# IMAGE AS SET OF POINTS

ICLR 2023在审

- 作者介绍
- 研究背景
- 本文方法
- 实验效果
- 总结反思

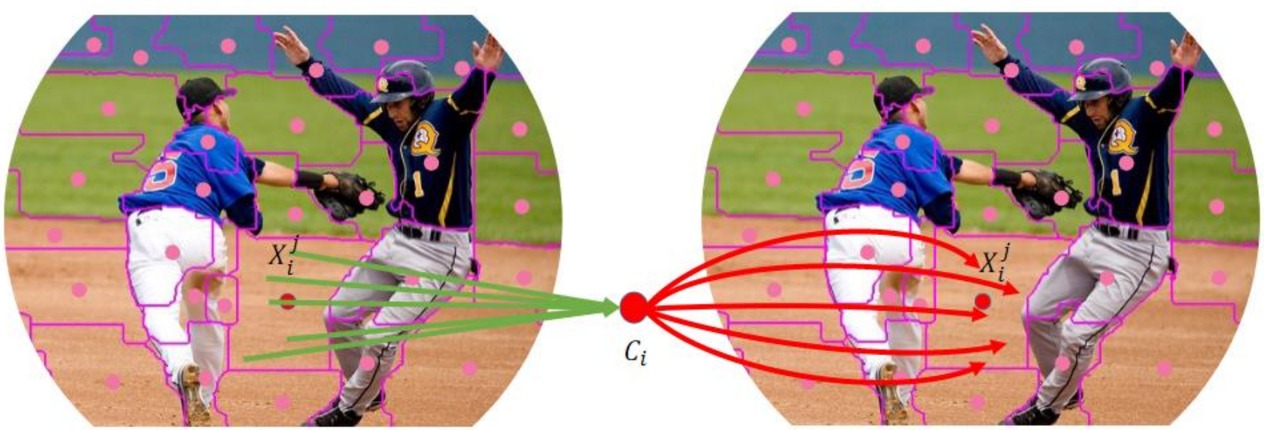
# 研究背景



- 图像是什么？怎样表示？
- ConvNet: pixels, 卷积提取局部特征
- ViT : sequence of patchs, 注意力机制学习全局表征
- Context Cluster: set of points, 聚类算法进行分组、聚合

---

- ConvNet，ViT 都是按照既定形状划分，本文是一种context-aware的划分方式。



# 研究背景



- 内容感知的patch划分
  - Deformable Conv
  - Deformable DETR
  - Deformable Attention

- 多部位特征的提取
- 可解释性

- 作者介绍
- 研究背景
- 本文方法
- 实验效果
- 总结反思

# Architecture



- 类似Swin Transformer的层次结构
- 从MetaFormer的角度看是用cluster替代self-attention

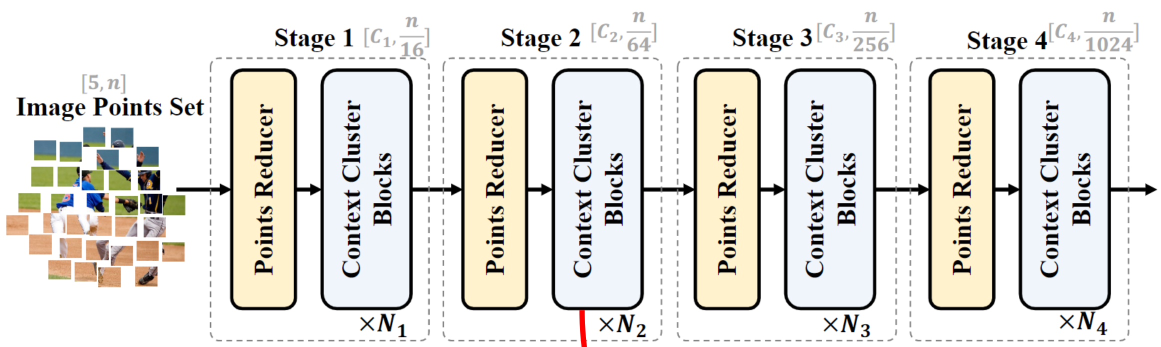
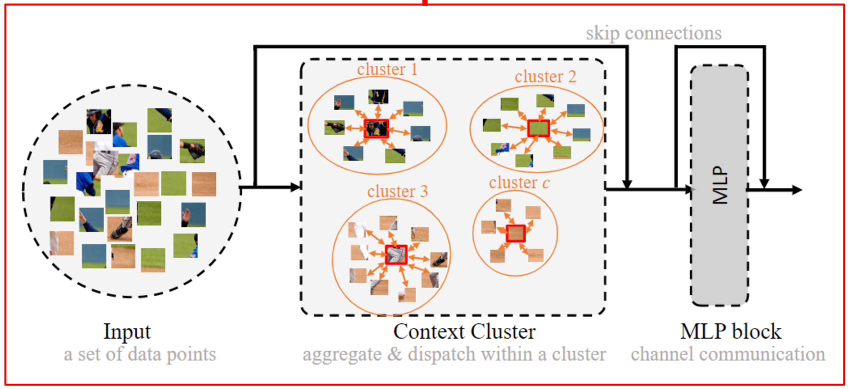



