CAESAR: CLUSTERING VIA AUTONOMOUS EMBEDDING-SPACE AGGLOMERATIVE REORGANIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Clustering algorithms that operate on nearest-neighbor graphs, such as FINCH (First Integer Neighbor Clustering Hierarchy), depend heavily on the quality of the embedding space they are given. However, pretrained vision and language model embeddings are not optimized for this purpose. We propose CAESAR, a method that reorganizes a pretrained embedding space: a reorganization network is trained to pull mutual nearest neighbors together and push non-neighbors apart, yielding a reorganized embedding space substantially better suited to clustering. CAESAR offers a second major advantage: it never requires the number of clusters K. This matters because in realistic unsupervised settings, K is typically unknown and discovering it is often part of the problem, yet most strong clustering methods take it as an input. We therefore design the entire CAESAR pipeline to infer K rather than assume it is known. Empirically, reorganizing the embeddings consistently improves clustering over the raw space on both text and image datasets. Since the few deep clustering methods that also infer K do not release their code, we complement controlled comparisons with methods that infer K on the same embedding space by comparisons with strong deep clustering methods that are given the true K, giving them a substantial oracle advantage. Even so, CAESAR outperforms all of them on text, achieves the best results on the most challenging image benchmark and remains competitive on the others. Reorganizing pretrained embeddings thus emerges as a simple and powerful route to clustering realistic data, where classes overlap and the number of clusters is unknown.

1 INTRODUCTION

Clustering is central to exploratory data analysis, yet most modern pipelines rely on embeddings from pretrained encoders that were never trained with clustering as an explicit objective. The so-obtained *pretrained embedding space* is typically built for optimizing objectives such as next-token prediction or contrastive alignment across augmentations. These objectives do not guarantee that the pretrained embedding space exposes a neighborhood structure that clustering algorithms can exploit effectively. This is particularly acute for large language and vision-language models, whose embeddings are known to be anisotropic (Mu et al., 2017; Gao et al., 2019; Ethayarajh, 2019) and optimized for generation or retrieval rather than for grouping semantically similar points together.

A second difficulty in any realistic clustering setting is that the number of clusters is unknown. In genuinely unsupervised applications, such as exploring an unlabeled corpus or discovering categories in new data, one does not know in advance how many groups exist. Determining this number is one of the main goals of the analysis. Most strong clustering methods sidestep this by requiring the number of clusters as an input, which quietly assumes away the hardest part of the problem. Graph-based algorithms such as FINCH (Sarfraz et al., 2019), by contrast, build their partitions directly from first-neighbor relations and do not need the number of clusters in advance. This makes them a natural choice in this setting. The price of this flexibility is a strong dependence on the neighborhood structure of the pretrained embedding space, a property that, as discussed above, pretrained encoders do not guarantee. FINCH has no global objective to compensate for noisy or inconsistent neighborhoods, so a few spurious first-neighbor links can propagate through the entire hierarchy.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Centroid-based methods such as k-means are more robust in this respect, but only because they are given the number of clusters in advance. The two problems thus compound: the methods best suited to an unknown number of clusters are also those most vulnerable to embeddings not designed for clustering.

We propose CAESAR (Clustering via Autonomous Embedding-Space Agglomerative Reorganization), a method that addresses both problems jointly. CAESAR takes the pretrained embedding space and reorganizes it to make it more amenable to graph-based clustering. It does so without retraining or modifying the encoder and without ever requiring the number of clusters. To address the first problem, we train a lightweight *reorganization network* that maps the pretrained embedding space to a new space, the *reorganized embedding space*. The reorganization network is trained to pull mutual nearest neighbors in the pretrained embedding space together and push non-neighbors apart. A memory bank allows each point to be compared against the full dataset rather than only the current batch. The reorganized embedding space is then clustered with FINCH. For the second problem, every stage of the pipeline is designed to operate without knowing the number of clusters: the neighborhood size is estimated from the geometry of the pretrained embedding space, the balance between the pull and push terms is set to a single value shared across all datasets and the final partition is chosen from the FINCH hierarchy by an unsupervised criterion, never by matching a target number of clusters.

We evaluate CAESAR on text and image datasets with varying levels of class separability. CAESAR consistently improves over FINCH applied directly to the pretrained embedding space. Since existing deep clustering methods that infer K are not reproducible from publicly available code, we complement our controlled comparisons with published results from recent deep clustering methods that use different encoders and are given the true number of clusters. Despite this and without ever being given the number of clusters, CAESAR outperforms every competing method on text and remains competitive on images, sometimes surpassing methods that know the true number of clusters. CAESAR is thus particularly well suited to the realistic setting of overlapping classes and an unknown number of clusters.

Our contributions are as follows:

- We propose a pull–push objective over mutual k-nearest neighbors, computed against a memory bank of the full dataset, to reorganize the pretrained embedding space of frozen LLM/VLM encoders for graph-based clustering.
- The entire pipeline operates without knowing the number of clusters: the neighborhood size is estimated from the data, the pull–push balance is shared across all datasets and the final partition is selected from the FINCH hierarchy by an unsupervised criterion.
- Across text and image datasets and without ever being given the number of clusters, CAESAR outperforms all competing methods on text and remains competitive on images, where it obtains the best NMI on the most entangled dataset—ahead even of baselines given the true number of clusters.

