VGN: A Graph Based Computer Vision Model

CaoShuo Li[1],Zheyi Cai[2],HuaDong Chen[3],Qia Hu[4],Ling Jiang[5]

[1]23020231154144,Artificial Intelligence Class
[2]23020231154166,Artificial Intelligence Class
[3]23020231154168,Artificial Intelligence Class
[4]23020231154186,Artificial Intelligence Class
[5]33320231150351,Information College Class

Abstract

In the field of computer vision, traditional methods mainly view images as grids or sequences. However, the potential of representing images as graph structures has not been fully explored. Compared to traditional representation methods, representing images in the form of graphs offers better flexibility, captures more non-local information, and reduces redundant information. In this paper, we propose a method to represent graphics as graphs and construct the VGN model for feature extraction from the graph structure. We first divide the input image into patches using traditional methods, where each patch is regarded as a node in the graph. Subsequently, we train the KNeighborsClassifier to obtain the neighboring nodes for each node. The VGN network consists of two basic modules, GPM and FFN. The GPM module is used for information aggregation and node update. The FFN module is for feature mapping, reducing the over-smoothing phenomenon, and preserving the diversity of node features. We represent the VGN model using both pyramid and isotropic structures, demonstrating excellent performance in image classification and object detection, thereby validating the effectiveness of our proposed model.

Introduction

Computer vision is a field that studies how to enable computers to understand visual data like humans. It encompasses tasks such as image classification, object detection, and instance segmentation.With the development of deep learning, Lecun(LeCun et al. 1998) and others introduced convolutional neural networks. Kaiming and others proposed ResNet(He et al. 2016a), introducing the concept of residual modules. In recent years, transformers have also been extensively applied in the computer vision domain, achieving outstanding results in many tasks. VIT, DETP, and SETR are among the representative achievements.

In convolutional neural networks, images are represented as grids in Euclidean space based on pixels. For instance, in vision transformers, images are split into fixed-size patches and then linearized into vector sequences. In computer vision tasks, it's often necessary to recognize irregularly shaped complex objects, which might also be occluded. In such cases, viewing images as grids or sequences lacks flexibility and struggles to capture this irregularity. Representing images with a graph structure can better capture non-local and nonuniform information, reduce redundant information, and is conducive to learning irregular structures and complex patterns.

We first convolve the input image to obtain patches, with each patch treated as a node in the graph structure. Subsequently, we train a KNeighborsClassifier to obtain the neighboring nodes for each node. After constructing the graph, we propose a graph-based computer vision model, VGN, which includes two basic modules: GPM and FFN. The GPM module uses Max-Relative GraphConv(Li et al. 2019b) to aggregate node information. Node updates utilize Multi-head to capture information from multiple subspaces. Considering that traditional GCNs experience over-smoothing as the network depth increases, the FFN module uses feature mapping and borrows the skip connection structure from ResNet to alleviate this issue. Based on the VGN model, we construct both pyramid and isotropic structures and conduct comparative experiments with computer vision models using both structures. Experimental results show that our proposed model exhibits excellent performance in image classification and object detection tasks. Since our model combines graph neural networks with computer vision, the latest research results from both fields can be further improved upon this foundation, promising a bright future.

Related Work

A graph is a fundamental concept in computer science, used to represent relationships between objects. A graph consists of a set of nodes and edges connecting them. MarcoGori(Scarselli et al. 2008) and others published a paper titled "The Graph Neural Network Model" in 2009, introducing the concept of graph neural networks for the first time, using a recursive method to update node information. Influenced by convolutional neural networks, Thomas Kipf and others proposed Graph Convolutional Neural Networks(Kipf and Welling 2016). Based on this, a series of variations were developed. For example, Velivckovic(Veličković et al. 2017) and others introduced the attention mechanism into the graph structure, proposing Graph Attention Networks (GATs). Simonovsky(Simonovsky and Komodakis 2017) and others incorporated edge information into the graph convolution process, introducing Edge-Conditioned Convolutions. Graph neural networks can predict missing parts in knowledge graphs, aiding in the construction of knowledge graphs. Through graph convolution, patterns and relationships between nodes can be captured, enabling tasks such as node classification, link prediction, recommendation systems, and more.

