CONVERGE TO SURPRISE: EVOLUTIONARY SELF-SUPERVISED IMAGE CLUSTERING

Anonymous authors
Paper under double-blind review

ABSTRACT

A variety of self-supervised image clustering approaches are invented in the past years. However, all dominant approaches are *exploitative*: The direction of parameter updates is determined by known states (observed input samples and existing parameters). We propose an *explorative* self-supervised learning framework that steps out of this zone. We define a *surprise score* that measures how unlikely the model's output representation is, assuming that all pixels are i.i.d. random noise. Maximizing the surprise score forces the deep learning model to reject the random noise null hypothesis, or equivalently, to discover *non-randomness* from data. Also, we propose a fundamental assumption: a surprise score cannot, in general, be fully optimized by exploitative optimization approaches. Thus, we propose the *converge-to-surprise* scheme to optimize a model: an evolution-strategy (ES) outer loop, which maximizes the surprise score using the mutation-selection mechanism, paired with a periodic gradient-descent inner loop, which uses the surprising clusters already discovered by ES as surrogate targets. On simple image benchmarks, our framework trained from scratch achieves new state-of-the-art results in non-parametric self-supervised image clustering — the strictest deep-clustering setting, where the number of classes is unknown during training.

1 INTRODUCTION

Self-supervised image clustering aims at grouping unlabeled images into distinct, semantically meaningful categories without human intervention (Chen et al., 2020; He et al., 2022). This is achieved by unifying self-supervised representation learning and unsupervised clustering into a cohesive pipeline (Caron et al., 2018; 2021).

A long line of work has approached this problem with different efforts. Patch-arrangement methods predict structural corruptions: a jigsaw permutation (Noroozi & Favaro, 2016), an applied rotation (Gidaris et al., 2018), or the relative position of two crops (Doersch et al., 2015). Contrastive learning pulls together two augmented views of one image and pushes apart views of different images (Chen et al., 2020; He et al., 2020). Masked autoencoders reconstruct randomly masked pixels (He et al., 2022). Deep clustering alternates between training on pseudo-labels and re-clustering the output features (Caron et al., 2018; 2020). Self-distillation trains a student to match a teacher whose weights are an exponential moving average of the student's (Grill et al., 2020; Caron et al., 2021). Latent-space prediction, the joint-embedding-predictive-architecture (JEPA) family, predicts the embedding of one part of an image from that of another (Assran et al., 2023). Information-theoretic methods optimize differentiable surrogates for mutual information between views, or for de-correlation across feature dimensions (Zbontar et al., 2021; Bardes et al., 2021; Hjelm et al., 2018; Ji et al., 2019).

In all these approaches, impacts made by random perturbations (such as data augmentation (Perez & Wang, 2017) and dropout (Srivastava et al., 2014)) are only up to an incidental level: There is no mutation-selection mechanism (Bell, 2008) that compares the model's performances under different perturbations, and then updates the parameters based on the results. The loss, or in general the direction of parameter updates, is determined by the observed samples in the current step and the model parameters in the current stage. That is, all these approaches are *exploitative* (Hill, 1993).

In contrast, this paper introduces a self-supervised learning (SSL) framework that applies a mutation-selection mechanism to parameter updates, leading to an *explorative* (He & Wong, 2004) image clustering approach. Initially, we ask one question: *what is not random in an image?* That is, in the

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.

absence of prior knowledge, the Principle of Maximum Entropy (Guiasu & Shenitzer, 1985) dictates the most conservative assumption on the image distribution — pixels are i.i.d. random noise. We take this assumption as our null hypothesis $\mathcal{H}_0$, and define a *surprise score* that measures how unlikely the model's output representation would be under $\mathcal{H}_0$. The higher the surprise score, the less plausible $\mathcal{H}_0$ becomes. Thus, maximizing the surprise score forces the model to reject the null hypothesis $\mathcal{H}_0$. Equivalently, this makes the model extract meaningful information, or *non-randomness*, from images, which naturally prevents representation collapse and is used for clustering.

Then, by introducing formal definitions for exploitative and explorative SSL approaches, we propose a fundamental hypothesis of this paper: *a surprise score cannot, in general, be fully optimized by exploitative optimization approaches*. Although proving this hypothesis is beyond our scope, detailed analysis is provided.

Accordingly, we propose the *converge-to-surprise* scheme: An evolution-strategy (ES) (Salimans et al., 2017) outer loop, which maximizes the surprise score by mutation-selection mechanism, is paired with a periodic gradient-descent inner loop, which uses the surprising clusters already discovered by ES as surrogate targets. Experiments on simple image datasets (MNIST (Deng, 2012), FashionMNIST (Xiao et al., 2017) and USPS (Hull, 2002)) show that our scheme achieves new state-of-the-art results in non-parametric self-supervised image clustering — the strictest deep-clustering task, in which case the number of ground-truth classes is unknown to the model during training.

Here are our contributions:

1. We define a surprise score that measures how unlikely the output of a deep learning model should occur, under the random noise null hypothesis. Maximizing the surprise score makes the model discover non-randomness from data, which naturally prevents representation collapse. Then, we formally define whether an SSL approach is exploitative or explorative. We show that all dominant SSL approaches for image clustering are exploitative. Finally, we propose a fundamental hypothesis: a surprise score cannot, in general, be fully optimized by exploitative approaches.
2. We propose converge-to-surprise, an explorative optimization scheme that combines evolution strategy with gradient-descent training to maximize the surprise score.
3. On simple image benchmarks, models trained from scratch using our scheme achieve new state-of-the-art results in non-parametric self-supervised image clustering tasks — the strictest image clustering scenario. To the edge of our knowledge, our model is the first explorative SSL approach being trained from scratch for self-supervised image clustering.

The remainder of the paper is organized as follows: Section 2 surveys related work in self-supervised image clustering. Section 3 describes the converge-to-surprise framework with all details. Section 4 reports experimental results, and Section 5 concludes.

2 Related Work

We organize self-supervised image clustering methods by what the network outputs, and how that output becomes a cluster assignment at test time.

Embedding-based clustering: Most self-supervised image clustering methods produce a dense feature embedding in $\mathbb{R}^d$, which is further clustered or linearly probed at test time. These span contrastive methods (SimCLR (Chen et al., 2020), MoCo (He et al., 2020)), self-distillation (BYOL (Grill et al., 2020), DINO (Caron et al., 2021)), latent-space prediction (I-JEPA (Assran et al., 2023)), masked autoencoders (He et al., 2022), information-theoretic objectives (Barlow Twins (Zbontar et al., 2021), VICReg (Bardes et al., 2021), Deep InfoMax (Hjelm et al., 2018)), and pseudo-label clustering (DeepCluster (Caron et al., 2018), SwAV (Caron et al., 2020)). They differ in the training signal but agree on the output: a continuous embedding, not a hard cluster assignment.

These approaches first apply a deep encoder that maps images to a continuous embedding space. Then, a shallow decoder (usually a multi-layer perceptron (Popescu et al., 2009)) maps the embedding to the output representation. In test time, classification is made based on nearest-neighbor evaluation (Mucherino et al., 2009) or via fine-tuned linear projections using the embeddings (Caron et al., 2018). This is the most dominant yet mildest setting: The network is not required to produce hard representations of the image, only an embedding that captures semantic meaning.

Parametric hard clustering: A second family outputs a cluster index directly, mapping each input to one of K clusters with K equals to the number of true classes. At evaluation, clusters are matched to class labels by the Kuhn-Munkres (Hungarian) algorithm (Kuhn, 1955). DEC (Xie et al., 2016) sharpens a soft assignment over K centroids, and DAC (Chang et al., 2017) recasts clustering as pairwise same/different classification. SCAN (Van Gansbeke et al., 2020) decouples feature learning from cluster assignment by classifying images based on semantic nearest neighbors. CLUDI (Uziel et al., 2025) distills stochastic diffusion-based cluster assignments from a teacher into a student, on top of a pretrained DINO model (Caron et al., 2021). Although being a parametric approach, IIC (Ji et al., 2019) is the closest one to our framework, which maximizes softmax mutual information (Kraskov et al., 2004) between the cluster predictions of two augmented views.

Non-parametric hard clustering: The third family outputs a cluster index without knowing the number of true classes in advance. This is the strictest image clustering regime, and the setting for which we report results. Few enough approaches can be found in this family. DeepDPM (Ronen et al., 2022) changes K during training by split/merge operations inspired by Dirichlet-Process Mixture (DPM) models (Neal, 2000). UNSEEN (Leiber et al., 2024) applies existing deep-clustering backbones (DCN (Yang et al., 2017), DEC (Xie et al., 2016), DKM (Fard et al., 2020)), then starts from an upper bound $K_{\max}$ and lets unused clusters atrophy. Classical algorithms such as moVB (Hughes & Sudderth, 2013) and DBSCAN (Ester et al., 1996) operate on learned features. All these models serve as baselines in this paper.

3 Main Theory

In this section, we first introduce our complementary masking strategy, based on which the surprise score is calculated. Then, we formally define exploitative and explorative optimization approaches. Finally, we combine evolution strategy with gradient-descent to produce our explorative approach.

3.1 Complementary masking strategy

Suppose we have a distribution $\mathbf{P}$ producing images of shape (H, W, C). That is, $\mathbf{X} \sim \mathbf{P}$ with $\mathbf{X} \in \mathbb{R}^{H,W,C}$. In a self-supervised learning scenario, we have no prior knowledge about $\mathbf{P}$. According to the Principle of Maximum Entropy (Guiasu & Shenitzer, 1985), we have to assume that each pixel (h, w) in $\mathbf{X}$ is independent of every other; In other words, we have to assume that $\mathbf{P}$ is the maximum-entropy distribution over $\mathbb{R}^{H,W,C}$, producing totally random noise. This is our null hypothesis $\mathcal{H}_0$.

Then, given a sampled image $\mathbf{X} \sim \mathbf{P}$, we partition its $H \times W$ pixel grid into two disjoint subsets using a *chessboard pattern*. Let

$$\mathcal{I} = \{(h, w) : (h + w) \bmod 2 = 0\}, \quad \mathcal{J} = \{(h, w) : (h + w) \bmod 2 = 1\} \tag{1}$$

denote the white and black chessboard positions, respectively. Let $\mathbf{M}^{(\mathcal{I})}, \mathbf{M}^{(\mathcal{J})} \in \{0, 1\}^{H,W}$ be the corresponding binary masks, defined by $\mathbf{M}^{(\mathcal{I})}_{h,w} = \mathbf{1}[(h + w) \bmod 2 = 0]$ and $\mathbf{M}^{(\mathcal{J})} = \mathbf{1} - \mathbf{M}^{(\mathcal{I})}$, where $\mathbf{1}[\cdot]$ is the indicator function (McLean & Jernigan, 1991). Applying these masks pixel-wise to $\mathbf{X}$, we obtain two complementary masked views of the same image:

$$\mathbf{X}^{(i)} = \mathbf{X} \odot \mathbf{M}^{(\mathcal{I})}, \quad \mathbf{X}^{(j)} = \mathbf{X} \odot \mathbf{M}^{(\mathcal{J})}, \tag{2}$$

where $\odot$ denotes element-wise multiplication. The i-side view $\mathbf{X}^{(i)}$ retains only the pixels at white chessboard positions and zeros out the rest, while the j-side view $\mathbf{X}^{(j)}$ retains only those at black positions. Every pixel of $\mathbf{X}$ appears in exactly one of the two views, and the two views share no pixel in common. Figure 1 (a) shows our masking strategy on an MNIST image (Deng, 2012).

As mentioned, the null hypothesis $\mathcal{H}_0$ assumes pixels of $\mathbf{X}$ to be independent of one another. Since $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$ share no common pixel, they are themselves independent tensors and therefore share no mutual information (Batina et al., 2011). In addition, applying data augmentation (Perez & Wang, 2017) independently on $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$ will not break this zero-mutual-information statement. We use $\tilde{\mathbf{X}}^{(i)}$ and $\tilde{\mathbf{X}}^{(j)}$ to denote the augmented views. Therefore, we have $\mathcal{H}_0 \Longrightarrow I\left(\tilde{\mathbf{X}}^{(i)}\,;\,\tilde{\mathbf{X}}^{(j)}\right) = 0$, where $I(\cdot\,;\,\cdot)$ denotes the mutual-information functional.

If $I\left(\tilde{\mathbf{X}}^{(i)}\,;\,\tilde{\mathbf{X}}^{(j)}\right) = 0$ is rejected, then $\mathcal{H}_0$ cannot be true. This will be our approach to reject $\mathcal{H}_0$.

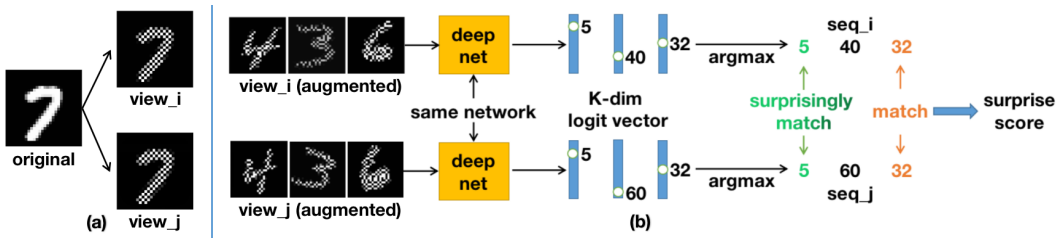


Figure 1: **(a) Chessboard masking**: i-side can only view pixels in white chessboard positions; j-side can only view pixels in black chessboard positions. The two views share no pixel in common. **(b) Overview of our pipeline**: The deep network is implemented on complementary masked and independently augmented views. Then, argmax is obtained from each output logit vector. Finally, surprise score sums across over-matching clusters.

3.2 Cluster co-occurrence as a surprise score

Again, given an image $\mathbf{X} \sim \mathbf{P}$, we obtain the two complementary masked and independently augmented views $\tilde{\mathbf{X}}^{(i)}$ and $\tilde{\mathbf{X}}^{(j)}$. Then, a deep learning model $f_\theta : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^K$ assigns each view to one of K candidate clusters, according to the argmax dimension in its output logits:

$$\hat{y}^{(i)} = \arg\max_k \left[f_\theta(\tilde{\mathbf{X}}^{(i)})\right]_k, \quad \hat{y}^{(j)} = \arg\max_k \left[f_\theta(\tilde{\mathbf{X}}^{(j)})\right]_k. \tag{3}$$

Both $\hat{y}^{(i)}$ and $\hat{y}^{(j)}$ take values in $\{0, 1, \ldots, K-1\}$. Because each $\hat{y}^{(\cdot)}$ is a deterministic function of the corresponding view, the data-processing inequality (Beaudry & Renner, 2011) gives $I(\hat{y}^{(i)}; \hat{y}^{(j)}) \le I\left(\tilde{\mathbf{X}}^{(i)}; \tilde{\mathbf{X}}^{(j)}\right) = 0$ under $\mathcal{H}_0$. That says, the cluster pseudo labels from i-side and j-side are statistically independent.