2 Related Work

Deep clustering. A large body of work jointly learns representations and cluster assignments. DEC (Xie et al., 2016) and IDEC (Guo et al., 2017) refine an autoencoder latent space toward a cluster-friendly structure by minimizing a KL divergence to a sharpened assignment distribution. DeepCluster (Caron et al., 2018) alternates between clustering features and using the assignments as pseudo-labels. SCAN (Van Gansbeke et al., 2020) first mines nearest neighbors with self-supervised pretraining, then trains a classification head that encourages consistent predictions among neighbors. CC (Li et al., 2021) performs contrastive learning simultaneously at both the instance and the cluster level. SPICE (Niu et al., 2022) splits the clustering network into a feature model and a clustering head and trains them in successive stages with semantics-aware pseudo-labels. These methods learn their representations end-to-end from raw inputs, typically rely on image augmentations and require the number of clusters K as an architectural parameter (e.g., the width of the cluster head). In contrast, we operate on the pretrained embedding space of frozen foundation models, use no augmentations and never require K at any stage.

Clustering with external textual guidance. A recent line of work improves image clustering by exploiting textual knowledge from vision-language models such as CLIP (Radford et al., 2021). SIC (Cai et al., 2023) maps images into a semantic space built from WordNet nouns, derives pseudo-labels from image–text similarities and enforces consistency in both the image and the semantic space. TAC (Li et al., 2023) selects the WordNet nouns that best discriminate the images, then mutually distills neighborhood information between the text and image modalities. Both methods keep the pretrained encoder frozen but rely on an external vocabulary and require K. In contrast, CAESAR uses no external textual supervision and never requires K.

Nearest-neighbor supervision for representation learning. Closest to our objective are methods that use nearest neighbors as positives in a contrastive loss. NNCLR (Dwibedi et al., 2021) retrieves the nearest neighbor of each view from a support queue and treats it as a positive, showing that cross-sample positives provide richer semantic variation than augmentations alone. NNM (Dang et al., 2021) matches samples to their nearest neighbors at both the batch (local) and dataset (global) level for deep clustering. Our loss follows this general pull–push principle, with three differences. First, we rely on *mutual* k-nearest neighbors – pairs that select each other – which prunes asymmetric, boundary-crossing pairs from the positive set. Second, our memory bank stores the embeddings of the *entire* dataset in the reorganized embedding space, so the repulsion term is computed against all points rather than a sampled queue. Third, whereas NNM and NNCLR learn a full encoder from images with augmentations, we train only a reorganization network on the pretrained embedding space of frozen LLM/VLM encoders, which makes the approach applicable to any modality for which a pretrained encoder exists.

Attraction–repulsion embedding methods. Our objective is also closely related to the attraction–repulsion principle at the heart of neighbor-graph embedding methods for dimensionality reduction and visualization. t-SNE (Van der Maaten & Hinton, 2008) and UMAP (McInnes et al., 2018) build a k-nearest-neighbor graph in the input space and optimize a low-dimensional layout in which neighboring points attract while non-neighbors repel. More recently, DiRE (Kolpakov & Rivin, 2025) makes this attraction–repulsion trade-off explicit. CAESAR differs in intent and in setting: rather than projecting to two or three dimensions for visualization, we learn a reorganization into a moderate-dimensional space whose sole purpose is to improve downstream graph-based clustering. Moreover, our repulsion is computed against a full memory bank rather than through the sampling or negative-approximation schemes these methods use for scalability.

Clustering without a predefined number of clusters. FINCH (Sarfraz et al., 2019), which serves as the clustering engine of our pipeline, builds a hierarchy of partitions from first-neighbor relations alone, without distance thresholds or a target number of clusters. Density-based methods such as DBSCAN and HDBSCAN (Ester et al., 1996; Campello et al., 2013) also avoid specifying K but replace it with density hyperparameters that are notoriously hard to tune in high dimensions.

EVoC (Bot et al., 2025), designed specifically for embedding vectors, combines a UMAP-like node embedding with HDBSCAN-style density-based clustering and selects stable partitions through persistence analysis, also without a target number of clusters; unlike CAESAR, it optimizes a non-parametric low-dimensional layout rather than training a reorganization network. Our contribution is complementary to this line of work: rather than proposing a new K-free clusterer, we show that a learned reorganization of the embedding space substantially improves the partitions that such graph-based methods discover, while preserving their K-free property end to end – the final partition is selected from the FINCH hierarchy with an internal validity criterion (silhouette; Rousseeuw, 1987) rather than by matching a known K.

Geometry of pretrained embeddings. Pretrained language-model embeddings are known to be anisotropic (Mu et al., 2017; Gao et al., 2019; Ethayarajh, 2019), which motivates global post-processing such as BERT-flow (Li et al., 2020), whitening or contrastive fine-tuning (Gao et al., 2021). These corrections are task-agnostic. Graph-based clustering instead requires a clean *local neighborhood structure*, which CAESAR targets directly by preserving mutual-neighbor relations.