Convolutional neural networks mainly used for processing grid-structured data. They extract local features by sliding through convolutional blocks, eventually learning complex patterns. In recent years, convolutional neural networks have developed rapidly, with representative works including ResNet and Nas(Zoph and Le 2016). CNNs in computer vision can be constructed using a pyramid structure, gradually reducing the resolution of the image, which better represents image features at multiple scales and enhances the network's ability to recognize objects of different sizes. Moreover, the self-attention in the Transformer structure can capture long-distance relationships in images, so it has been applied to the computer vision field, achieving excellent performance in various tasks. ViT and DETP are among the representative achievements. Unlike CNNs, where images are represented as grids, images in Transformer structures are represented as sequences.

Proposed Solution

First, we select different datasets based on different computer vision tasks. For classification tasks, we choose the representative dataset ImageNet ILSVRC 2012(Russakovsky et al. 2015), which contains 1.2 million training images, 100,000 test images, and 50,000 validation images, totaling 1,000 categories. For object detection tasks, we choose COCO 2017(Tsung-Yi Lin 2014), which includes 118k training images, 5k validation images, and about 41k unlabeled test images, with a total of 80 categories.

We divided an image with size of $H \times W \times 3$ into N patches. By transforming each patch into a feature vector $X_i \in R^D$, we have $X = [x_1, x_2, \ldots, x_N]$ where D is the feature dimension and $i = 1, 2, \ldots, N$. These features can be viewed as a set of unordered nodes which are denoted as $V = v_1, v_2, \ldots, v_N$. For each node v_i, we find its K nearest neighbors $N(v_i)$ and add an edge e_{ji} directed from v_j to v_i for all $v_j N(v_i)$. Then we obtain a graph $G = (V, \xi)$ where ξ denote all the edges. We denote the graph construction process as $G = G(X)$ in the following. We start from the features$X \in R^{N\times D}$. We construct a graph based on the features: $G = G(X)$. A graph convolutional layer can exchange information between nodes by aggregating features from its neighbor nodes. Specifically, graph convolution operates as follows:

$$g_{\prime = F(G,W) = Update(Aggregate(g, W_{agg}), W_{update})} (1)$$

where W_{agg} and W_{update} are the learnable weights of the aggregation and update operations, respectively.

We represent images with a graph structure and implement the GPM and FFN, two basic modules in the VGN model. Figure 1 is the framework of the proposed VGN model. GPM chooses Max-Relative of graph convolution as the benchmark, performs feature mapping before and after input, and borrows the skip connection structure from ResNet to alleviate the over-smoothing problem. To compare with convolutional neural networks based on grid representation and Transformers based on sequence representation, we construct pyramid structures and isotropic architectures for VGN, respectively. These two structures also correspond to the mainstream representations of convolutional neural networks and Transformers in computer vision tasks. Considering that different tasks have different requirements for model size, we further subdivide the model based on the number of model parameters. Then we evaluate the model on various visual tasks to verify the effectiveness of our proposed model.

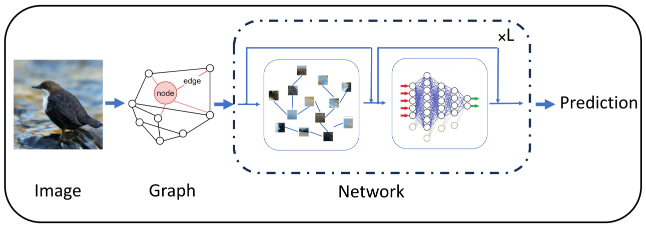


Figure 1: The framework of the proposed VGN model.

Considering that the GPM module is designed from the graph convolution module, we can refer to various variants of graph convolution, such as Max-Relative Graph-Conv, EdgeConv, and choose the most suitable graph convolution variant through experiments. Secondly, the neighborsof each node are obtained through the k-Nearest Neighbors algorithm, where k is a manually selected hyperparameter.Therefore, we will select different values for ablation experiments.

Finally, we will summarize the entire project, analyze the application prospects of our proposed model, and discuss where the model can be improved in the future.

Experiments

In this chapter, experiments are carried out to validate the performance of the ViG models in visual computing tasks such as image classification and object detection.