Then, given N images $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$ sampled i.i.d. from $\mathbf{P}$, we obtain their complementary masked and independently augmented views $\{\tilde{\mathbf{X}}_1^{(i)}, \ldots, \tilde{\mathbf{X}}_N^{(i)}\}$ and $\{\tilde{\mathbf{X}}_1^{(j)}, \ldots, \tilde{\mathbf{X}}_N^{(j)}\}$, respectively. We implement our model f_θ on each view and obtain the predicted cluster, leading to two integer sequences of length N. That is, $\mathrm{seq}^{(i)} = \left(\hat{y}_1^{(i)}, \ldots, \hat{y}_N^{(i)}\right)$ and $\mathrm{seq}^{(j)} = \left(\hat{y}_1^{(j)}, \ldots, \hat{y}_N^{(j)}\right)$.

In fact, we implicitly assume that the distributions of nearby pixels in an original image $\mathbf{X}$ are almost identical, although pixels are assumed to be i.i.d. under $\mathcal{H}_0$. Hence, after chessboard masking, $\mathbf{X}^{(i)}$ and $\mathbf{X}^{(j)}$ will have identical distribution. After independent augmentation with the same hyper-parameters, $\tilde{\mathbf{X}}^{(i)}$ and $\tilde{\mathbf{X}}^{(j)}$ will also have identical distribution. Thus, although statistically independent, $\mathrm{seq}^{(i)}$ and $\mathrm{seq}^{(j)}$ will have identical distribution as well under $\mathcal{H}_0$.

For each cluster $k \in \{0, 1, \ldots, K-1\}$, let $n_k^{(i)}$ be the number of times k appears in $\mathrm{seq}^{(i)}$, and define $n_k^{(j)}$ analogously. Then, the empirical marginal probability of clustering a view (either i-side or j-side) to cluster k is $p_k = (n_k^{(i)} + n_k^{(j)}) \,/\, (2N)$.

Again, since $\mathrm{seq}^{(i)}$ and $\mathrm{seq}^{(j)}$ are independent under $\mathcal{H}_0$, the null probability that both views of one image are predicted to the same cluster k (i.e. $\hat{y}_n^{(i)} = \hat{y}_n^{(j)} = k$ at any index n) is $q_k = p_k \cdot p_k = p_k^2$. Hence, under $\mathcal{H}_0$, the expected number of view matching at cluster k over the whole batch is $N \cdot q_k$.

On the other hand, we denote the observed number of view matching at cluster k as t_k, which is the actual count of indices $n \in \{1, \ldots, N\}$ for which $\hat{y}_n^{(i)} = \hat{y}_n^{(j)} = k$. Accordingly, the observed view-matching probability is $\hat{q}_k = t_k / N$.

Considering cluster k in isolation, t_k is then the number of successes in N independent Bernoulli trials, each with success probability q_k (Feller, 1968). That is, $t_k \sim \mathrm{Binomial}(N, q_k)$ under $\mathcal{H}_0$.

The K output dimensions are of course coupled: each view is predicted to exactly one cluster. So, the counts $t_1, \ldots, t_K$ are correlated. But this coupling has only a very weak effect on the distribution of any individual t_k. Treating each t_k on its own as binomial is the form we will use throughout.

Then, we apply the binomial distribution formula (Feller, 1991):

$$\mathbb{P}_{\mathcal{H}_0}[t_k \text{ or more matches at } k] = \sum_{s=t_k}^{N} \binom{N}{s} q_k^s \, (1-q_k)^{N-s}. \tag{4}$$

The sum on the right is exact but awkward to compute. A standard Chernoff upper bound on the binomial upper tail (Cover, 1999) turns it into the compact inequality

$$\mathbb{P}_{\mathcal{H}_0}[t_k \text{ or more matches at } k] \le \exp\big(-N \cdot D(\hat{q}_k \,\|\, q_k)\big), \tag{5}$$

where $D(\hat{q}_k \,\|\, q_k)$ is the binary Kullback–Leibler divergence (Shlens, 2014) between the observed view-matching probability $\hat{q}_k = t_k/N$ and the null view-matching probability $q_k = p_k^2$:

$$D(\hat{q}_k \,\|\, q_k) \;=\; \hat{q}_k \log \frac{\hat{q}_k}{q_k} \;+\; (1-\hat{q}_k) \log \frac{1-\hat{q}_k}{1-q_k}. \tag{6}$$

After taking logarithm, formula 5 becomes

$$-\log(\mathbb{P}_{\mathcal{H}_0}[t_k \text{ or more matches at } k]) \;\ge\; N \, D(\hat{q}_k \,\|\, q_k), \tag{7}$$

where the left side is the amount of information, or *level of surprise*, obtained by observing the view-matching result at cluster k (Shannon, 1948). The larger $N \cdot D(\hat{q}_k \,\|\, q_k)$ is, the more surprising and thus unlikely the result will be under $\mathcal{H}_0$. We call $D(\hat{q}_k \,\|\, q_k)$ the *surprise score at cluster* k.

Formula 7 shows the probability of observing t_k or more view matching at cluster k under the null hypothesis $\mathcal{H}_0$. So, we will ignore the surprise score if $\hat{q}_k < q_k$ for any cluster k (under-matching). Then, summing across all over-matching clusters, we define the final *surprise score*

$$\mathcal{S}(\theta) \;=\; \sum\nolimits_{k \,:\, \hat{q}_k > q_k} D(\hat{q}_k \,\|\, q_k)\,. \tag{8}$$

$\mathcal{S}(\theta)$ measures in total how strongly the view-clustering results reject $\mathcal{H}_0$, or how surprising the view-matching results look like. To achieve a large $\mathcal{S}(\theta)$, the model needs to discover reliable features shared by both views, which are essentially the meaningful information, or *non-randomness*, in data distribution $\mathbf{P}$. Hence, the optimization goal of our model is to maximize $\mathcal{S}(\theta)$. We call this framework *converge-to-surprise*. Figure 1 (b) briefly illustrates the pipeline of our framework.

If the argmax of all logit vectors collapse to one fixed dimension, then it is easy to see that $\mathcal{S}(\theta) = 0$. In contrast, if the pixels are really random noise, we can also have $\mathcal{S}(\theta) \approx 0$ since the two views will be independent. Hence, maximizing $\mathcal{S}(\theta)$ naturally prevents representation collapse (Li et al., 2022). Also, the argmax operation makes $\mathcal{S}(\theta)$ irrelative to K in its formula, enabling non-parametric clustering. In Appendix B, we show the connection between our surprise score $\mathcal{S}(\theta)$ and the IIC model training target (Ji et al., 2019).

However, optimizing $\mathcal{S}(\theta)$ is not straightforward. We first provide the following definition:

Definition 1. *Let $\mathbf{P}$ be a data distribution over $\mathbb{R}^{H\times W\times C}$. Let $\Theta = \{\theta_i\}_{i=1}^{I}$ be a set of parameters, where each θ_i is either network parameters (such as in a main model $f_{\theta_i} : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^K$ or in a latent projector $f_{\theta_i} : \mathbb{R}^{K_i^1} \to \mathbb{R}^{K_i^2}$) or supportive parameters (such as codebook vectors (Hu et al., 2023) or memory queue (He et al., 2020)). Let $\mathcal{B} = \{\mathbf{X}_n\}_{n=1}^{N}$ be a batch of samples with $\mathbf{X}_n \sim \mathbf{P}$. Let $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^{N} \to \{\tilde{\mathbf{X}}_{n'}\}_{n'=1}^{N'}$ be the random perturbation applied to $\mathcal{B}$ (such as masking or augmentation), and let ϵ be the random perturbation applied to Θ (such as dropout). Finally, let Φ be a self-supervised learning (SSL) approach, and let $\phi(\Theta, \mathcal{B}) \in \mathbb{R}$ be its fitness score.*

*We say that Φ is **exploitative**, if there exists a map $\Omega(\phi, \Theta, \mathcal{B}, \mathcal{E})$ taking values in the parameter-update space, such that the actual parameter updates can be written as*

$$\delta\Theta \;=\; \Omega(\phi, \Theta, \mathcal{B}, \mathcal{E}) \;+\; \epsilon, \tag{9}$$

where both $\mathcal{E}$ and ϵ are applied directly without performance-based selection. That is, ϕ is never evaluated at multiple $\Theta + \epsilon$ settings or multiple $\mathcal{E}(\mathcal{B})$ settings in order to decide $\delta\Theta$.

*Otherwise (if such conditions on Ω, $\mathcal{E}$ and ϵ cannot be satisfied), we say that Φ is **explorative**.*

Intuitively, an exploitative approach determines its future by its past, whereas an explorative approach refuses to be confined by its past and embraces any change that benefits its learning goal.

From this perspective, explorative SSL approaches are more general. In Appendix A, we show that: (i) **all dominant self-supervised image clustering approaches are exploitative;** (ii) DeepDPM (Ronen et al., 2022) and UNSEEN (Leiber et al., 2024), the two baseline models in the next section, are explorative and exploitative, respectively; (iii) if we expand our scope, then generative adversarial networks (GAN) (Goodfellow et al., 2020) and diffusion models (Ho et al., 2020; Peebles & Xie, 2023) are also exploitative, while reinforcement learning (Sutton et al., 1998) is explorative.

Now, we propose another fundamental hypothesis in this paper:

Hypothesis 1. *Let $\mathbf{P}$, Θ, $\mathcal{B}$, $\mathcal{E}$ and ϵ be the data distribution, set of model parameters, set of samples and perturbations defined as in above. Let $\mathcal{S}(\Theta, \mathcal{B}, \mathcal{E}, \epsilon) \in \mathbb{R}$ be a surprise score measuring, in any manner, how non-random the model's output representation is. Let Φ be an SSL approach.*

If Φ is exploitative, then $\mathcal{S}(\Theta, \mathcal{B}, \mathcal{E}, \epsilon)$ cannot, in general, be fully optimized by Φ.

In Appendix C, we transform $\mathcal{S}(\theta)$ into two differentiable formats and train the model by gradient descent, so that both are exploitative approaches. Both attempts perform less satisfying than our scheme. Although not a proof, such results are experimental evidences supporting Hypothesis 1.

Intuitively, a surprise score increases both alignment and uniformity (Wang & Isola, 2020; Mikulasch & Zenke, 2026) at the same time. More discussions are provided in Appendix D.

3.3 Optimization

Evolution strategy. We maximize the surprise score $\mathcal{S}(\theta)$ with the evolution strategy (ES) described in (Salimans et al., 2017). That is, we treat $\mathcal{S}(\theta)$ as the fitness score of a black-box optimization problem over the model's flat parameter vector $\theta \in \mathbb{R}^D$.

At each ES step, we sample a population of m perturbed parameter vectors around the current θ. To be specific, we split the population into $m/2$ mirrored pairs: For each pair, we draw $\epsilon_i \sim \mathcal{N}(0, \sigma^2 I_D)$, which forms two models with parameter $\theta + \epsilon_i$ and $\theta - \epsilon_i$, respectively. Here, $\mathcal{N}(0, \sigma^2 I_D)$ is the D-dim Gaussian distribution with variance σ (MacKay et al., 1998).

Given N images $\{\mathbf{X}_n\}_{n=1}^N \sim \mathbf{P}$, we obtain the complementary masked and independently augmented views $\{\tilde{\mathbf{X}}_n^{(i)}\}_{n=1}^N$ and $\{\tilde{\mathbf{X}}_n^{(j)}\}_{n=1}^N$, which are shared across all models in all $m/2$ pairs. We implement both models in each pair to obtain the fitness scores $\mathcal{S}(\theta + \epsilon_i)$ and $\mathcal{S}(\theta - \epsilon_i)$ for $i = 1, \cdots, m/2$. Then, we rank-shape (Wierstra et al., 2014) all m scores into centered ranks $r_i \in [-\frac{1}{2}, \frac{1}{2}]$, and apply a weighted update to θ:

$$\theta \leftarrow (1 - \eta\lambda)\,\theta + \frac{\eta}{m\sigma} \sum\nolimits_{i=1}^{m} r_i\,\epsilon_i. \tag{10}$$

Here, η is the learning rate and λ is the weight decay rate. Note that each r_i evolves performance-based selection (ranking), which is directly multiplied by each random perturbation ϵ_i. Hence, we cannot find a map Ω in Definition 1 to represent formula 10. Thus, our approach is explorative.

Mirrored pairs, rank-shaping and weight-decay are commonly used tricks in deep evolution strategies Wierstra et al. (2014); Salimans et al. (2017). More discussions are in (Salimans et al., 2017).

Surrogate training using surprising results. ES on its own is sufficient to discover clusters with surprising view-matching results. But it gives equal credit to every output dimension k contributing to $\mathcal{S}(\theta)$, including tiny, noisy micro-clusters that share similar features with larger clusters.

So, we first apply T_0 epochs of pure ES optimization to warm-up, during which clusters with surprising view-matching results will emerge. Starting from epoch $T_0 + 1$, we apply $T_{1,a}$ epochs of gradient-descent training every $T_{1,b}$ epochs of ES optimization. They form a complete period with $T_{1,a} + T_{1,b}$ epochs. We implement multiple such periods.

At the beginning of a gradient-descent training epoch, we identify the set of *contributing positions* among the sampled N original images $\{\mathbf{X}_1, \ldots, \mathbf{X}_N\}$:

$$\mathcal{C} = \Big\{ n \in \{1, \ldots, N\} : \hat{y}_n^{(i)} = \hat{y}_n^{(j)} = k \text{ and } D(\hat{q}_k \,\|\, q_k) \geq \tau \Big\}, \tag{11}$$

where $\hat{y}_n^{(i)} = \arg\max_k [f_\theta(\mathbf{X}_n^{(i)})]_k$ is the predicted cluster of the un-augmented view $\mathbf{X}_n^{(i)}$ of the original image $\mathbf{X}_n$, $\hat{y}_n^{(j)}$ is obtained accordingly on the un-augmented view $\mathbf{X}_n^{(j)}$, $D(\hat{q}_k \,\|\, q_k)$ is the

surprise score at cluster k defined in formula 6, and τ is a pre-defined threshold on the per-cluster surprise score. To be specific, we regard cluster k as *surprising* if $D(\hat{q}_k \,\|\, q_k) \geq \tau$.

That says, n is a contributing position when both views of $\mathbf{X}_n$ are predicted to the same cluster, and the view-matching result of the predicted cluster is surprising enough under $\mathcal{H}_0$. We obtain $\mathcal{C}$ on un-augmented views to avoid unnecessary perturbations during view-matching observation.

Then, we group the original images at contributing positions by their predicted cluster:

$$\mathcal{C}_k = \{\mathbf{X}_n : n \in \mathcal{C} \text{ and } \hat{y}_n^{(i)} = k\}.$$

Given the per-cluster count $n_k = |\mathcal{C}_k|$, let M be the median of $\{n_k\}_{D(\hat{q}_k \,\|\, q_k) \geq \tau}$. Then, within each $\mathcal{C}_k$, we select $\min(n_k, M)$ images uniformly without replacement: large clusters are sub-sampled down to M images, whereas small clusters contribute all their images. Combining all selected images, we obtain a training set $\mathcal{B}$ with no single dominant cluster.