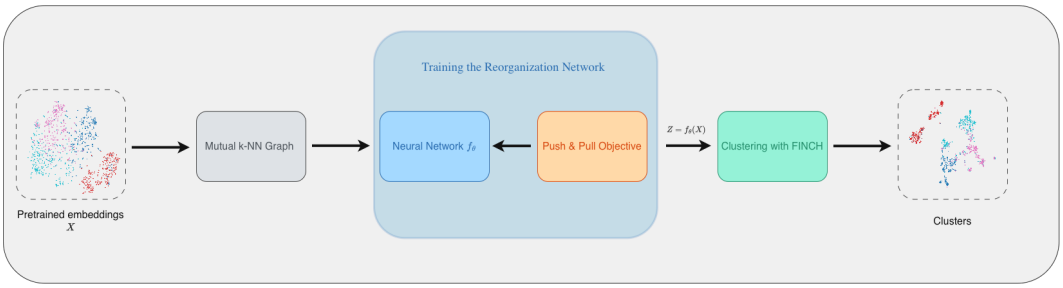


Figure 1: Overview of CAESAR. Starting from a pretrained embedding space X, we build a mutual k-NN graph (Section 3.2) and train a reorganization network f_θ (Section 3.3) with a pull–push objective (Section 3.4). The resulting embeddings $Z = f_\theta(X)$ are partitioned by FINCH (Section 3.5). No stage of the pipeline uses the number of clusters.

3 METHOD

3.1 OVERVIEW

Our approach starts from a pretrained embedding space $X \in \mathbb{R}^{N\times d}$ produced by a foundation model and learns a reorganization network $f_\theta : \mathbb{R}^d \to \mathbb{R}^{d'}$ (Section 3.3) that maps the data into a reorganized embedding space $Z = f_\theta(X)$, with $d' < d$. Figure 1 gives an overview of the full pipeline. Rather than learning representations from scratch, we aim to reshape the geometry of the pretrained embedding space while preserving its semantic structure. To this end, we build a neighborhood graph on X (Section 3.2) and train f_θ to preserve and sharpen these local relations in the reorganized embedding space: neighbors are pulled together, non-neighbors are pushed apart (Section 3.4). This is done with a two-term loss (pull term and push term). The neighborhood size is estimated from the geometry of X. The embeddings Z are then passed to FINCH and the final partition is chosen from its hierarchy by an unsupervised criterion (Section 3.5), never by matching a target number of clusters. This makes the framework directly applicable to realistic unsupervised settings, where the number of clusters is unknown.

Note that every choice in the pipeline is made without knowing the number of clusters.

3.2 THE MUTUAL k-NN GRAPH

Before training f_θ, we define the structure it must preserve: a neighborhood graph built, once for all, from the pretrained embedding space X. For each point i, we take its k nearest neighbors $\mathcal{N}_k(i)$ under cosine similarity.

However, directed neighbors are unreliable for our objective. A point j can be among the nearest neighbors of i without i being among those of j, especially in dense regions. We therefore keep only *mutual* neighbors, $\mathcal{M}_k(i) = \{j \in \mathcal{N}_k(i) : i \in \mathcal{N}_k(j)\}$. If $\mathcal{M}_k(i)$ is empty, we fall back to the single closest neighbor so that every point keeps at least one anchor. If fewer than k mutual neighbors are found, $\mathcal{M}_k(i)$ is padded to k entries by repeating its last element for batching.

Choice of k. Rather than fixing k by hand, we estimate it from the geometry of the pretrained embedding space. On a random subsample of X, we compute the mean distance $\bar{d}(r)$ to the r-th nearest neighbor for $r = 1, \ldots, k_{\max}$. This curve describes how fast the local neighborhood expands with rank. Its elbow marks the rank beyond which additional neighbors become markedly less informative. We locate this elbow as the rank of maximum discrete curvature,

$$k = \arg\max_r \ \Delta^2 \bar{d}(r), \qquad \Delta^2 \bar{d}(r) = \bar{d}(r{+}2) - 2\bar{d}(r{+}1) + \bar{d}(r). \tag{1}$$

The neighborhood size is thus set by the data geometry alone and is never tuned against clustering performance.

The graph $\mathcal{M}_k$ is computed once and kept fixed throughout training: it encodes the local structure of X that f_θ must preserve and serves as the target of the objective defined in Section 3.4.

3.3 THE REORGANIZATION NETWORK f_θ

The reorganization network f_θ is a lightweight neural network. Unlike a full encoder trained from raw data, it operates directly on the pretrained embedding space $X \in \mathbb{R}^{N\times d}$: its input dimension d is set by the chosen pretrained model rather than by the raw modality. It outputs embeddings $Z \in \mathbb{R}^{N\times d'}$ in the reorganized embedding space, with $d' = 256$ across all datasets and modalities. Operating on an already informative space rather than learning an encoder from scratch keeps the number of trainable parameters small and restricts f_θ to reorganizing the geometry of X instead of rediscovering the semantic content the foundation model has already captured.

The network is a three-layer MLP, $d \to 1024 \to 512 \to d' = 256$, with the same hidden sizes for all pretrained models, regardless of their input dimension d. Each hidden layer is followed by a SELU activation (Klambauer et al., 2017) and the output layer is linear.

3.4 TRAINING f_θ WITH THE PULL–PUSH OBJECTIVE

We train f_θ with an objective designed to preserve the neighborhoods defined by $\mathcal{M}_k$, which brings mutual neighbors together in the reorganized embedding space and keeps non-neighbors apart. Let $z_i = f_\theta(x_i)$ be the embedding of point i in the reorganized embedding space and $\hat{z}_i = z_i/\|z_i\|_2$ its ℓ_2-normalized version, so that $\hat{z}_i^\top \hat{z}_j$ is the cosine similarity between i and j.

