ CvT-21 [57] | 224×224 | 32 | 7.1 | 82.5 | - |
| ♦ CvT-21 [57] | 384×384 | 32 | 24.9 | 83.3 | - |
| ♦ Swin-T [33] | 224×224 | 29 | 4.5 | 81.3 | 95.5 |
| ♦ Swin-S [33] | 224×224 | 50 | 8.7 | 83.0 | 96.2 |
| ♦ Swin-B [33] | 224×224 | 88 | 15.4 | 83.5 | 96.5 |
| ■ CycleMLP-B2 [4] | 224×224 | 27 | 3.9 | 81.6 | - |
| ■ CycleMLP-B3 [4] | 224×224 | 38 | 6.9 | 82.4 | - |
| ■ CycleMLP-B4 [4] | 224×224 | 52 | 10.1 | 83.0 | - |
| ■ Poolformer-S12 [64] | 224×224 | 12 | 2.0 | 77.2 | 93.5 |
| ■ Poolformer-S36 [64] | 224×224 | 31 | 5.2 | 81.4 | 95.5 |
| ■ Poolformer-M48 [64] | 224×224 | 73 | 11.9 | 82.5 | 96.0 |

# Network Architecture

**Pyramid architecture.** Pyramid architecture considers the multi-scale property of images by extracting features with gradually smaller spatial size as the layer goes deeper, such as ResNet [17] and PVT [57]. Empirical evidences show that pyramid architecture is effective for visual tasks [57]. Thus, we utilize the advanced design and build four versions of pyramid ViG models. The details are shown in Table 2. Note that we utilize the spatial reduction [57] in the first two stages to handle large number of nodes.

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

| Stage | Output size | PyramidViG-Ti | PyramidViG-S | PyramidViG-M | PyramidViG-B |
|---|---|---|---|---|---|
| Stem | $\frac{H}{4} \times \frac{W}{4}$ | Conv×3 | Conv×3 | Conv×3 | Conv×3 |
| Stage 1 | $\frac{H}{4} \times \frac{W}{4}$ | $\begin{bmatrix} D=48 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=80 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=96 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=128 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ |
| Downsample | $\frac{H}{8} \times \frac{W}{8}$ | Conv | Conv | Conv | Conv |
| Stage 2 | $\frac{H}{8} \times \frac{W}{8}$ | $\begin{bmatrix} D=96 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=160 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=192 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=256 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ |
| Downsample | $\frac{H}{16} \times \frac{W}{16}$ | Conv | Conv | Conv | Conv |
| Stage 3 | $\frac{H}{16} \times \frac{W}{16}$ | $\begin{bmatrix} D=240 \\ E=4 \\ K=9 \end{bmatrix} \times 6$ | $\begin{bmatrix} D=400 \\ E=4 \\ K=9 \end{bmatrix} \times 6$ | $\begin{bmatrix} D=384 \\ E=4 \\ K=9 \end{bmatrix} \times 16$ | $\begin{bmatrix} D=512 \\ E=4 \\ K=9 \end{bmatrix} \times 18$ |
| Downsample | $\frac{H}{32} \times \frac{W}{32}$ | Conv | Conv | Conv | Conv |
| Stage 4 | $\frac{H}{32} \times \frac{W}{32}$ | $\begin{bmatrix} D=384 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=640 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=768 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ | $\begin{bmatrix} D=1024 \\ E=4 \\ K=9 \end{bmatrix} \times 2$ |
| Head | $1 \times 1$ | Pooling & MLP | Pooling & MLP | Pooling & MLP | Pooling & MLP |
| Parameters (M) | | 10.7 | 27.3 | 51.7 | 92.6 |
| FLOPs (B) | | 1.7 | 4.6 | 8.9 | 16.8 |

# Experiments

**Dataset**

- Image classification - ImageNet ILSVRC 2012
- Object detection - COCO 2017

**Main Results on ImageNet**

- Isotropic VIG
- Object detection - COCO 2017

Table 4: Results of ViG and other isotropic networks on ImageNet. ♠ CNN, ■ MLP, ♦ Transformer, ★ GNN.