For each selected image $\mathbf{X}_n \in \mathcal{B}$, we use its predicted cluster $\hat{y}_n^{(i)}$ as the surrogate label, denoted as $\hat{y}_n$. We apply chessboard masking and independent augmentation again to obtain new $\tilde{\mathbf{X}}_n^{(i)}$ and $\tilde{\mathbf{X}}_n^{(j)}$ from $\mathbf{X}_n$. The model is trained by gradient descent via cross-entropy loss (Mao et al., 2023):

$$\mathcal{L}_{\text{ft}}(\theta) = \sum_{n \in \mathcal{B}} \Big[\mathrm{CE}\big(f_\theta(\tilde{\mathbf{X}}_n^{(i)}),\, \hat{y}_n\big) + \mathrm{CE}\big(f_\theta(\tilde{\mathbf{X}}_n^{(j)}),\, \hat{y}_n\big) \Big]. \tag{12}$$

Finally, algorithm 1 in Appendix E summarizes the full optimization procedure.

4 Experiments

We evaluate our converge-to-surprise scheme in non-parametric self-supervised image clustering, the strictest image clustering task in which the number of ground-truth classes is unknown in training. We report results on three simple benchmarks and compare against the leading non-parametric deep-clustering methods, DeepDPM (Ronen et al., 2022) and UNSEEN (Leiber et al., 2024).

4.1 Experimental setup

Model and data: Our model f_θ is a ResNet-9 (He et al., 2016) whose linear head returns $K = 64$ logits. A view's predicted cluster is the argmax over the ℓ_2-normalized logits. Since the model is never told the number of classes, K acts only as an upper bound, and the number of active clusters is discovered during training. The same architecture is used for all three benchmarks, each with ten ground-truth classes: **MNIST** (Deng, 2012), which contains 70,000 training images of handwritten digits at a 28×28 resolution; **Fashion-MNIST** (Xiao et al., 2017), which matches MNIST in size and resolution but depicts clothing items in grayscale (and hence more challenging than MNIST); and **USPS** (Hull, 2002), which contains 7,291 training images at a 16×16 resolution and a notable class imbalance. After the chessboard masking of Section 3.1, the two complementary views are augmented independently — random rotation, cropping and brightness/contrast jitter, plus a horizontal flip on Fashion-MNIST and an anisotropic zoom-out on USPS that decorrelates the aspect-ratio variation of its dominant digit-0 class. More details can be found in Appendix F.

Optimization and protocol: We maximize $\mathcal{S}(\theta)$ with an ES outer loop of population size $m = 32$, paired with a gradient-descent training inner loop on the discovered surprising clusters ($\tau = 0.005$ in Eq. 11). MNIST and Fashion-MNIST use a two-stage schedule: 2000 epochs of pure ES, then 4 gradient-descent epochs every 25 ES epochs until epoch 3000. USPS inserts an intermediate weak stage (2 gradient-descent epochs every 500 ES epochs for 4000 epochs) after 4000 pure ES epochs, followed by the same last 1000 epochs as in MNIST. This is because USPS has fewer samples, so models need more epochs to evolve. The surprise score is computed over batches of $N = 3000$ images (3650 for USPS, chosen so that each epoch is split into two nearly equal steps, producing almost the same statistics). We report the mean $\pm$ standard deviation over ten independent runs from fresh initializations. At test time, one masked view (without augmentation) of each image is passed through the network and assigned to its argmax cluster. Training one model takes roughly 5 hours on MNIST or Fashion-MNIST and 2 hours on USPS, using a single NVIDIA H200 GPU.

Metrics and baselines: We report clustering accuracy (ACC) under the Kuhn-Munkres (Hungarian) algorithm (Kuhn, 1955), Normalized Mutual Information (NMI) (Kvålseth, 2017) and the Adjusted

Table 1: Non-parametric clustering results (%, mean ± std) on MNIST, Fashion-MNIST, and USPS. For each metric (NMI, ARI, ACC) higher is better, and the best mean per column is in **bold**. Baselines are the values reported by (Ronen et al., 2022) (DBSCAN (Ester et al., 1996), moVB (Hughes & Sudderth, 2013), DPM Sampler and DeepDPM (Ronen et al., 2022); on pretrained-autoencoder features) and (Leiber et al., 2024) (UNSEEN+DCN (Yang et al., 2017), UNSEEN+DEC (Xie et al., 2016), UNSEEN+DKM (Fard et al., 2020); on an autoencoder backbone). Our results are over ten independent runs from fresh initializations.

	MNIST			Fashion-MNIST			USPS		
Method	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
DBSCAN	$92.0_{\pm 0.0}$	$86.0_{\pm 0.0}$	$89.0_{\pm 0.0}$	$63.0_{\pm 0.0}$	$32.0_{\pm 0.0}$	$39.0_{\pm 0.0}$	$72.0_{\pm 0.0}$	$46.0_{\pm 0.0}$	$57.0_{\pm 0.0}$
moVB	$93.0_{\pm 0.0}$	$94.0_{\pm 0.0}$	$97.0_{\pm 0.0}$	$66.0_{\pm 2.0}$	$47.0_{\pm 3.0}$	$55.0_{\pm 3.0}$	$87.0_{\pm 2.0}$	$86.0_{\pm 4.0}$	$90.0_{\pm 4.0}$
DPM Sampler	$92.0_{\pm 1.0}$	$91.0_{\pm 4.0}$	$93.0_{\pm 5.0}$	$67.0_{\pm 1.0}$	$49.0_{\pm 2.0}$	$59.0_{\pm 3.0}$	$87.0_{\pm 1.0}$	$82.0_{\pm 2.0}$	$83.0_{\pm 3.0}$
DeepDPM	$94.0_{\pm 0.0}$	$95.0_{\pm 0.0}$	$\mathbf{98.0_{\pm 0.0}}$	$\mathbf{68.0_{\pm 1.0}}$	$51.0_{\pm 2.0}$	$62.0_{\pm 3.0}$	$88.0_{\pm 0.0}$	$86.0_{\pm 1.0}$	$89.0_{\pm 2.0}$
UNSEEN+DCN	$87.8_{\pm 1.9}$	$83.7_{\pm 4.5}$	$86.0_{\pm 5.4}$	$63.7_{\pm 1.0}$	$46.9_{\pm 1.5}$	$57.3_{\pm 2.4}$	$76.5_{\pm 1.4}$	$69.8_{\pm 4.9}$	$77.5_{\pm 5.6}$
UNSEEN+DEC	$80.4_{\pm 1.8}$	$70.9_{\pm 3.5}$	$71.0_{\pm 4.4}$	$58.6_{\pm 2.6}$	$42.3_{\pm 3.4}$	$54.4_{\pm 3.8}$	$80.8_{\pm 0.9}$	$75.9_{\pm 2.4}$	$81.3_{\pm 2.4}$
UNSEEN+DKM	$84.2_{\pm 2.6}$	$78.5_{\pm 5.2}$	$83.0_{\pm 5.5}$	$62.2_{\pm 1.0}$	$43.9_{\pm 1.4}$	$53.1_{\pm 2.0}$	$73.3_{\pm 5.3}$	$62.6_{\pm 10.6}$	$71.5_{\pm 9.4}$
Ours	$\mathbf{95.6_{\pm 0.7}}$	$\mathbf{96.1_{\pm 1.0}}$	$97.9_{\pm 1.3}$	$64.0_{\pm 1.7}$	$\mathbf{51.9_{\pm 2.2}}$	$\mathbf{64.1_{\pm 1.8}}$	$\mathbf{89.4_{\pm 1.7}}$	$\mathbf{89.1_{\pm 3.0}}$	$\mathbf{93.4_{\pm 3.1}}$

Rand Index (ARI) (Santos & Embrechts, 2009) — all higher-is-better, defined as in (Ronen et al., 2022) — together with the inferred number of active clusters $\hat{K}$. Baselines are the values reported by (Ronen et al., 2022) and (Leiber et al., 2024). Again, more details are in Appendix F.

4.2 Results

Table 1 reports the comparison. Our framework achieves state-of-the-art performances on USPS across all three metrics. On MNIST, our model leads in NMI and ARI while almost matching DeepDPM in ACC (97.9% against 98.0%, though with a larger variance). On Fashion-MNIST, we obtain the best ACC and ARI scores, while remaining competitive in NMI. The largest improvement occurs on USPS, where our model improves ACC from DeepDPM's 89% to 93.4%, and lifts NMI/ARI by roughly 1.5% and 3%, respectively. On Fashion-MNIST, we reach 64.1% ACC and 51.9% ARI, both above the strongest baselines, whereas our NMI of 64.0% is second to DeepDPM's 68%. In addition, our model is trained from scratch, rather than from pre-trained features.

Crucially, these results are obtained without telling the model the number of true classes. As shown in Table 5 of Appendix G, the number of active clusters our method discovers stays close to the ground-truth value of ten: $10.1_{\pm 0.3}$ on MNIST, $11.2_{\pm 1.0}$ on Fashion-MNIST, and $10.2_{\pm 0.8}$ on USPS. These results are comparable to the values inferred by DeepDPM ($10.0_{\pm 0.0}$, $10.2_{\pm 0.8}$, $9.2_{\pm 0.4}$) and largely beat the values by other baselines. The remaining $K - \hat{K}$ output dimensions naturally die out during our training. Appendix H shows how the number of active clusters changes across epochs.

In addition, DeepDPM is explorative, but applies pre-trained autoencoder features. The autoencoder in UNSEEN is trained from scratch, but its 'dying cluster' mechanism is exploitative (further discussions are provided in Appendix A.7). To the edge of our knowledge, our model is the first explorative SSL approach being trained from scratch for self-supervised image clustering.

In summary, simply by maximizing the surprise score — without pre-trained models or prior knowledge — a deep network can naturally discover semantically meaningful hard representation from simple digit or garment images. Visualizations of the clustered images are exhibited in appendix I.

Models trained by our scheme perform less satisfying on more complicated datasets such as CIFAR10 (Krizhevsky et al., 2009). More details are provided in Appendix J.

4.3 Ablation studies

In this sub-section, we first replace the explorative optimizer (converge-to-surprise scheme) by two exploitative ones. Then, we ablate three major components of our approach: the independent two-

Table 2: Ablation studies (%, mean ± std; the full model over ten runs, each ablation over five runs). The first two rows replace the evolution strategy by gradient descent on a differentiable surrogate of the surprise score; each remaining row removes one component from the full model (GDT is short for gradient-descent training). The anisotropic zoom-out augmentation is USPS-specific ('–').

	MNIST			Fashion-MNIST			USPS		
Setting	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
conv.-to-surprise	$\mathbf{95.6}_{\pm 0.7}$	$\mathbf{96.1}_{\pm 1.0}$	$\mathbf{97.9}_{\pm 1.3}$	$64.0_{\pm 1.7}$	$\mathbf{51.9}_{\pm 2.2}$	$\mathbf{64.1}_{\pm 1.8}$	$\mathbf{89.4}_{\pm 1.7}$	$\mathbf{89.1}_{\pm 3.0}$	$\mathbf{93.4}_{\pm 3.1}$
plain-softmax	$91.3_{\pm 3.7}$	$81.0_{\pm 9.0}$	$82.8_{\pm 8.4}$	$60.6_{\pm 3.5}$	$46.5_{\pm 4.4}$	$58.5_{\pm 6.2}$	$80.4_{\pm 4.3}$	$70.6_{\pm 9.0}$	$72.1_{\pm 8.1}$
Gumbel-softmax	$93.3_{\pm 2.0}$	$87.6_{\pm 6.0}$	$88.2_{\pm 7.4}$	$\mathbf{64.3}_{\pm 2.3}$	$50.2_{\pm 3.9}$	$60.5_{\pm 4.2}$	$84.4_{\pm 5.3}$	$77.0_{\pm 9.7}$	$82.7_{\pm 8.3}$
no augmentation	$49.1_{\pm 2.5}$	$25.1_{\pm 1.9}$	$30.1_{\pm 1.5}$	$49.8_{\pm 0.6}$	$23.5_{\pm 1.1}$	$26.9_{\pm 1.7}$	$65.3_{\pm 3.0}$	$47.7_{\pm 5.1}$	$55.2_{\pm 5.9}$
no inner-loop GDT	$77.4_{\pm 3.0}$	$74.3_{\pm 5.3}$	$80.9_{\pm 6.1}$	$58.1_{\pm 2.1}$	$44.9_{\pm 2.2}$	$56.6_{\pm 1.5}$	$66.2_{\pm 5.1}$	$51.7_{\pm 9.2}$	$59.3_{\pm 9.3}$
no anisotropic aug.	–	–	–	–	–	–	$80.4_{\pm 2.8}$	$73.0_{\pm 4.8}$	$83.4_{\pm 3.3}$

view augmentation, the inner-loop gradient-descent training, and the anisotropic (zoom-out) augmentation used only on the USPS dataset. Table 2 reports the effect.

Exploitative approaches do not match evolution strategy. The surprise score can be made differentiable and then maximized by gradient descent. We replace the argmax of Eq. 3 by either a plain softmax head, or a Gumbel softmax head (see Appendix C for more details). Both approaches lead to differentiable learning targets, and hence are exploitative by Definition 1. We train both models using gradient descent, with every other setting unchanged. Table 2 shows that although both approaches produce non-trivial results, our converge-to-surprise scheme still surpasses them in eight out of nine columns. Although not a strict proof, such experimental results are highly supportive to Hypothesis 1, saying that exploitative approaches cannot fully optimize a surprise score. Again, more details regarding these two differentiable transformations are provided in Appendix C.

Augmentation is essential. Without augmentation, evolution strategy (ES) maximizes the surprise score by latching onto low-level features that the two views happen to share — stroke thickness, brightness, and overall shape — rather than the semantic class. Augmenting the two views independently breaks this shortcut: the two views now differ in brightness, contrast, scale (cropping) and orientation (rotation). So, only augmentation-invariant features can make the two views agree, which are more likely the true underlying topology of the digits or items.

Inner-loop gradient-descent training is important. After several thousand epochs of pure ES, the model consistently produces more clusters than the number of ground-truth classes. Gradient-descent training then consolidates discovered clusters: if two clusters hold similar image views, the larger one will provide stronger pseudo-label signals, which finally pulls across the views from the smaller one. Augmenting both views again before each training step further reinforces the clusters discovered by ES. The two-view agreement rate rises from 80–85% after pure ES to nearly 99%.

Anisotropic augmentation is necessary for USPS. The USPS dataset is class-imbalanced, and its digit-zero samples vary widely in aspect ratio (tall-and-thin versus short-and-wide). The single-axis-zoom-out randomly stretches or compresses each view along one axis, so the two views of an image differ in width and height. Aspect ratio is then no longer a salient feature shared by the two views. This forces the network to explore more essential, topology-related structures of digits.

5 Conclusion

In this paper, we introduce converge-to-surprise, a self-supervised learning framework aiming at discovering non-randomness from images. Our null hypothesis states that pixels are i.i.d. random noise. Then, we build two complementary masked views from one image, and define a surprise score to measure how strongly a deep learning model's view-clustering results reject the null hypothesis. Assuming that a surprise score cannot be fully optimized by exploitative SSL approaches, we optimize our model combining evolution strategy with gradient descent, leading to an explorative approach. Experiments show that our model trained from scratch attains state-of-the-art performances on simple image benchmarks like MNIST, Fashion-MNIST and USPS, under the strictest non-parametric self-supervised image clustering setting. To the edge of our knowledge, our model is the first explorative SSL approach being trained from scratch for self-supervised image clustering.