Datasets and Experimental Settings Datasets

For image classification, the standard ImageNet ILSVRC 2012 dataset is employed, which contains 1.2 million training images and 50,000 validation images across 1,000 different classes. The ImageNet dataset's licensing information is available at its official website: serves for object detection experiments, featuring 80 categories with 118,000 training images and 5,000 validation images. Licensing details for COCO can be found on its website: https://cocodataset.org/home. In terms of experimental configuration, ViG models use dilated aggregation[30] in their Grapher modules, with a dilation rate set as [l/4] for the l-th layer. The GELU(Hendrycks and Gimpel 2016) function is the chosen non-linear activation for specific equations. For ImageNet classification, the training strategy is derived from the DeiT (Touvron et al. 2020) protocol, and includes a suite of data augmentation techniques. For object detection

(Pyramid) ViG	Ti S M B
Epochs	300
Optimizer	AdamW(Loshchilov and Hutter 2017)
Batch size	1024
Start learning rate (LR)	2e-3
Learning rate schedule	Cosine
Warmup epochs	20
Weight decay	0.05
Label smoothing(Szegedy et al. 2015)	0.1
Stochastic path(Lin et al. 2014)	0.1 0.1 0.1 0.3
Repeated augment(Hoffer et al. 2019)	✓
RandAugment(Cubuk et al. 2020)	✓
Mixup prob.(Zhang et al. 2017)	0.8
Cutmix prob.(Yun et al. 2019)	1.0
Random erasing prob.(Zhong et al. 2017)	0.25
Exponential moving average	0.99996

Table 1: Training hyper-parameters for ImageNet

Model	Resolution	Params(M)	FLOPs(B)	Top-1	Top-5
ViG-Ti	224×224	7.1	1.3	**73.9**	**92**
ViG-S	224×224	22.7	4.5	**80.4**	**95.2**
ViG-B	224×224	86.8	17.7	**82.3**	**95.9**

Table 2: Results of ViG on ImageNet

Model	Resolution	Params(M)	FLOPs(B)	Top-1	Top-5
ResNet-50(He et al. 2016b)	224×224	25.6	4.1	**81.7**	**95**
BoTNet-T3(Srinivas et al. 2021)	224×224	33.5	7.3	**82.1**	**-**
PVT-Small(Wang et al. 2022)	224×224	24.5	3.8	**79.8**	**-**
CvT-13(Wu et al. 2021)	224×224	20	4.5	**81.6**	**-**
Swin-T(Liu et al. 2021)	224×224	29	4.5	**81.3**	**95.5**
CycleMLP-B2(Chen et al. 2021)	224×224	27	3.9	**81.6**	**-**
Poolformer-S12(Yu et al. 2021)	224×224	12	2	**77.2**	**93.5**
Pyramid VGN-Ti (ours)	224×224	10.7	1.7	**78.2**	**94.2**
Pyramid VGN-S (ours)	224×224	27.3	4.6	**82.1**	**96**

Table 3: Results of Pyramid VGN and other pyramid networks on ImageNet

GraphConv	Params(M)	FLOPs(B)	Top-1
EdgeConv(Wang et al. 2019)	7.2	2.4	74.3
GIN(Xu et al. 2018)	7.0	1.3	72.8
GraphSAGE(Hamilton, Ying, and Leskovec 2017)	7.3	1.6	74.0
Max-Relative GraphConv(Li et al. 2019a)	7.1	1.3	73.9

Table 4: ImageNet results of different types of graph convolution. The basic architecture is ViG-Ti

GraphConv	FC in Grapher module	FFN module	FLOPs(B)	FLOPs (B)	Top-1
✓	✗	✗	5.8	1.4	67
✓	✓	✗	4.4	1.4	73.4
✓	✗	✓	7.7	1.3	73.6
✓	✓	✓	7.1	1.3	73.9

Table 5: The effects of modules in ViG on ImageNet

K	3.00	6.00	9.00	12.00	15.00	20.00	9 to 18
Top-1	72.20	73.40	73.60	73.60	73.50	73.30	73.90

Table 6: Top-1 accuracy vs. K on ImageNet

h	1	2	4	6	8
FLOPs / Top-1	1.6B / 74.2	1.4B / 74.0	1.3B / 73.9	1.2B / 73.7	1.2B / 73.7

Table 7: Top-1 accuracy vs. h on ImageNet

on COCO, RetinaNet(Lin et al. 2017) and Mask R-CNN(He et al. 2017) frameworks are employed with Pyramid ViG as the backbone. Models are trained on NVIDIA 4080 GPUs using a “1×” schedule, with implementation carried out in both PyTorch and MindSpore.The details are shown in Table 1.

