Hyperbolic Hierarchical Clustering for Visual Representation Learning

Jianan Wei[1], Guikun Chen[1], Zhiyuan Weng[1], Chunchao Guo[2], Yujia Wang[3†], and Wenguan Wang[1]

[1]Zhejiang University [2]Tencent Hunyuan [3]Zhejiang Sci-Tech University
https://github.com/weijianan1/HCFormer

Abstract. We investigate the token mixer in vision backbones by revisiting clustering, one of the most classic approaches in machine learning. An effective token mixer is a fundamental component of modern vision backbones like vision Transformers, facilitating information exchange between image patches. Mainstream token mixers, which rely on convolution, attention, MLP, or their hybrids, primarily focus on navigating the trade-off between accuracy and computational cost. However, a significant drawback of these methods is their black-box nature; their encoding process is opaque and lacks interpretability. Diverging from these opaque designs, we introduce *ClusterMixer*, a transparent token mixer that is grounded in a clustering paradigm and interpretable by design. *ClusterMixer* explicitly formulates the token mixing process through a hierarchical clustering mechanism. To model the natural, tree-like relationships inherent in visual data, the clustering is performed in hyperbolic space, which is well-suited for embedding hierarchies with low distortion. Building on this innovation, we present HCFormer, a new backbone architecture that integrates *ClusterMixer* with a series of meticulously designed clustering strategies to ensure robust performance across tasks. Extensive experiments demonstrate that HCFormer consistently outperforms its counterparts across diverse tasks, including image classification, object detection, instance segmentation, and semantic segmentation. Considering its transparency and efficacy, we hope HCFormer can facilitate a paradigm shift toward interpretable backbones.

Keywords: Representation learning · Hyperbolic geometry · Clustering

1 Introduction

Convolutional Networks (ConvNets) and Vision Transformers (ViTs) are the dominant paradigms in computer vision. ConvNets serve as the *de facto* standard since the pioneering success of AlexNet [31], owing to their strong inductive bias (*i.e.*, locality and translation equivariance). Marking a paradigm shift, ViTs [16] introduce self-attention mechanisms with fewer inductive biases, enabling effective scaling

† Corresponding author: Yujia Wang.

and superior generalization over ConvNets. Building on this, MLP-Mixer [59] conceptualizes *token mixing* and proposes a convolution- and attention-free alternative, which replaces the self-attention layers with spatial MLPs. This work spurs research into alternative token mixers in the vision community. For instance, VAN [21] employs convolution as the token mixer, while Uniformer [32] integrates both convolution and self-attention. Despite their incremental performance gains, the reasoning process of these models remains opaque to humans. Recently, there has been growing research interest in clustering-based vision backbones [9, 46], which can provide enhanced interpretability. Albeit promising, it has not yet fully exploited the capabilities of clustering algorithms for visual representation learning, as evidenced by their suboptimal performance, indicating a notable gap that merits further investigation.

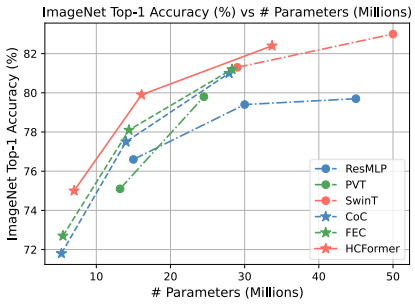

Fig. 1: Top-1 accuracy of HCFormer and other SOTAs on ImageNet-1K [13].

In this work, we introduce an efficient token mixer based on clustering algorithms, dubbed *ClusterMixer*, which operates via: **i)** initializing data points and cluster centers based on token representations, **ii)** assigning data points to centers to form distinct clusters, and **iii)** performing token mixing based on the established clusters. Nevertheless, the direct application of *ClusterMixer* presents a dual challenge in terms of computational efficiency and model performance. **First**, it faces analogous challenges of computational complexity as self-attention and spatial MLPs, stemming from data point and cluster center counts. This efficiency limitation becomes more pronounced for vision tasks that require extensive token sets to accomplish dense prediction or high-resolution image representation. A straightforward approach is to reduce the number of cluster centers; however, it leads to an undesirable tradeoff by degrading model performance. **Second**, the similarity estimation of clustering algorithms is primarily conducted in Euclidean geometry [46, 52]. However, the *flat geometry* of Euclidean space limits its ability in modeling hierarchical relation [34], leading to distortions in the semantic distance and suboptimal performance. In other words, even if two embeddings are close in Euclidean geometry, they may be distant in the semantic hierarchy.

To bridge these gaps, we propose HCFormer, a framework for visual representation learning that enhances *ClusterMixer* via two meticulously designed strategies: **i)** *hierarchical clustering*. Here input patches are partitioned into several non-overlapping windows, with token mixing restricted to local windows to reduce computational complexity in clustering operations. To capture global contextual information, we further apply a mixing operation over these windows, which introduces only minimal computational overhead. **ii)** *hyperbolic hierarchical clustering*. Unlike Euclidean geometry, which excels in flat, simple structures, hyperbolic geometry naturally models hierarchical relationships as a continuous analog of trees [29], thereby better capturing abstract and complex semantics.

To leverage these complementary properties, we extend the *ClusterMixer* to use Euclidean space for fine-grained patch-level clustering and hyperbolic space for abstract window-level clustering.

Despite its attractive properties (*e.g.*, grasping abstract and complex semantic relationships), hyperbolic geometry is numerically unstable, whereas Euclidean geometry demonstrates superior performance and stability in simple scenarios. Besides, the operations defined in hyperbolic geometry incur greater computational overhead compared to their Euclidean counterparts. To address these trade-offs, we extend the *ClusterMixer* to perform clustering across dual geometries: Euclidean similarity is computed at the finer-grained and computationally demanding patch-level, while hyperbolic similarity is estimated at the abstract and computationally efficient window-level.

HCFormer enjoys several desirable virtues: **First**, *efficiency*. Based on the proposed clustering strategies, HCFormer can reduce the quadratic computational complexity associated with increasing data points and cluster centers to a linear one. This reduction in complexity is especially beneficial for downstream tasks, *e.g.*, semantic segmentation and object detection, which facilitates the use of a greater number of cluster centers for improved performance. **Second**, *holistic hierarchy*. HCFormer conducts *ClusterMixer* in both Euclidean and Hyperbolic geometries, enabling the model to handle similarity estimation across diverse scenarios, from simple (*e.g.*, flat relation) to complex (*e.g.*, tree-like structure). **Third**, *flexibility*. Owing to its non-parametric and clustering-based design, *ClusterMixer* enables HCFormer to be effectively applied to various downstream tasks (see §4).

For comprehensive evaluation, HCFormer is benchmarked on three datasets covering diverse application scenarios, including ImageNet-1K [13] for image classification, ADE20K [76] for semantic segmentation, COCO [38] for instance detection and segmentation. In §4.1, by training from scratch, HCFormer outperforms mainstream counterparts with similar parameter counts, *e.g.*, exceeding ResNet-50 by 2.6% and Swin-Tiny by 1.1% in `top-1` accuracy. Notably, HCFormer also improves over prior clustering-based backbones across model scales, with gains of 1.2–3.3% over CoC [46] and 0.3–2.4% over FEC [9]. In §4.2, as the backbone with Semantic FPN [30], HCFormer achieves performance gains of 0.7%~2.8% mIoU for semantic segmentation. In §4.3, HCFormer integrated with Mask R-CNN [23] demonstrates competitive performance for instance detection and segmentation. These results are impressive for a clustering-based backbone like HCFormer, which also enjoys built-in interpretability.