Since a mini-batch $\mathcal{B}$ covers only a small fraction of the dataset, most neighbors of a point i are not in $\mathcal{B}$ and restricting the objective to pairs within the batch would ignore most of $\mathcal{M}_k$. We therefore maintain a memory bank $\mathrm{Bank} \in \mathbb{R}^{N\times d'}$ (Wu et al., 2018), whose j-th row $\hat{b}_j$ stores the most recent ℓ_2-normalized embedding $\hat{z}_j$ of point j in the reorganized embedding space; after each optimization step, the rows of the points in $\mathcal{B}$ are updated. Each point in the batch can thus be compared to all its neighbors and to all other points, with a single forward pass of f_θ per batch.

The objective has two terms. Let $n_{i,1}, \ldots, n_{i,k}$ denote the elements of $\mathcal{M}_k(i)$ ordered by increasing distance to i in the pretrained embedding space. The pull term encourages each point i to be similar to its neighbors:

$$\mathcal{L}_{\text{pull}} = \frac{1}{|\mathcal{B}|} \sum_{i\in\mathcal{B}} \sum_{r=1}^{k} w_r \left(1 - \hat{z}_i^\top \hat{b}_{n_{i,r}}\right), \tag{2}$$

where the weights $w_r \propto 1/r$, normalized to sum to one, give more influence to the nearest neighbors, which are more reliable positives than farther ones.

The push term discourages i from becoming similar to points that are *not* its neighbors, over the entire bank rather than only the current batch:

$$\mathcal{L}_{\text{push}} = \frac{\lambda}{|\mathcal{B}|\, N} \sum_{i\in\mathcal{B}} \sum_{j=1}^{N} (1 - m_{i,j}) \max\left(\hat{z}_i^\top \hat{b}_j,\ 0\right), \tag{3}$$

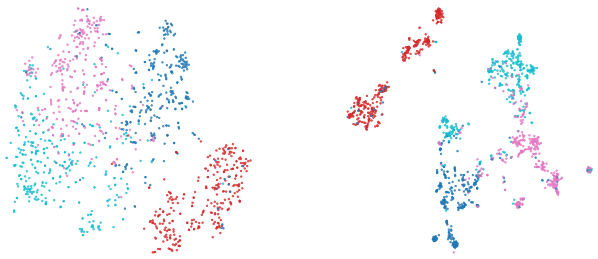
where $m_{i,j} = 1$ if $j = i$ or $j \in \mathcal{M}_k(i)$ and 0 otherwise and λ balances push against pull (Section 4.1). The full objective is $\mathcal{L} = \mathcal{L}_{\text{pull}} + \mathcal{L}_{\text{push}}$.

The clamp at zero penalizes a non-neighbor pair only while its similarity is positive; once their similarity is zero or negative, it no longer contributes to the gradient. Without the clamp, the objective would push every non-neighbor pair toward maximal dissimilarity. This is geometrically impossible: the average pairwise cosine similarity of N unit vectors is at least $-1/(N-1)$. It would also let the push term, which sums over far more pairs than the pull term, dominate the gradient.

Figure 2 illustrates the effect of this objective on AG News: in the pretrained embedding space the four classes overlap, while in the reorganized embedding space they form compact, well-separated groups.

3.5 CLUSTERING WITH FINCH

The embeddings $Z = f_\theta(X)$ in the reorganized embedding space are ℓ_2-normalized and clustered with FINCH (Sarfraz et al., 2019), which links each point to its first nearest neighbor and merges the resulting connected components recursively. It therefore requires no number of clusters in advance,



(a) Pretrained embedding space

(b) Reorganized embedding space

Figure 2: Effect of the pull–push objective on AG News, visualized with t-SNE (Van der Maaten & Hinton, 2008). Colors denote the four ground-truth classes and are used only for visualization, never during training. In the pretrained embedding space (left) the classes overlap; in the reorganized embedding space (right) they form compact, well-separated groups.

consistent with the rest of our pipeline. Moreover, since FINCH relies on the same local neighbor relations that our objective reinforces, it directly reflects whether f_θ has achieved its intended effect.

FINCH returns a hierarchy of partitions ranging from fine to coarse and a single level must be selected. To avoid using the number of clusters, we select this level with an internal validity criterion: we compute the silhouette score (Rousseeuw, 1987) of each level in the reorganized embedding space and keep the level that maximizes it. As a baseline given the true number of clusters, we also report FINCH on the pretrained embedding space with oracle level selection: FINCH builds its hierarchy without K and we then use the true K to select the level whose number of clusters is closest to it.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate CAESAR on four datasets spanning two modalities. For text, we use AG News (Zhang et al., 2015) (4 topic classes). For images, we use CIFAR-10 (Krizhevsky et al., 2009) (10 classes), CIFAR-20 (the 20 superclasses of CIFAR-100 (Krizhevsky et al., 2009)) and STL-10 (Coates et al., 2011) (10 classes). Text is encoded with Qwen3-Embedding-8B and images with Qwen3-VL-Embedding-8B; both produce 4096-dimensional embeddings, which we ℓ_2-normalize. All embeddings are extracted once and kept frozen throughout. Since none of these datasets provides an official validation split, we hold out a stratified 10% of the official training set for validation, leaving the test split untouched. Split sizes are given in Appendix A.