Main Results on ImageNet

Isotropic Vision Graph (ViG) Architecture: The isotropic design of the ViG maintains consistent feature dimensions throughout its core processing stages, facilitating scalability and hardware acceleration compatibility. Such design principles are frequently applied in the domain of NLP transformers and are increasingly being utilized in vision-based neural networks like ConvMixer (Tolstikhin et al. 2021), Vision Transformer (ViT)(Dosovitskiy et al. 2020), and ResMLP. In Table 2, we list the performance of vig in each image with better results.

Pyramidal Vision Graph (ViG) Structure: Adopting a pyramidal shape, the ViG progressively condenses the spatial dimensions of the feature maps as it delves deeper, which is an effective approach for capturing multi-scale features and exploiting the inherent scale invariance of images. This kind of architecture is a hallmark of leading-edge networks such as the ResNet, Swin Transformer(Liu et al. 2021), and CycleMLP(Chen et al. 2021). In Table3, we draw a comparison between our Pyramidal VGN models and these well-known pyramidal frameworks. The results indicate that our Pyramidal VGN lineup either surpasses or rivals the performance of contemporary pyramid-structured networks across CNNs, MLPs, and transformers. These findings suggest that the graph neural network framework is not only effective for image-related tasks but also has the potential to become an integral element of future computer vision systems.

Evaluating the Variations

Our ablation studies are centered on the ImageNet classification task utilizing the isotropic ViG-Ti as our foundational design.

Graph Convolution Varieties. We explored several prominent graph convolution techniques, such as EdgeConv(Wang et al. 2019), GIN(Xu et al. 2018), GraphSAGE(Hamilton, Ying, and Leskovec 2017), and Max-Relative GraphConv(Li et al. 2019a). The results, shown in Table 4, reveal that all tested graph convolution variants exceed the top-1 accuracy of the DeiT-Ti model. This illustrates the adaptability of the ViG framework. Notably, Max-Relative GraphConv delivers the most balanced results in terms of FLOPs and accuracy. Hence, we've made it the default graph convolution method for the remainder of our studies.

Impact of ViG Modules. In adapting graph neural networks for visual tasks, we've integrated FC layers within the Grapher module and employed FFN blocks for feature transformation. We assessed the influence of these components through a series of ablation tests. We equalized the FLOPs across models for a fair comparison by adjusting their feature dimensions. As delineated in Table 5, solely relying on graph convolution for image classification yields suboptimal results. However, incorporating FC and FFN enhances accuracy progressively.

Neighbor Count. When constructing the graph, the parameter K specifies the count of neighboring nodes for aggregation. A low K hampers information flow, while a high K can cause over-smoothing. We experimented with K values ranging from 3 to 20 and present our findings in Table 6. The optimal neighbor count for ImageNet classification appears to be between 9 and 15.

Heads Count. The multi-head update mechanism within the Grapher module allows for the processing of node features across various subspaces. The number of heads, h, governs the diversity of transformations within these subspaces and also affects the FLOPs. After testing head counts from 1 to 8, as detailed in Table 7, we found that the impact on both FLOPs and top-1 ImageNet accuracy is marginal across different h values. Consequently, we have chosen h = 4 as the standard setting to achieve the best balance between computational cost and performance.

Assessing ViG in Object Detection

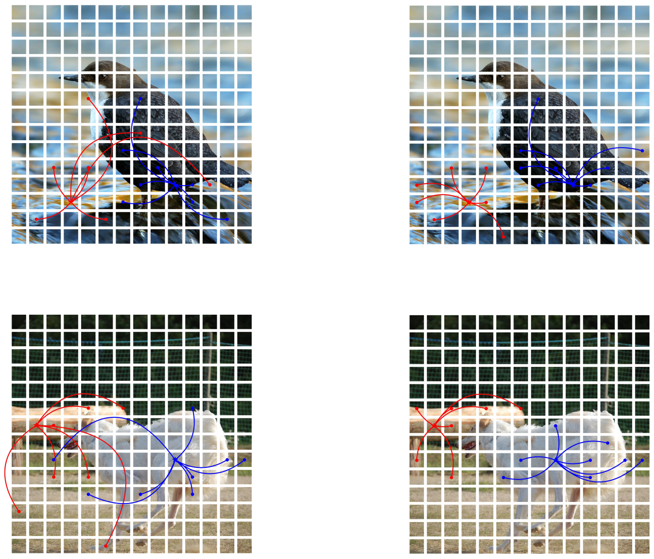
To determine the adaptability of ViG, we extend its application to object detection. Utilizing a Pyramid ViG-S architecture pre-trained on ImageNet as a foundational component, we integrate it within RetinaNet (Lin et al. 2017) and Mask R-CNN(He et al. 2017) - two well-established detection frameworks. We adhere to the standard ”1x” training schedule and calculate FLOPs based on an input size of 1280×800. The outcomes, presented in Table 8, highlight Pyramid VGN-S outperforming several benchmark backbones such as ResNet (He et al. 2016b), CycleMLP(Chen et al. 2021), and Swin Transformer(Liu et al. 2021) in both RetinaNet and Mask R-CNN frameworks. These promising results underscore the robustness and versatility of the ViG model.