REFERENCES

Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

Neena Aloysius and M Geetha. A review on deep convolutional neural networks. In *2017 international conference on communication and signal processing (ICCSP)*, pp. 0588–0592. IEEE, 2017.

Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 15619–15629, 2023.

Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

L. Batina, B. Gierlichs, E. Prouff, M. Rivain, F.-X. Standaert, and Veyrat-Charvillon N. Web text corpus for natural language processing. *Journal of Cryptology*, 24(2):269–291, 2011.

Normand J Beaudry and Renato Renner. An intuitive proof of the data processing inequality. *arXiv preprint arXiv:1107.0740*, 2011.

ALD Beckers and Arnold WM Smeulders. Optimization of length measurements for isotropic distance transformations in three dimension. *CVGIP: Image Understanding*, 55(3):296–306, 1992.

Graham Bell. *Selection: the mechanism of evolution*. OUP Oxford, 2008.

Mathilde Caron, Piotr Bojanowski, Armand Joulin, and Matthijs Douze. Deep clustering for unsupervised learning of visual features. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 132–149, 2018.

Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in neural information processing systems*, 33:9912–9924, 2020.

Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Jason Chang and John W Fisher III. Parallel sampling of dp mixture models using sub-cluster splits. *Advances in Neural Information Processing Systems*, 26, 2013.

Jianlong Chang, Lingfeng Wang, Gaofeng Meng, Shiming Xiang, and Chunhong Pan. Deep adaptive image clustering. In *Proceedings of the IEEE international conference on computer vision*, pp. 5879–5887, 2017.

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

Thomas M Cover. *Elements of information theory*. John Wiley & Sons, 1999.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. *Advances in neural information processing systems*, 26, 2013.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39(1): 1–22, 1977.

L. Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE Signal Processing Magazine*, pp. 141–142, 2012.

Carl Doersch, Abhinav Gupta, and Alexei A Efros. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, pp. 1422–1430, 2015.

Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

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.

Maziar Moradi Fard, Thibaut Thonet, and Eric Gaussier. Deep k-means: Jointly clustering with k-means and learning representations. *Pattern Recognition Letters*, 138:185–192, 2020.

W Feller. An introduction to probability theory and its applications. *Third edition. New York: Wiley. Theorem is in section VI.3*, 1968.

William Feller. *An introduction to probability theory and its applications, Volume 2*, volume 2. John Wiley & Sons, 1991.

Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Z. Ghahramani, M. Welling, C. Cortes, N. Lawrence, and K.Q. Weinberger (eds.), *Advances in Neural Information Processing Systems*, volume 27, pp. 2672–2680. Curran Associates, Inc., 2014. URL `https://nips.cc`.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63(11):139–144, 2020.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020.

S. Guiasu and A. Shenitzer. The principle of maximum entropy. *The Mathematical Intelligencer. doi.org/10.1007/BF03023004*, 7:42–48, 1985.

W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. 1970.

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*, pp. 770–778, 2016.

Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9729–9738, 2020.

Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 16000–16009, 2022.

Zi-Lin He and Poh-Kam Wong. Exploration vs. exploitation: An empirical test of the ambidexterity hypothesis. *Organization science*, 15(4):481–494, 2004.

John Lawrence Hill. Exploitation. *Cornell L. Rev.*, 79:631, 1993.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Qiyu Hu, Guangyi Zhang, Zhijin Qin, Yunlong Cai, Guanding Yu, and Geoffrey Ye Li. Robust semantic communications with masked vq-vae enabled codebook. *IEEE Transactions on Wireless Communications*, 22(12):8707–8722, 2023.

Michael C Hughes and Erik Sudderth. Memoized online variational inference for dirichlet process mixture models. *Advances in neural information processing systems*, 26, 2013.

Jonathan J. Hull. A database for handwritten text recognition research. *IEEE Transactions on pattern analysis and machine intelligence*, 16(5):550–554, 2002.

Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Tony Jebara. *Machine learning: discriminative and generative*. Springer Science & Business Media, 2012.

Xu Ji, Joao F Henriques, and Andrea Vedaldi. Invariant information clustering for unsupervised image classification and segmentation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9865–9874, 2019.

D. P. Kingma and J. L. Ba. Adam: A method for stochastic optimization. *arXiv preprint, arXiv:1412.6980*, 2014.

Alexander Kraskov, Harald Stögbauer, and Peter Grassberger. Estimating mutual information. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 69(6):066138, 2004.

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.

Tarald O Kvålseth. On normalized mutual information: measure derivations and properties. *Entropy*, 19(11):631, 2017.

Nikolay Kyurkchiev and Svetoslav Markov. Sigmoid functions: some approximation and modelling aspects. *LAP LAMBERT Academic Publishing, Saarbrucken*, 4, 2015.

Yann LeCun et al. A path towards autonomous machine intelligence version 0.9. 2, 2022-06-27. *Open Review*, 62(1):1–62, 2022.

Collin Leiber, Niklas Strauß, Matthias Schubert, and Thomas Seidl. Dying clusters is all you need-deep clustering with an unknown number of clusters. In *2024 IEEE International Conference on Data Mining Workshops (ICDMW)*, pp. 726–733. IEEE, 2024.

Alexander C Li, Alexei A Efros, and Deepak Pathak. Understanding collapse in non-contrastive siamese representation learning. In *European conference on computer vision*, pp. 490–505. Springer, 2022.

Stuart Lloyd. Least squares quantization in pcm. *IEEE transactions on information theory*, 28(2): 129–137, 1982.

David JC MacKay et al. Introduction to gaussian processes. *NATO ASI series F computer and systems sciences*, 168:133–166, 1998.

Anqi Mao, Mehryar Mohri, and Yutao Zhong. Cross-entropy loss functions: Theoretical analysis and applications. In *International conference on Machine learning*, pp. 23803–23828. pmlr, 2023.

Filipe J Marques, Carlos A Coelho, and Miguel De Carvalho. On the distribution of linear combinations of independent gumbel random variables. *Statistics and Computing*, 25(3):683–701, 2015.

GF McLean and ME Jernigan. Indicator functions for adaptive image processing. *Journal of the Optical Society of America A*, 8(1):141–156, 1991.

Fabian A Mikulasch and Friedemann Zenke. Understanding self-supervised learning via latent distribution matching. *arXiv preprint arXiv:2605.03517*, 2026.

Daniel Morales-Brotons, Thijs Vogels, and Hadrien Hendrikx. Exponential moving average of weights in deep learning: Dynamics and benefits. *arXiv preprint arXiv:2411.18704*, 2024.

Antonio Mucherino, Petraq J Papajorgji, and Panos M Pardalos. K-nearest neighbor classification. In *Data mining in agriculture*, pp. 83–106. Springer, 2009.

Radford M Neal. Markov chain sampling methods for dirichlet process mixture models. *Journal of computational and graphical statistics*, 9(2):249–265, 2000.

Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pp. 69–84. Springer, 2016.

Maxime Oquab, Timothée Darcet, Théo Moutakanni, Huy Vo, Marc Szafraniec, Vasil Khalidov, Pierre Fernandez, Daniel Haziza, Francisco Massa, Alaaeldin El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4172–4182. IEEE, 2023.

Luis Perez and Jason Wang. The effectiveness of data augmentation in image classification using deep learning. *arXiv preprint arXiv:1712.04621*, 2017.

Marius-Constantin Popescu, Valentina E Balas, Liliana Perescu-Popescu, and Nikos Mastorakis. Multilayer perceptron and neural networks. *WSEAS transactions on circuits and systems*, 8(7): 579–588, 2009.

Meitar Ronen, Shahaf E Finder, and Oren Freifeld. Deepdpm: Deep clustering with an unknown number of clusters. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 9861–9870, 2022.

Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

Jorge M Santos and Mark Embrechts. On the use of the adjusted rand index as a metric for evaluating supervised classification. In *International conference on artificial neural networks*, pp. 175–184. Springer, 2009.

Claude Elwood Shannon. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423, 1948.

Jonathon Shlens. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014.

Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):60, 2019.

Oriane Siméoni, Huy V Vo, Maximilian Seitzer, Federico Baldassarre, Maxime Oquab, Cijo Jose, Vasil Khalidov, Marc Szafraniec, Seungeun Yi, Michaël Ramamonjisoa, et al. Dinov3. *arXiv preprint arXiv:2508.10104*, 2025.

Jonathan So. Estimating the normal-inverse-wishart distribution. *arXiv preprint arXiv:2405.16088*, 2024.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014.

Richard S Sutton, Andrew G Barto, and Andrew Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

Yee Whye Teh et al. Dirichlet process. *Encyclopedia of machine learning*, 1063:280–287, 2010.

Roy Uziel, Irit Chelly, Oren Freifeld, and Ari Pakman. Clustering via self-supervised diffusion. *arXiv preprint arXiv:2507.04283*, 2025.

Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

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.

Tongzhou Wang and Phillip Isola. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International conference on machine learning*, pp. 9929–9939. PMLR, 2020.

Christopher JCH Watkins and Peter Dayan. Q-learning. *Machine learning*, 8(3):279–292, 1992.

Daan Wierstra, Tom Schaul, Tobias Glasmachers, Yi Sun, Jan Peters, and Jürgen Schmidhuber. Natural evolution strategies. *The Journal of Machine Learning Research*, 15(1):949–980, 2014.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

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

Bo Yang, Xiao Fu, Nicholas D Sidiropoulos, and Mingyi Hong. Towards k-means-friendly spaces: Simultaneous deep learning and clustering. In *international conference on machine learning*, pp. 3861–3870. PMLR, 2017.

Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International conference on machine learning*, pp. 12310–12320. PMLR, 2021.

A Evaluating existing self-supervised learning approaches for exploitative and explorative

In this section, we apply Definition 1 to the dominant families of self-supervised learning. In each case, we explicitly define Ω. For every gradient-based approach below, ϕ is the training loss up to a sign, and Ω is one gradient step on it.

Note that in Definition 1, Θ denotes the complete current state of one SSL approach — including network parameters, codebook vectors, memory queue, optimizer momentum, etc. This is also why $\Theta = \{\theta_i\}_{i=1}^{I}$ is defined as a set of parameters rather than a single vector. With that reading, none of the approaches below has parameters not covered by Θ.

A.1 SELF-DISTILLATION

We first evaluate the self-distillation family, which includes BYOL (Grill et al., 2020) and the DINO family: DINO (Caron et al., 2021), DINOv2 (Oquab et al., 2023) and DINOv3 (Siméoni et al., 2025). Here, the state consists of two networks, a student and a teacher:

$$\Theta \;=\; \{\theta_s,\; \theta_t,\; c\}\,, \tag{13}$$

where θ_s and θ_t are the student and teacher network parameters, and c is the centering vector (not a constant) used by the DINO family (absent in BYOL, which uses a predictor head instead).

The input perturbation. From each image, several views are extracted at different scales — two global crops and a number of smaller local crops, together with photometric jitter. This is precisely the random perturbation $\mathcal{E}$ in Definition 1 being applied to the batch $\mathcal{B}$, and it is the reason that $\mathcal{E}$ is allowed to change the number of samples, $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^{N} \to \{\tilde{\mathbf{X}}_{n'}\}_{n'=1}^{N'}$. We use $\mathcal{V} = \mathcal{E}(\mathcal{B})$ to denote the resulting set of views and use $\mathcal{V}_g \subset \mathcal{V}$ to denote the global ones.

The training loss. The student is trained to match the teacher's output. In DINO, both networks end in a K-dimensional head, which is turned into a distribution by a softmax. The softmax is further sharpened by temperatures $\tau_s, \tau_t > 0$, with the teacher's output additionally centered by c:

$$\mathbf{P}_s(\tilde{\mathbf{X}}) = \mathrm{softmax}\left(\frac{f_{\theta_s}(\tilde{\mathbf{X}})}{\tau_s}\right), \qquad \mathbf{P}_t(\tilde{\mathbf{X}}) = \mathrm{softmax}\left(\frac{f_{\theta_t}(\tilde{\mathbf{X}}) - c}{\tau_t}\right). \tag{14}$$

The loss is the cross-entropy between the teacher's distribution on a global view and the student's distribution on every other view:

$$\mathcal{L}(\theta_s,\, \theta_t,\, c,\, \mathcal{E}(\mathcal{B})) \;=\; -\sum_{\tilde{\mathbf{X}}\in\mathcal{V}_g}\;\sum_{\substack{\tilde{\mathbf{X}}'\in\mathcal{V}\\ \tilde{\mathbf{X}}'\neq\tilde{\mathbf{X}}}}\;\sum_{k=1}^{K}\Big[\mathbf{P}_t(\tilde{\mathbf{X}})\Big]_k \log\Big[\mathbf{P}_s(\tilde{\mathbf{X}}')\Big]_k. \tag{15}$$

BYOL differs in the form of the target but not in its structure: a predictor q_{θ_s} is applied on the student branch and the loss is the squared distance between ℓ_2-normalized projections, $\left\|\overline{q_{\theta_s}(z_s)} - \overline{z_t}\right\|_2^2$, with z_s and z_t the student and teacher embeddings of two views of the same image. In both cases, the loss is a deterministic function of θ_s, θ_t, c, and the views $\mathcal{E}(\mathcal{B})$ — and nothing else.

The updates. The student takes one gradient step. The teacher is not trained at all, but is an Exponential Moving Average (EMA) (Morales-Brotons et al., 2024) of the student, with momentum $\lambda \in (0, 1)$. The centering vector is an EMA of the teacher's own output over the global views, with momentum $\mu \in (0, 1)$:

$$\delta\theta_s = -\eta\, \nabla_{\theta_s}\mathcal{L}(\theta_s;\, \theta_t,\, c,\, \mathcal{E}(\mathcal{B}))\,, \tag{16}$$

$$\delta\theta_t = (1-\lambda)\,(\theta_s - \theta_t)\,, \tag{17}$$

$$\delta c = (1-\mu)\left(\frac{1}{|\mathcal{V}_g|}\sum\nolimits_{\tilde{\mathbf{X}}\in\mathcal{V}_g} f_{\theta_t}(\tilde{\mathbf{X}}) \;-\; c\right). \tag{18}$$

Reading Eq. 16–18, every component of $\delta\Theta$ is a function of the current state $\Theta = \{\theta_s, \theta_t, c\}$, the batch $\mathcal{B}$, and the perturbation $\mathcal{E}$. Also, it is easy to see that the fitness score in Definition 1 is just the final loss, $\phi = \mathcal{L}(\theta_s;\, \theta_t,\, c,\, \mathcal{E}(\mathcal{B}))$. Collecting them, we have

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) \;=\; \Big(\delta\theta_s,\;\; \delta\theta_t,\;\; \delta c\Big), \qquad \delta\Theta \;=\; \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon, \tag{19}$$

where ϵ collects the parameter-level noise such as dropout, and is applied directly. Depending on different training settings of different approaches, dropout may or may not be actually applied. Without loss of generality, we always keep the term ϵ in this section.