2 Related Work

Clustering in Vision. The objective of clustering is to partition finite, unlabeled data into discrete subsets of inherent groupings or clusters using a similarity measure (*e.g.*, Euclidean distance) [47, 68], operating as an essential tool in pattern recognition. Traditional clustering approaches are adopted in computer vision as an effective image preprocessing method [1, 35, 53], which groups pixels

into perceptually meaningful atomic regions. Modern clustering-based methods are further developed for downstream vision tasks, such as semantic segmentation [15, 33, 36, 37, 70, 71], visual relationship understanding [25, 66] and trajectory prediction [58, 67]. Due to its flexibility and interpretability, clustering methods are increasingly utilized for modeling complex data types, *e.g.*, point clouds [17, 45, 50, 69] and protein [2, 51], and employed to elucidate the decisions derived from the neural network [26, 64, 77]. Motivated by these compelling virtues, our work endeavors to explore the adaptation of clustering methods for basic visual representation learning.

Learning in Hyperbolic Geometry. Hyperbolic geometry has attracted significant attention owing to its effectiveness in modeling data with tree-like structures [6, 48]. These desirable properties render hyperbolic embeddings a superior alternative in diverse modalities, *e.g.*, graphs [4, 7], images [3, 18] and videos [28, 42]. Another important line of research involves leveraging hyperbolic geometry for representation learning. For instance, Hyperbolic Neural Networks [19] introduce a set of corresponding hyperbolic neural network layers and demonstrates superior performance over their Euclidean counterparts in various downstream tasks, such as text entailment and noisy prefix prediction. Subsequent works further expand this framework: Hyperbolic Neural Networks++ [56] proposes hyperbolic convolutional layers, while Hyperbolic attention networks [40] present a Graph Neural Network architecture that operates in hyperbolic space. In contrast to these works, this paper incorporates hyperbolic geometry into clustering algorithms for visual representation learning, aiming to enhance token mixing by better capturing hierarchical relationships between features.

Generic Vision Backbone. Revolutions in computer vision can be characterized as a paradigm shift in feature extraction. In early research [11, 44, 55], feature extraction typically relies on manually crafted features (*e.g.*, geometric structure or color statistics) derived from the predefined rules. Beginning with AlexNet [31], Convolutional Networks (ConvNets) [24, 57] evolve this paradigm from hand-crafted feature engineering to data-driven feature learning, which offers greater versatility and performance. ConvNets utilize sliding windows to partition the entire image into a set of rectangular patches, where convolutional kernels conduct pixel mixing via weighted summations in the local region. In contrast, Vision Transformers (ViTs) [16] provide an attention-based alternative, which reduces the image-specific inductive biases in ConvNets for feature extraction. ViTs split images into a collection of non-overlapping patches and conduct token mixing across them in a global range, facilitating model scalability. Beyond these two paradigms, Context Cluster (CoC) [46] proposes an innovative idea to group the points into clusters, where pixel features are aggregated and then dispatched within a cluster. This simple design is convolution- and attention-free, which only relies on a clustering algorithm to provide interpretability. FEC [9] extends this by introducing clustering-based feature pooling for downsampling, thereby achieving a fully clustering-based feature extraction.

Nevertheless, the potential of clustering methodologies for representation learning remains underexplored within the research community. Our work achieves

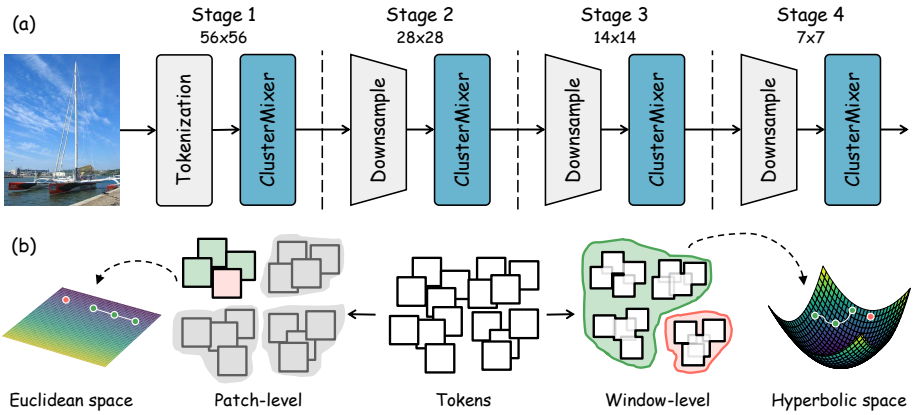

Fig. 2: (a) **Overall pipeline of HCFormer.** Following the MetaFormer paradigm [72], HCFormer adopts a hierarchical architecture with 4 stages. (b) **Hierarchical clustering**. *ClusterMixer* conducts patch-level clustering in Euclidean space and window-level clustering in hyperbolic space.

strong performance on various vision tasks, and we hope it will contribute to a paradigm shift toward clustering-based vision backbones.

3 Method

3.1 Overall Architecture

An overview of the HCFormer is presented in Figure 2, which is built upon the MetaFormer paradigm [72]. The input image $\boldsymbol{I}$ is first split into non-overlapping patches, following the patch splitting module of ViT [16]. Given the fundamental role of spatial proximity in visual clustering [27], these patches are equipped with their coordinates and processed into embedding tokens. Then, a series of residual blocks incorporating token mixers are applied on these tokens. Instead of attention mechanism (e.g., Transformer [63]), spatial MLP (e.g., MLP-Mixer [59]), or average pooling (e.g., PoolFormer [72]), we implement interpretable clustering algorithms as the token mixer, dubbed *ClusterMixer*. To produce a hierarchical representation akin to ConvNets [24, 57], the number of tokens is reduced by concatenating the spatially adjacent tokens, which is achieved by a convolutional operation. This design enables seamless adaptation to downstream tasks such as semantic segmentation and object detection. The following section first delineates the implementation of *ClusterMixer* (§3.2) and then describes the two strategies developed to enhance it (§3.3 & §3.4).

3.2 ClusterMixer

Considering data points $\boldsymbol{X} \in \mathbb{R}^{N \times C}$, the *ClusterMixer* is performed as following:
Center Estimation. The estimation of cluster centers from data points can be implemented through various methods, such as the K-means [53] or the Sinkhorn-Knopp algorithm [43, 64]. However, these methods are computationally intensive

and thus not suitable for token mixing. In this paper, we initialize the data points $\boldsymbol{X} \in \mathbb{R}^{N\times C}$ with feature embeddings and employ average pooling over neighboring patches to estimate cluster centers $\boldsymbol{C} \in \mathbb{R}^{M\times C}$. Inspired by [72], the cluster centers $\boldsymbol{C}\in\mathbb{R}^{M\times C}$ are estimated via a pooling operator, defined as:

$$\boldsymbol{C}_i = \frac{1}{K^2}\sum_{p=1}^{K}\sum_{q=1}^{K}\boldsymbol{F}_{[:,r\cdot K+p,c\cdot K+q]}, \quad s.t.\ \ r=\lfloor i/M_w \rfloor, \quad c = i \bmod M_w, \tag{1}$$

where K is the pooling size, M_w denotes the number of cluster centers in a row, *i.e.*, $M_w{=}W/K$. This approach incurs a computational complexity linear in the sequence length while involving no learnable parameters.

