Unsupervised Feature Learning with Emergent Data-Driven Prototypicality

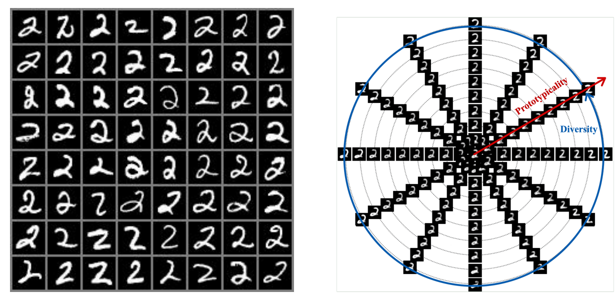
Yunhui Guo[1] Youren Zhang[2] Yubei Chen[3] Stella X. Yu[2,4]
[1]The University of Texas at Dallas [2]University of Michigan [3]UC Davis [4]UC Berkeley

Abstract

Given a set of images, our goal is to map each image to a point in a feature space such that, not only point proximity indicates visual similarity, but where it is located directly encodes how prototypical the image is according to the dataset.

Our key insight is to perform unsupervised feature learning in hyperbolic instead of Euclidean space, where the distance between points still reflects image similarity, yet we gain additional capacity for representing prototypicality with the location of the point: The closer it is to the origin, the more prototypical it is. The latter property is simply emergent from optimizing the metric learning objective: The image similar to many training instances is best placed at the center of corresponding points in Euclidean space, but closer to the origin in hyperbolic space.

We propose an unsupervised feature learning algorithm in Hyperbolic space with sphere pACKing. HACK first generates uniformly packed particles in the Poincaré ball of hyperbolic space and then assigns each image uniquely to a particle. With our feature mapper simply trained to spread out training instances in hyperbolic space, we observe that images move closer to the origin with congealing - a warping process that aligns all the images and makes them appear more common and similar to each other, validating our idea of unsupervised prototypicality discovery. We demonstrate that our data-driven prototypicality provides an easy and superior unsupervised instance selection to reduce sample complexity, increase model generalization with atypical instances and robustness with typical ones.



MNIST 2's (samples) $\Longrightarrow$ our image embedding

Figure 1. Given a dataset (left), we aim to learn an image feature that encodes not only visual similarity between instances but also data-driven prototypicality (right). Additionally, the angular arrangement of the features can naturally serve as a measure of diversity. Our feature encoder (in 2D hyperbolic space) is learned without any labels. We can then learn a decoder to map each point in the feature space back to an image. The right plot visualizes images located at the origin and those moving away in different directions, automatically revealing that 2's with loops are most common and the whole dataset can be grasped as the cursive style systematically varies.

1. Introduction

Not all instances are created equal. For example, the MNIST dataset of handwritten digits contain almost 6,000 samples of 2's; some are close to textbook versions that we are taught to follow, whereas others have idiosyncratic cursive styles, varying in proportions and stroke weights (Fig. 1). Given such a set of natural data, we are interested in dataset summarization and organization such that we can automatically discover which instances are more representative and which ones are anomalies. In other words, we aim to computationally rationalize *graded membership* [8, 36] of a category in a purely data-driven manner, putting each instance on a feature map that reflects not only their prototypicality in a particular dataset but also their visual similarity with each other.

Such unsupervised feature learning is useful for not only data organization, but those discovered representative instances or prototypes can be used for interpretable machine learning [4], curriculum learning [3], and learning better decision boundaries [5]. Prototypes also allow us to classify with as few as or even one example [31].

If the image feature is given, it is relatively easy to find prototypes: we just need to identify density peaks of the feature distribution of the image set. Otherwise, discovering prototypical instances without supervision is difficult: There is no universal definition or simple metric to assess the prototypicality of the examples.

One way to address this problem is to examine the gradient magnitude [5]. However, this approach is shown to have a high variance which results from different training setups [5]. Some methods address this problem using adversarial

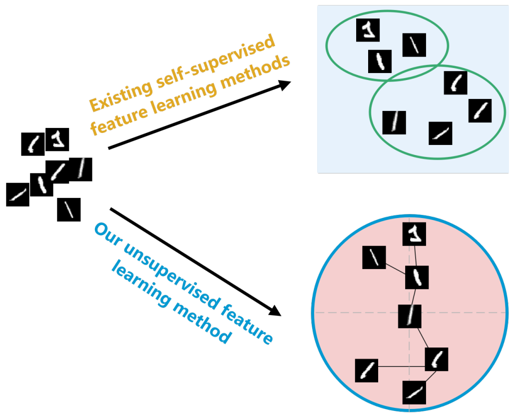


Figure 2. Existing self-supervised feature learning methods focus on visual similarity only in Euclidean space, whereas our unsupervised feature learning method embeds images in hyperbolic space which automatically encodes prototypicality with respect to the origin: Images are organized hierarchically, with typical images at the center and atypical ones near the boundary of the Poincaré ball.

robustness [5, 39]: Prototypical examples should be more adversarially robust. However, selecting these examples is dependent on the adversarial method and the metric used in the adversarial attack. Several methods exist but they are either based on heuristics or lack a proper justification [5].

Naturally, given a feature space, prototypical examples can be identified as density peaks. However, prototypicality varies with the feature. We propose an unsupervised feature learning method in hyperbolic space, which, unlike the shift-invariant Euclidean space, naturally embeds a continuous version of hierarchical trees rooted at the origin of the space.

Hyperbolic space is non-Euclidean space with constant non-negative curvature [1]. Different from Euclidean space, hyperbolic space can represent hierarchical relations with low distortion. Poincaré ball model is one of the most commonly used models for hyperbolic space [35]. One notable property of Poincaré ball model is that the distance to the origin grows exponentially as we move towards the boundary. Thus, points located in the center of the ball are close to all the other points while the points located close to the boundary are infinitely far away from other points. With unsupervised learning in hyperbolic space, we can learn features that capture both visual similarity and hierarchical proximity among instances.

Our key insight is that a typical image is similar (closest) to more nearby instances than atypical ones, and such an image would be at the center of the neighbourhood in Euclidean space but the parent/root node in a tree in hyperbolic space, closer to the origin. We develop a new learning procedure that first places all the target locations evenly in the Poincaré ball to reflect the desire of mapping uniformity [43], and then we optimize which images should be mapped to which target locations in a batch-wise Hungarian matching manner, where those similar to most instances naturally moving closer to the origin.

Our work makes the following contributions. **1)** We propose the first unsupervised feature learning method to learn features which capture visual similarity with distance between features and prototypicalitywith the distance to the origin. **2)** We develop a new learning paradigm that sits between supervised learning (with known targets) and unsupervised metric learning (with unknown targets and constrained metric distances). We want to map images to known targets uniformly packed and maximally distant in hyperbolic space, but we learn to optimize the specific image to target assignment. **3)** We validate our joint feature learning and data prototypicality discovery on congealing [31], where the consensus is that images after congealing are perceived to be more typical images. **4)** We demonstrate two practical usages of data-driven prototypicality: Prototypical and atypical examples are shown to reduce sample complexity for learning and increase the robustness of the model respectively.