The map Ω required by Definition 1 therefore exists, and the loss is evaluated once — at the single setting (θ_s, θ_t, c) and on the single view set $\mathcal{E}(\mathcal{B})$. No candidate teacher, candidate student or candidate view set is ever scored and then selected against another. Hence, **self-distillation is exploitative**.

It is worth being explicit about why the teacher does not make the approach explorative. The teacher is a second network, and one might read Eq. 17 as a form of selection between two candidates. It is not: Eq. 17 is a fixed formula being applied unconditionally, with a constant decay factor λ. The teacher is never compared against the student on ϕ in order to keep the better one.

A.2 CONTRASTIVE LEARNING

We now turn to contrastive learning, taking SimCLR (Chen et al., 2020) as the example. Here, we have $\Theta = \{\theta\}$, with θ to be a single encoder combined with a projection head.

The input perturbation. Each image in the batch is augmented into two views. That is, $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^{N} \to \{\tilde{\mathbf{X}}_{n'}\}_{n'=1}^{2N}$. The two views of image n form a positive pair and all other views act as negatives. Again, this is exactly the perturbation described in Definition 1.

The training loss. Each view is mapped to a projection $\mathbf{z}_{n'} = g_\theta\Big(f_\theta(\tilde{\mathbf{X}}_{n'})\Big)$, and views in each pair are compared by cosine similarity, $\mathrm{sim}(\mathbf{u}, \mathbf{v}) = \mathbf{u}^\top \mathbf{v} / (\|\mathbf{u}\|\|\mathbf{v}\|)$. With temperature $\tau > 0$, the loss for a positive pair (a, b) and the total loss are respectively

$$\ell_{a,b} \;=\; -\log \frac{\exp(\mathrm{sim}(\mathbf{z}_a, \mathbf{z}_b)/\tau)}{\sum_{m \neq a} \exp(\mathrm{sim}(\mathbf{z}_a, \mathbf{z}_m)/\tau)}, \qquad \mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B})) = \frac{1}{2N} \sum_{(a,b)} \left(\ell_{a,b} + \ell_{b,a}\right). \tag{20}$$

Here, the numerator in $\ell_{a,b}$ pulls the two views of one image together and the denominator pushes every other view away. In $\mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B}))$, the sum runs over the N positive pairs.

The update. Every projection $\mathbf{z}_{n'}$ in Eq. 20 is determined by the current parameters θ and the current views $\mathcal{E}(\mathcal{B})$. The loss is a function of the pairwise similarities among those projections, so it too is determined by θ, $\mathcal{B}$ and $\mathcal{E}$. Again, the fitness score in Definition 1 is the final loss, $\phi = \mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B}))$. This will be true throughout this appendix section, unless with specify otherwise.

Finally, the update is a single gradient step,

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) = -\eta\, \nabla_\theta \mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B})), \qquad \delta\Theta = \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon, \tag{21}$$

where ϵ is any potential perturbation. Thus, conditions in Definition 1 are satisfied again, with $\mathcal{E}$ and ϵ being applied directly without performance-based selection. Hence, **contrastive learning is exploitative**. MoCo (He et al., 2020) adds a momentum encoder and a queue of stored negatives, which simply enlarges the parameter set Θ without introducing any performance-based selection.

A.3 LATENT-SPACE PREDICTION

We now evaluate the joint-embedding predictive architecture (JEPA) family (LeCun et al., 2022), taking I-JEPA (Assran et al., 2023) as the example; V-JEPA (Bardes et al., 2024) differs only in that the blocks are drawn from a video clip rather than from a single image. Unlike the two families above, no photometric view augmentation is required here: $\mathcal{E}$ is masking alone.

The parameter set consists of three networks,

$$\Theta \;=\; \{\theta_c,\, \theta_t,\, \theta_p\}, \tag{22}$$

where θ_c is the context encoder, θ_t is the target encoder, and θ_p is the predictor. Here, θ_p maps a context representation — conditioned on the position of a queried block — to the predicted representation of that block.

The input perturbation. Each image $\mathbf{X}_n$ is cut into a grid of patches. From that grid, M target blocks $B_{n,1}, \ldots, B_{n,M}$ are sampled with random scale and aspect ratio. Then, one context block A_n is sampled and has every patch overlapping any target block removed. Writing $\mathcal{E}$ for this block sampling, $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^{N} \to \{(A_n,\, B_{n,1}, \ldots, B_{n,M})\}_{n=1}^{N}$, we recover exactly the perturbation in Definition 1. We use $\mathbf{X}_n[A_n]$ to represent the block in its corresponding image.

The training loss. The target encoder is applied to the full image and produces one representation per patch, where the target for patch j is $[f_{\theta_t}(\mathbf{X}_n)]_j$. The predictor is applied to the context representation and asked to reproduce that target, in latent space, for every patch of every target block:

$$\mathcal{L}(\theta_c, \theta_p;\, \theta_t,\, \mathcal{E}(\mathcal{B})) \;=\; \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{j \in B_{n,m}} \Big\| g_{\theta_p}(f_{\theta_c}(\mathbf{X}_n[A_n])\,,\, j) \;-\; [f_{\theta_t}(\mathbf{X}_n)]_j \Big\|_2^2, \tag{23}$$

with a stop-gradient on the target branch. Only the position index j is passed to the predictor, whereas the content of patch j is never passed to the predictor. Thus, the loss is a deterministic function of θ_c, θ_p, θ_t and $\mathcal{E}(\mathcal{B})$, and of nothing else.

The updates. The context encoder and the predictor take one gradient step. The target encoder is an EMA of the context encoder with momentum $\lambda \in (0, 1)$:

$$\delta\theta_c = -\eta\, \nabla_{\theta_c} \mathcal{L}, \qquad \delta\theta_p = -\eta\, \nabla_{\theta_p} \mathcal{L}, \qquad \delta\theta_t = (1-\lambda)\,(\theta_c - \theta_t)\,. \tag{24}$$

Collecting the three components, we have

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) \;=\; \Big(\, \delta\theta_c,\;\; \delta\theta_p,\;\; \delta\theta_t \,\Big), \qquad \delta\Theta \;=\; \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon. \tag{25}$$

The map Ω therefore exists, and ϕ is evaluated once — at the single setting $(\theta_c, \theta_t, \theta_p)$ and on the single sample batch $\mathcal{E}(\mathcal{B})$. As in Appendix A.1, the EMA target encoder never goes through performance-based selection. Hence, **latent-space prediction is exploitative**.

A.4 Deep clustering with pseudo-labels

We now turn to deep clustering approaches, taking DeepCluster (Caron et al., 2018) as the example. This is the family in which one would most expect Definition 1 to break, because its central step, k-means clustering, is neither a gradient step nor a differentiable operation.

The parameter set is

$$\Theta \;=\; \{\theta,\; W,\; C\}\,, \tag{26}$$

where θ is the backbone model parameters, W is a linear classification head over K pseudo-classes, and $C = [\mathbf{c}_1, \ldots, \mathbf{c}_K]$ is the centroid matrix (Caron et al., 2018). We can see that C is the supportive parameters in Definition 1, in the same role as a codebook.

The input perturbation. Standard augmentation (random resized crop, flip, photometric jitter) is applied on each image in the batch, $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^N \to \{\tilde{\mathbf{X}}_n\}_{n=1}^N$. Two details of DeepCluster are worth noting, because both are sometimes mistaken for selection: (i) images are drawn uniformly over the current pseudo-classes rather than uniformly over the dataset; and (ii) the assignment process described below runs on un-augmented (or mildly augmented) images. Neither involves comparison among candidates.

The assignment step. Periodically (once per epoch in DeepCluster), features $\mathbf{z}_n = f_\theta(\mathbf{X}_n)$ are extracted, reduced by principle component analysis (PCA) (Abdi & Williams, 2010), and then ℓ_2-normalized, denoted as $\tilde{\mathbf{z}}_n$. After that, the processed representations $\{\tilde{\mathbf{z}}_n\}_{n=1}^N$ are clustered by Lloyd's algorithm (Lloyd, 1982):

$$\Big(C^+,\; \{y_n\}_{n=1}^N\Big) \;\in\; \operatorname*{arg\,min}_{C,\,\{y_n\}} \sum_{n=1}^{N} \|\tilde{\mathbf{z}}_n - \mathbf{c}_{y_n}\|_2^2\,, \qquad y_n \in \{1, \ldots, K\}\,, \tag{27}$$

which yields the pseudo-label y_n of each image. Because cluster indices are permuted arbitrarily from one round to the next, the head W is re-initialized whenever Eq. 27 is recomputed.

The training step and the updates. With the pseudo-labels being fixed, the network is trained by multinomial logistic regression on the augmented images,

$$\mathcal{L}(\theta,\, W;\, \{y_n\}\,,\, \mathcal{E}(\mathcal{B})) \;=\; -\frac{1}{N} \sum_{n=1}^{N} \log \Big[\mathrm{softmax}\Big(W f_\theta(\tilde{\mathbf{X}}_n)\Big)\Big]_{y_n}. \tag{28}$$

Thus, the complete update is

$$\delta\theta = -\eta\, \nabla_\theta \mathcal{L}, \qquad \delta W = -\eta\, \nabla_W \mathcal{L} \;\text{ (or a re-initialization)}, \qquad \delta C = C^+ - C. \tag{29}$$

Collecting these components, we have

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) = (\delta\theta,\, \delta W,\, \delta C)\,, \qquad \delta\Theta = \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon. \tag{30}$$

Every component of Eq. 29 is computed from the current parameter set $\Theta = \{\theta, W, C\}$, the batch $\mathcal{B}$ and the perturbation $\mathcal{E}$. To be specific, the pseudo-labels of Eq. 27 are a functional of the current output features, hence of θ and $\mathcal{B}$. The loss of Eq. 28 is evaluated once without performance-based selection or comparison. Hence, **deep clustering is exploitative**.

SwAV (Caron et al., 2020) replaces the offline k-means by an online Sinkhorn–Knopp (Cuturi, 2013) balanced assignment over a set of prototypes that are themselves trained by gradient descent. The Sinkhorn step is a deterministic function of the current scores, so the verdict is unchanged.

A.5 MASKED AUTOENCODERS

Now, we evaluate masked autoencoders, taking MAE (He et al., 2022) as the example. The parameter set is

$$\Theta \;=\; \{\theta_e,\; \theta_d,\; \mathbf{m}\}\,, \tag{31}$$

where θ_e is the encoder, θ_d is the decoder (discarded after pre-training), and $\mathbf{m}$ is the learned mask token together with the positional embeddings (Dosovitskiy et al., 2020) (supportive parameters in the sense of Definition 1).

The input perturbation. Each image is cut into L patches $\mathbf{x}_{n,1}, \ldots, \mathbf{x}_{n,L}$, of which a fraction ρ (in MAE, $\rho = 0.75$) is masked uniformly at random. Let $\mathcal{M}_n$ be the masked index set of image n, and $\mathcal{V}_n$ its complement. Then,

$$\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^N \to \left\{\tilde{\mathbf{X}}_n\right\}_{n=1}^N, \qquad \tilde{\mathbf{X}}_n \;=\; \Big(\, \{\mathbf{x}_{n,j}\}_{j\in\mathcal{V}_n},\;\; \mathcal{M}_n \,\Big). \tag{32}$$

That is, each image is perturbed into its visible patches together with the record of which positions were removed. The encoder sees only $\{\mathbf{x}_{n,j}\}_{j\in\mathcal{V}_n}$, and the decoder receives the encoded visible tokens together with one copy of $\mathbf{m}$ per position in $\mathcal{M}_n$. The format of $\mathcal{E}$ in MAE is slightly more complicated than those in the above families, which still conforms to Definition 1.

The training loss and the update. The decoder predicts the pixels of the masked patches, and the loss is the mean squared error on those patches only, computed on (optionally per-patch normalized) pixel values:

$$\mathcal{L}(\theta_e, \theta_d, \mathbf{m};\, \mathcal{E}(\mathcal{B})) \;=\; \frac{1}{\sum_n |\mathcal{M}_n|} \sum_{n=1}^{N} \sum_{j\in\mathcal{M}_n} \|\hat{\mathbf{x}}_{n,j} - \mathbf{x}_{n,j}\|_2^2\,, \tag{33}$$

where $\hat{\mathbf{x}}_{n,j}$ is the reconstruction of patch j.

Again, all three components of Θ are updated in one gradient step,

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) \;=\; -\eta\,(\,\nabla_{\theta_e}\mathcal{L},\;\; \nabla_{\theta_d}\mathcal{L},\;\; \nabla_{\mathbf{m}}\mathcal{L}\,)\,, \qquad \delta\Theta \;=\; \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon, \tag{34}$$

which obeys the format proposed in Definition 1. In summary, exactly one mask is drawn per image per step. There is no comparison among multiple masked candidates. Hence, **masked autoencoders are exploitative**.

BEiT (Bao et al., 2021) replaces the pixel target by discrete visual tokens. The tokens come from a discrete variational autoencoder (Van Den Oord et al., 2017), which is trained beforehand and then frozen. So, its parameter has $\delta\theta = 0$. The rest training process in BEiT is the same as that in MAE. Hence, BEiT is exploitative as well.

A.6 INFORMATION-THEORETIC APPROACHES

Finally, we evaluate information-theoretic approaches, taking IIC (Ji et al., 2019) and VICReg (Bardes et al., 2021) as the two examples. In this family, the loss is not a sum over individual samples. Instead, the loss is a function of statistics aggregated over the whole batch, such as the joint distribution in IIC and the covariance matrix in VICReg.

The IIC model. The parameter set is $\Theta = \{\theta\}$, where θ is the parameters of an encoder followed by a softmax clustering head, $f_\theta : \mathbb{R}^{H\times W\times C} \to [0,1]^K$.

Each image is augmented into two views. That is, $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^N \to \{(\tilde{\mathbf{X}}_n,\, \tilde{\mathbf{X}}'_n)\}_{n=1}^N$ (the way we describe $\mathcal{E}$ here differs from that in the Contrastive Learning sub-section A.2, which is intended to easy the description in below. But both $\mathcal{E}$ are essentially the same, which augments one image into a pair of views).