Cluster Assignment. We first partition all data points into distinct clusters by assigning them to different cluster centers $c_j \in \boldsymbol{C}$ based on the feature similarity. Notably, in contrast to hard clustering (*i.e.*, each token is exclusively assigned to a single cluster), we employ a soft clustering approach, where data points can belong to multiple clusters with varying degrees of probability. This process resembles the Expectation-step (E-step) in Expectation-Maximization (EM) clustering [12], with the distinction that our cluster centers are precomputed via average pooling over neighboring patches, avoiding iterative updates. The i-th row $a_i \in \mathbb{R}^{1\times M}$ of the assignment matrix $\boldsymbol{A} \in \mathbb{R}^{N\times M}$ for each token $x_i \in \boldsymbol{X}$ is computed as:

$$a_i = \mathrm{Softmax}_{c_j\in\boldsymbol{C},b_{i,j}\in\mathbf{B}}(\alpha\cdot D(x_i,c_j)+b_{i,j}), \tag{2}$$

where D denotes the similarity metric (see §3.4), α is a learnable scalar for feature similarity, and $\boldsymbol{B} \in \mathbb{R}^{N\times M}$ parameterizes the learnable relative positional bias. Notably, D is the only component evaluated in Euclidean or hyperbolic geometry; the cluster centers c_j and token features x_i used by center estimation and token mixing remain Euclidean feature vectors.

Token Mixing. We perform token mixing in two steps: **i)** feature aggregation from data points and **ii)** feature propagation back to data points. Specifically, it adaptively gathers contextual information from data points based on the assignment matrix to compute the aggregated feature $g_i \in \mathbb{R}^{1\times C}$:

$$g_i = \sum\nolimits_{j=1}^{M}\boldsymbol{A}_{i,j}\cdot c'_j, \quad s.t.\ \ c'_j = (c_j + \sum\nolimits_{i=1}^{N}\boldsymbol{A}_{i,j}\cdot x_i)/(1+\sum\nolimits_{i=1}^{N}\boldsymbol{A}_{i,j}). \tag{3}$$

The aggregated feature g_i is then employed to update the corresponding data point via:

$$x'_i = x_i + \mathrm{FC}(g_i), \tag{4}$$

where FC denotes a fully-connected layer to maintain the feature dimension of token embeddings.

Our *ClusterMixer* offers several advantages: **i)** the soft clustering (*vs.* CoC's hard clustering [46]) enables our method to benefit from constructing more clusters across various tasks (see §4.4). **ii)** the relative positional bias effectively assists the clustering algorithm in establishing relational dependencies.

3.3 Hierarchical Clustering

To tackle the challenge of computational complexity, we perform clustering operations within non-overlapping local windows, where each input is partitioned into S windows, each containing K patches. However, such disconnected windows are recognized to limit the acquisition of global contextual information for representation learning. While this limitation can be mitigated via shifted windows in Swin Transformer [41], our clustering-based token mixer offers an alternative approach by performing mixing at both the *patch-level* and the *window-level*. These two strategies differ solely in *how data points and cluster centers are formulated* in the *ClusterMixer*:

- *Patch-level clustering*. Each patch serves as a data point, while cluster centers are estimated within each localized window, and the *ClusterMixer* is performed in parallel across all windows.
- *Window-level clustering*. Each window is modeled as a data point, represented by the mean feature embedding of its patches, while cluster centers are estimated globally over the entire input.

This formulation transforms the quadratic complexity of *ClusterMixer* into linear complexity. Empirically, this hierarchical clustering mechanism is better suited for the proposed *ClusterMixer* compared to the shifted windowing scheme (§4.4).

3.4 Hyperbolic Hierarchical Clustering

The choice of similarity metric in Equation 2 is critical for cluster assignment. Currently, similarity estimation is primarily conducted in Euclidean geometry [46, 52]: given data points $x_{\mathbb{E}} \in \mathbb{R}^C$ and a cluster center $c_{\mathbb{E}} \in \mathbb{R}^C$, the feature distance is estimated by pair-wise cosine similarity:

$$D_{\mathbb{E}}(x_{\mathbb{E}}, c_{\mathbb{E}}) := \text{sim}(x_{\mathbb{E}}, c_{\mathbb{E}}) = \frac{x_{\mathbb{E}} \cdot c_{\mathbb{E}}}{\|x_{\mathbb{E}}\| \|c_{\mathbb{E}}\|}, \tag{5}$$

where $\|\cdot\|$ denotes the L2 norm. To better model abstract and complex semantic relationships, we next perform similarity estimation in hyperbolic space.

Defining Hyperboloid. Hyperbolic spaces are Riemannian manifolds characterized by a constant negative curvature, whereas Euclidean spaces exhibit zero-curvature (*i.e.*, flat) geometry. There are five well-known isometric models of hyperbolic geometry, including the Lorentz model, the Poincaré ball model, the Poincaré half-space model, the Klein model, and the Hemisphere model [5]. Among these, the Lorentz model is widely adopted due to its numerical stability and computational efficiency.

The Lorentz model is an n-dimensional hyperbolic space on the upper half of a two-sheeted hyperboloid in $n+1$-dimensional Minkowski space. In the Lorentz space, every vector $x \in \mathbb{R}^{n+1}$ can be written as $[x_{\text{time}}, x_{\text{space}}]$, where $x_{\text{time}} \in \mathbb{R}$ denotes the first dimension as the *time dimension* and $x_{\text{space}} \in \mathbb{R}^n$ denotes the remaining n dimensions as the *space dimension*. The model is described as:

$$\mathbb{L}^n = \{x \in \mathbb{R}^{n+1} : \langle x, x \rangle_{\mathbb{L}} = -1/\kappa, \kappa > 0\}, \quad s.t.\ x_{\text{time}} = \sqrt{1/\kappa + \|x_{\text{space}}\|^2}, \tag{6}$$

where $-\kappa \in \mathbb{R}$ is the curvature of the space, typically set to $\kappa = 1$ for simplicity, and $\langle \cdot, \cdot \rangle_{\mathbb{L}}$ denotes the *Lorentzian inner product*, defined as:

$$\langle x, y \rangle_{\mathbb{L}} := -x_{\text{time}} y_{\text{time}} + x_{\text{space}}^{\top} y_{\text{space}}. \tag{7}$$

Lifting onto Hyperboloid. To project a vector from Euclidean space to hyperbolic space, we first define a mapping from the *tangent space* $T_z\mathbb{L}^n$ onto the *hyperbolic manifold* $\mathbb{L}^n$, where $T_z\mathbb{L}^n$ is a Euclidean space of vectors that are orthogonal to some points $z \in \mathbb{L}^n$ on the hyperboloid. Given a tangent vector $v \in T_z\mathbb{L}^n$, the *exponential mapping* $T_z\mathbb{L}^n \to \mathbb{L}^n$ can be defined as:

$$\exp_z^{\kappa}(v) = \cosh(\sqrt{\kappa}\|v\|_{\mathbb{L}})z + \sinh(\sqrt{\kappa}\|v\|_{\mathbb{L}})\frac{v}{\sqrt{\kappa}\|v\|_{\mathbb{L}}}, \quad s.t. \|v\|_{\mathbb{L}} = \sqrt{\langle v, v \rangle_{\mathbb{L}}}, \tag{8}$$

By treating Euclidean vectors as tangent vectors at the origin $\mathbf{0} = (\sqrt{1/\kappa}, 0, \ldots, 0)^{\top}$ of hyperbolic space, one can map vectors from Euclidean space to hyperbolic space via $\exp_0(\cdot)$.