2. Related Work

Prototypicality. The study of prototypical examples in machine learning has a long history. In Zhang [47], the authors select typical instances based on the fact that typical instances should be representative of the cluster. In Kim et al. [22], prototypical examples are defined as the examples that have maximum mean discrepancy within the data. Li et al. [27] propose to discover prototypical examples by architectural modifications: project the dataset onto a low-dimensional manifold and use a prototype layer to minimize the distance between inputs and the prototypes on the manifold. The robustness to adversarial attacks is also used as a criterion for prototypicality [39]. In Carlini et al. [5], the authors propose multiple metrics for prototypicality discovery. For example, the features of prototypical examples should be consistent across different training setups. However, these metrics usually depend heavily on the training setups and hyperparameters. The idea of prototypicality is also extensively studied in meta-learning for one-shot or few-shot classification [38]. No existing works address the prototypicality discovery problem in a data-driven fashion. Our proposed HACK naturally exploits hyperbolic space to organize the images based on prototypicality.

Unsupervised Learning in Hyperbolic Space. Learning features in hyperbolic space have shown to be useful for many machine learning problems [11, 34]. One useful property is that hierarchical relations can be embedded in hyperbolic space with low distortion [34]. Wrapped normal distribution, which is a generalized version of the normal distribution for modeling the distribution of points in hyperbolic space [33], is used as the latent space for constructing hyperbolic variational autoencoders (VAEs)

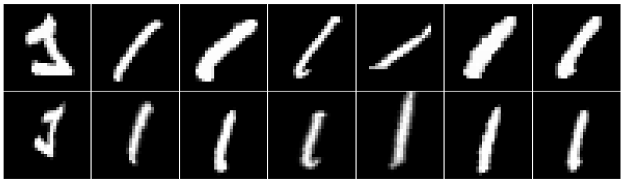


Figure 3. **Congealed images are more typical than the original images.** First row: sampled original images. Second row: the corresponding congealed images.

[23]. Poincaré VAEs is constructed in Mathieu et al. [30] with a similar idea to Nagano et al. [33] by replacing the standard normal distribution with hyperbolic normal distribution. Unsupervised 3D segmentation [20] and instance segmentation [44] are conducted in hyperbolic space via hierarchical hyperbolic triplet loss. CO-SNE [15] is recently proposed to visualize high-dimensional hyperbolic features in a two-dimensional hyperbolic space. Although hyperbolic distance facilitates the learning of hierarchical structure, how to leverage hyperbolic space for unsupervised prototypicality discovery is not explored in the current literature.

3. Sample Hierarchy

Sample Hierarchy VS. Class Hierarchy. While most of the existing works in hierarchical image classification focus on using label hierarchy [9, 14], there also exists a natural hierarchy among different samples. In Khrulkov et al. [21], the authors conducted an experiment to measure the δ-hyperbolicity of the various image datasets and showed that common image datasets such as CIFAR10 and CUB exhibit natural hierarchical structure among the samples. Amongst a collection of images representing digit 1, suppose $\mathbf{x}$ is used for representing an image with a digit '1' that is upright, $\mathbf{x}'$ is used for representing an image with a digit 1 that leaning left and $\mathbf{x}''$ is used for representing an image with a digit '1' that leaning right. Given a metric $d(\cdot,\cdot)$, if we assume that $d(\mathbf{x}'',\mathbf{x}') \approx d(\mathbf{x}'',\mathbf{x}) + d(\mathbf{x}',\mathbf{x})$, in this context, we can naturally view the sample $\mathbf{x}$ as the root, and consider the other samples as its children in an underlying tree.

Compared with class hierarchy which can be extracted based on the pre-defined label relations, sample hierarchy is much harder to construct due to the lack of ground truth. Once a sample hierarchy is established, there are currently no existing methods available for verifying the accuracy of the hierarchy. Additionally, just like with class hierarchies, there may be ambiguities when constructing a sample hierarchy since multiple samples could potentially serve as the root.

Building Sample Hierarchy from Density Peaks. Given existing features $\{f(v_i)\}$ obtained by applying a feature extractor for each instance v_i, prototypical examples can be found by examining the density peaks via techniques from density estimation. For example, the K-nearest neighbor density (K-NN) estimation [10] is defined as $p_{knn}(v_i, k) = \frac{k}{n}\frac{1}{A_d \cdot D^d(v_i, v_{k(i)})}$, where d is the feature dimension, $A_d = \pi^{d/2}/\Gamma(d/2+1)$, $\Gamma(x)$ is the Gamma function and $k(i)$ is the kth nearest neighbor of example v_i. The nearest neighbors can be found by computing the distance between the features. Therefore, the process of constructing sample hierarchy through density estimation can be conceptualized as a two-step procedure involving: 1) feature learning and 2) detecting density peaks.

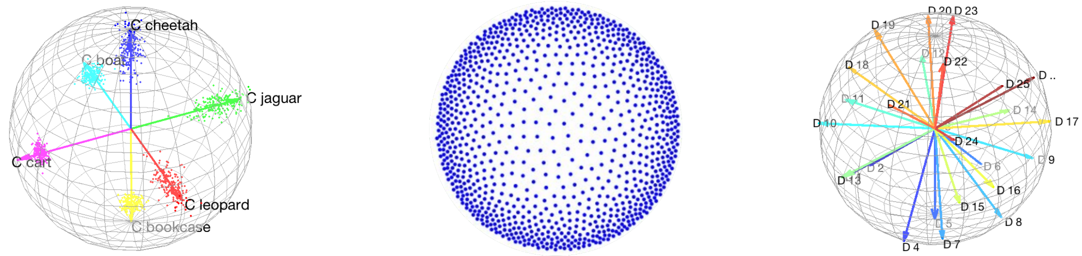
In the density estimation approach outlined above, the level of prototypicality depends on the learned features. Varying training setups can induce diverse feature spaces, resulting in differing conclusions on prototypicality. Nevertheless, prototypicality is an inherent attribute of the dataset and should remain consistent across various features. The aim of this paper is to extract features that intrinsically showcase the hierarchical organization of the samples. Specifically, by examining the feature alone within the feature space, we should be able to identify the example's prototypicality.

Construct a Sample Hierarchy from Congealing. To determine whether the feature truly captures prototypicality, it is necessary to identify which sample is the prototype. We ground our concept of prototypicality based on congealing [31]. In particular, we define prototypical examples in the *pixel space* by examining the distance of the images to the average image in the corresponding class. Our idea is based on a traditional computer vision technique called image alignment [40] that aims to find correspondences across images. During congealing [31], a set of images are transformed to be jointly aligned by minimizing the joint pixel-wise entropies. The congealed images are more prototypical: they are better aligned with the average image. Thus, we have a simple way to transform an atypical example into a typical example (see Figure 3). This is useful since given an unlabeled image dataset the typicality of the examples is unknown, congealing examples can be naturally served as examples with known typicality and be used as a validation for the effectiveness of our method.

4. Unsupervised Hyperbolic Feature Learning

4.1. Poincaré Ball Model for Hyperbolic Space

Euclidean space has a curvature of zero and a hyperbolic space is a Riemannian manifold with constant negative curvature.. There are several isometrically equivalent models for visualizing hyperbolic space with Euclidean representation. The Poincaré ball model is the commonly used one in hyperbolic representation learning [35]. The n-dimensional Poincaré ball model is defined as $(\mathbb{B}^n, \mathfrak{g}_{\mathbf{x}})$, where $\mathbb{B}^n = \{\mathbf{x} \in \mathbb{R}^n : \|\mathbf{x}\| < 1\}$ and $\mathfrak{g}_{\mathbf{x}} = (\gamma_{\mathbf{x}})^2 I_n$ is the Riemannian metric tensor. $\gamma_{\mathbf{x}} = \frac{2}{1-\|\mathbf{x}\|^2}$ is the conformal factor and I_n is the Euclidean metric tensor.



a) Supervised classification with fixed known targets

b) Our unsupervised feature learning with fixed but *unknown* targets

c) Metric feature learning with unknown targets

Figure 4. **The proposed HACK has a predefined geometrical arrangement and allows the images to be freely assigned to any particle.** a) Standard supervised learning has predefined targets. The image is only allowed to be assigned to the corresponding target. b) HACK packs particles uniformly in hyperbolic space to create initial seeds for the organization. The images are assigned to the particles based on their prototypicality and semantic similarities. c) Standard unsupervised learning has no predefined targets and images are clustered based on their semantic similarities.