| Model | Resolution | Params (M) | FLOPs (B) | Top-1 | Top-5 |
|---|---|---|---|---|---|
| ♠ ResMLP-S12 conv3x3 [50] | 224×224 | 16.7 | 3.2 | 77.0 | - |
| ♠ ConvMixer-768/32 [52] | 224×224 | 21.1 | 20.9 | 80.2 | - |
| ♠ ConvMixer-1536/20 [52] | 224×224 | 51.6 | 51.4 | 81.4 | - |
| ♦ ViT-B/16 [9] | 384×384 | 86.4 | 55.5 | 77.9 | - |
| ♦ DeiT-Ti [51] | 224×224 | 5.7 | 1.3 | 72.2 | 91.1 |
| ♦ DeiT-S [51] | 224×224 | 22.1 | 4.6 | 79.8 | 95.0 |
| ♦ DeiT-B [51] | 224×224 | 86.4 | 17.6 | 81.8 | 95.7 |
| ■ ResMLP-S24 [50] | 224×224 | 30 | 6.0 | 79.4 | 94.5 |
| ■ ResMLP-B24 [50] | 224×224 | 116 | 23.0 | 81.0 | 95.0 |
| ■ Mixer-B/16 [49] | 224×224 | 59 | 11.7 | 76.4 | - |
| ★ ViG-Ti (ours) | 224×224 | 7.1 | 1.3 | **73.9** | **92.0** |
| ★ ViG-S (ours) | 224×224 | 22.7 | 4.5 | **80.4** | **95.2** |
| ★ ViG-B (ours) | 224×224 | 86.8 | 17.7 | **82.3** | **95.9** |

**ViG-Ti achieves 73.9% top-1 accuracy, 1.7% higher than the DeiT-Ti model, at a similar computational cost.**

# Experiments

## Pyramid ViG

Table 5: Results of Pyramid ViG and other pyramid networks on ImageNet. ♠ CNN, ■ MLP, ◆ Transformer, ★ GNN.

| Model | Resolution | Params (M) | FLOPs (B) | Top-1 | Top-5 |
|---|---|---|---|---|---|
| ♠ ResNet-18 [17, 59] | 224×224 | 12 | 1.8 | 70.6 | 89.7 |
| ♠ ResNet-50 [17, 59] | 224×224 | 25.6 | 4.1 | 79.8 | 95.0 |
| ♠ ResNet-152 [17, 59] | 224×224 | 60.2 | 11.5 | 81.8 | 95.9 |
| ♠ BoTNet-T3 [46] | 224×224 | 33.5 | 7.3 | 81.7 | - |
| ♠ BoTNet-T3 [46] | 224×224 | 54.7 | 10.9 | 82.8 | - |
| ♠ BoTNet-T3 [46] | 256×256 | 75.1 | 19.3 | 83.5 | - |
| ◆ PVT-Tiny [57] | 224×224 | 13.2 | 1.9 | 75.1 | - |
| ◆ PVT-Small [57] | 224×224 | 24.5 | 3.8 | 79.8 | - |
| ◆ PVT-Medium [57] | 224×224 | 44.2 | 6.7 | 81.2 | - |
| ◆ PVT-Large [57] | 224×224 | 61.4 | 9.8 | 81.7 | - |
| ◆ CvT-13 [60] | 224×224 | 20 | 4.5 | 81.6 | - |
| ◆ CvT-21 [60] | 224×224 | 32 | 7.1 | 82.5 | - |
| ◆ CvT-21 [60] | 384×384 | 32 | 24.9 | 83.3 | - |
| ◆ Swin-T [35] | 224×224 | 29 | 4.5 | 81.3 | 95.5 |
| ◆ Swin-S [35] | 224×224 | 50 | 8.7 | 83.0 | 96.2 |
| ◆ Swin-B [35] | 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 [71] | 224×224 | 12 | 2.0 | 77.2 | 93.5 |
| ■ Poolformer-S36 [71] | 224×224 | 31 | 5.2 | 81.4 | 95.5 |
| ■ Poolformer-M48 [71] | 224×224 | 73 | 11.9 | 82.5 | 96.0 |
| ★ Pyramid ViG-Ti (ours) | 224×224 | 10.7 | 1.7 | **78.2** | **94.2** |
| ★ Pyramid ViG-S (ours) | 224×224 | 27.3 | 4.6 | **82.1** | **96.0** |
| ★ Pyramid ViG-M (ours) | 224×224 | 51.7 | 8.9 | **83.1** | **96.4** |
| ★ Pyramid ViG-B (ours) | 224×224 | 92.6 | 16.8 | **83.7** | **96.5** |

- It showed similar or better performance than SOTA using CNN, MLP, and transformer.
- GNNs have the potential to become basic components in visual tasks.

# Conclusion

- Image를 graph로 표현하고 graph neural network를 visual task에 이용하는 방법을 조사했습니다.
- 이미지를 여러 개의 패치로 나누고 , 노드처럼 처리했어요
- 이러한 node를 기반으로 graph를 만드는 것은 불규칙하고 복잡합 대상을 표현하는데 더 적합하다.
- Information diversity를 위해 각각의 node 안에 추가적인 feature transformation를 실시한다.
- Image recognition과 Object detection의 실험해서 ViG의 우수함을 보였다.