Given a data point $x_{\mathbb{E}} \in \mathbb{R}^C$ and a cluster center $c_{\mathbb{E}} \in \mathbb{R}^C$, we first project both onto the Lorentz hyperboloid $\mathbb{L}^n$, yielding embeddings $x_{\mathbb{L}} \in \mathbb{L}^n$ and $c_{\mathbb{L}} \in \mathbb{L}^n$ in the hyperbolic space:

$$x_{\mathbb{L}} = \exp_0^{\kappa}(x_{\mathbb{E}}), \qquad c_{\mathbb{L}} = \exp_0^{\kappa}(c_{\mathbb{E}}). \tag{9}$$

Estimating Hyperbolic Similarity. Given two hyperbolic embeddings $x_{\mathbb{L}}, c_{\mathbb{L}} \in \mathbb{L}^n$, we estimate hyperbolic similarity via the Lorentzian distance, as:

$$D_{\mathbb{L}}^{\kappa}(x_{\mathbb{L}}, c_{\mathbb{L}}) := \sqrt{1/\kappa} \cdot \cosh^{-1}(-\kappa \langle x_{\mathbb{L}}, c_{\mathbb{L}} \rangle_{\mathbb{L}}). \tag{10}$$

Reformulation of ClusterMixer. Despite its attractive properties (*e.g.*, grasping abstract and complex semantic relationships), hyperbolic geometry is numerically unstable, whereas Euclidean geometry demonstrates superior performance and stability in simple scenarios. Besides, the operations defined in hyperbolic geometry incur greater computational overhead compared to their Euclidean counterparts. To address these trade-offs, we extend the *ClusterMixer* to perform clustering across dual geometries: Euclidean similarity is computed at the finer-grained and computationally demanding patch-level, while hyperbolic similarity is estimated at the abstract and computationally efficient window-level. Drawing upon this principle, Equation 4 is rewritten as:

$$x' = x + \text{FC}(\text{Norm}([g_W, g_P])), \tag{11}$$

where $[\cdot, \cdot]$ denotes the concatenation operation. Here, g_W represents features aggregated from windows via hyperbolic similarity, while g_P does so for patches using Euclidean similarity. Prior to the FC layer, g_W is upsampled to match the dimensions of g_P. The computations for both are performed in parallel and remain decoupled from one another.

3.5 Network Configuration

Following conventional protocols [16, 72], the network input for image classification is set to 224×224, yielding $224/4\times224/4=56\times56$ patches. The network architecture comprises four stages, with the number of patches reduced by a factor of four, while each cluster center remains derived from 4 data points throughout the forward process. In this way, global patch-level mixing would impose a significant computational burden on the clustering algorithm. For instance, at stage 1, there would be $56/2\times56/2$ cluster centers and 56×56 data points, resulting in a 784×3136 assignment matrix. Given $S=49$ partitioned windows at this stage, our hyperbolic hierarchical clustering strategy reduces the patch-level complexity to $49\times16\times64$, at the cost of an additional window-level complexity 16×49. For downstream tasks such as segmentation and detection with variable-sized inputs, reflect padding is applied to maintain this configuration. Additionally, the relative positional bias is linearly interpolated before use to accommodate variable input sizes. See Supplementary for more details of network configuration.

4 Experiment

4.1 Image Classification

Dataset. The evaluation for image classification is carried out on ImageNet-1K [13], which contains 1.3M training and 50K validation images across 1K classes.
Setup. We adopt `Timm` as the codebase and the experiments are run on 4 A100 GPUs with a batch size of 256. Following [46, 72], all of our models are trained for 310 epochs using a momentum of 0.9 and a weight decay of 0.05. The learning rate is set to 1e-3 and adjusted via a cosine schedule with 5 warmup epochs. For data augmentation, we use Mixup [74], CutMix [73], CutOut [75], and RandAugment [10]. `Top-1` classification accuracy is reported.
Results. Table 1 compares HCFormer with other widely-used baselines on image classification. HCFormer achieves superior results over counterparts with similar parameter counts, demonstrating the effectiveness of the proposed method. For example, with 16M parameters, HCFormer outperforms ResMLP-12 by 3.2%, PVT-Tiny by 4.7%, and ConvMixer-1024/12 by 2.0% in `top-1` accuracy. HCFormer also demonstrates superior performance compared to other models adhering to the MetaFormer paradigm, as evidenced by HCFormer-Medium outperforming Swin-Tiny (82.4% *vs.* 81.3%) and MLP-Mixer-B/16 (82.4% *vs.* 76.4%). These results are impressive, considering the transparent, clustering-based interpretable nature of HCFormer. With a marginal increase in parameters and FLOPs, HCFormer yields a notable improvement compared to existing cluster-based approaches; for instance, it achieves 2.4%/1.7%/1.2% gains in `top-1` accuracy across three configurations of model size relative to FEC [9]. To further substantiate the performance of our method, we introduce a smaller variant of HCFormer with only 5.1M parameters, *i.e.*, HCFormer-Nano, which has the fewest parameters among all methods listed in Table 1. Despite its compact size, HCFormer-Nano achieves 73.0% `top-1` accuracy, exceeding CoC-Tiny by 1.2%

Table 1: Classification `top-1` accuracy on ImageNet-1K [13] `val`. Throughput (images/s) measured on a single V100 GPU at batch size 128, averaged over last 500 iterations.