Exploring ViG's Graph Visualization

To gain insights into ViG's operational mechanism, we examine its graph structure by visualizing the links it forms. Figure 2 outlines the graphs from two sample images at varying network depths (the 1st and the 12th blocks). In these illustrations, the center node is marked by a pentagram, and nodes sharing the same color represent its first-order neighbors. We focus on visualizing two center nodes

Backbone	RetinaNet 1×							
	Param	FLOPs	mAP	AP50	AP75	APS	APM	APL
ResNet50(He et al. 2016b)	37.7M	239.3B	36.3	55.3	38.6	19.3	40	48.8
CycleMLP-B2(Chen et al. 2021)	36.5M	230.9B	40.6	61.4	43.2	22.9	44.4	54.5
Swin-T(Liu et al. 2021)	38.5M	244.8B	41.5	62.1	44.2	25.1	44.9	55.5
Pyramid ViG-S (ours)	36.2M	240.0B	41.8	63.1	44.7	28.5	45.4	53.4
Backbone	Mask R-CNN 1×							
	Param	FLOPs	AP(b)	AP50(b)	AP75(b)	AP(n)	AP50(m)	AP75(3)
ResNet50(He et al. 2016b)	44.2M	260.1B	38	58.6	41.4	34.4	55.1	36.7
CycleMLP-B2(Chen et al. 2021)	46.5M	249.5B	42.1	64	45.7	38.9	61.2	41.8
Swin-T(Liu et al. 2021)	47.8M	264.0B	42.2	64.6	46.2	39.1	61.6	42
Pyramid ViG-S (ours)	45.8M	258.8B	42.6	65.2	46	39.4	62.4	41.6

Table 8: Object detection and instance segmentation results on COCO val2017. Our Pyramid VGN is compared with other backbones on RetinaNet and Mask R-CNN frameworks



(a)Graph connection in the 1st block(b)Graph connection in the 12th block

Figure 2: Visualization of the constructed graph structure. The pentagram is the center node, and the nodes with the same color are its neighbors in the graph.

to avoid clutter from displaying all connections. Initial observations suggest that the ViG model is adept at selecting nodes related to content as its neighbors. At the initial stages, neighboring nodes are chosen based on superficial attributes like color and texture. As the network goes deeper, the connections between the center node and its neighbors become more semantically aligned, often within the same category. This ability of the ViG to progressively form associations between nodes based on content and semantic characteristics contributes significantly to its object recognition capabilities.

Conclusion

In this study, we explore algorithms such as VIG, which uses graph data structures to represent images, and apply graph neural networks (GNNs) to a variety of visual tasks. By segmenting an image into patches and treating each patch as a node, we are able to construct a graph that more accurately captures the intricate irregular shapes of real-world objects. However, the direct use of graph convolution on such image-induced graphs tends to create an over-smoothing problem, leading to suboptimal performance. To address this issue, the algorithm incorporates enhanced feature transformations in each node to promote rich diversity of information. With this graph-based image representation and an improved graph processing unit, we use the Visual GNN (ViG) framework in both isotropic and pyramidal formats. The benefits of ViG's architecture have been clearly demonstrated through rigorous testing on image classification and object detection benchmarks.

References

Chen, S.; Xie, E.; Ge, C.; Liang, D.; and Luo, P. 2021. CycleMLP: A MLP-like Architecture for Dense Prediction.

Cubuk, E. D.; Zoph, B.; Shlens, J.; and Le, Q. 2020. RandAugment: Practical Automated Data Augmentation with a Reduced Search Space. In *Neural Information Processing Systems*.

Dosovitskiy, A.; Beyer, L.; Kolesnikov, A.; Weissenborn, D.; and Houlsby, N. 2020. An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale.