Hyperbolic Distance. Given two points $\boldsymbol{u} \in \mathbb{B}^n$ and $\boldsymbol{v} \in \mathbb{B}^n$, the hyperbolic distance is defined as,

$$d_{\mathbb{B}^n}(\boldsymbol{u}, \boldsymbol{v}) = \operatorname{arcosh}\left(1 + 2\frac{\|\boldsymbol{u} - \boldsymbol{v}\|^2}{(1 - \|\boldsymbol{u}\|^2)(1 - \|\boldsymbol{v}\|^2)}\right) \quad (1)$$

where arcosh is the inverse hyperbolic cosine function and $\|\cdot\|$ is the usual Euclidean norm.

Hyperbolic distance has the unique property that it grows exponentially as we move towards the boundary of the Poincaré ball. In particular, the points on the circle represent points in infinity. Hyperbolic space is naturally suitable for embedding hierarchical structure [35, 37] and can be regarded as a continuous representation of trees [6]. The hyperbolic distance between samples implicitly reflects their hierarchical relation. Thus, by embedding images in hyperbolic space we can naturally organize images based on their semantic similarity and prototypicality.

4.2. Build Instance Hierarchy in Hyperbolic Space

Hyperbolic space is naturally suitable for embedding tree structure. However, in order to leverage hyperbolic space to build a sample hierarchy in an unsupervised manner, a suitable objective function is still missing. There are two challenges in designing the objective function. First, the underlying tree structure of the samples is unknown. Second, how to perform feature learning such that hierarchy can naturally emerge is unclear. In this paper, we propose Hyperbolic space with sphere pACKing, also called HACK, to address the two challenges.

To address the first challenge, instead of creating a predefined tree structure that might not faithfully represent the genuine hierarchical organization, we leverage sphere packing in hyperbolic space for building a skeleton for placing the samples. We specify where the particles should be located ahead of training with uniform packing, which by design are maximally evenly spread out in hyperbolic space. The uniformly distributed particles guide feature learning to achieve maximum instance discrimination [45] while enabling us to extract a tree structure from the samples.

To address the second challenge, HACK figures out which instance should be mapped to which target through bipartite graph matching as a global optimization procedure. During training, HACK minimizes the total hyperbolic distances between the mapped image point (in the feature space) and the target, those that are more typical naturally emerge closer to the origin of Poincaré ball. HACK differs from the existing learning methods in several aspects (Figure 4). Different from supervised learning, HACK allows the image to be assigned to *any* target (particle). This enables the exploration of the natural organization of the data. Different from unsupervised learning method, HACK specifies a predefined geometrical organization which encourages the corresponding structure to be emerged from the dataset.

4.3. Sphere Packing in Hyperbolic Space

Given n particles, our goal is to pack the particles into a two-dimensional hyperbolic space as densely as possible. We derive a simple repulsion loss function to encourage the particles to be equally distant from each other. The loss is derived via the following steps. First, we need to determine the radius of the Poincaré ball used for packing. We use a curvature of 1.0 so the radius of the Poincaré ball is 1.0. The whole Poincaré ball cannot be used for packing since the volume is infinite. We use $r < 1$ to denote the actual radius used for packing. Thus, our goal is to pack n particles in a compact subspace of Poincaré ball. Then, the Euclidean radius r is further converted into hyperbolic radius $r_{\mathbb{B}}$. Let $s = \frac{1}{\sqrt{c}}$, where c is the curvature. The relation between r and $r_{\mathbb{B}}$ is $r_{\mathbb{B}} = s \log \frac{s+r}{s-r}$. Next, the total hyperbolic area $A_{\mathbb{B}}$ of a Poincaré ball of radius $r_{\mathbb{B}}$ can

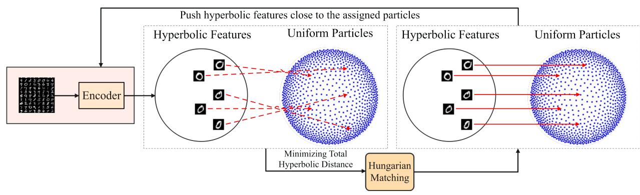


Figure 5. **HACK conducts unsupervised learning in hyperbolic space with sphere packing.** The images are mapped to particles by minimizing the total hyperbolic distance. HACK learns features that can capture both visual similarities and prototypicality.

Algorithm 1 HACK: Unsupervised Learning in Hyperbolic Space. ised Learning in Hyperbolic Space.

Require: # of images: $n \geq 0$. Radius for packing: $r < 1$. An encoder with parameters θ: f_θ

Generate uniformly distributed particles in hyperbolic space by minimizing the repulsion loss in Equation 2

Given $\{(\mathbf{x}_1, s_1), (\mathbf{x}_2, s_2), ..., (\mathbf{x}_b, s_b)\}$, optimize f_θ by minimizing the total hyperbolic distance via Hungarian algorithm.

be computed as $A_{\mathbb{B}} = 4\pi s^2 \sinh^2(\frac{r_{\mathbb{B}}}{2s})$, where sinh is the hyperbolic sine function. Finally, the area per point A_n can be easily computed as $\frac{A_{\mathbb{B}}}{n}$, where n is the total number of particles. Given A_n, the radius per point can be computed as $r_n = 2s\sinh^{-1}(\sqrt{\frac{A_n}{4\pi s^2}})$. We use the following loss to generate uniform packing in hyperbolic space. Given two particles i and j, the repulsion loss V is defined as,

$$V(i,j) = \{\frac{1}{[2r_n - \max(0, 2r_n - d_{\mathbb{B}}(i,j))]^k} - \frac{1}{(2r_n)^k}\} \cdot C(k) \tag{2}$$

where $C(k) = \frac{(2r_n)^{k+1}}{k}$ and k is a hyperparameter. Intuitively, if the particle i and the particle j are within $2r_n$, the repulsion loss is positive. Minimizing the repulsion loss would push the particles i and j away. If the repulsion is zero, this indicates all the particles are equally distant. Also the repulsion loss grows significantly when two particles become close. We also adopt the following boundary loss to prevent the particles from escaping the ball,

$$B(i;r) = \max(0, \text{norm}_i - r + \text{margin}) \tag{3}$$

where norm_i is the ℓ_2 norm of the representation of the particle i. Figure 4 b) shows an example of the generated particles that are uniformly packed in hyperbolic space.

4.4. Hyperbolic Instance Assignment

HACK learns the features by optimizing the assignments of the images to particles (Figure 5). The assignment should be one-to-one, i.e., each image is assigned to one particle and each particle is allowed to be associated with one image. We cast the instance assignment problem as a bipartite matching problem [12] and solve it with Hungarian algorithm [32].

