# Image as Set of Points

# Motivation

What is an image and how to extract features?

1. **Convolutional Networks**
   - Image is organized pixels in a rectangular shape
   - Extract features via convolutional operation in local region
2. **Vision Transformers**
   - Image is a sequence of patches
   - Extract features via attention mechanism in global region
3. **Context Clusters (CoCs)**
   - Image is a set of unorganized points
   - Extract features via simplified clustering algorithm

# SuperPixel

- Segment an image into regions by grouping a set of pixels that share common characteristics
- Common practice for image preprocessing
- Viewed as a type of *over-segmentation*

Figure 1: SuperPixel

# Context Cluster (Overview)

1. View image as a set of points
2. Sample $c$ centers
3. Aggregate point features within a cluster
4. Dispatch within a cluster

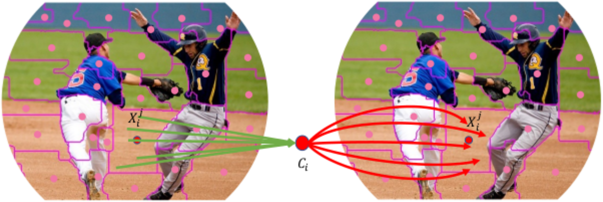


Figure 2: Overview of CoC

# Context Cluster Block (Overview)

1. Group a set of unorganized points
2. Communicate the points within a cluster
3. Apply MLP block

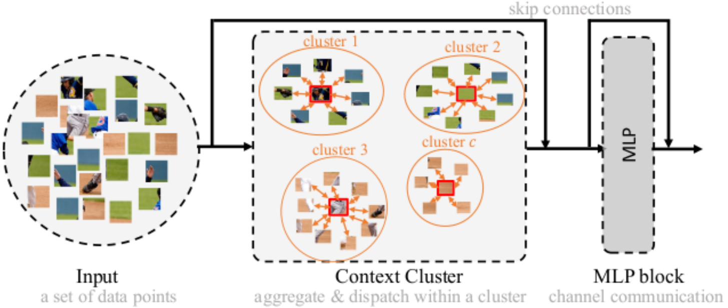


Figure 3: Context Cluster block

# Context Cluster Architecture (Overview)

1. Reduce the number of points for computational efficiency
2. Apply context cluster blocks to extract the features
3. Apply task-specific head to the features

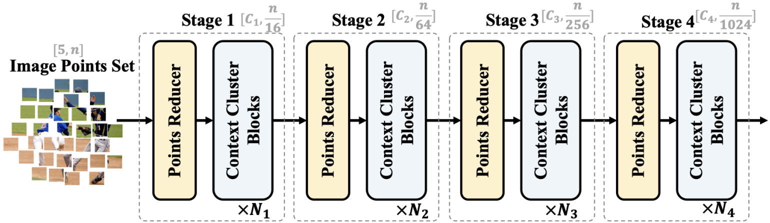


Figure 4: CoC architecture with four stages

# From Image to Set of Points

Given an image $\mathbf{I} \in \mathbb{R}^{3\times w\times h}$,

1. enhance the image by adding positional channel from the 2D coordinates information of each pixel $\mathbf{I}_{i,j}$
2. convert the augmented image to a a collection of points $\mathbf{P} \in \mathbb{R}^{5\times n}$, where $n = w \times h$

For example, if $(i, j)$-th pixel of $\mathbf{I}$ is given by

$$\mathbf{I}_{i,j} = (r, g, b),$$

it is converted to a point $\mathbf{p} \in \mathbb{R}^5$ in $P$, where

$$\mathbf{p} = (r, g, b, p_1, p_2),$$

with $[p_1, p_2] = \left[\frac{i}{w} - 0.5, \frac{j}{h} - 0.5\right]$.

# Point Reducer

1. Generate anchors evenly in the space
2. Take $k$ nearest neighbors and concatenate them along the channel dimension
3. Use FC layer to lower the dimension number

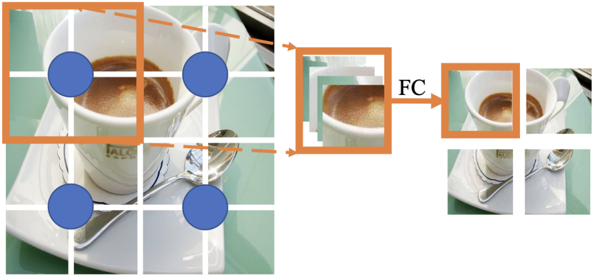


Figure 5: Point reducer

# Context Clustering

Given $\mathbf{P} \in \mathbb{R}^{n \times d}$,

1. Linearly project $\mathbf{P}$ to $\mathbf{P}_s$
2. Evenly propose $c$ centers (red blocks) in space
3. Get center features by averaging $k$ nearest points (blue circle)
4. Calculate similarity matrix $\mathbf{S} \in \mathbb{R}^{c \times n}$ between $\mathbf{P}_s$ and centers
5. Allocate each point to the most similar center