Using $\mathbf{p}_n = f_\theta(\tilde{\mathbf{X}}_n)$ and $\mathbf{p}'_n = f_\theta(\tilde{\mathbf{X}}'_n)$ to represent the two soft assignments, IIC forms the symmetrized joint distribution of the two views over the $K \times K$ grid of cluster pairs. In the meanwhile, the marginal distribution is simply a sum over matrix rows. That is,

$$P_{k,k'} \;=\; \frac{1}{2N} \sum_{n=1}^{N} \left(p_{n,k}\, p'_{n,k'} \;+\; p_{n,k'}\, p'_{n,k}\right), \qquad P_k \;=\; \sum\nolimits_{k'=1}^{K} P_{k,k'}. \tag{35}$$

The training target is to maximize the mutual information between the two views' assignments:

$$I(P) \;=\; \sum_{k=1}^{K}\sum_{k'=1}^{K} P_{k,k'} \log \frac{P_{k,k'}}{P_k\, P_{k'}}, \qquad \mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B})) \;=\; -\,I(P). \tag{36}$$

The matrix P of Eq. 35 is a deterministic function of θ and $\mathcal{E}(\mathcal{B})$. The training target $I(P)$ is evaluated once on that single matrix. The parameter updates come from one gradient step,

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) \;=\; -\eta\, \nabla_\theta \mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B}))\,, \qquad \delta\Theta \;=\; \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon. \tag{37}$$

The full IIC model adds an auxiliary over-clustering head, which enlarges Θ by that head's parameters and adds a second loss of the same form as Eq. 36. No comparison among candidates is introduced. Thus, the IIC model satisfies the conditions in Definition 1, **which is exploitative.**

The VICReg model. The parameter set is $\Theta = \{\theta\}$, where θ is the parameters of an encoder followed by an expander. Each image is again augmented into two views, $\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^{N} \to \{(\tilde{\mathbf{X}}_n,\, \tilde{\mathbf{X}}'_n)\}_{n=1}^{N}$. The two views, in one batch of N images, produce embedding matrices $Z, Z' \in \mathbb{R}^{N\times d}$, whose n-th rows are respectively $\mathbf{z}_n = f_\theta(\tilde{\mathbf{X}}_n)$ and $\mathbf{z}'_n = f_\theta(\tilde{\mathbf{X}}'_n)$.

The loss has three terms: the invariance term pulls the two views of one image together; the variance term keeps every embedding dimension from collapsing, using a hinge at a target value $\gamma > 0$; and the covariance term de-correlates distinct embedding dimensions. That is,

$$\begin{aligned} s(Z, Z') &= \frac{1}{N}\sum_{n=1}^{N} \|\mathbf{z}_n - \mathbf{z}'_n\|_2^2\,, \qquad v(Z) = \frac{1}{d}\sum_{a=1}^{d} \max\!\left(0,\; \gamma - \sqrt{\mathrm{Var}(Z_{:,a})}\right), \\ c(Z) &= \frac{1}{d}\sum_{a\neq b} [\mathrm{Cov}(Z)]_{a,b}^2\,. \end{aligned} \tag{38}$$

With positive coefficients $\alpha_1, \alpha_2, \alpha_3$, the total loss is

$$\mathcal{L}(\theta;\, \mathcal{E}(\mathcal{B})) \;=\; \alpha_1\, s(Z, Z') \;+\; \alpha_2\, (v(Z) + v(Z')) \;+\; \alpha_3\, (c(Z) + c(Z'))\,. \tag{39}$$

Once again, the parameter updates come from one gradient step, with exactly the same form as Eq. 37. **Hence, VICReg is exploitative.**

Barlow Twins (Zbontar et al., 2021) replaces the variance and covariance terms by driving the cross-correlation matrix between the two views toward the identity matrix. This changes the statistic being computed, but not the parameter updates process. Hence, Barlow Twins is exploitative as well.

One point deserves emphasis here, because it concerns our own approach. Our surprise score $\mathcal{S}(\theta)$ is also a batch-level statistic, and IIC's mutual information is its closest relative: Appendix B derives the exact connection between the two. However, $\mathcal{S}(\theta)$ cannot be fully optimized by exploitative approaches, due to its own particularity.

According to all the formulas and descriptions in Section 3.2, we can see that each cluster (output dimension) in our model has its own 'freedom' to either produce surprising view-matching result, or die out throughout the batch. That says, the final surprise score $\mathcal{S}(\theta)$ places no restriction on which clusters, or how many clusters, should finally produce surprising view-matching results. A cluster can either contribute to $\mathcal{S}(\theta)$, or die out quietly. With such a high freedom, there are astronomical possibilities for $\mathcal{S}(\theta)$ to select surprising clusters. Among these astronomical possibilities, the optimal one may not be achievable simply by exploiting observed states. Thus, explorative methods involving random perturbation and performance-based selection become necessary. But once again, this is an intuitive explanation, rather than a strict proof of Hypothesis 1.

Due to the overwhelming number of self-supervised learning approaches available today, it is impossible for us to evaluate every one of them in detail. Researchers can apply the evaluations above to the approaches they are interested in: First, write down the parameter set Θ, the input perturbation $\mathcal{E}$ and the update $\delta\Theta$. Then, ask whether the fitness ϕ (or loss $\mathcal{L}$ in many cases) is ever evaluated at multiple $\Theta + \epsilon$ settings or multiple $\mathcal{E}(\mathcal{B})$ settings in order to decide $\delta\Theta$.

In general, based on all six subsections above, we claim that all dominant SSL approaches for image clustering are exploitative.

A.7 THE TWO NON-PARAMETRIC BASELINES: DEEPDPM AND UNSEEN

Now, we evaluate the two baseline models used in Section 4: DeepDPM (Ronen et al., 2022) and UNSEEN (Leiber et al., 2024). To be honest, both approaches are non-parametric clustering algorithms applied to given features, whereas the features in both approaches are obtained separately. Note that neither approach augments input images. That is, $\mathcal{E} = \mathbf{I}$.

The DeepDPM model. DeepDPM performs inference in a Dirichlet Process Mixture (DPM) model (Neal, 2000) with a dynamic architecture. Its parameter set is

$$\Theta \;=\; \Big\{\, \theta_{\mathrm{cl}},\;\; \big(\theta^{k}_{\mathrm{sub}}\big)_{k=1}^{K},\;\; \big(\pi_k,\, \boldsymbol{\mu}_k,\, \Sigma_k\big)_{k=1}^{K},\;\; \big(\tilde{\pi}_{k,j},\, \tilde{\boldsymbol{\mu}}_{k,j},\, \tilde{\Sigma}_{k,j}\big)_{k,\,j\in\{1,2\}} \,\Big\}, \tag{40}$$

where θ_{cl} is a clustering network with K output clusters, and $\theta^{k}_{\mathrm{sub}}$ is a sub-clustering network for cluster k with two further output clusters (sub-clusters) that maintains a two-component split of cluster k. Then, $\big(\pi_k,\, \boldsymbol{\mu}_k,\, \Sigma_k\big)_{k=1}^{K}$ are respectively the weights, means and covariances of the K main clusters; $\big(\tilde{\pi}_{k,j},\, \tilde{\boldsymbol{\mu}}_{k,j},\, \tilde{\Sigma}_{k,j}\big)_{k,\,j\in\{1,2\}}$ are those of the two sub-clusters in cluster k, indexed by $j \in \{1, 2\}$.

For a fixed K, DeepDPM is exploitative. In each step, the clustering network takes one gradient step, with the loss to be the KL divergence (Shlens, 2014) between the network's soft assignments and the assignments that an E-step in an EM algorithm (Dempster et al., 1977) would produce from the current mixture parameters. The sub-clustering network takes one gradient step on an isotropic distance loss (Beckers & Smeulders, 1992). The mixture parameters are then updated by weighted maximum-a-posteriori estimates under a Normal-Inverse-Wishart prior (So, 2024). All of these quantities are computed from the current Θ and the current batch $\mathcal{B}$ without performance-based selection or comparison. So, they are of the form $\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E})$ required by Definition 1.

Before continue, we note that this is a re-description of another model. One can refer to the original DeepDPM paper (Ronen et al., 2022) for more details.

However, K itself is changed by a mutation-selection mechanism. With every certain number of iterations, the split of each cluster k into its two sub-clusters is proposed, and that proposal is accepted with probability $\min(1, H_{\mathrm{s}})$, where

$$H_{\mathrm{s}} \;=\; \frac{\alpha\,\Gamma(N_{k,1})\,f_x(X_{k,1})\,\Gamma(N_{k,2})\,f_x(X_{k,2})}{\Gamma(N_k)\,f_x(X_k)} \tag{41}$$

is the Metropolis–Hastings ratio (Hastings, 1970; Chang & Fisher III, 2013). Here, α is the concentration parameter of the Dirichlet process (Teh et al., 2010), Γ is the Gamma function, X_k denotes the points in cluster k, $N_k = |X_k|$ denotes their amount, $X_{k,j}$ and $N_{k,j}$ are the same quantities for sub-cluster $j \in \{1, 2\}$, and $f_x(\cdot)$ is the marginal likelihood under the Normal-Inverse-Wishart prior. Merge proposals are handled in the same way, with $H_{\mathrm{m}} = 1/H_{\mathrm{s}}$, considered sequentially between each cluster and its three nearest neighbors.

When a proposal is accepted, output units of the clustering network are added or removed, new sub-clustering networks are initialized, and K changes accordingly. Once again, more details are provided in the original paper (Ronen et al., 2022).

As a result, DeepDPM randomly perturbs the number of main clusters K, and then decides whether to keep a perturbation based on the performance. Hence, **DeepDPM is explorative**.

It is worth noting where that exploration acts. DeepDPM directly performs on the output features of pre-trained vision models. In contrast, our converge-to-surprise scheme explores the entire parameter space of a network, which is trained from scratch on raw images.

The UNSEEN framework. UNSEEN is built upon existing deep clustering algorithms — DCN (Yang et al., 2017), DEC (Xie et al., 2016) or DKM (Fard et al., 2020) — and starts from an initial number of clusters k_{init} that acts as an upper bound. Using C^j_i to denote the set of samples assigned to cluster i in epoch j, the parameter set of UNSEEN is

$$\Theta \;=\; \Big\{\, \theta_{\mathrm{enc}},\;\; \Theta_{\mathrm{base}},\;\; k^j,\;\; \big(\big|C^0_i\big|\big)_i \,\Big\}, \tag{42}$$

where θ_{enc} is the parameters of the encoder, Θ_{base} collects the parameters of the clustering model built upon the encoder, k^j is the number of active clusters in epoch j, and $\big|C^0_i\big|$ is the recorded

creation size (initial size) of cluster i. The last two are supportive parameters: they are bookkeeping quantities that the update reads and writes.

The network is optimized by Adam (Kingma & Ba, 2014) on the clustering algorithm's own loss, together with UNSEEN's nearest-neighbor term:

$$\mathcal{L}_{\mathrm{UNSEEN}}(\mathcal{B},\, l) \;=\; \frac{1}{l\,|\mathcal{B}|} \sum_{\mathbf{X}\in\mathcal{B}} \sum_{\mathbf{Y}\in \mathrm{nn}(\mathcal{B},\,\mathbf{X},\,l)} \|\mathrm{enc}(\mathbf{X}) - \mathrm{enc}(\mathbf{Y})\|_2^2 \,. \tag{43}$$

Such a loss pulls each sample toward its l nearest neighbors within the batch, so that samples can still change clusters late in training. Note that the encoder is trained from scratch with freshly initialized θ_{enc}. So, unlike DeepDPM, UNSEEN does not apply pre-trained models.

Then, after each epoch, cluster i is declared dead whenever its current size has shrunk below a fraction t of its creation size. That is, if

$$\left|C_i^j\right| / \left|C_i^0\right| \;<\; t, \qquad t = 0.5 \;\text{ by default,} \tag{44}$$

then cluster i will be removed completely. Its samples are reassigned to the closest remaining cluster. The creation sizes of the receiving clusters are increased by the number of samples they absorb, and k^j is set to the number of survivors. Eq. 44 is the mechanism by which the number of clusters is inferred. There is no split operation. Once again, more details can be discovered from the original paper (Leiber et al., 2024).

To write the parameter updates explicitly, we first fix the notation of the assignment. Let $\mathcal{X}$ be the dataset, and let A^j be the index set of active clusters in epoch j (so $k^j = \left|A^j\right|$). Let $a^j : \mathcal{X} \to A^j$ be the hard assignment produced by the clustering algorithm, so that $C_i^j = \{\mathbf{X} \in \mathcal{X} : a^j(\mathbf{X}) = i\}$. Then, Eq. 44 defines the set of dead clusters and the set of survivors,

$$D^j \;=\; \left\{\, i \in A^j \;:\; \left|C_i^j\right| / \left|C_i^0\right| < t \,\right\}, \qquad A^{j+1} \;=\; A^j \setminus D^j. \tag{45}$$

Every sample of a dead cluster is then reassigned by the same rule restricted to the survivors, $\hat{a}^j : \mathcal{X} \to A^{j+1}$. We write

$$n_i^j \;=\; \left|\left\{\, \mathbf{X} \in \bigcup\nolimits_{i'\in D^j} C_{i'}^j \;:\; \hat{a}^j(\mathbf{X}) = i \,\right\}\right|, \qquad i \in A^{j+1}, \tag{46}$$

to denote the number of samples that the surviving cluster i absorbs. Note that $\sum_{i\in A^{j+1}} n_i^j = \sum_{i'\in D^j} \left|C_{i'}^j\right|$: no sample is lost.

With $\mathcal{L} = \mathcal{L}_{\mathrm{base}} + \mathcal{L}_{\mathrm{UNSEEN}}$ (again, UNSEEN is built upon existing deep clustering algorithms. So, $\mathcal{L}_{\mathrm{base}}$ is the loss of that base model), the updates of the four components of Θ come in two stages:

Within epoch j, each batch produces one gradient step and touches nothing else (of course there are multiple steps in one batch):

$$\delta\theta_{\mathrm{enc}} = -\eta\, \nabla_{\theta_{\mathrm{enc}}} \mathcal{L}(\Theta;\, \mathcal{B})\,, \qquad \delta\Theta_{\mathrm{base}} = -\eta\, \nabla_{\Theta_{\mathrm{base}}} \mathcal{L}(\Theta;\, \mathcal{B})\,, \qquad \delta k^j = 0, \qquad \delta\left|C_i^0\right| = 0. \tag{47}$$

At the end of epoch j, the pruning stage applies Eq. 45–46:

$$\delta\theta_{\mathrm{enc}} = 0, \qquad \delta\Theta_{\mathrm{base}} = \Pi_{A^{j+1}}\Theta_{\mathrm{base}} - \Theta_{\mathrm{base}}, \qquad \delta k^j = -\left|D^j\right|, \qquad \delta\left|C_i^0\right| = n_i^j \;\; \left(i \in A^{j+1}\right), \tag{48}$$

where $\Pi_{A^{j+1}}$ keeps only the entries indexed by the surviving clusters, so that the cluster parameters of every $i \in D^j$ — together with their recorded creation sizes — are deleted from Θ. Two features of Eq. 48 are worth naming. The count k^j decreases by exactly the number of dead clusters and never increases, since there is no split operation. And a creation size never tracks the current size of its cluster: by Eq. 46, $\left|C_i^0\right|$ changes only when cluster i absorbs samples from a dead cluster, and is otherwise frozen at the value it was given when the cluster was created.

Every quantity in Eq. 47–48 is determined by the current parameter set Θ and the current data. The dead set D^j is read off the counts $\left|C_i^j\right|$ and the recorded $\left|C_i^0\right|$. The absorbed counts n_i^j come from the clustering algorithm's own assignment rule. In particular, Eq. 44 is a threshold evaluated at the single current Θ.