Initially, we randomly assign the particles to the images, thus there is a random one-to-one correspondence between the images to the particles (not optimized). Given a batch of samples $\{(\mathbf{x}_1, \mathbf{s}_1), (\mathbf{x}_2, \mathbf{s}_2), ..., (\mathbf{x}_B, \mathbf{s}_B)\}$, where $\mathbf{x}_i$ is an image and $\mathbf{s}_i$ is the corresponding particle, and an encoder f_θ, we generate the hyperbolic feature for each image $\mathbf{x}_i$ as $f_\theta(\mathbf{x}_i) \in \mathbb{B}^2$, where $\mathbb{B}^2$ is a two-dimensional Poincaré ball. For a given hyperbolic feature $f_\theta(\mathbf{x})$, with fixed particle locations, the distance between the hyperbolic feature and the particles signifies the hierarchical level of the associated sample. Thus, to determine the hierarchical levels for all samples within the hierarchy, we must establish a one-to-one mapping between all the samples and the particles. This can be cast as the following bipartite matching problem in hyperbolic space,

$$\ell(\theta, \pi) = \sum_{i=1}^{B} d_{\mathbb{B}^n}(f_\theta(\mathbf{x}_i), \mathbf{s}_{\pi(f_\theta(\mathbf{x}_i))}) \tag{4}$$

where $\pi : f_\theta(\mathbf{x}) \rightarrow \mathbb{N}$ is a projection function which projects hyperbolic features to a particle index. Minimizing $\ell(\theta, \pi)$ with respect to π is a combinatorial optimization problem, which can hardly be optimized with θ using gradient-based algorithms. Thus, we adopt a joint optimization strategy which optimizes θ and π alternatively. In each batch, we first leverage the Hungarian algorithm [32] to find the optimal matching π^* based on the current hyperbolic features. Then we minimize Eq. 4 with respect to θ based on the current assignment π^*. This process is repeated for a certain number of epochs until convergence is achieved. On the other hand, the feature encoder can serve as an image prior for assigning similar images to nearby particles [41].

The Hungarian algorithm [32] has a complexity of $\mathcal{O}(x^3)$, where x is the number of items. As we perform the particle assignment in the batch level, the time and memory complexity is tolerable. Also, the one-to-one correspondence between the images and particles is always maintained during training. After training, based on the assigned particle, the level of the sample in the hierarchy can be easily retrieved. The details of HACK are shown in Algorithm 1.

5. Experiments

We design several experiments to show the effectiveness of HACK for the semantic and hierarchical organization. First, we first construct a dataset with known hierarchical structure using the congealing algorithm [31]. Then, we apply HACK to datasets with unknown hierarchical structure to organize the samples based on the semantic and prototypical structure.

Finally, we show that the prototypical structure can be used to reduce sample complexity and increase model robustness.

Datasets. We first construct a dataset called *Congealed MNIST*. To verify the efficacy of HACK for unsupervised prototypicality discovery, we need a benchmark with known prototypical examples. However, currently there is no standard benchmark for this purpose. To construct the benchmark, we use the congealing algorithm from Miller et al. [31] to align the images in each class of MNIST [25]. The congealing algorithm is initially used for one-shot classification. During congealing, the images are brought into correspondence with each other jointly. The congealed images are more prototypical: they are better aligned with the average image. The synthetic data is generated by replacing 500 original images with the corresponding congealed images. In the Appendix, we show the results of changing the number of replaced original images. We expect HACK to discover the congealed images and place them in the center of the Poincaré ball. We also aim to discover the prototypical examples from each class of the standard MNIST dataset [25] and CIFAR10 [24]. CIFAR10 consists of 60000 from 10 object categories ranging from airplane to truck. CIFAR10 is more challenging than MNIST since it has larger intra-class variations. Moreover, to better visualize how HACK arranges the samples according to their prototypicality, we run HACK on 10k US Adult Faces [2] (hereafter referred to as USA10kF), which contains 10,168 natural face photographs.

Baselines. We consider several existing metrics proposed in Carlini et al. [5] for prototypicality discovery, the details can be found in the Appendix.

- Holdout Retraining [5]: We consider the Holdout Retraining proposed in Carlini et al. [5]. The idea is that the distance of features of prototypical examples obtained from models trained on different datasets should be close.
- Model Confidence [5]: Intuitively, the model should be confident in prototypical examples. Thus, it is natural to use the confidence of the model prediction as the criterion for prototypicality.
- UHML [46]: UHML is an unsupervised hyperbolic learning method that aims to discover the natural hierarchies of data by taking advantage of hyperbolic metric learning and hierarchical clustering.

Implementation Details. We implement HACK in PyTorch and the code will be made public. To generate uniform particles, we first randomly initialize the particles and then run the training for 1000 epochs with a 0.01 learning rate to minimize the repulsion loss and boundary loss. The curvature of the Poincaré ball is 1.0 and the r is 0.76 which is used to alleviate the numerical issues [16]. The hyperparameter k is 1.55 which is shown to generate uniform particles well. For the assignment, we use a LeNet [26] for MNIST, a ResNet20 [17] for CIFAR10, and a ResNet18 for USA10kF as the encoder. We apply HACK to each class separately.

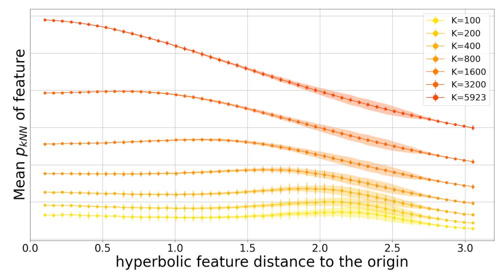
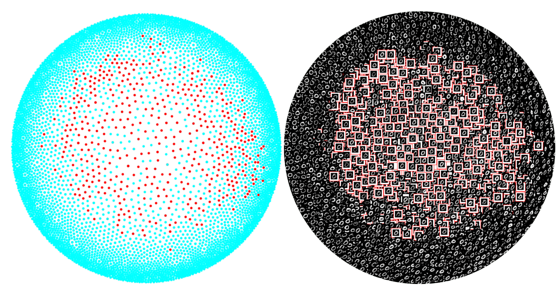


Figure 6. **Hyperbolic space can capture the prototypicality inherently.** The error bar of each point is given by the variance of density within the corresponding portion, and the width of the shaded band indicates the number of features within the portion.



a) b)

Figure 7. **Congealed images are located in the center of the Poincaré ball.** a) Red dots denote congealed images and cyan dots denote original images. b) Typical images are in the center and atypical images are close to the boundary. Images are also clustered together based on visual similarity. Congealed images are shown in red boxes.

We attach a fully connected layer to project the feature into a two-dimensional Euclidean space. The image features are then projected onto hyperbolic space via an exponential map. We run the training for 200 epochs using a cosine learning rate scheduler [29] with an initial learning rate of 0.1. We optimize the assignment *every other* epoch. All the experiments are run on an NVIDIA TITAN RTX GPU.

5.1. Prototypicality in the Hyperbolic Feature Norm

We explicitly show that the hyperbolic space can capture prototypicality by analyzing the relation between hyperbolic norms and the K-NN density estimation. Taking the learned hyperbolic features, we first divide the range of norms of hyperbolic features into numerous portions with equal length (50 portions for this plot). The mean K-NN density is calculated by averaging the density estimation of features within each portion. Figure 6 shows that the mean density drops as the norm increases, which shows that the prototypicality emerges automatically within the norms, the inherent characteristic of hyperbolic space. This validates that prototypicality is reflected in the hyperbolic feature norm.

a) MNIST b) CIFAR10

Figure 8. **Our unsupervised learning methods conform to our visual perception. The images are organized from left to right, top to bottom to cover the 360 degrees at the same radius.**

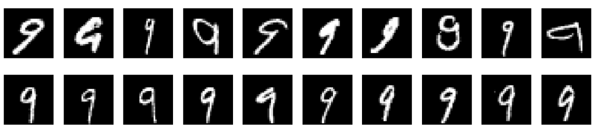
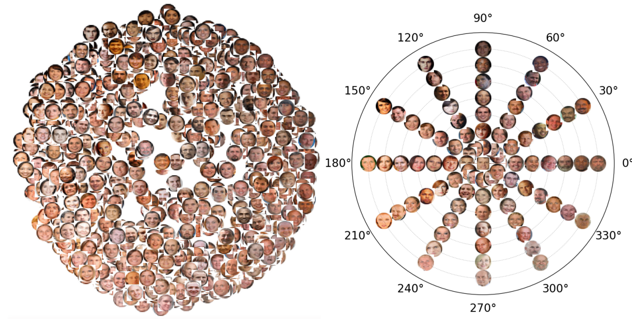


Figure 9. **HACK discovers typical and atypical images from the data. First row**: atypical images with large hyperbolic norm. **Second row**: typical images with small hyperbolic norm.