	Method		Param. (M)	GFLOPs (G)	Top-1(%) ↑	Throughput (images/s)
CLIP	MERU-S/16	[14][ICML23]	-	-	34.3	-
	MERU-B/16	[14][ICML23]	-	-	37.5	-
	MERU-L/16	[14][ICML23]	-	-	38.8	-
	HyCoCLIP-S/16	[49][ICLR24]	-	-	41.7	-
	HyCoCLIP-B/16	[49][ICLR24]	-	-	45.8	-
	HCL	[20][CVPR23]	-	-	58.5	-
MLP	ResMLP-12	[60][TPAMI22]	15.0	3.0	76.6	-
	ResMLP-24	[60][TPAMI22]	30.0	6.0	79.4	-
	ResMLP-36	[60][TPAMI22]	45.0	8.9	79.7	-
	MLP-Mixer-B/16	[59][NeurIPS21]	59.0	12.7	76.4	-
	MLP-Mixer-L/16	[59][NeurIPS21]	207.0	44.8	71.8	-
	gMLP-Ti	[39][NeurIPS21]	6.0	1.4	72.3	-
	gMLP-S	[39][NeurIPS21]	20.0	4.5	79.6	-
Attention	ViT-B/16	[16][ICLR20]	86.0	55.5	77.9	-
	ViT-L/16	[16][ICLR20]	307	190.7	76.5	-
	PVT-Tiny	[65][ICCV21]	13.2	1.9	75.1	-
	PVT-Small	[65][ICCV21]	24.5	3.8	79.8	-
	DeiT-Tiny/16	[61][ICML21]	5.7	1.3	72.2	-
	DeiT-Small/16	[61][ICML21]	22.1	4.6	79.8	-
	Swin-Tiny	[41][ICCV21]	29	4.5	81.3	-
	Swin-Small	[41][ICCV21]	50	8.7	83.0	-
Conv.	ResNet-18	[24][CVPR16]	12	1.8	69.8	-
	ResNet-50	[24][CVPR16]	26	4.1	79.8	-
	ConvMixer-512/16	[62][TMLR23]	5.4	-	73.8	-
	ConvMixer-1024/12	[62][TMLR23]	14.6	-	77.8	-
	ConvMixer-768/32	[62][TMLR23]	21.1	-	80.2	-
Cluster	CoC-Tiny	[46][ICLR23]	5.3	1.0	71.8	792.5
	CoC-Small	[46][ICLR23]	14.0	2.6	77.5	581.8
	CoC-Medium	[46][ICLR23]	27.9	5.5	81.0	473.8
	FEC-Small	[9][CVPR24]	5.5	1.4	72.7	742.1
	FEC-Base	[9][CVPR24]	14.4	3.4	78.1	532.1
	FEC-Large	[9][CVPR24]	28.3	6.5	81.2	478.2
	HCFormer-Nano	(ours)	5.1	0.9	73.0±0.29	719.0
	HCFormer-Tiny	(ours)	7.0	1.0	75.1±0.14	536.0
	HCFormer-Small	(ours)	16.1	2.9	79.8±0.06	324.7
	HCFormer-Medium	(ours)	33.7	6.4	82.4±0.03	235.7

with 0.2M fewer parameters and FEC-Small by 0.3% with 0.4M fewer parameters. Notably, our 16M model exhibits performance comparable to ConvMixer-768/32 [62] (21.1M) and DeiT-Small/16 [61] (22.1M). In conclusion, these promising results manifest the effectiveness and wide benefit of our algorithm.

4.2 Semantic Segmentation

Dataset. The evaluation for semantic segmentation is carried out on ADE20K [76], which includes 20K training and 2K validation images across 150 classes.
Setup. We adopt `mmsegmentation` as the codebase and the experiments are run on 4 A100 GPUs with a batch size of 16. HCFormer serves as the backbone equipped with Semantic FPN [30]. For a fair comparison, all models are trained for 80k iterations using AdamW. The learning rate starts at 2e-4, with polynomial decay (power=0.9). The backbones are initialized with ImageNet pre-trained weights and the added layers employ Xavier initialization. During training, we use random scale jittering with a factor in $[0.5, 2.0]$ and a crop size of 512×512 for training. During inference, we use one input image scale with shorter side as 512 pixels. Mean intersection-over-union (mIoU) is reported.
Results. Table 2 illustrates our compelling results over semantic segmentation. In terms of `mIoU`, our HCFormer exceeds CoC [46] and FEC [9] by significant improvements with comparable parameter counts: 40.4% *vs*. 36.6% *vs*. 37.7% at 19M parameters, 43.3% *vs*. 40.2% *vs*. 40.5% at 35M parameters. Notably, HCFormer-Small achieves performance comparable to CoC-Medium/4 and FEC-Large, with 6.3M and 13.0M fewer parameters, respectively. Furthermore, our HCFormer-Nano outperforms FEC-Small by 1.5% with 0.3M fewer parameters, while even surpassing CoC-Small (17.6M parameters). These benchmarking results are significant, which provide solid evidence that our method serves as an effective backbone architecture for semantic segmentation. We attribute this advancement to the adopted soft clustering mechanism and hierarchical clustering strategies, which enable HCFormer to capture multi-scale features essential for semantic segmentation.

Table 2: Segmentation `mIoU` score of different backbones with Semantic FPN [30] on ADE20K [76] `val`.

Backbone		Param. (M)	mIoU(%) ↑
ResNet-18	[24][CVPR16]	15.5	32.9
ResNet-50	[24][CVPR16]	28.5	36.7
PVT-Tiny	[65][ICCV21]	17.0	35.7
PVT-Small	[65][ICCV21]	28.2	39.8
CoC-Small/4	[46][ICLR23]	17.6	36.6
CoC-Medium/4	[46][ICLR23]	25.2	40.2
FEC-Small	[9][CVPR24]	9.1	35.3
FEC-Base	[9][CVPR24]	18.0	37.7
FEC-Large	[9][CVPR24]	31.9	40.5
HCFormer-Nano	(ours)	8.8	36.8
HCFormer-Tiny	(ours)	10.3	37.3
HCFormer-Small	(ours)	18.9	40.4
HCFormer-Medium	(ours)	35.7	43.3

4.3 Object Detection and Instance Segmentation

Dataset. The evaluation for object detection and instance segmentation is carried out on MS COCO 2017 [38], which has 118K training and 5K validation images.
Setup. We adopt `mmdetection` as the codebase and the experiments are run on 4 A100 GPUs with a batch size of 16. HCFormer is adopted as the backbone of Mask R-CNN [23] for both object detection and instance segmentation tasks. The backbones are initialized with ImageNet pre-trained weights and the added

Table 3: Object detection and instance segmentation results using Mask R-CNN [23] on COCO [38] `val2017`. AP^{box} and AP^{mask} denote bounding box AP and mask AP.

Method		Param. (M)	AP^{box}	AP^{box}_{50}	AP^{box}_{75}	AP^{mask}	AP^{mask}_{50}	AP^{mask}_{75}
ResNet-18	[24][CVPR16]	31.2	34.0	54.0	36.7	31.2	51.0	32.7
ResNet-50	[24][CVPR16]	44.2	38.0	58.6	41.4	34.4	55.1	36.7
PVT-Tiny	[65][ICCV21]	32.9	36.7	59.2	39.3	35.1	56.7	37.3
PVT-Small	[65][ICCV21]	44.1	40.4	62.9	43.8	37.8	60.1	40.3
CoC-Small/4	[46][ICLR23]	33.6	35.9	58.3	38.3	33.8	55.3	35.8
CoC-Small/25	[46][ICLR23]	33.6	37.5	60.1	40.0	35.4	57.1	37.9
CoC-Small/49	[46][ICLR23]	33.6	37.2	59.8	39.7	34.9	56.7	37.0
CoC-Medium/4	[46][ICLR23]	42.1	38.6	61.1	41.5	36.1	58.2	38.0
CoC-Medium/25	[46][ICLR23]	42.1	40.1	62.8	43.6	37.4	59.9	40.0
CoC-Medium/49	[46][ICLR23]	42.1	40.6	63.3	43.9	37.6	60.1	39.9
FEC-Small	[9][CVPR24]	24.3	35.6	57.5	38.2	33.6	54.7	35.7
FEC-Base	[9][CVPR24]	33.1	37.9	60.1	40.8	35.5	57.5	37.8
FEC-Large	[9][CVPR24]	47.1	39.9	62.5	43.2	37.3	59.5	39.5
HCFormer-Nano	(ours)	24.0	36.0	57.8	38.4	34.0	55.1	36.0
HCFormer-Tiny	(ours)	25.5	36.4	58.5	38.6	34.5	55.8	36.5
HCFormer-Small	(ours)	34.1	38.7	60.9	41.8	36.0	58.0	38.3
HCFormer-Medium	(ours)	50.8	40.9	63.0	44.0	37.6	59.8	40.0

layers employ Xavier initialization. All models are trained for 12 epochs (1× schedule) using AdamW optimizer with an initial learning rate of 1e-4. During training, images are resized with the shorter side at 800 pixels and the longer side $\leq$ 1,333 pixels. During inference, the shorter side is also scaled to 800 pixels. Mean Average Precision (mAP) is adopted for evaluation.