Baselines. Ideally, CAESAR would be compared with other deep clustering methods that infer the number of clusters K. However, such methods are rare and their code is not publicly available. Reimplementing them is not a viable option either: each one is tied to its own encoder, architecture and training pipeline, so it cannot be faithfully reproduced on our pretrained embedding space. We therefore do not include them. Instead, we compare CAESAR with two groups of methods:

First, on both text and images, we consider methods operating on the same pretrained embedding space as CAESAR. Two baselines, like CAESAR, infer the number of clusters without access to K: FINCH with the hierarchy level selected by silhouette and EVoC (Bot et al., 2025), a density-based method designed for embedding vectors. In contrast, a third baseline is a FINCH variant that uses the ground-truth K to select, among the levels of the hierarchy, the one whose number of clusters is closest to K, which gives it an oracle advantage over CAESAR. Since the hierarchy only offers a few discrete levels, the selected number of clusters may differ from K (Appendix B). Since all methods operate on exactly the same embeddings, these comparisons are controlled.

Second, on images only, we compare CAESAR with recent deep image clustering methods that are given the true number of clusters: CC (Li et al., 2021), SPICE (Niu et al., 2022), SCAN (Van Gans-

Table 1: Clustering performance on AG News (text, $\times 100$). All methods use the same pretrained embedding space. Without using the number of clusters, CAESAR outperforms all methods, including the oracle FINCH and recovers a number of clusters close to the true K (Appendix B). raw: FINCH on the pretrained embedding space. sil.: level selected by silhouette. K: level whose number of clusters is closest to the true K. †: oracle method, knows the true number of clusters K. **Bold**: best.

Method	NMI	ACC
FINCH (raw, sil.)	33.6	4.5
EVoC	36.2	21.8
FINCH (raw, K)†	49.6	58.5
CAESAR (ours)	**61.5**	**78.3**

beke et al., 2020), SIC (Cai et al., 2023) and TAC (Li et al., 2023). We report their published numbers, as compiled in the benchmark of Li et al. (2023). These methods use a ResNet trained from scratch or frozen CLIP features and SIC and TAC also use external textual knowledge. Since the encoders differ, this comparison is not controlled. Moreover, knowing the number of clusters gives these methods a clear oracle advantage over CAESAR which must infer it. We nevertheless include them as a demanding reference point, to assess how CAESAR fares against methods that know the true number of clusters.

Methods that use the true number of clusters, either as oracle information or as an input, are marked with † in the result tables.

Choice of the push weight. We fix $\lambda = 0.5$ for all datasets and encoders, without dataset-specific tuning or access to the number of clusters K. Appendix C reports a sensitivity analysis over $\lambda \in \{0.1, 0.2, 0.5, 0.8\}$ on the validation splits.

Metrics. We report normalized mutual information (NMI), Adjusted Rand index (ARI; Hubert & Arabie, 1985) and clustering accuracy (ACC), with clusters matched to classes using the Hungarian algorithm (Kuhn, 1955). All metrics are computed on the test split and averaged over three seeds for methods evaluated in our experiments. The main tables report NMI and ACC; ARI is reported in Appendix B. Architecture, training and hyperparameter details are provided in Appendix A.

4.2 Main Results

We use NMI as our primary metric, as it naturally compares partitions with different numbers of clusters, whereas ACC relies on matching predicted clusters to ground-truth classes. In all tables, methods are ordered by increasing NMI and † marks methods that use the true number of clusters K, either as input or for model selection.

Text. Table 1 compares CAESAR with standard clustering methods on AG News, all applied to the same pretrained embedding space. CAESAR outperforms every other method on both metrics, including FINCH with its level selected using the true number of clusters: it improves NMI by 11.9 points (61.5 vs. 49.6) and ACC by 19.8 points (78.3 vs. 58.5), without ever using K. It also recovers a number of clusters very close to the true one (5 vs. 4; Appendix B). The other methods that do not use the number of clusters are far behind: EVoC reaches only 36.2 NMI and silhouette-selected FINCH 33.6.

Images. Table 2 compares CAESAR with FINCH and EVoC applied to the pretrained embedding space of Qwen3-VL-Embedding, which is the direct starting point that CAESAR reorganizes and with recent deep image clustering methods. All these deep clustering methods are oracles: they are given the true number of clusters. CAESAR uses neither. Despite this, CAESAR obtains the best NMI and ACC on CIFAR-20, the most challenging of these benchmarks, ahead of TAC and SIC. On the more separable CIFAR-10 and STL-10, the methods given K keep an edge, especially in ACC, but CAESAR remains competitive in NMI and outperforms CC, SPICE and SCAN on both datasets. Among the methods applied to the same pretrained embedding space, CAESAR improves

Table 2: Comparison with deep image clustering methods ($\times 100$). All baselines except EVoC and FINCH (raw, sil.) are oracle methods (†): they know the true number of clusters K. Although CAESAR uses neither, it obtains the best NMI and ACC on CIFAR-20 and remains competitive in NMI on CIFAR-10 and STL-10, where it outperforms CC, SPICE and SCAN. Literature methods use ResNet or CLIP encoders and their scores are taken from the published papers (Li et al., 2023); CAESAR, FINCH and EVoC use Qwen3-VL-Embedding features. raw: FINCH on the pretrained embedding space. sil.: level selected by silhouette. K: level whose number of clusters is closest to the true K (Appendix B). **Bold**: best.