5.2. Visual Prototypicality

Congealed MNIST. We further apply HACK for visual feature learning on congealed MNIST. Figure 7 shows that HACK can discover the congealed images from all images. In Figure 7 a), the red particles denote the congealed images and cyan particles denote the original images. We can observe that the congealed images are assigned to the particles located in the center of the Poincaré ball. This verifies that HACK can *indeed* discover prototypical examples from the original dataset. In the Appendix, we show that the features of atypical examples gradually move to the boundary of the Poincaré ball during training. In Figure 7 b), we show the actual images that are embedded in the two-dimensional hyperbolic space. We can observe that the images in the center of Poincaré ball are more prototypical and images close to the boundary are more atypical. Also, the images are naturally organized by their semantic similarity. In summary, HACK can discover prototypicality and also organize the images based on their semantic and hierarchical structure. To the best of our knowledge, this is the first unsupervised learning method that can be used to discover prototypical examples in a data-driven fashion.

USA10kF. Figure 10 a) shows the assignment of 2000 images sampled from USA10kF. Compared to MNIST, the variation in faces is much more complex. A facial image is also subject to various factors (such as race, facial expression, environmental condition, etc.). Therefore, the results from USA10kF are less intuitive than those from MNIST. However, Figure 10 b) illustrates the evolutionary process of images originating from the center of hyperbolic space and progressing along different directions. In the space, we selected 12 directions at equal angular intervals and chose five equally spaced sampling points in each direction. The images closest to these sampling points are displayed at their respective locations. Although the detailed organization is unclear, the evolutionary process reveals a tendency of HACK to cluster different features together, such as darker skin tones appearing in the 0-90 degree range and lighter skin tones in the 180-270 degree range.



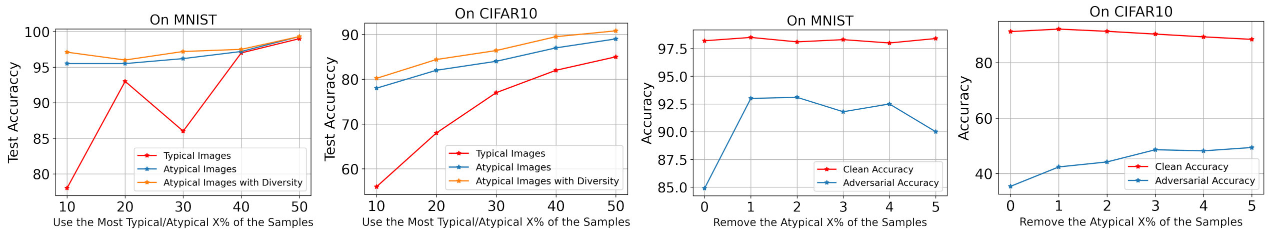
a) b)

Figure 10. **HACK captures prototypicality of faces**. a) The assignment of 2000 images sampled from USA10kF. b) The angles indicate the uniform division of the space. For each orientation, 8 equidistant sample points were selected and the nearest faces are shown at each point. The face located in the center is most symmetrical with clear smiling

MNIST and CIFAR10. Figure 8 shows the embedding of class 0 from MNIST and class "airplane" from CIFAR10 arranged to cover 360 degrees of at the same radius. The visual similarity of the images has a smooth transition as we move around angularly. Figure 9 shows the typical images and atypical images discovered by HACK. This further illustrates that HACK captures the semantic similarity of the images which enables prototypicality discovery. Please see the Appendix for more results.

5.3. Prototypicality for Instance Selection

Figure 12 shows the comparison of the baselines with HACK. With HACK, typical images are characterized by the smallest hyperbolic norms, whereas atypical images are associated with the largest hyperbolic norms. We can observe that both HACK and Model Confidence (MC) can discover typical and atypical images. Compared with MC, HACK defines prototypicality as the distance of the sample to other samples which is more aligned with human intuition. Moreover, in addition to prototypicality, HACK can also be used to organize examples by semantic similarities. Holdout Retraining



a) Reducing sample complexity   b) Increasing model robustness

Figure 11. **HACK can be used to construct sample prototypical hierarchy which is useful for several downstream tasks.** a) Training with atypical examples achieves higher accuracy than training with typical examples. b) The adversarial accuracy greatly improves after removing the X% of most atypical examples.

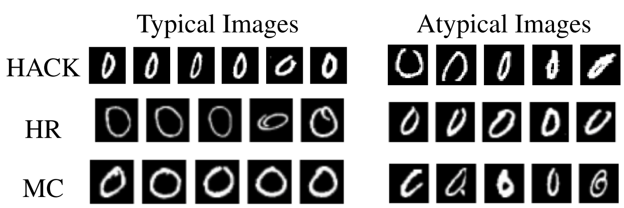


Figure 12. **HACK can capture both prototypicality and diversity in the dataset.** Each row shows the typical and atypical images discovered by HACK/HR [5]/MC [5].

(HR) is not effective for prototypicality discovery due to the randomness of model training.

5.4. Application of Prototypicality

Reducing Sample Complexity. The proposed HACK can discover prototypical images as well as atypical images. We show that with *atypical* images we can reduce the sample complexity for training the model. Prototypical images are representative of the dataset but lack variations. Atypical examples contain more variations and it is intuitive that models trained on atypical examples should generalize better to the test samples. To verify this hypothesis, we select a subset of samples based on the norm of the features which indicates prototypicality of the examples. In particular, typical samples correspond to the samples with smaller norms and atypical samples correspond to the samples with larger norms. The angular layout of the hyperbolic features naturally captures sample diversity, thus for selecting atypical examples, we consider introducing more diversity by sampling images with large norms along the angular direction.

Figure 11 a) shows that training with atypical images can achieve much higher accuracy than training with typical images. In particular, training with the most atypical 10% of the images achieves 22.67% higher accuracy than with the most typical 10% of the images on CIFAR10. Similar results can be observed on MNIST. Thus, HACK provides an easy solution to reduce sample complexity. We also compared UHML [46], which is an unsupervised metric learning in hyperbolic space, with HACK on the MNIST dataset. By incorporating 10% atypical samples based on feature norm, HACK outperformed UHML by 10.2%. Also by excluding the 1% atypical examples, HACK achieved an additional 5.7% improvement over UHML.