Collecting Eq. 47–48 into Ω, we have

$$\Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) = (\delta\theta_{\text{enc}},\ \ \delta\Theta_{\text{base}},\ \ \delta k^j,\ \ \left(\delta\left|C_i^0\right|\right)_i), \qquad \delta\Theta = \Omega(\phi,\, \Theta,\, \mathcal{B},\, \mathcal{E}) + \epsilon. \tag{49}$$

One minor issue is that, the fitness score ϕ of k^j and $\left(\left|C_i^0\right|\right)_i$ is not explicitly defined. But since we can explicitly describe Eq. 49, we do not need to be bothered by this minor issue. In summary, UNSEEN never instantiates or scores a second parameter set, and never keeps one candidate over another according to performances. Hence, **UNSEEN is exploitative**.

The contrast between these two baselines is informative. Both infer the number of clusters, both operate on a learned embedding, and both are trained without labels. Yet the explorative one, DeepDPM, outperforms all three exploitative UNSEEN variants on every one of the three benchmarks in Table 1. Our converge-to-surprise scheme is explorative as well, which explores the full parameter set of a network and outperforms DeepDPM on most metrics in Table 1. This may be regarded as an indirect supportive evidence of Hypothesis 1: On the same benchmarks with the same non-parametric clustering tasks, explorative approaches in general performs better than exploitative ones, indicating that explorative approaches are better at 'exploring' the parameter space to find an optimum. Hence, explorative approaches should also be better at optimizing the surprise score, which requires exploring the deep network's parameter space. But once again, this is another intuitive analysis.

Remark (scope of Definition 1). The following sub-sections no more deal with discriminative learning on images. Thus, some assumptions in Definition 1 need to be expanded or modified.

First, we allow $\mathbf{P}$ to be a distribution over an arbitrary sample space.

Second, a single fitness score ϕ is not required. When an approach partitions Θ into blocks and updates separately, each block carries its own fitness score (or scores).

Third, ϵ and $\mathcal{E}$ stand for whatever random factors the approach draws, and need not be the only two. The criterion itself is unchanged: the approach is exploitative when no update is decided by evaluating fitness scores at multiple settings of those random factors, and explorative otherwise.

A.8 Expanding our scope: generative adversarial networks

Every approach evaluated above is discriminative (Jebara, 2012): one parameter set is trained toward one fitness (explicit or implicit). Generative adversarial networks (GAN) (Goodfellow et al., 2014) differ in two aspects. They are generative models, and they are trained by a min-max game between two parameter blocks with opposing objectives.

We take the original GAN of (Goodfellow et al., 2014) as the example. Its parameter set is

$$\Theta \;=\; \{\theta_g,\ \theta_d\}\,, \tag{50}$$

where θ_g is the parameters of a generator $G_{\theta_g} : \mathbb{R}^{d_z} \to \mathbb{R}^{H\times W\times C}$, which maps a latent code to an image, and θ_d is the parameters of a discriminator $D_{\theta_d} : \mathbb{R}^{H\times W\times C} \to [0,1]$, which estimates the probability that its input is a real image rather than a generated one. We keep the standard symbols G and D here. The discriminator $D_{\theta_d}(\cdot)$ takes a single argument, and is not to be confused with the binary Kullback–Leibler divergence $D(\cdot\|\cdot)$ of Eq. 8.

The input perturbation. The original GAN applies no augmentation to the real images. Its random factor is the latent code instead: N codes are drawn independently from a fixed prior p_z, such as a Gaussian, and passed to the generator. This is the perturbation of Definition 1, which attaches one code to each sample of the batch:

$$\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^N \to \{(\mathbf{X}_n,\, \mathbf{z}_n)\}_{n=1}^N\,, \qquad \mathbf{z}_n \sim p_z. \tag{51}$$

The training target. The two blocks (generator and discriminator) share a single value function, which the discriminator maximizes (ascends) and the generator minimizes (descends):

$$\min_{\theta_g}\max_{\theta_d}\ V(\theta_d,\, \theta_g) \;=\; \mathbb{E}_{\mathbf{X}\sim\mathbf{P}}\left[\log D_{\theta_d}(\mathbf{X})\right] \;+\; \mathbb{E}_{\mathbf{z}\sim p_z}\left[\log\!\big(1 - D_{\theta_d}\big(G_{\theta_g}(\mathbf{z})\big)\big)\right]. \tag{52}$$

Read block-wise, the fitness of the discriminator is $\phi_d = V$ and the fitness of the generator is $\phi_g = -V$.

The updates. Training alternates between a fixed number of ascent steps on θ_d and one descent step on θ_g over the current batch:

$$\begin{aligned}\delta\theta_d &= \eta\,\nabla_{\theta_d}\frac{1}{N}\sum_{n=1}^{N}\left[\log D_{\theta_d}(\mathbf{X}_n) + \log\big(1 - D_{\theta_d}\big(G_{\theta_g}(\mathbf{z}_n)\big)\big)\right],\\ \delta\theta_g &= -\eta\,\nabla_{\theta_g}\frac{1}{N}\sum_{n=1}^{N}\log\big(1 - D_{\theta_d}\big(G_{\theta_g}(\mathbf{z}_n)\big)\big).\end{aligned} \tag{53}$$

In practice, the generator maximizes $\log D_{\theta_d}(G_{\theta_g}(\mathbf{z}_n))$ instead, which avoids vanishing gradients early in training. This changes the gradient, but not the structure of the update. Collecting the two blocks, we have that

$$\Omega(\phi,\,\Theta,\,\mathcal{B},\,\mathcal{E}) \;=\; (\delta\theta_g,\;\delta\theta_d)\,, \qquad \delta\Theta \;=\; \Omega(\phi,\,\Theta,\,\mathcal{B},\,\mathcal{E}) + \epsilon. \tag{54}$$

Both entries are computed once, at the current (θ_g, θ_d) and on the single perturbed batch $\mathcal{E}(\mathcal{B})$ of Eq. 51. Hence, **GAN is exploitative**.

The min-max game may look like an explorative process, so we clarify it here. Both blocks in Eq. 53 are updated by gradient operators, applied unconditionally and with opposite signs. Neither block ever proposes a candidate parameter setting, evaluates V at it, and then keeps or discards it according to the evaluation. The discriminator does score the generated samples, but that is through a direct training signal rather than an acceptance test. After the discriminator ascends the fitness V, the generator descends it, whether or not the step improves the ability of the generator. In contrast, the centered rank r_i in Eq. 10 of our converge-to-surprise scheme multiplies the perturbation ϵ_i itself, so there are performance-based selection (ranking) on perturbations in each optimization step. In summary, the min-max game in GAN does not involve a mutation-selection mechanism.

A.9 Expanding our scope: diffusion models

Diffusion models are generative as well, but they are trained without an adversary. So, Definition 1 applies to them directly. We take the denoising diffusion probabilistic model (DDPM) of (Ho et al., 2020) as the example. Its parameter set is simply

$$\Theta \;=\; \{\theta\}\,. \tag{55}$$

Here, θ is the parameters of a network $\boldsymbol{\xi}_\theta(\cdot,\,t) : \mathbb{R}^{H\times W\times C} \to \mathbb{R}^{H\times W\times C}$, which receives a noised image together with its noise level t, and predicts the noise that was added to it. Here, we use $\boldsymbol{\xi}$ to denote the Gaussian noise, rather than using $\boldsymbol{\epsilon}$ of (Ho et al., 2020), since ϵ already denotes the parameter-level perturbation in Definition 1. One will not confuse $\boldsymbol{\xi}$ with the network symbol $\boldsymbol{\xi}_\theta$.

The input perturbation. The random factor of a diffusion model is the forward (noising/diffusion) process, which is applied to the batch and is itself the pretext task. For each image, a time step t_n is drawn uniformly from $\{1, \ldots, T\}$, and a noise tensor $\boldsymbol{\xi}_n$ is drawn from a standard Gaussian. Then, the image is corrupted accordingly:

$$\mathcal{E} : \{\mathbf{X}_n\}_{n=1}^{N} \to \left\{\left(\tilde{\mathbf{X}}_n,\,t_n,\,\boldsymbol{\xi}_n\right)\right\}_{n=1}^{N}, \qquad \tilde{\mathbf{X}}_n = \sqrt{\bar{\alpha}_{t_n}}\,\mathbf{X}_n + \sqrt{1-\bar{\alpha}_{t_n}}\,\boldsymbol{\xi}_n. \tag{56}$$

Here, $t_n \sim \mathrm{Uniform}\{1, \ldots, T\}$, $\boldsymbol{\xi}_n \sim \mathcal{N}(0, I)$, and $\bar{\alpha}_t \in (0, 1)$ is a fixed decreasing variance schedule, which is not learned. Eq. 56 plays the same role as the random masking in MAE (Eq. 32): a random rule destroys part of the information in each sample, and the model is trained to undo it.

The training target and the update. The network is trained to predict the noise that Eq. 56 added, under the simplified objective of (Ho et al., 2020):

$$\mathcal{L}(\theta;\,\mathcal{E}(\mathcal{B})) \;=\; \frac{1}{N}\sum_{n=1}^{N}\left\|\boldsymbol{\xi}_n - \boldsymbol{\xi}_\theta\left(\tilde{\mathbf{X}}_n,\,t_n\right)\right\|_2^2. \tag{57}$$

Similar as in the discriminative models, the fitness is $\phi = \mathcal{L}(\theta;\,\mathcal{E}(\mathcal{B}))$. The parameter updates come from one gradient step,

$$\Omega(\phi,\,\Theta,\,\mathcal{B},\,\mathcal{E}) \;=\; -\eta\,\nabla_\theta\mathcal{L}(\theta;\,\mathcal{E}(\mathcal{B}))\,, \qquad \delta\Theta \;=\; \Omega(\phi,\,\Theta,\,\mathcal{B},\,\mathcal{E}) + \epsilon. \tag{58}$$

Exactly one perturbation pair $(t_n, \boldsymbol{\xi}_n)$ is drawn per image per step, and it is directly applied to the sample image without performance-based selection. Hence, **diffusion models (DDPMs) are exploitative**.

During inference, the iterative reverse process may look like an explorative process, so we clarify it here. Starting from pure noise, $\boldsymbol{\xi}_\theta$ is applied for T steps to produce an image. There is no parameter update at all, which is therefore outside the scope of Definition 1. Later diffusion models, such as the transformer-based DiT (Peebles & Xie, 2023), change the architecture and the space in which the diffusion is run, but leave the fundamental denoising mechanism unchanged.

DDIM (Song et al., 2020) differs from DDPM in the reverse process rather than in the training. It replaces the Markovian reverse chain by a family of non-Markovian ones that share the same forward marginals as Eq. 56, so the training target remains Eq. 57 and a DDPM-trained network can be reused unchanged. What DDIM gains is at inference time: the sampling becomes deterministic given the initial noise, and it may jump along a subsequence of the T steps, which is considerably faster. Once again, there is no performance-based selection. The noise is applied directly to the samples, and all parameter updates come from one gradient step. Hence, DDIMs are also exploitative.

A.10 Expanding our scope: reinforcement learning

Reinforcement learning (RL) (Sutton et al., 1998) needs two further relaxations of the setting in Definition 1. First, the samples are not drawn from a distribution that is fixed and independent of Θ: the agent generates its own data, so the sample distribution follows the current parameters. Second, the random factor is neither a perturbation $\mathcal{E}$ of a given batch nor a perturbation ϵ of the parameters, but the agent's own random choice of action. The criterion of Definition 1 is unchanged: we ask whether the fitness is evaluated at multiple settings of that random factor in order to decide $\delta\Theta$.

We take tabular Q-learning (Watkins & Dayan, 1992) as the example, which is among the simplest RL approaches. Its parameter set is

$$\Theta \;=\; \{\theta_Q\}\,, \qquad Q : \mathcal{S} \times \mathcal{A} \to \mathbb{R}, \tag{59}$$

where $\mathcal{S}$ is the set of states, $\mathcal{A}$ is the set of actions, $Q(s, a)$ is the current estimate of the value of taking action a in state s, and θ_Q is the parameters of the deep Q-learning network.

The random factor. In state s_t, the agent selects an action a_t using an ϵ-greedy behavior policy: it chooses a uniformly random action with a small, fixed probability ϵ, and selects the greedy action (the one with the highest current value) otherwise. The environment then returns a reward R_t and a next state s_{t+1}. The transition $(s_t,\, a_t,\, R_t,\, s_{t+1})$ is the sample, and it exists only because that random choice was made.

The update. With learning rate η and discount factor $\gamma \in [0, 1)$, only the entry of the action that was actually taken is updated:

$$\delta Q(s_t,\, a_t) \;=\; \eta\Big[\, R_t \;+\; \gamma \max_{a' \in \mathcal{A}} Q(s_{t+1},\, a') \;-\; Q(s_t,\, a_t)\Big], \tag{60}$$

while $\delta Q(s, a) = 0$ for every other state-action pair.

Eq. 60 applies performance-based selection to a random factor. The bracketed term is the outcome of the randomly chosen action a_t, measured against the value that was expected of it, and the value of exactly that action is then updated. If the random choice turned out better than expected, its value is raised, and the behavior policy will take it more often; if it turned out worse, its value is lowered and the choice is suppressed. So, the random factor is retained or discarded according to the performance it produced, and the samples of the following steps are generated accordingly. This means that we cannot find a map Ω, with only directly applied perturbations, to represent parameter updates in reinforcement learning. Hence, **reinforcement learning is explorative**.

Once again, there is a wide range of self-supervised learning approaches: generative, discriminative and behavioral. We cannot evaluate each and every of them. Researchers can apply similar evaluations to the approaches they are interested in.

B CONNECTION BETWEEN OUR SURPRISE SCORE $\mathcal{S}(\theta)$ AND THE IIC MODEL TRAINING TARGET

Recall from Section 3.2 that, given N images, the two complementary views of image n produce logit vectors $\mathbf{y}_n^{(i)} = f_\theta(\tilde{\mathbf{X}}_n^{(i)})$ and $\mathbf{y}_n^{(j)} = f_\theta(\tilde{\mathbf{X}}_n^{(j)})$, from which the hard assignments $\hat{y}_n^{(i)}, \hat{y}_n^{(j)}$ are taken by argmax. But now, if we replace each argmax by a softmax at temperature $\tau > 0$, we will obtain soft assignments on the probability simplex,

$$\boldsymbol{\pi}_n = \mathrm{softmax}\Big(\mathbf{y}_n^{(i)}/\tau\Big), \qquad \boldsymbol{\rho}_n = \mathrm{softmax}\Big(\mathbf{y}_n^{(j)}/\tau\Big). \tag{61}$$

Every count in Section 3.2 becomes a soft expectation:

$$\hat{q}_k = \frac{1}{N}\sum_{n=1}^{N} \pi_{n,k}\,\rho_{n,k}, \qquad p_k = \frac{1}{2N}\sum_{n=1}^{N}(\pi_{n,k} + \rho_{n,k}), \qquad q_k = p_k^2. \tag{62}$$

Also, the over-matching indicator $\mathbf{1}[\hat{q}_k > q_k]$ of Eq. 8 can be replaced by a smooth gate, namely the logistic sigmoid (Kyurkchiev & Markov, 2015):

$$\sigma(u) \;=\; \frac{1}{1+e^{-u}}, \qquad u \in \mathbb{R}. \tag{63}$$

By a routine derivation, we can prove that

$$\lim_{\varepsilon \to 0^+} \sigma\!\left(\frac{\hat{q}_k - q_k}{\varepsilon}\right) \;=\; \mathbf{1}[\hat{q}_k > q_k]\,. \tag{64}$$

So, the hard selection of over-matching clusters is recovered as $\varepsilon \to 0^+$. Then, we define the soft version of the surprise score as

$$\mathcal{S}_{\mathrm{soft}}(\theta;\tau,\varepsilon) = \sum\nolimits_k \sigma\!\left(\frac{\hat{q}_k - q_k}{\varepsilon}\right) D(\hat{q}_k \,\|\, q_k)\,. \tag{65}$$

Again, by a routine derivation, we can prove that

$$\lim_{\tau,\varepsilon \to 0^+} \mathcal{S}_{\mathrm{soft}}(\theta;\tau,\varepsilon) = \mathcal{S}(\theta). \tag{66}$$

$\mathcal{S}_{\mathrm{soft}}$ is differentiable in θ, so it can be maximized directly by Adam (Kingma & Ba, 2014). This is worth stating plainly: we do not claim that the surprise score cannot be made differentiable. It can, and Eq. 65 is one way to do it.