	CIFAR-20		CIFAR-10		STL-10	
Method	NMI	ACC	NMI	ACC	NMI	ACC
FINCH (raw, sil.)	29.6	14.1	39.1	19.9	44.7	20.0
CC†	43.1	42.9	70.5	79.0	76.4	85.0
SPICE†	44.8	46.8	73.4	83.8	81.7	90.8
SCAN†	48.6	50.7	79.7	88.3	69.8	80.9
FINCH (raw, K)†	49.2	38.4	63.3	60.5	82.1	84.1
EVoC	56.0	36.0	71.5	76.0	82.6	82.4
SIC†	59.3	58.3	**84.7**	**92.6**	95.3	98.1
TAC†	61.1	60.7	83.3	91.9	**95.5**	**98.2**
CAESAR (ours)	**61.5**	**60.9**	81.5	57.5	83.0	66.8

markedly over silhouette-selected FINCH on every dataset (e.g., from 29.6 to 61.5 NMI on CIFAR-20) and obtains a higher NMI than EVoC on all three datasets, although EVoC reaches a higher ACC on CIFAR-10 and STL-10. In NMI, CAESAR even outperforms FINCH with its level selected using the true number of clusters on all three datasets. This confirms that the reorganization, not the encoder alone, drives the results. Finally, on CIFAR-20, the silhouette criterion selects 21 clusters for 20 ground-truth classes, almost recovering the correct granularity without ever being given it.

4.3 Component Ablation

To isolate the contribution of each component of CAESAR, we build it up cumulatively, starting from FINCH applied to the pretrained embedding space and adding one component at a time: the reorganization network trained with a pull-only objective, then the push term computed within each mini-batch and finally the push term computed against the full memory bank, which corresponds to our complete method. As in all experiments, $\lambda = 0.5$. Table 3 reports the results, averaged over three seeds and over all four datasets.

Training f_θ with a pull-only objective leads to a complete collapse: with no repulsive term, all points are drawn toward one another and the geometry of the space is destroyed. The average NMI drops from 36.8 (raw FINCH) to 2.4. Explicit repulsion is therefore not an optional refinement but a necessary condition for learning any usable structure. Adding the push term computed within each mini-batch prevents the collapse and already improves over raw FINCH (45.6 NMI). However, contrasting each point only against the few examples in its mini-batch provides too few repulsive comparisons to realize the full benefit of the reorganization. Replacing the in-batch push with a push against the full memory bank yields the decisive gain, raising the average NMI to 71.9, with ARI and ACC following the same trend. Each component is therefore necessary. Contrasting every point against the entire dataset, rather than only the current mini-batch, is the main driver of CAESAR's performance.

4.4 Robustness to the Encoder

Since CAESAR reorganizes a pretrained embedding space, its benefit should not depend on the encoder that produced it. We therefore rerun the full pipeline on a much smaller encoder, CLIP ViT-B/32 (512-dimensional, versus 4096 for Qwen3-VL-Embedding); results are reported in Table 10 (Appendix D).

Table 3: Cumulative ablation of CAESAR's components ($\times 100$, mean over three seeds, averaged over CIFAR-20, CIFAR-10, STL-10 and AG News). Starting from FINCH on the pretrained embedding space, each row adds one component; the last row is our complete method.

Configuration	NMI	ARI	ACC
FINCH (raw, sil.)	36.8	26.7	14.6
+ f_θ (pull only)	2.4	0.7	15.6
+ push (in-batch)	45.6	30.3	41.6
+ memory bank (**ours**)	**71.9**	**52.9**	**65.9**

On CLIP, CAESAR lifts the NMI of silhouette-selected FINCH from 24.3, 32.5 and 44.8 to 56.5, 80.0 and 90.1 on CIFAR-20, CIFAR-10 and STL-10, gains comparable to those obtained with Qwen3-VL-Embedding. On STL-10, CAESAR even performs better with CLIP than with Qwen3-VL-Embedding (90.1 vs. 83.0 NMI) and on CIFAR-10 the two are close (80.0 vs. 81.5). The gain thus comes from the reorganization itself rather than from a particular encoder. We keep the Qwen3 models as our main encoders because they cover both text and images, allowing a uniform evaluation across modalities.

5 Discussion and Limitations

Why CAESAR is most competitive on entangled data. When classes are more separable (CIFAR-10, STL-10), the pretrained embedding space already exposes a clean neighborhood structure and methods given K retain an edge. When classes overlap (CIFAR-20, AG News), the first-neighbor graph of the pretrained embedding space is noisy and reshaping its geometry pays off: CAESAR matches or outperforms all baselines in NMI, including those given K. This is the realistic setting CAESAR targets.

Accuracy, granularity and the role of K. On CIFAR-10 and STL-10, CAESAR is stronger in NMI than in ACC. Although the silhouette criterion selects a number of clusters close to the true one on every dataset (Appendix B), ACC penalizes any mismatch through its one-to-one assignment, while NMI is less sensitive to it. On STL-10, for instance, selecting 8 clusters for 10 classes leaves at least two classes unmatched, which caps ACC at 80%. Conversely, on CIFAR-20, where 21 clusters are selected for 20 classes, CAESAR obtains the best ACC of all methods. A better level-selection rule than the silhouette could therefore improve ACC without changing the reorganized embedding space.

Scope. CAESAR assumes a pretrained encoder that already produces semantically meaningful embeddings: it reshapes the pretrained embedding space but cannot create structure that the encoder failed to capture. When no strong pretrained encoder is available, we expect the benefit of the reorganization to be smaller.

6 Conclusion

We introduced CAESAR, a lightweight method that reorganizes the pretrained embedding space of a frozen LLM/VLM encoder for graph-based clustering. A reorganization network is trained with a pull–push objective over mutual nearest neighbors, computed against a memory bank of the entire dataset and the reorganized embedding space is clustered with FINCH. Unlike most competing methods, CAESAR never uses the number of clusters.