Figure 3: Context Cluster architecture with four stages. Given a set of image points, Context Cluster gradually reduces the point number and extracts deep features. Each stage begins with a points reducer, after which a succession of context cluster blocks is used to extract features.



# Point Reducer

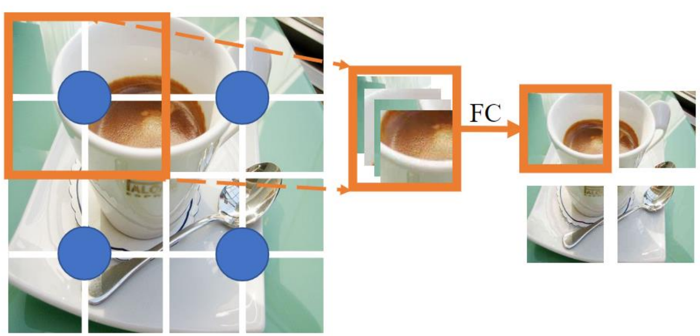


- image2point
  - 将输入图像的每个像素表示成（feature，position）向量。

| |
|---|
| R |
| G |
| B |
| x |
| y |

$$\mathbf{P} \in \mathbb{R}^{5\times n} \begin{cases} \mathbf{I} \in \mathbb{R}^{3\times w\times h}, \\ \left[\frac{i}{w} - 0.5, \frac{j}{h} - 0.5\right]. \end{cases}$$

- Feature extraction（与SwinT中的操作一致）
  - 先将k近邻点concat再经过Linear层变换，使得分辨率下降



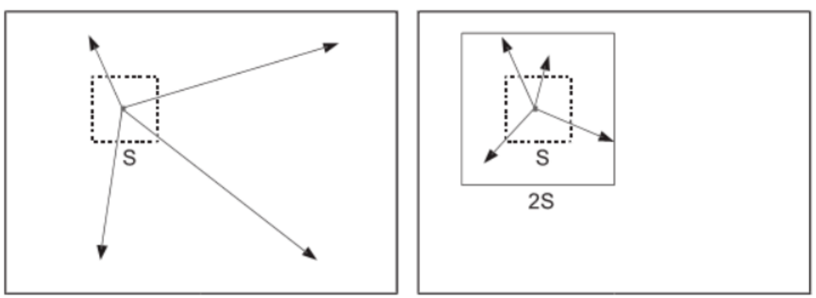

(a) Illustration of anchors for points reduction.

# Context Cluster



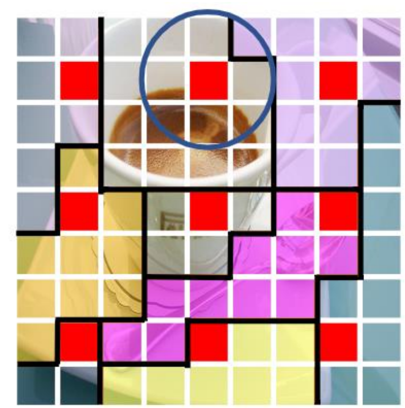
## Context Cluster

- 聚类算法：SLIC（super linear iterative clustering）
  - 设置均匀的聚类中心（S*S个），在设定的邻域内进行搜索。
  - 用余弦相似度衡量样本点的相似度。
  - 算法与Kmeans类似，区别就在于Kmeans在全局搜索，SLIC在局域搜索。




(*a*) standard *k*-means searches the entire image

(*b*) SLIC searches a limited region



(b) Demo of centers in CoC.