Increasing Model Robustness. Training models with atypical examples can lead to a vulnerable model to adversarial attacks [5, 28]. Intuitively, atypical examples lead to a less smooth decision boundary thus a small perturbation to examples is likely to change the prediction. With HACK, we can easily identify atypical samples to improve the robustness of the model. We use MNIST and CIFAR 10 as the benchmark and use FGSM [13] to attack the model with an ϵ of 0.07 on MNIST and 8/(255*std) on CIFAR10, where std is the standard deviation used for normalization. More details of the attack settings can be found in the appendix. We identify the atypical examples based on the norm of the features with HACK and remove the most atypical X% of the examples. Figure 11 b) shows that discarding atypically examples greatly improves the robustness of the model: the adversarial accuracy is improved by 8.7% when excluding the most atypical 1% of examples on MNIST and 7.3% on CIFAR10. It is worth noting that the clean accuracy remains the same after removing a small number of atypical examples.

6. Summary

We propose HACK, an unsupervised learning method that organizes images in hyperbolic space using sphere packing. By optimizing image assignments to uniformly distributed particles, HACK leverages the inherent properties of hyperbolic space, leading to the natural emergence of prototypical and semantic structures through feature learning. We validate HACK on synthetic data and standard datasets, demonstrating its ability to discover prototypical examples for reducing sample complexity and increasing model robustness.

Acknowledgements. This project was supported, in part, by NSF 2131111, NSF 2215542, and NSF 2313151 to S. Yu, and a grant from UT Dallas to Y. Guo.

References

[1] James W Anderson. *Hyperbolic geometry*. Springer Science & Business Media, 2006. 2

[2] Wilma A Bainbridge, Phillip Isola, and Aude Oliva. The intrinsic memorability of face photographs. *Journal of Experimental Psychology: General*, 142(4):1323, 2013. 6

[3] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48, 2009. 1

[4] Jacob Bien and Robert Tibshirani. Prototype selection for interpretable classification. *The Annals of Applied Statistics*, 5(4):2403–2424, 2011. 1

[5] Nicholas Carlini, Ulfar Erlingsson, and Nicolas Papernot. Prototypical examples in deep learning: Metrics, characteristics, and utility. 2018. 1, 2, 6, 8

[6] Ines Chami, Albert Gu, Vaggos Chatziafratis, and Christopher Ré. From trees to continuous embeddings and back: Hyperbolic hierarchical clustering. *Advances in Neural Information Processing Systems*, 33:15065–15076, 2020. 4

[7] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. 2

[8] Lieven Decock and Igor Douven. What is graded membership? *Noûs*, 48(4):653–82, 2014. 1

[9] Ankit Dhall, Anastasia Makarova, Octavian Ganea, Dario Pavllo, Michael Greeff, and Andreas Krause. Hierarchical image classification using entailment cone embeddings. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 836–837, 2020. 3

[10] Evelyn Fix and Joseph Lawson Hodges. Discriminatory analysis. nonparametric discrimination: Consistency properties. *International Statistical Review/Revue Internationale de Statistique*, 57(3):238–247, 1989. 3

[11] Octavian-Eugen Ganea, Gary Bécigneul, and Thomas Hofmann. Hyperbolic neural networks. *arXiv preprint arXiv:1805.09112*, 2018. 2

[12] Alan Gibbons. *Algorithmic graph theory*. Cambridge university press, 1985. 5

[13] Ian J Goodfellow, Jonathon Shlens, and Christian Szegedy. Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*, 2014. 8, 1

[14] Yanming Guo, Yu Liu, Erwin M Bakker, Yuanhao Guo, and Michael S Lew. Cnn-rnn: a large-scale hierarchical image classification framework. *Multimedia tools and applications*, 77(8):10251–10271, 2018. 3

[15] Yunhui Guo, Haoran Guo, and Stella Yu. Co-sne: Dimensionality reduction and visualization for hyperbolic data. *arXiv preprint arXiv:2111.15037*, 2021. 3

[16] Yunhui Guo, Xudong Wang, Yubei Chen, and Stella X Yu. Clipped hyperbolic classifiers are super-hyperbolic classifiers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11–20, 2022. 6

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 6

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross B. Girshick. Momentum contrast for unsupervised visual representation learning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9726–9735, 2019. 1

[19] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *CVPR*, pages 9729–9738, 2020. 2

[20] Joy Hsu, Jeffrey Gu, Gong-Her Wu, Wah Chiu, and Serena Yeung. Learning hyperbolic representations for unsupervised 3d segmentation. *arXiv preprint arXiv:2012.01644*, 2020. 3

[21] Valentin Khrulkov, Leyla Mirvakhabova, Evgeniya Ustinova, Ivan Oseledets, and Victor Lempitsky. Hyperbolic image embeddings. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6418–6428, 2020. 3

[22] Been Kim, Rajiv Khanna, and Oluwasanmi O Koyejo. Examples are not enough, learn to criticize! criticism for interpretability. *Advances in neural information processing systems*, 29, 2016. 2

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013. 3

[24] Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images. 2009. 6

[25] Yann LeCun. The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*, 1998. 6, 1

[26] 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. 6, 1

[27] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. 2

[28] Yongshuai Liu, Jiyu Chen, and Hao Chen. Less is more: Culling the training set to improve robustness of deep neural networks. In *International Conference on Decision and Game Theory for Security*, pages 102–114. Springer, 2018. 8

[29] Ilya Loshchilov and Frank Hutter. Sgdr: Stochastic gradient descent with warm restarts. *arXiv preprint arXiv:1608.03983*, 2016. 6, 1

[30] Emile Mathieu, Charline Le Lan, Chris J Maddison, Ryota Tomioka, and Yee Whye Teh. Continuous hierarchical representations with poincar\'e variational auto-encoders. *arXiv preprint arXiv:1901.06033*, 2019. 3

[31] Erik G Miller, Nicholas E Matsakis, and Paul A Viola. Learning from one example through shared densities on transforms. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No. PR00662)*, pages 464–471. IEEE, 2000. 1, 2, 3, 5, 6

[32] James Munkres. Algorithms for the assignment and transportation problems. *Journal of the society for industrial and applied mathematics*, 5(1):32–38, 1957. 5

[33] Yoshihiro Nagano, Shoichiro Yamaguchi, Yasuhiro Fujita, and Masanori Koyama. A wrapped normal distribution on

hyperbolic space for gradient-based learning. In *International Conference on Machine Learning*, pages 4693–4702. PMLR, 2019. 2, 3

[34] Maximilian Nickel and Douwe Kiela. Poincar\'e embeddings for learning hierarchical representations. *arXiv preprint arXiv:1705.08039*, 2017. 2

[35] Maximillian Nickel and Douwe Kiela. Poincaré embeddings for learning hierarchical representations. *Advances in neural information processing systems*, 30, 2017. 2, 3, 4

[36] Eleanor Rosch. Cognitive representations of semantic categories. *Journal of experimental psychology: General*, 104(3): 192, 1975. 1

[37] Rik Sarkar. Low distortion delaunay embedding of trees in hyperbolic plane. In *International Symposium on Graph Drawing*, pages 355–366. Springer, 2011. 4

[38] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017. 2

[39] Pierre Stock and Moustapha Cisse. Convnets and imagenet beyond accuracy: Understanding mistakes and uncovering biases. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 498–512, 2018. 2

[40] Richard Szeliski et al. Image alignment and stitching: A tutorial. *Foundations and Trends® in Computer Graphics and Vision*, 2(1):1–104, 2007. 3

[41] Dmitry Ulyanov, Andrea Vedaldi, and Victor Lempitsky. Deep image prior. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 9446–9454, 2018. 5

[42] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 1

[43] Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, pages 9929–9939. PMLR, 2020. 2

[44] Zhenzhen Weng, Mehmet Giray Ogut, Shai Limonchik, and Serena Yeung. Unsupervised discovery of the long-tail in instance segmentation using hierarchical self-supervision. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2603–2612, 2021. 3