Despite this disadvantage, CAESAR outperforms every baseline on text, including those given the true number of clusters. On images, it obtains the best NMI and ACC on CIFAR-20, ahead of methods that know the number of clusters or rely on external textual knowledge and remains competitive on CIFAR-10 and STL-10, where it still outperforms several such methods in NMI. These gains hold across encoders. Our results suggest that reorganizing pretrained embeddings is a simple and effective route to clustering realistic data when the number of clusters is unknown.

AI USE STATEMENT

In this work, we used generative AI tools for none of the tasks with required disclosure. We have not used generative AI tools for developing theoretical models or conceptual frameworks, proposing or refining hypotheses, designing or providing feedback on research methodology or experiments, implementing methods or interpreting results and generating synthetic data sets, formulating mathematical claims, providing ingredients for or writing proofs, translation, cleaning and reformatting datasets and qualitative or thematic data analysis are not applicable to this work. Additionally, we used generative AI tools for identifying relevant literature and drafting parts of the related work section. We have reviewed all AI-assisted work. All references identified with generative AI were checked by the authors against the original papers to verify that they exist and that the claims attributed to them are accurate and all AI-assisted text in the related work section was revised by the authors. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work does not involve human subjects or the collection of new data. All experiments use publicly available benchmark datasets (AG News, CIFAR-10, CIFAR-100, STL-10 and MNIST) and publicly released pretrained encoders. Since CAESAR reorganizes embeddings produced by these encoders, it may inherit their biases and clusters discovered on sensitive data should not be interpreted without human review. We are not aware of any other ethical concern raised by this work.

REPRODUCIBILITY STATEMENT

The method is fully described in Section 3. All hyperparameters and training details are given in Appendix A and the choice of the push weight λ in Appendix C. All datasets and pretrained encoders are publicly available and the data splits are described in Section 4.1. Full results, including ARI and the number of selected clusters, are reported in Appendix B. We will release our code upon publication.

REFERENCES

Daniël Bot, Leland McInnes, and Jan Aerts. Persistent multiscale density-based clustering. *arXiv preprint arXiv:2512.16558*, 2025.

Shaotian Cai, Liping Qiu, Xiaojun Chen, Qin Zhang, and Longteng Chen. Semantic-enhanced image clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 6869–6878, 2023.

Ricardo JGB Campello, Davoud Moulavi, and Jörg Sander. Density-based clustering based on hierarchical density estimates. In *Pacific-Asia conference on knowledge discovery and data mining*, pp. 160–172. Springer, 2013.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *European conference on computer vision*, pp. 139–156. Springer, 2018.

Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pp. 215–223. JMLR Workshop and Conference Proceedings, 2011.

Zhiyuan Dang, Cheng Deng, Xu Yang, Kun Wei, and Heng Huang. Nearest neighbor matching for deep clustering. In *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 13688–13697. IEEE, 2021.

Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. With a little help from my friends: Nearest-neighbor contrastive learning of visual representations. In *2021 IEEE/CVF international conference on computer vision (ICCV)*, pp. 9568–9577. IEEE, 2021.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pp. 226–231, 1996.

Kawin Ethayarajh. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 55–65, 2019.

Jun Gao, Di He, Xu Tan, Tao Qin, Liwei Wang, and Tie-Yan Liu. Representation degeneration problem in training natural language generation models. *arXiv preprint arXiv:1907.12009*, 2019.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 conference on empirical methods in natural language processing*, pp. 6894–6910, 2021.

Xifeng Guo, Long Gao, Xinwang Liu, and Jianping Yin. Improved deep embedded clustering with local structure preservation. In *Ijcai*, volume 17, pp. 1753–1759, 2017.

Lawrence Hubert and Phipps Arabie. Comparing partitions. *Journal of classification*, 2(1):193–218, 1985.

Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

Alexander Kolpakov and Igor Rivin. Dimensionality reduction for homological stability and global structure preservation. *arXiv preprint arXiv:2503.03156*, 2025.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009.

Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

Bohan Li, Hao Zhou, Junxian He, Mingxuan Wang, Yiming Yang, and Lei Li. On the sentence embeddings from pre-trained language models. In *Proceedings of the 2020 conference on empirical methods in natural language processing (EMNLP)*, pp. 9119–9130, 2020.

Yunfan Li, Peng Hu, Zitao Liu, Dezhong Peng, Joey Tianyi Zhou, and Xi Peng. Contrastive clustering. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pp. 8547–8555, 2021.

Yunfan Li, Peng Hu, Dezhong Peng, Jiancheng Lv, Jianping Fan, and Xi Peng. Image clustering with external guidance. *arXiv preprint arXiv:2310.11989*, 2023.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018.

Jiaqi Mu, Suma Bhat, and Pramod Viswanath. All-but-the-top: Simple and effective postprocessing for word representations. *arXiv preprint arXiv:1702.01417*, 2017.

Chuang Niu, Hongming Shan, and Ge Wang. Spice: Semantic pseudo-labeling for image clustering. *IEEE Transactions on Image Processing*, 31:7264–7278, 2022.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PmLR, 2021.

Peter J Rousseeuw. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. *Journal of computational and applied mathematics*, 20:53–65, 1987.

Saquib Sarfraz, Vivek Sharma, and Rainer Stiefelhagen. Efficient parameter-free clustering using first neighbor relations. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 8926–8935. IEEE, 2019.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008.