# Context Cluster



- 备注
  - $s_i$表示cluster中各点与中心的相似度
  - $p_i$表示各个点，对$p_i$做线性映射得到value空间
- Feature Aggregating
  - Cluster内特征聚合

$$g = \frac{1}{C}\left(v_c + \sum_{i=1}^{m} \mathrm{sig}\left(\alpha s_i + \beta\right) * v_i\right), \quad \text{s.t.,} \quad C = 1 + \sum_{i=1}^{m} \mathrm{sig}\left(\alpha s_i + \beta\right).$$

- Feature Dispatching
  - Cluster内特征传播

$$p_i' = p_i + \mathrm{FC}\left(\mathrm{sig}\left(\alpha s_i + \beta\right) * g\right).$$

# Details



- 聚类中心是固定的（考虑到计算效率）
- Clustering互不重叠

- 作者介绍
- 研究背景
- 本文方法
- 实验效果
- 总结反思

# Classification



Table 1: Comparison with representative small backbones on ImageNet-1k benchmark. Throughput (images / s) is measured on a single V100 GPU with a batch size of 128, and is averaged by the last 500 iterations. All models are trained and tested at 224×224 resolution, except ViT-B and ViT-L.

| | Method | Param. | GFLOPs | Top-1 | Throughputs (images/s) |
|---|---|---|---|---|---|
| MLP | ♣ ResMLP-12 (Touvron et al., 2021a) | 15.0 | 3.0 | 76.6 | 511.4 |
| | ♣ ResMLP-24 (Touvron et al., 2021a) | 30.0 | 6.0 | 79.4 | 509.7 |
| | ♣ ResMLP-36 (Touvron et al., 2021a) | 45.0 | 8.9 | 79.7 | 452.9 |
| | ♣ MLP-Mixer-B/16 (Tolstikhin et al., 2021) | 59.0 | 12.7 | 76.4 | 400.8 |
| | ♣ MLP-Mixer-L/16 (Tolstikhin et al., 2021) | 207.0 | 44.8 | 71.8 | 125.2 |
| | ♣ gMLP-Ti (Liu et al., 2021a) | 6.0 | 1.4 | 72.3 | 511.6 |
| | ♣ gMLP-S (Liu et al., 2021a) | 20.0 | 4.5 | 79.6 | 509.4 |
| Attention | ♦ ViT-B/16 (Dosovitskiy et al., 2020) | 86.0 | 55.5 | 77.9 | 292.0 |
| | ♦ ViT-L/16 (Dosovitskiy et al., 2020) | 307 | 190.7 | 76.5 | 92.8 |
| | ♦ PVT-Tiny (Wang et al., 2021) | 13.2 | 1.9 | 75.1 | - |
| | ♦ PVT-Small (Wang et al., 2021) | 24.5 | 3.8 | 79.8 | - |
| | ♦ T2T-ViT-7 (Yuan et al., 2021a) | 4.3 | 1.1 | 71.7 | - |
| | ♦ DeiT-Tiny/16 (Touvron et al., 2021b) | 5.7 | 1.3 | 72.2 | 523.8 |
| | ♦ DeiT-Small/16 (Touvron et al., 2021b) | 22.1 | 4.6 | 79.8 | 521.3 |
| Convolution | ♠ ResNet18 (He et al., 2016) | 12 | 1.8 | 69.8 | 584.9 |
| | ♠ ResNet50 (He et al., 2016) | 26 | 4.1 | 79.8 | 524.8 |
| | ♠ ConvMixer-512/16 (Trockman et al., 2022) | 5.4 | - | 73.8 | - |
| | ♠ ConvMixer-1024/12 (Trockman et al., 2022) | 14.6 | - | 77.8 | - |
| | ♠ ConvMixer-768/32 (Trockman et al., 2022) | 21.1 | - | 80.16 | 142.9 |
| Cluster | ♥ Context-Cluster-Ti (ours) | 5.3 | 1.0 | 71.8 | 518.4 |
| | ♥ Context-Cluster-Ti‡ (ours) | 5.3 | 1.0 | 71.7 | 510.8 |
| | ♥ Context-Cluster-Small (ours) | 14.0 | 2.6 | 77.5 | 513.0 |
| | ♥ Context-Cluster-Medium (ours) | 27.9 | 5.5 | 81.0 | 325.2 |

# 3D point cloud Classification



Table 3: Classification results on ScanObjectNN. All results are reported on the most challenging variant (PB_T50_RS).