Results. Table 3 reports the numerical results for both object detection and instance segmentation. Empirically, our method achieves superior performance over other competitors under identical network sizes across all evaluation metrics. For instance, at the 34M model scale, HCFormer outperforms recent clustering-based advancements (*i.e.*, CoC [46] and FEC [9]), achieving 38.7% *vs.* 37.2% *vs.* 37.9% AP^{box} for object detection and 36.0% *vs.* 35.4% *vs.* 35.5% AP^{mask} for instance segmentation, respectively. With fewer parameters, HCFormer-Nano outperforms FEC-Small by 0.4% AP^{box} and 0.4% AP^{mask}. Moreover, HCFormer-Medium achieves 40.9% AP^{box} in object detection and 37.6% AP^{mask} in instance segmentation, surpassing all other methods except PVT-Small, to which it is only marginally inferior (by 0.1% AP^{mask}) in instance segmentation. However, its advantage narrows when compared to CoC-Medium/49. We posit that this is because object detection performance benefits from the number of cluster centers, which enables CoC-Medium/49 to achieve results competitive with ours. Crucially, this accuracy comes at the expense of efficiency; HCFormer-Medium achieves a 37% higher throughput (14.0 vs. 10.2) than CoC-Medium/49 (as shown in Supplementary Table S2), underscoring its superior computational economy.

Table 4: A set of ablative experiments on ImageNet [54] `val`. *Hier. Clus.* indicates hierarchical clustering; *Shift.* implies shifted windows; *Hyp. Geo.* represents hyperbolic geometry; *Rel. Pos.* denotes relative position; *Euc.* and *Hyp.* are Euclidean and hyperbolic geometry, respectively.

Strategy	top-1(%)↑	top-5(%)↑
Shift.	72.7	91.1
Hier. Clus.	73.4	91.4

(a) Hierarchical clustering for *ClusterMixer*.

Window	Patch	top-1(%)↑	top-5(%)↑
Euc.	*Euc.*	73.4	91.4
Hyp.	*Hyp.*	74.7	92.3
Hyp.	*Euc.*	75.1	92.5

(b) Geometry for clustering.

curvature κ	top-1(%)↑	top-5(%)↑
0.1	74.9	92.4
1.0	75.1	92.5
10.0	74.8	92.4

(c) Curvature κ in hyperbolic space.

cluster number	top-1(%)↑	top-5(%)↑
9	74.9	92.4
16	75.1	92.5
25	74.6	92.3

(d) Cluster number for window-level clustering.

Hier. Clus.	*Hyp. Geo.*	*Rel. Pos.*	top-1(%)↑	top-5(%)↑	mIoU(%) ↑	AP^{box}	AP^{mask}
			71.9	90.8	35.1	34.3	32.7
	✓	✓	72.7	91.1	34.6	34.4	32.8
✓		✓	73.4	91.4	35.2	34.7	33.1
✓	✓		74.4	92.2	36.4	35.9	34.2
✓	✓	✓	75.1	92.5	37.3	36.4	34.5

(e) Key Component Analysis.

4.4 Diagnostic Experiment

For thorough evaluation, we perform a series of ablative studies on ImageNet-1K [13] `val` for image classification to investigate the following aspects. HCFormer-Tiny is utilized as the baseline.

Shifted Windows or Hierarchical Clustering. We first investigate the effectiveness of the proposed hierarchical clustering strategies for acquiring global contextual information in representation learning, compared with the shifted window mechanism [41]. As outlined in Table 4a, while the shifted window mechanism remains effective, our hierarchical clustering strategies are better suited for *ClusterMixer* (*i.e.*, 72.7% → 73.4%). This may arise because the shifted operation only enables localized interaction between adjacent windows, thereby remaining constrained by the limited receptive field.

Hyperbolic or Euclidean Distance. We next examine the impact of hyperbolic versus Euclidean geometry on window-level clustering. As summarized in Table 4b, we find that the use of hyperbolic geometry yields a notable performance gain over Euclidean geometry by 1.6% `top-1` accuracy. When clustering is performed entirely in hyperbolic space, a marginal performance degradation (0.3% `top-1`) is observed, accompanied by a substantial reduction in computational efficiency (throughput decreases from 536.0 to 319.2). We posit this may be because local visual neighborhoods are usually nearly flat, making Euclidean space more suitable for preserving local geometry.

Curvature in Hyperbolic Space. As shown in Table 4c, it can be seen that the variation in curvature exhibits minimal impact. We hypothesize that this

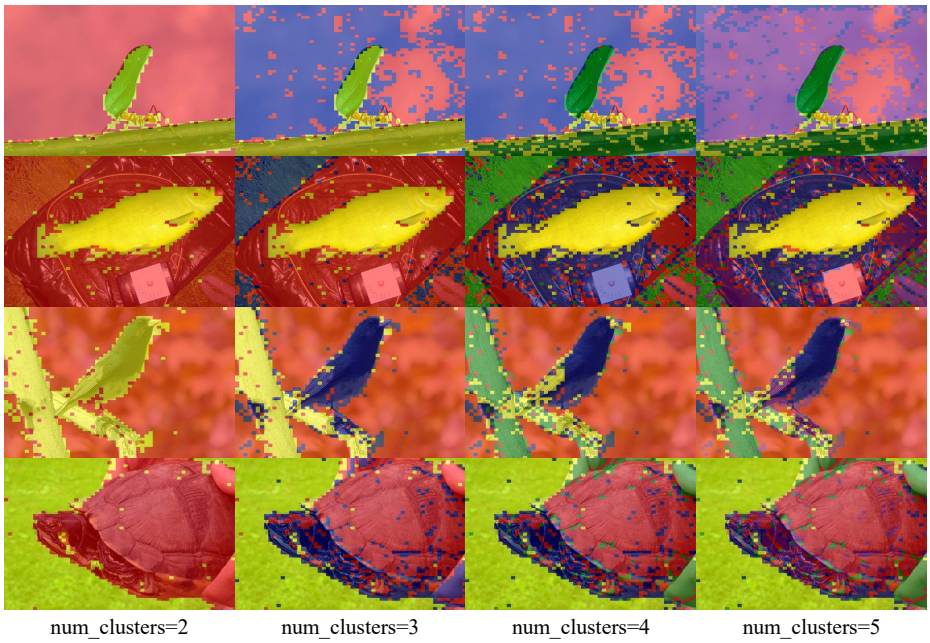

Fig. 3: Visualization of clustering maps for our HCFormer-Tiny on ImageNet [54] `val`. Different colored masks indicate different clusters, ranging from 2 to 5.

may be attributed to the fact that the current head dimension (24 for Tiny, 32 for Small and Medium) is sufficient to capture the relational structure in the embedding space across varying curvature regimes. We attribute the numerical safeguards of our method from two aspects: i) In our assignment logits, the learnable scale α Eq. 2 helps absorb curvature-scale changes. This aligns with prior work demonstrating that representations across different curvatures can be related via scaling transformations [8]. ii) We clip all features to a bounded norm before hyperbolic mapping, which limits the input magnitude of the cosh / sinh terms, following [22].