Wouter Van Gansbeke, Simon Vandenhende, Stamatios Georgoulis, Marc Proesmans, and Luc Van Gool. Scan: Learning to classify images without labels. In *European conference on computer vision*, pp. 268–285. Springer, 2020.

Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3733–3742, 2018.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487. PMLR, 2016.

Xiang Zhang, Junbo Zhao, and Yann LeCun. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28, 2015.

A Implementation Details

Embeddings. All embeddings are extracted once from frozen pretrained encoders (Qwen3-Embedding-8B for text, Qwen3-VL-Embedding-8B for images and CLIP ViT-B/32 for the experiments of Section 4.4), ℓ_2-normalized and stored on disk. They are never updated during training.

Datasets. Table 4 summarizes the datasets and the sizes of the training, validation and test splits.

Table 4: Datasets used in our main experiments. All embeddings are 4096-dimensional and kept fixed.

Dataset	Modality	Encoder	K	Train	Val	Test
AG News	Text	Qwen3-Embedding-8B	4	108,000	12,000	7,600
CIFAR-10	Image	Qwen3-VL-Embedding-8B	10	45,000	5,000	10,000
CIFAR-20	Image	Qwen3-VL-Embedding-8B	20	45,000	5,000	10,000
STL-10	Image	Qwen3-VL-Embedding-8B	10	4,500	500	8,000

Training. The reorganization network f_θ is trained with AdamW (Loshchilov & Hutter, 2017) (learning rate 3×10^{-4}, weight decay 10^{-4}) and a batch size of 256, for at most 600 epochs with early stopping (patience 20) on the validation objective $\mathcal{L}$. The memory bank is initialized with the outputs of f_θ before training and stores the most recent ℓ_2-normalized embedding of each training point in the reorganized embedding space; after each optimization step, the rows of the points in the current batch are updated.

Clustering and evaluation. The neighborhood size k is estimated as described in Section 3.2. The silhouette score used to select the FINCH level is computed on the full test split. Results of the methods we run are averaged over three seeds. All experiments were run on a single NVIDIA A100 GPU.

Table 5 summarizes all hyperparameters.

Table 5: Hyperparameters of CAESAR. All values are shared across datasets and encoders, except k, which is estimated from each dataset.

Hyperparameter	Value
Architecture of f_θ	$d \to 1024 \to 512 \to 256$
Activations	SELU after each hidden layer, linear output
Output dimension d'	256
Optimizer	AdamW, learning rate 3×10^{-4}, weight decay 10^{-4}
Batch size	256
Epochs	at most 600, early stopping (patience 20)
Neighborhood size k	estimated by maximum curvature (Section 3.2)
Push weight λ	0.5 for all datasets (Appendix C)
Seeds	3

B Full Results

Table 6 extends Table 1 with the adjusted Rand index (ARI), Table 7 reports the ARI of CAESAR on the image datasets and Table 8 compares the number of clusters selected by CAESAR with the true number of classes.

Table 6: Clustering performance on AG News (text, $\times 100$), ordered by increasing NMI. †: given the true number of clusters. **Bold**: best in each column.

Method	NMI	ACC	ARI
FINCH (raw, sil.)	33.6	4.5	7.6
EVoC	36.2	21.8	30.6
FINCH (raw, K)†	49.6	58.5	42.2
CAESAR (ours)	**61.5**	**78.3**	**58.3**

Table 7: ARI of CAESAR ($\times 100$) on the three image datasets: CIFAR-20 (the 20 superclasses of CIFAR-100), CIFAR-10 and STL-10, all encoded with Qwen3-VL-Embedding-8B.

Method	CIFAR-20	CIFAR-10	STL-10
CAESAR (ours)	38.7	54.4	60.3

Table 8: Number of clusters selected by the silhouette criterion on the FINCH hierarchy of CAESAR, compared with the true number of classes K. The number of clusters is never given to the method.

	AG News	CIFAR-20	CIFAR-10	STL-10
True K	4	20	10	10
Selected clusters	5	21	9	8

C Effect of the Push Weight

Table 9 shows the effect of the push weight λ on clustering quality, measured on validation splits. In addition to AG News, CIFAR-10 and STL-10, the sweep includes MNIST (LeCun et al., 1998), which we do not report in the main evaluation because it is not covered by the benchmark of deep clustering results we compare against (Li et al., 2023). A small weight under-penalizes non-neighbors and leaves clusters insufficiently separated, while a large weight spreads points too much. We therefore use the intermediate value $\lambda = 0.5$ for all datasets.

Table 9: Effect of the push weight λ (NMI $\times 100$ on validation splits, averaged over AG News, CIFAR-10, STL-10 and MNIST).

λ	0.1	0.2	0.5	0.8
NMI	41.4	44.5	**68.8**	62.0

D Robustness to the Encoder

Table 10: Robustness to the encoder (NMI $\times 100$). With both a large (4096-d Qwen3-VL-Embedding) and a much smaller (512-d CLIP) encoder, CAESAR yields a large improvement over silhouette-selected FINCH on the pretrained embedding space, confirming that the gain comes from the reorganization rather than from the encoder.

Encoder	Method	CIFAR-20	CIFAR-10	STL-10
Qwen3-VL-Embedding	FINCH (raw, sil.)	29.6	39.1	44.7
	CAESAR (ours)	**61.5**	**81.5**	**83.0**
CLIP ViT-B/32	FINCH (raw, sil.)	24.3	32.5	44.8
	CAESAR (ours)	**56.5**	**80.0**	**90.1**