| Method | mAcc(%) | OA(%) |
|---|---|---|
| ♠ SpiderCNN (Xu et al., 2018) | 69.8 | 73.7 |
| ♠ DGCNN (Wang et al., 2019) | 73.6 | 78.1 |
| ♠ PointCNN (Li et al., 2018) | 75.1 | 78.5 |
| ♠ GBNet (Qiu et al., 2021) | 77.8 | 80.5 |
| ♦ PointBert (Yu et al., 2022d) | - | 83.1 |
| ♦ Point-MAE (Pang et al., 2022) | - | 85.2 |
| ♦ Point-TnT (Berg et al., 2022) | 81.0 | 83.5 |
| ♣ PointNet (Qi et al., 2017a) | 63.4 | 68.2 |
| ♣ PointNet++ (Qi et al., 2017b) | 75.4 | 77.9 |
| ♣ BGA-PN++ (Uy et al., 2019) | 77.5 | 80.2 |
| ♣ PointMLP (Ma et al., 2022) | 83.9 | 85.4 |
| ♣ PointMLP-elite (Ma et al., 2022) | 81.8 | 83.8 |
| ♥ PointMLP-CoC (ours) | **84.4**$_{\uparrow 0.5}$ | **86.2**$_{\uparrow 0.8}$ |

# Detection & Segmentation



Table 4: COCO object detection and instance segmentation results using Mask-RCNN (1×).

| Family | Backbone | Params | $AP^{box}$ | $AP^{box}_{50}$ | $AP^{box}_{75}$ | $AP^{mask}$ | $AP^{mask}_{50}$ | $AP^{mask}_{75}$ |
|---|---|---|---|---|---|---|---|---|
| Conv. | ♠ ResNet-18 | 31.2M | 34.0 | 54.0 | 36.7 | 31.2 | 51.0 | 32.7 |
| Attention | ♦ PVT-Tiny | 32.9M | 36.7 | 59.2 | 39.3 | 35.1 | 56.7 | 37.3 |
| Cluster | ♥ CoC-Small/4 | 33.6M | 35.9 | 58.3 | 38.3 | 33.8 | 55.3 | 35.8 |
| | ♥ CoC-Small/25 | 33.6M | **37.5** | **60.1** | **40.0** | **35.4** | **57.1** | **37.9** |
| | ♥ CoC-Small/49 | 33.6M | 37.2 | 59.8 | 39.7 | 34.9 | 56.7 | 37.0 |

Table 5: Semantic segmentation performance of different backbones with Semantic FPN on the ADE20K validation set.

| Backbone | Params | mIoU(%) |
|---|---|---|
| ♠ ResNet18 | 15.5M | 32.9 |
| ♦ PVT-Tiny | 17.0M | 35.7 |
| ♥ CoC-Small/4 | 17.7M | **36.6** |
| ♥ CoC-Small/25 | 17.7M | **36.4** |
| ♥ CoC-Small/49 | 17.7M | **36.3** |

Table 7: Semantic segmentation results of different backbones with Semantic-FPN on the ADE20K validation set.

| Family | Backbone | Params | mIoU(%) |
|---|---|---|---|
| Conv. | ♠ ResNet50 | 28.5M | 36.7 |
| Atten. | ♦ PVT-Small | 28.2M | 39.8 |
| Cluster | ♥ CoC-Medium/4 | 25.2M | **40.2** |
| Cluster | ♥ CoC-Medium/25 | 25.2M | **40.6** |
| Cluster | ♥ CoC-Medium/49 | 25.2M | **40.8** |