[45] 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*, pages 3733–3742, 2018. 4, 2

[46] Jiexi Yan, Lei Luo, Cheng Deng, and Heng Huang. Unsupervised hyperbolic metric learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12465–12474, 2021. 6, 8

[47] Jianping Zhang. Selecting typical instances in instance-based learning. In *Machine learning proceedings 1992*, pages 470–479. Elsevier, 1992. 2

Unsupervised Feature Learning with Emergent Data-Driven Prototypicality

Supplementary Material

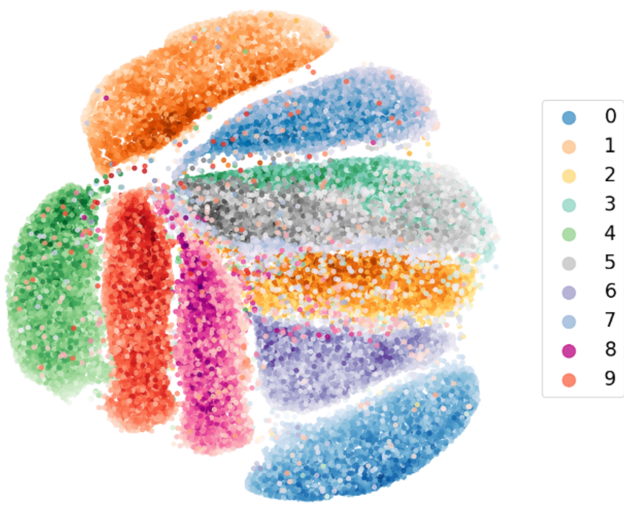


Figure 13. The KNN density estimation on MoCo [18] features of MNIST [25]. The shades of color represent the density value: the darker the color, the higher the density.

7. More Details on K-NN Density Estimation on MNIST

Feature Extraction: We use a LeNet [26] without classifier as the encoder and follow the scheme of MoCo [18] to train the feature extractor. We run the training for 200 epochs and the initial learning rate is 0.06. We use a cosine learning rate scheduler [29].

Visualization: Figure 13 visualize the KNN density estimation on MoCo [18] features of MNIST [25]. The output features have the dimension of 64. To visualize the features, we use t-SNE [42] with the perplexity of 40 and 300 iterations for optimization.

8. More Details on Hyperbolic Instance Assignment

A more detailed description of the hyperbolic instance assignment is given.

Initially, we randomly assign the particles to the images. Given a batch of samples $\{(\mathbf{x}_1, s_1), (\mathbf{x}_2, s_2), ..., (\mathbf{x}_b, s_b)\}$, where $\mathbf{x}_i$ is an image and s_i is the corresponding particle. Given an encoder f_θ, we generate the hyperbolic feature for each image $\mathbf{x}_i$ as $f_\theta(\mathbf{x}_i) \in \mathbb{B}^2$, where $\mathbb{B}^2$ is a two-dimensional Poincaré ball.

we aim to find the minimum cost bipartite matching of the images to the particles. The cost to minimize is the total hyperbolic distance of the hyperbolic features to the particles. We first compute all the pairwise distances between the hyperbolic features and the particles. This is the cost matrix of the bipartite graph. Then we use the Hungarian algorithm to optimize the assignment (Figure 14).

Suppose we train the encoder f_θ for T epochs. We run the hyperbolic instance assignment every other epoch to avoid instability during training. **We optimize the encoder f_θ to minimize the hyperbolic distance of the hyperbolic feature to the assigned particle in each batch**.

9. Details of Adversarial Attacks

For adversarial attacks, we use MNIST and CIFAR 10 as the benchmark and use FGSM [13] to attack the model. For MNIST, we leverage an ϵ of 0.07. For CIFAR10, as the range of the pixel values is from 0 to 255, we leverage an ϵ of 8. For model training, we standardize the pixel values by removing the mean and scaling to unit variance. Thus, the final ϵ on CIFAR10 is 8/(255*std), where std is the standard deviation used for normalization.

10. Details of Baselines

Holdout Retraining: We consider the Holdout Retraining proposed in [5]. The idea is that the distance of features of prototypical examples obtained from models trained on different datasets should be close. In Holdout Retraining, multiple models are trained on the same dataset. The distances of the features of the images obtained from different models are computed and ranked. The prototypical examples are those examples with the closest feature distance.

Model Confidence: Intuitively, the model should be confident on prototypical examples. Thus, it is natural to use the confidence of the model prediction as the criterion for prototypicality. Once we train a model on the dataset, we use the confidence of the model to rank the examples. The prototypical examples are those examples that the model is most

11. Gradually Adding More Congealed Images

We gradually increase the number of original images replaced by congealed images from 100 to 500. Still, as shown in Figure 15, HACK can learn a representation that captures the concept of prototypicality regardless of the number of congealed images. This again confirms the effectiveness of HACK for discovering prototypicality.

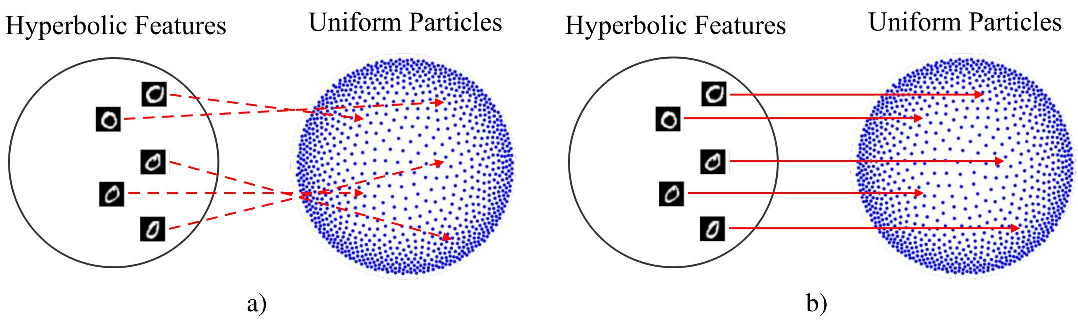


Figure 14. **Hyperbolic Instance Assignment minimizes the total hyperbolic distances between the image features and the particles.** a) Initial assignment. b) Optimized assignment.

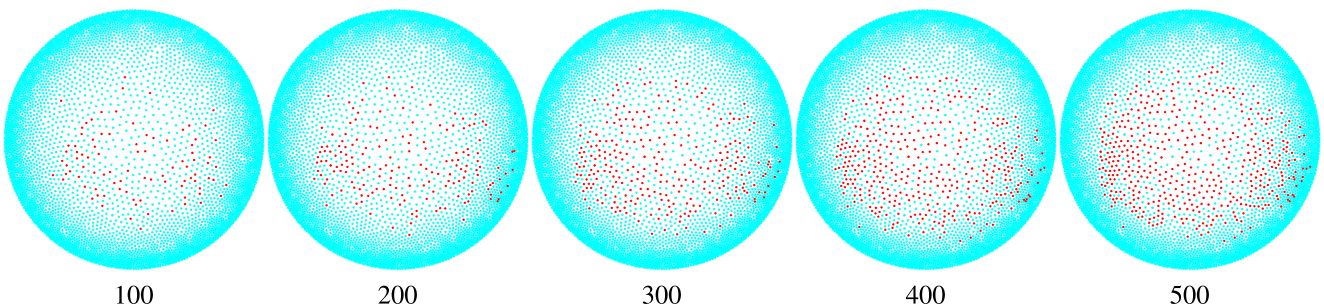


Figure 15. **HACK consistently places congealed images in the center of the Poincaré ball**. We gradually increase the number of original images replaced by congealed images from 100 to 500. The congealed images are marked with red dots and the original images are marked with cyan dots.