Cluster Number for Window-level Clustering. Table 4d presents that a higher cluster count in the hyperbolic space yields performance gains, peaking at 16 clusters. This may be because 25 cluster centers are excessive for the number of windows, given the 49 windows defined in our experimental setup.

Key Component Analysis. We finally ablate the key design elements in the proposed HCFormer. As shown in Table 4e, the baseline of our model achieves only 71.9% without the proposed components. Empirical results demonstrate that all components provide complementary benefits, as the absence of either leads to performance degradation: -2.3% without hierarchical clustering, -1.6% without hyperbolic geometry, and -0.6% without relative position.

4.5 Visualization of Clustering

Our configuration ensures that the feature representation at each center of the network architecture (except the final stage) is derived from a 2×2 token grid, while tokens are progressively merged to reduce their count by a factor of 2×2 by downsampling operator (See Supplementary Table S3). This mechanism indicates that HCFormer develops gradually expanding clusters during feature extraction, ultimately forming 16 clusters at the final stage. Following FEC's visualization protocol [9], we also use K-means to reduce the number of clusters for better visualization. As illustrated in Figure 3, the clustering results demonstrate that our HCFormer is able to capture intrinsic relational patterns among tokens.

5 Conclusion

Clustering represents a promising yet underexplored direction in architectural design, lacking an effective and efficient framework that enables it to emerge as a competitive alternative to mainstream architectures. This paper proposes a new token-mixer design, termed *ClusterMixer*, which leverages clustering algorithms to enhance token aggregation. To leverage this design effectively, we advocate a universal vision framework, designated as HCFormer, along with a set of training strategies tailored for *ClusterMixer*. Empirical results demonstrate that this framework improves interpretability over conventional convolution- and attention-based methods while achieving superior performance to existing cluster-based approaches. Given its favorable balance of interpretability and performance, we expect that this approach will potentially benefit a wider range of visual tasks.

Acknowledgements. This work was supported by Zhejiang Provincial Natural Science Foundation of China (No. LD25F020001), National Natural Science Foundation of China (No. 62302046, T25B2005), Beijing Natural Science Foundation (L252036), and the State Key Laboratory of Brain Cognition and Brain-inspired Intelligence Technology (No. SKLBI-K2025004).

Bibliography

[1] Achanta, R., Shaji, A., Smith, K., Lucchi, A., Fua, P., Süsstrunk, S.: Slic superpixels compared to state-of-the-art superpixel methods. IEEE TPAMI (2012)

[2] Alley, E.C., Khimulya, G., Biswas, S., AlQuraishi, M., Church, G.M.: Unified rational protein engineering with sequence-based deep representation learning. Nature methods (2019)

[3] Atigh, M.G., Schoep, J., Acar, E., Van Noord, N., Mettes, P.: Hyperbolic image segmentation. In: CVPR (2022)

[4] Bai, Y., Ying, Z., Ren, H., Leskovec, J.: Modeling heterogeneous hierarchies with relation-specific hyperbolic cones. NeurIPS (2021)

[5] Cannon, J.W., Floyd, W.J., Kenyon, R., Parry, W.R., et al.: Hyperbolic geometry. Flavors of geometry (1997)

[6] Chamberlain, B.P., Clough, J., Deisenroth, M.P.: Neural embeddings of graphs in hyperbolic space. arXiv preprint arXiv:1705.10359 (2017)

[7] Chami, I., Wolf, A., Juan, D.C., Sala, F., Ravi, S., Ré, C.: Low-dimensional hyperbolic knowledge graph embeddings. arXiv preprint arXiv:2005.00545 (2020)

[8] Chami, I., Ying, Z., Ré, C., Leskovec, J.: Hyperbolic graph convolutional neural networks. NeurIPS (2019)

[9] Chen, G., Li, X., Yang, Y., Wang, W.: Neural clustering based visual representation learning. In: CVPR (2024)

[10] Cubuk, E.D., Zoph, B., Shlens, J., Le, Q.V.: Randaugment: Practical automated data augmentation with a reduced search space. In: CVPRW (2020)

[11] Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. In: CVPR (2005)

[12] Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the em algorithm. Journal of the royal statistical society: series B (methodological) **39**(1), 1–22 (1977)

[13] Deng, J., Dong, W., Socher, R., Li, L.J., Li, K., Fei-Fei, L.: Imagenet: A large-scale hierarchical image database. In: CVPR (2009)

[14] Desai, K., Nickel, M., Rajpurohit, T., Johnson, J., Vedantam, S.R.: Hyperbolic image-text representations. In: ICML (2023)

[15] Ding, Y., Li, L., Wang, W., Yang, Y.: Clustering propagation for universal medical image segmentation. In: CVPR (2024)

[16] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., et al.: An image is worth 16x16 words: Transformers for image recognition at scale. ICLR (2020)

[17] Feng, T., Quan, R., Wang, X., Wang, W., Yang, Y.: Interpretable3d: An ad-hoc interpretable classifier for 3d point clouds. In: AAAI (2024)

[18] Franco, L., Mandica, P., Kallidromitis, K., Guillory, D., Li, Y.T., Darrell, T., Galasso, F.: Hyperbolic active learning for semantic segmentation under domain shift. arXiv preprint arXiv:2306.11180 (2023)