# 可视化



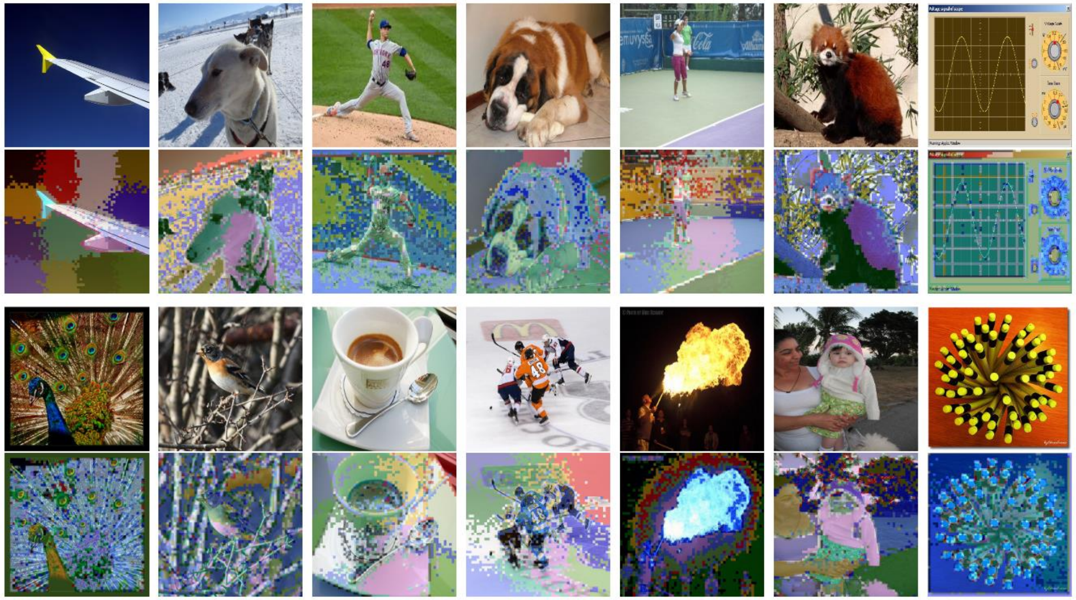
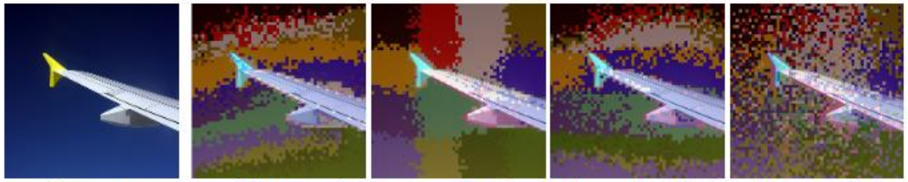


Figure 8: The clustering results of the last context cluster block in the first CoC-Tiny stage (without region partition). Without region partition, Our Context Cluster astonishingly displays "superpixel"-like clustering results, even in the early stage. we pick the most intriguing one out of the four heads.



Input Clustering results in 4 heads (16 clusters)

# 可视化



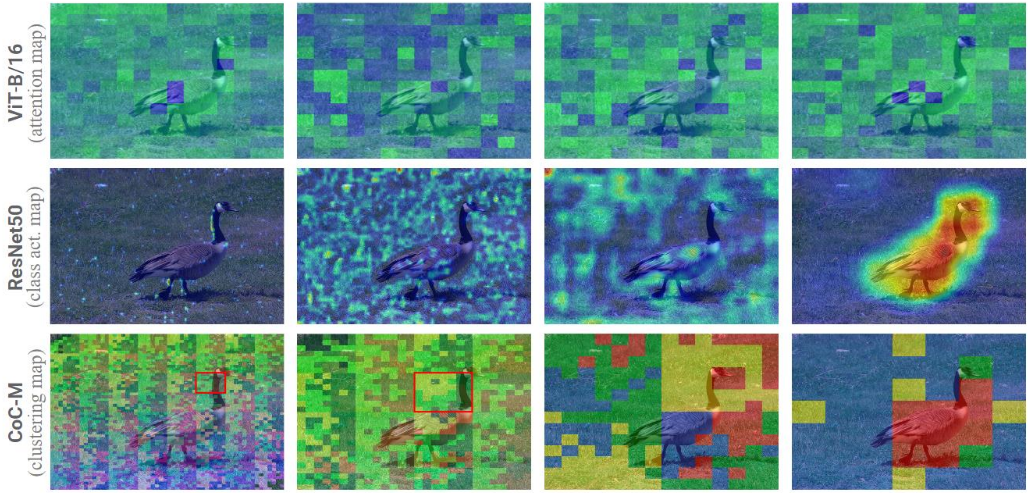



Figure 4: Visualization of activation map, class activation map, and clustering map for ViT-B/16, ResNet50, and our CoC-M, respectively. We plot the results of the last block in the four stages from left to right. For ViT-B/16, we select the [3rd, 6th, 9th, 12th] blocks, and show the cosine (instead of dot-product) attention map for the `cls-token`. We randomly select a head for both ViT-B/16 and our CoC-M. The clustering map shows that our Context Cluster is able to cluster similar contexts together (please zoom in to see details), showing what model learned visually.

- 作者介绍
- 研究背景
- 本文方法
- 实验效果
- 总结反思

# 总结



- 点集表示具有很好的通用性（feature+position）
- 聚类使得模型有好的可解释性

Thanks!