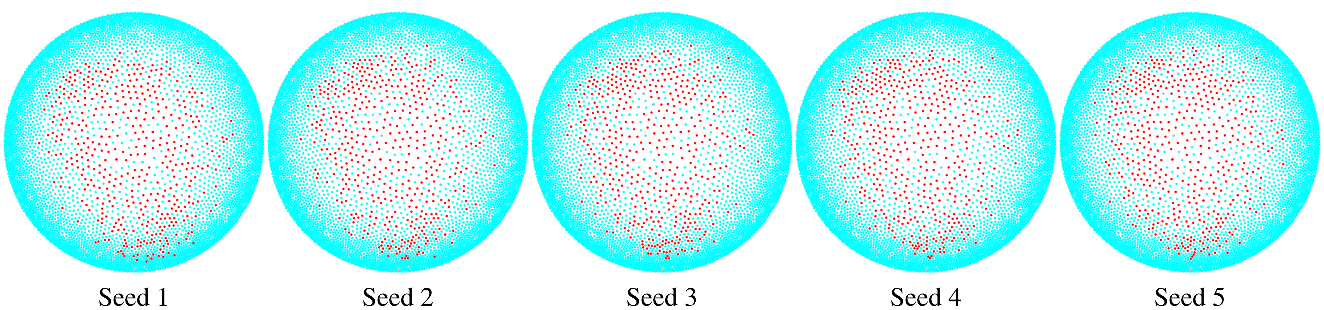


Figure 16. **HACK consistently places congealed images in the center of the Poincaré ball in multiple runs with different random seeds.**. The congealed images are marked with red dots and the original images are marked with cyan dots.

12. Different Random Seeds

We further run the assignment 5 times with different random seeds. The results are shown in Figure 16. We observe that the algorithm does not suffer from high variance and the congealed images are always assigned to the particles in the center of the Poincaré ball. This further confirms the efficacy of the proposed method for discovering prototypicality.

13. Emergence of Prototypicality in the Feature Space

Existing unsupervised learning methods mainly focus on learning features for differentiating different classes or samples [7, 19, 45]. The learned representations are transferred to various downstream tasks such as segmentation and detection. In contrast, the features learned by HACK aim at capturing prototypicality within a single class.

To investigate the effectiveness of HACK in revealing prototypicality, we can include or exclude congealed images in the training process. When the congealed images are included in the training process, we expect the congealed images to be located in the center of the Poincaré ball while the original images to be located near the boundary of the Poincaré ball. When the congealed images are excluded from the training process, we expect the features of congealed images produced via the trained network to be located in the center of the Poincaré ball.

13.1. Training with congealed images and original images

We follow the same setups as in Section 4.3.1 of the main text. Figure 17 shows the hyperbolic features of the congealed images and original images in different training epochs. The features of the congealed images stay in the center of the Poincaré ball while the features of the original images gradually expand to the boundary.

13.2. Training only with original images

Figure 18 shows the hyperbolic features of the congealed images **when the model is trained only with original images**. As we have shown before, congealed images are naturally more typical than their corresponding original images since they are aligned with the average image. The features of congealed images are all located close to the center of the Poincaré ball. This demonstrates that prototypicality naturally emerges in the feature space.

Without using congealed images during training, we exclude any artifacts and further confirm the effectiveness of HACK for discovering prototypicality. We also observe that the features produced by HACK also capture the fine-grained similarities among the congealing images despite the fact that all the images are aligned with the average image.

14. Discussions on Societal Impact and Limitations.

We address the problem of unsupervised learning in hyperbolic space. We believe the proposed HACK should not raise any ethical considerations. We discuss current limitations below,

Applying to the Whole Dataset Currently, HACK is applied to each class separately. Thus, it would be interesting to apply HACK to all the classes at once without supervision. This is much more challenging since we need to differentiate between examples from different classes as well as the prototypical and semantic structure.

Exploring other Geometrical Structures We consider uniform packing in hyperbolic space to organize the images. It is also possible to extend HACK by specifying other geometrical structures to encourage the corresponding organization to emerge from the dataset.

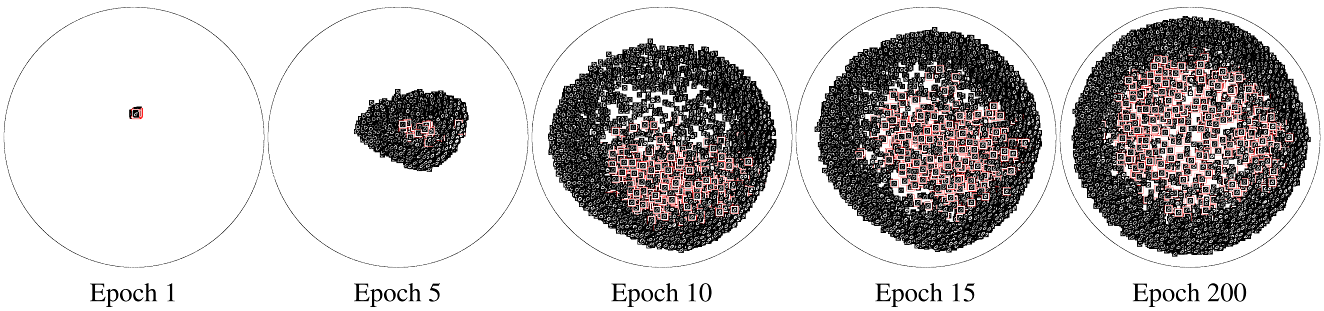


Figure 17. **Atypical images gradually move to the boundary of the Poincaré ball**. This shows that the representations learned by HACK capture prototypicality. Congealed images are in red boxes which are more typical. The network is trained with *both* the congealed images and original images.

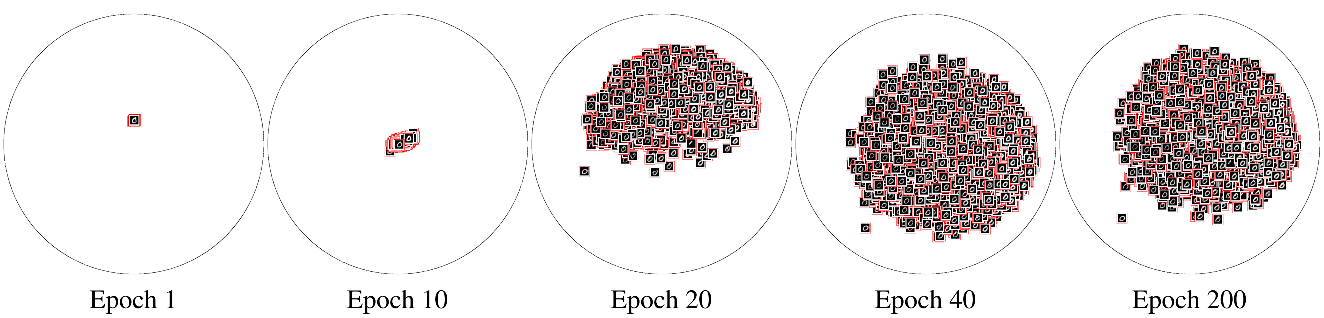


Figure 18. **The representations learned by HACK gradually capture prototypicality during the training process.** Congealed images are in red boxes which are more typical. We produce the features of the congealed images with the trained network in different epochs. The network is *only* trained with original images.