Hamilton, W. L.; Ying, R.; and Leskovec, J. 2017. Inductive Representation Learning on Large Graphs.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask R-CNN. *IEEE Transactions on Pattern Analysis Machine Intelligence*.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016a. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 770–778.

He, K.; Zhang, X.; Ren, S.; and Sun, J. 2016b. Deep Residual Learning for Image Recognition. *IEEE*.

Hendrycks, D.; and Gimpel, K. 2016. Gaussian Error Linear Units (GELUs).

Hoffer, E.; Ben-Nun, T.; Hubara, I.; Giladi, N.; Hoefler, T.; and Soudry, D. 2019. Augment your batch: better training with larger batches.

Kipf, T. N.; and Welling, M. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Li, G.; Mueller, M.; Thabet, A.; and Ghanem, B. 2019a. DeepGCNs: Can GCNs Go as Deep as CNNs?

Li, G.; Muller, M.; Thabet, A.; and Ghanem, B. 2019b. Deepgcns: Can gcns go as deep as cnns? In *Proceedings of the IEEE/CVF international conference on computer vision*, 9267–9276.

Lin, T. Y.; Goyal, P.; Girshick, R.; He, K.; and Dollár, P. 2017. Focal Loss for Dense Object Detection. *IEEE Transactions on Pattern Analysis Machine Intelligence*, PP(99): 2999–3007.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y.; Zhang, Z.; Lin, S.; and Guo, B. 2021. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows.

Loshchilov, I.; and Hutter, F. 2017. Decoupled Weight Decay Regularization.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115: 211–252.

Scarselli, F.; Gori, M.; Tsoi, A. C.; Hagenbuchner, M.; and Monfardini, G. 2008. The graph neural network model. *IEEE transactions on neural networks*, 20(1): 61–80.

Simonovsky, M.; and Komodakis, N. 2017. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3693–3702.

Srinivas, A.; Lin, T. Y.; Parmar, N.; Shlens, J.; and Vaswani, A. 2021. Bottleneck Transformers for Visual Recognition.

Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; and Wojna, Z. 2015. Rethinking the Inception Architecture for Computer Vision.

Tolstikhin, I.; Houlsby, N.; Kolesnikov, A.; Beyer, L.; and Dosovitskiy, A. 2021. MLP-Mixer: An all-MLP Architecture for Vision.

Touvron, H.; Cord, M.; Douze, M.; Massa, F.; and Jégou, H. 2020. Training data-efficient image transformers distillation through attention.

Tsung-Yi Lin, S. J. B. L. D. B. R. B. G. J. H. . C. L. Z., Michael Maire. 2014. Microsoft COCO: Common Objects in Context. *CoRR*, abs/1405.0312.

Veličković, P.; Cucurull, G.; Casanova, A.; Romero, A.; Lio, P.; and Bengio, Y. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903*.

Wang, W.; Xie, E.; Li, X.; Fan, D. P.; Song, K.; Liang, D.; Lu, T.; Luo, P.; and Shao, L. 2022. PVT v2: Improved baselines with Pyramid Vision Transformer. *:*, 8(3): 10.

Wang, Y.; Sun, Y.; Liu, Z.; Sarma, S. E.; Bronstein, M. M.; and Solomon, J. M. 2019. Dynamic Graph CNN for Learning on Point Clouds. *Association for Computing Machinery (ACM)*, (5).

Wu, H.; Xiao, B.; Codella, N.; Liu, M.; Dai, X.; Yuan, L.; and Zhang, L. 2021. CvT: Introducing Convolutions to Vision Transformers.

Xu, K.; Hu, W.; Leskovec, J.; and Jegelka, S. 2018. How Powerful are Graph Neural Networks?

Yu, W.; Luo, M.; Zhou, P.; Si, C.; Zhou, Y.; Wang, X.; Feng, J.; and Yan, S. 2021. MetaFormer is Actually What You Need for Vision.

Yun, S.; Han, D.; Oh, S. J.; Chun, S.; Choe, J.; and Yoo, Y. 2019. CutMix: Regularization Strategy to Train Strong Classifiers with Localizable Features.

Zhang, H.; Cisse, M.; Dauphin, Y. N.; and Lopez-Paz, D. 2017. mixup: Beyond Empirical Risk Minimization.

Zhong, Z.; Zheng, L.; Kang, G.; Li, S.; and Yang, Y. 2017. Random Erasing Data Augmentation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(7).

Zoph, B.; and Le, Q. V. 2016. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*.