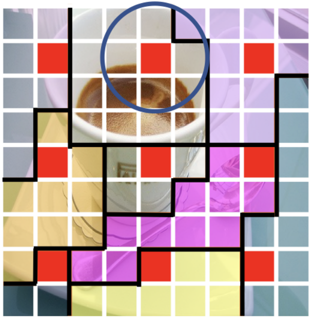

Figure 6: Context clustering

# Feature Aggregating

Assume a cluster contains $m$ points, and $s \in \mathbb{R}^m$ is the similarity between $m$ points and the center,

1. Map $m$ points to a value space to get $P_\nu \in \mathbb{R}^{m \times d'}$, where $d'$ is the value dimension
2. Propose a center $\nu_c$ as in the context clustering
3. The aggregated feature $g \in \mathbb{R}^{d'}$ is given by,

$$g = \frac{1}{\mathcal{C}} \left( \nu_c + \sum_{i=1}^{m} \mathrm{sig}(\alpha s_i + \beta) * \nu_i \right),$$

where

$$\mathcal{C} = 1 + \sum_{i=1}^{m} \mathrm{sig}(\alpha s_i + \beta),$$

with learnable scalars $\alpha, \beta$, $s_i, \nu_i$ being $i$th points in $s$ and $P_\nu$ respectively.

# Aggregated Feature

$$g = \frac{1}{\mathcal{C}} \left( \nu_c + \sum_{i=1}^{m} \mathsf{sig}(\alpha s_i + \beta) * \nu_i \right)$$

- *Why sigmoid?* Because empirically better with sigmoid, because no negative value is involved.
- *Why not Softmax?* Because points do not contradict with one another.
- *Why add $\nu_c$?* For numerical stability, and to further emphasize the locality.
- *Why scale by $\mathcal{C}$?* To control the magnitude.

# Feature Dispatching

As shown in Figure 2, the aggregated feature $g$ is dispatched to each point $p_i$ in a cluster as follows:

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

Through this, the points in a cluster can communicate and share the features.

# Technical Details

1. **Multi-head computing.** Use $h$ heads, and the outputs of multi-head operations are concatenated and fused by a FC layer
2. **Region partition.** To reduce computational overhead, split the points into several local regions and compute similarity locally:

$$\mathcal{O}(ncd) \to \mathcal{O}\left(r\frac{n}{r}\frac{c}{r}d\right),$$

where $n$ is the number of input points ($d$-dimensional), $c$ is the number of centers, and $r$ is the number of local regions

# Experiments

1. Image classification on ImageNet-1K
2. 3D point cloud classification on ScanObjectNN
3. Object detection and instance segmentation on MS-COCO
4. Semantic segmentation on ADE20K

# Image Classification (ImageNet-1K)

| | 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 |

Figure 7: Image classificaion on ImageNet-1K

# 3D Point Cloud Classification (ScanObjectNN)

| 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** $_{\uparrow 0.8}$ |

Figure 8: Classification on ScanObjectNN

# Object Detection vs Semantic Segmentation vs Instance Segmentation

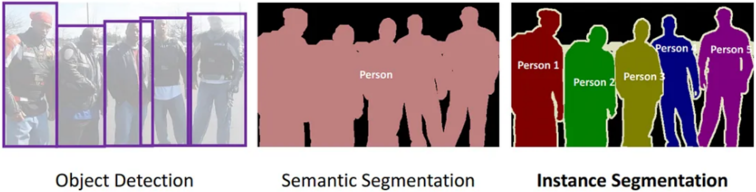


Figure 9: Object detection/semantic segmentation/instance segmentation

# Object Detection and Instance Segmentation on MS-COCO

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Intersection over Union (IoU)} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

$$\text{AP}_n = \text{Average Precision with } n\% \text{ IoU threshold}$$

| 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 |

Figure 10: Object detection and instance segmentation on MS-COCO

# Semantic Segmentation on ADE20K

| 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** |

Figure 11: Semantic segmentation on ADE20K

# Visualization of Context Clustering

1. Cluster the goose as one object context and group background grass together
2. Cluster the similar contexts even in the early stage (neck of the goose in the red box)
3. Most clusters emphasize the locality, while some show the globality a lot

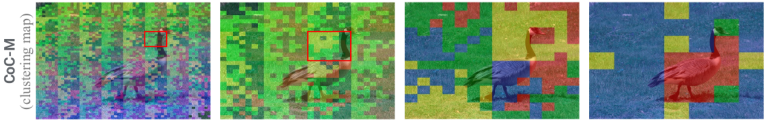


Figure 12: Visualization of clustering map

# Thank You

Q & A