[19] Ganea, O., Bécigneul, G., Hofmann, T.: Hyperbolic neural networks. NeurIPS (2018)
[20] Ge, S., Mishra, S., Kornblith, S., Li, C.L., Jacobs, D.: Hyperbolic contrastive learning for visual representations beyond objects. In: CVPR (2023)
[21] Guo, M.H., Lu, C.Z., Liu, Z.N., Cheng, M.M., Hu, S.M.: Visual attention network. Computational visual media (2023)
[22] Guo, Y., Wang, X., Chen, Y., Yu, S.X.: Clipped hyperbolic classifiers are super-hyperbolic classifiers. In: CVPR (2022)
[23] He, K., Gkioxari, G., Dollár, P., Girshick, R.: Mask r-cnn. In: ICCV (2017)
[24] He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: CVPR (2016)
[25] Hong, J., Wei, J., Wang, W.: Learning human-object interaction as groups. NeurIPS (2025)
[26] Huang, Z., Li, Y.: Interpretable and accurate fine-grained recognition via region grouping. In: CVPR (2020)
[27] Jolion, J.M., Meer, P., Bataouche, S.: Robust clustering with applications in computer vision. IEEE TPAMI (1991)
[28] Jun, L., Jinpeng, W., Chaolei, T., Niu, L., Long, C., Min, Z., Yaowei, W., Shu-Tao, X., Bin, C.: Hlformer: Enhancing partially relevant video retrieval with hyperbolic learning. ICCV (2025)
[29] Khrulkov, V., Mirvakhabova, L., Ustinova, E., Oseledets, I., Lempitsky, V.: Hyperbolic image embeddings. In: CVPR (2020)
[30] Kirillov, A., Girshick, R., He, K., Dollár, P.: Panoptic feature pyramid networks. In: CVPR (2019)
[31] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. NeurIPS (2012)
[32] Li, K., Wang, Y., Gao, P., Song, G., Liu, Y., Li, H., Qiao, Y.: Uniformer: Unified transformer for efficient spatiotemporal representation learning. ICLR (2022)
[33] Li, L., Wang, W., Zhou, T., Li, J., Yang, Y.: Unified mask embedding and correspondence learning for self-supervised video segmentation. In: CVPR (2023)
[34] Li, L., Zhou, T., Wang, W., Li, J., Yang, Y.: Deep hierarchical semantic segmentation. In: CVPR (2022)
[35] Li, Z., Chen, J.: Superpixel segmentation using linear spectral clustering. In: CVPR (2015)
[36] Liang, C., Wang, W., Miao, J., Yang, Y.: Gmmseg: Gaussian mixture based generative semantic segmentation models. NeurIPS (2022)
[37] Liang, J.C., Zhou, T., Liu, D., Wang, W.: Clustseg: Clustering for universal segmentation. In: ICML (2023)
[38] Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: ECCV (2014)
[39] Liu, H., Dai, Z., So, D., Le, Q.V.: Pay attention to mlps. NeurIPS (2021)
[40] Liu, Q., Nickel, M., Kiela, D.: Hyperbolic graph neural networks. NeurIPS (2019)

[41] Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., Guo, B.: Swin transformer: Hierarchical vision transformer using shifted windows. In: ICCV (2021)
[42] Long, T., Mettes, P., Shen, H.T., Snoek, C.G.: Searching for actions on the hyperbole. In: CVPR (2020)
[43] Long, T., van Noord, N.: Cross-modal scalable hyperbolic hierarchical clustering. In: ICCV (2023)
[44] Lowe, D.G.: Distinctive image features from scale-invariant keypoints. IJCV (2004)
[45] Ma, X., Qin, C., You, H., Ran, H., Fu, Y.: Rethinking network design and local geometry in point cloud: A simple residual mlp framework. ICLR (2022)
[46] Ma, X., Zhou, Y., Wang, H., Qin, C., Sun, B., Liu, C., Fu, Y.: Image as set of points. ICLR (2023)
[47] Madhulatha, T.S.: An overview on clustering methods. arXiv preprint arXiv:1205.1117 (2012)
[48] Nickel, M., Kiela, D.: Poincaré embeddings for learning hierarchical representations. NeurIPS **30** (2017)
[49] Pal, A., van Spengler, M., di Melendugno, G.M.D., Flaborea, A., Galasso, F., Mettes, P.: Compositional entailment learning for hyperbolic vision-language models. In: ICLR (2024)
[50] Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: Deep hierarchical feature learning on point sets in a metric space. NeurIPS (2017)
[51] Quan, R., Wang, W., Ma, F., Fan, H., Yang, Y.: Clustering for protein representation learning. In: CVPR (2024)
[52] Radford, A., Kim, J.W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al.: Learning transferable visual models from natural language supervision. In: ICML (2021)
[53] Ren, Malik: Learning a classification model for segmentation. In: ICCV (2003)
[54] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. IJCV (2015)
[55] Schmid, C., Mohr, R.: Local grayvalue invariants for image retrieval. IEEE TPAMI (2002)
[56] Shimizu, R., Mukuta, Y., Harada, T.: Hyperbolic neural networks++. arXiv preprint arXiv:2006.08210 (2020)
[57] Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. ICLR (2015)
[58] Sun, J., Li, Y., Fang, H.S., Lu, C.: Three steps to multimodal trajectory prediction: Modality clustering, classification and synthesis. In: ICCV (2021)
[59] Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., et al.: Mlp-mixer: An all-mlp architecture for vision. NeurIPS (2021)
[60] Touvron, H., Bojanowski, P., Caron, M., Cord, M., El-Nouby, A., Grave, E., Izacard, G., Joulin, A., Synnaeve, G., Verbeek, J., et al.: Resmlp: Feedforward

networks for image classification with data-efficient training. IEEE TPAMI (2022)
[61] Touvron, H., Cord, M., Douze, M., Massa, F., Sablayrolles, A., Jégou, H.: Training data-efficient image transformers & distillation through attention. In: ICML (2021)
[62] Trockman, A., Kolter, J.Z.: Patches are all you need? arXiv preprint arXiv:2201.09792 (2022)
[63] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. NeurIPS (2017)
[64] Wang, W., Han, C., Zhou, T., Liu, D.: Visual recognition with deep nearest centroids. ICLR (2022)
[65] Wang, W., Xie, E., Li, X., Fan, D.P., Song, K., Liang, D., Lu, T., Luo, P., Shao, L.: Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In: ICCV (2021)
[66] Wei, J., Zhou, T., Yang, Y., Wang, W.: Nonverbal interaction detection. In: ECCV (2024)
[67] Xu, C., Mao, W., Zhang, W., Chen, S.: Remember intentions: Retrospective-memory-based trajectory prediction. In: CVPR (2022)
[68] Xu, R., Wunsch, D.: Survey of clustering algorithms. IEEE TNNLS (2005)
[69] Yin, J., Zhou, D., Zhang, L., Fang, J., Xu, C.Z., Shen, J., Wang, W.: Proposalcontrast: Unsupervised pre-training for lidar-based 3d object detection. In: ECCV (2022)
[70] Yu, Q., Wang, H., Kim, D., Qiao, S., Collins, M., Zhu, Y., Adam, H., Yuille, A., Chen, L.C.: Cmt-deeplab: Clustering mask transformers for panoptic segmentation. In: CVPR (2022)
[71] Yu, Q., Wang, H., Qiao, S., Collins, M., Zhu, Y., Adam, H., Yuille, A., Chen, L.C.: k-means mask transformer. In: ECCV (2022)
[72] Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., Yan, S.: Metaformer is actually what you need for vision. In: CVPR (2022)
[73] Yun, S., Han, D., Oh, S.J., Chun, S., Choe, J., Yoo, Y.: Cutmix: Regularization strategy to train strong classifiers with localizable features. In: ICCV (2019)
[74] Zhang, H., Cisse, M., Dauphin, Y.N., Lopez-Paz, D.: mixup: Beyond empirical risk minimization. ICLR (2018)
[75] Zhong, Z., Zheng, L., Kang, G., Li, S., Yang, Y.: Random erasing data augmentation. In: AAAI (2020)
[76] Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., Torralba, A.: Scene parsing through ade20k dataset. In: CVPR (2017)
[77] Zhou, T., Wang, W., Konukoglu, E., Van Gool, L.: Rethinking semantic segmentation: A prototype view. In: CVPR (2022)