Now, we compare the above format with the training target of the IIC model. Note that the IIC model does not apply chessboard masking as in our approach. The IIC model directly augment the original image into two views. But the output of the model on the two views of image n can be represented by $\boldsymbol{\pi}_n$ and $\boldsymbol{\rho}_n$ all the same.

Then, given the same soft assignments $\boldsymbol{\pi}_n$ and $\boldsymbol{\rho}_n$, IIC forms the soft joint distribution of the two views' cluster assignments over the $K \times K$ grid of assignment pairs:

$$\tilde{P}_{k,k'} \;=\; \frac{1}{N}\sum_{n=1}^{N} \pi_{n,k}\,\rho_{n,k'}. \tag{67}$$

Entry (k, k') is the soft frequency with which the i-side view is assigned to cluster k while the j-side view is assigned to cluster k'. IIC then symmetrizes this matrix,

$$P \;=\; \tfrac{1}{2}\left(\tilde{P} + \tilde{P}^\top\right), \qquad \text{that is,} \qquad P_{k,k'} \;=\; \frac{1}{2N}\sum_{n=1}^{N}(\pi_{n,k}\,\rho_{n,k'} + \pi_{n,k'}\,\rho_{n,k})\,. \tag{68}$$

That is, the IIC model training target encodes the fact that the two augmented views play interchangeable roles. Neither view is "the original" and neither is "the transformed one". This is exactly as in our own chessboard-masking construction, where the i-side and the j-side are two halves of one chessboard with no side being original.

We now read off the two quantities of Section 3.2 from this matrix.

The diagonal. Setting $k' = k$ in Eq. 68, we have:

$$P_{k,k} \;=\; \frac{1}{2N}\sum_{n=1}^{N}\left(\pi_{n,k}\,\rho_{n,k} + \pi_{n,k}\,\rho_{n,k}\right) \;=\; \frac{1}{N}\sum_{n=1}^{N}\pi_{n,k}\,\rho_{n,k} \;=\; \hat{q}_k. \tag{69}$$

So, the k-th diagonal entry of IIC's joint distribution equals to our soft view-matching rate: the soft probability that both views of the same image are clustered to k.

The marginal. Summing the k-th row of Eq. 68 over k' recovers our marginal cluster probability p_k. That is,

$$\begin{aligned}
P_k \;&=\; \sum_{k'} P_{k,k'} && (70)\\
&=\; \sum_{k'}\frac{1}{2N}\sum_{n=1}^{N}\left(\pi_{n,k}\,\rho_{n,k'} + \pi_{n,k'}\,\rho_{n,k}\right) && (71)\\
&=\; \frac{1}{2N}\sum_{n=1}^{N}\sum_{k'}\left(\pi_{n,k}\,\rho_{n,k'} + \pi_{n,k'}\,\rho_{n,k}\right) && (72)\\
&=\; \frac{1}{2N}\sum_{n=1}^{N}\left(\pi_{n,k}\sum_{k'}\rho_{n,k'} \;+\; \rho_{n,k}\sum_{k'}\pi_{n,k'}\right) && (73)\\
&=\; \frac{1}{2N}\sum_{n=1}^{N}\left(\pi_{n,k}\cdot 1 \;+\; \rho_{n,k}\cdot 1\right) && (74)\\
&=\; \frac{1}{2N}\sum_{n=1}^{N}\left(\pi_{n,k} + \rho_{n,k}\right) \;=\; p_k. && (75)
\end{aligned}$$

The step from Eq. 73 to Eq. 74 is true because

$$\textstyle\sum_{k'}\pi_{n,k'} \;=\; \sum_{k'}\rho_{n,k'} \;=\; 1. \tag{76}$$

It is worth noting what would happen without the symmetrization of Eq. 68. Running the same computation on the raw joint of Eq. 67 gives $\sum_{k'}\tilde{P}_{k,k'} = \frac{1}{N}\sum_n \pi_{n,k}$ and $\sum_k \tilde{P}_{k,k'} = \frac{1}{N}\sum_n \rho_{n,k'}$. That says, the default symmetrization in both the IIC model and our approach ensures that the joint and marginal probabilities above are equal.

Collecting Eq. 69 and Eq. 75, we obtain the correspondence

$$P_{k,k} = \frac{1}{N}\sum_{n=1}^{N}\pi_{n,k}\,\rho_{n,k} = \hat{q}_k, \qquad P_k = \textstyle\sum_{k'} P_{k,k'} = \frac{1}{2N}\sum_{n=1}^{N}\left(\pi_{n,k} + \rho_{n,k}\right) = p_k. \tag{77}$$

That is, our observed view-matching rate $\hat{q}_k$ equals to the k-th diagonal entry of IIC's joint distribution, and our marginal distribution also equals to theirs.

Finally, the training target of the IIC model is the mutual information between the two views' assignments:

$$I(\boldsymbol{\pi};\boldsymbol{\rho}) \;=\; \textstyle\sum_{k,k'} P_{k,k'}\log\dfrac{P_{k,k'}}{P_k\,P_{k'}}. \tag{78}$$

Now, we compare the two training targets directly. Removing the smooth gate from Eq. 65 leaves the un-gated soft surprise score, which no longer depends on ε:

$$\tilde{\mathcal{S}}_{\text{soft}}(\theta;\tau) \;=\; \textstyle\sum_{k} D(\hat{q}_k \,\|\, q_k)\,. \tag{79}$$

Substituting the correspondence of Eq. 77 into Eq. 79, and subtracting Eq. 78, we have:

$$\begin{aligned}
\tilde{\mathcal{S}}_{\text{soft}}(\theta;\tau) - I(\boldsymbol{\pi};\boldsymbol{\rho}) &= \sum_k D(\hat{q}_k \,\|\, q_k) \;-\; \sum_{k,k'} P_{k,k'} \log \frac{P_{k,k'}}{P_k P_{k'}} \\
&= \sum_k \left[\hat{q}_k \log \frac{\hat{q}_k}{q_k} + (1-\hat{q}_k) \log \frac{1-\hat{q}_k}{1-q_k} \right] \;-\; \sum_{k,k'} P_{k,k'} \log \frac{P_{k,k'}}{P_k P_{k'}} \\
&= \sum_k \left[P_{k,k} \log \frac{P_{k,k}}{P_k P_k} + (1-P_{k,k}) \log \frac{1-P_{k,k}}{1-P_k P_k} \right] \;-\; \sum_{k,k'} P_{k,k'} \log \frac{P_{k,k'}}{P_k P_{k'}} \\
&= \sum_k P_{k,k} \log \frac{P_{k,k}}{P_k P_k} + \sum_k (1-P_{k,k}) \log \frac{1-P_{k,k}}{1-P_k P_k} - \sum_k P_{k,k} \log \frac{P_{k,k}}{P_k P_k} - \sum_{k \neq k'} P_{k,k'} \log \frac{P_{k,k'}}{P_k P_{k'}} \\
&= \sum_k (1-P_{k,k}) \log \frac{1-P_{k,k}}{1-P_k P_k} \;-\; \sum_{k \neq k'} P_{k,k'} \log \frac{P_{k,k'}}{P_k P_{k'}}.
\end{aligned} \tag{80}$$

For simplicity, we write $R_k = (1-P_{k,k}) \log \frac{1-P_{k,k}}{1-P_k P_k}$. Here is our intuitive analysis on Eq. 80:

1. The diagonal sum cancels between the two targets. That is, on the diagonal, our surprise score (without considering over-matching or under-matching) and the IIC model training target agree exactly. This is the precise sense in which our differentiable surprise score is the diagonal of the IIC model training target. Everything that distinguishes the two lives off the diagonal.
2. Off the diagonal, the two behave differently. Since $\sum_{k,k'} P_{k,k'} = 1$ and $\sum_{k,k'} P_k P_{k'} = 1$, we can write $1 - P_{k,k} = \sum_{(a,b)\neq(k,k)} P_{a,b}$ and $1 - P_k P_k = \sum_{(a,b)\neq(k,k)} P_a P_b$. So, R_k compares the total off-diagonal probability mass with the total mass predicted under independence, giving one lumped number for each k. In contrast, the mutual information resolves all $K^2 - K$ off-diagonal entries separately. Put plainly, for each cluster k, our surprise score asks only a binary question: whether both views are clustered to k more often than independence predicts. In contrast, the mutual information further asks which pair (k, k') the disagreement falls into.

Consequently, the residual $\sum_k R_k - \sum_{k \neq k'} P_{k,k'} \log \left(P_{k,k'} / P_k P_{k'} \right)$ is itself a function of P, and hence of θ. The two training targets are therefore not equal, and they are not equal up to an additive constant either. Maximizing one is not the same as maximizing the other, although the two coincide on the diagonal.

C Convert the surprise score into differentiable formats and train the model by exploitative approaches

Appendix B shows that the surprise score can be made differentiable. Here, we take that step twice — once with a plain softmax and once with a Gumbel-softmax (Jang et al., 2016) — and train the model end-to-end by Adam (Kingma & Ba, 2014). Then, we compare the obtained results against those from the converge-to-surprise scheme.

C.1 The plain softmax relaxation

Recall the construction of Appendix B. Each argmax of Eq. 3 is replaced by a softmax at temperature $\tau > 0$, so that the two views of image n produce soft assignments on the probability simplex,

$$\boldsymbol{\pi}_n = \text{softmax}\left(\mathbf{y}_n^{(i)} / \tau \right), \qquad \boldsymbol{\rho}_n = \text{softmax}\left(\mathbf{y}_n^{(j)} / \tau \right). \tag{81}$$

Every count of Section 3.2 becomes a soft expectation,

$$\hat{q}_k = \frac{1}{N} \sum_{n=1}^{N} \pi_{n,k}\, \rho_{n,k}, \qquad p_k = \frac{1}{2N} \sum_{n=1}^{N} \left(\pi_{n,k} + \rho_{n,k} \right), \qquad q_k = p_k^2. \tag{82}$$

Also, the over-matching indicator becomes the smooth gate of Eq. 63, giving the differentiable objective of Eq. 65,

$$\mathcal{S}_{\text{soft}}(\theta;\tau,\varepsilon)=\sum\nolimits_k \sigma\left(\frac{\hat{q}_k-q_k}{\varepsilon}\right) D(\hat{q}_k \,\|\, q_k)\,. \tag{83}$$

We maximize Eq. 83 directly, taking $-\mathcal{S}_{\text{soft}}$ as the loss. We refer to this as the plain softmax approach.

C.2 The Gumbel-softmax relaxation

A Gumbel variable (Marques et al., 2015) between 0 and 1, denoted as $\text{Gumbel}(0,1)$, is obtained from a uniform variable by

$$g=-\log(-\log u)\,, \qquad u\sim \text{Uniform}(0,1). \tag{84}$$

Let $\mathbf{g}_n^{(i)}, \mathbf{g}_n^{(j)}$ be drawn i.i.d. from $\text{Gumbel}(0,1)$, independently for each image, each cluster (dimension) and each of the two views, and re-drawn at every step. Then, the assignments of Eq. 81 are replaced by

$$\boldsymbol{\pi}_n=\text{softmax}\Big(\mathbf{y}_n^{(i)}/\tau+\mathbf{g}_n^{(i)}\Big)\,, \qquad \boldsymbol{\rho}_n=\text{softmax}\Big(\mathbf{y}_n^{(j)}/\tau+\mathbf{g}_n^{(j)}\Big)\,. \tag{85}$$

Here, the Gumbel noise is added after the logit vector being divided by τ. Adding it before, i.e. $\text{softmax}\big((\mathbf{y}_n+\mathbf{g}_n)/\tau\big)$, is the more popular form. But that is inappropriate in our case: we apply ℓ_2-normalization to the output logit vector $\mathbf{y}_n$. So, $\mathbf{y}_n$ has entries of order $1/\sqrt{K}$ with measured standard deviation 0.12, while $\text{Gumbel}(0,1)$ has standard deviation 1.28. The Gumbel noise will then overwhelm the logit signal by an order of magnitude, making the training process extremely unstable and not reliable. As a result, we add the Gumbel noise after the logit being divided by τ.

Everything downstream is unchanged: Eq. 82 and Eq. 83 are applied to these $\boldsymbol{\pi}_n, \boldsymbol{\rho}_n$ exactly as in the plain softmax approach. That is, dropping $\mathbf{g}$ recovers the plain softmax approach.

By Definition 1, both approaches are exploitative: the parameter updates come from one gradient step, and no random factor is retained by performance-based selection.

C.3 Results

Both approaches use the same chessboard masking, the same independent augmentations (with the same hyper-parameter settings as described in Table 4 of Appendix F), the same ResNet-9 network backbone with $K=64$, and the same batch size N on each dataset as our main experiments. Only the training objective and the optimizer differ. We set $\tau=0.02$ and $\varepsilon=0.005$, and use Adam (Kingma & Ba, 2014) as the training optimizer with a learning rate 10^{-3}. Note that $\tau=0.02$ is the temperature in the softmax Eq. 81 and Eq. 85. One should not confuse it with the surprising cluster threshold τ used in formula 11.

Then, we train the model for 300 epochs (900 on USPS, whose far smaller training set gives correspondingly fewer gradient steps per epoch). End-to-end training is much more efficient than evolution strategies. As a result, we intuitively regard one end-to-end training epoch as equivalent to ten evolution epochs. Thus, we use 300 training epochs here for MNIST and FashionMNIST, corresponding to the 3000 total epochs on these two datasets in the main paper. For the same reason, we use 900 epochs for USPS, corresponding to its 9000 epochs in the main paper. Here, we report the mean $\pm$ standard deviation over five independent runs from fresh initializations.

Table 3 reports the comparison. Both exploitative approaches do learn: the surprise score can be optimized to a good extent by gradient descent. However, neither approach matches the converge-to-surprise scheme applying evolution strategy, which leads on eight of the nine columns — by 9.7 ACC points on MNIST, 3.6 on Fashion-MNIST and 10.7 on USPS. The only exception is the NMI on Fashion-MNIST, where the Gumbel-softmax is 0.3 points higher.

Furthermore, we observe the number of active clusters discovered by both exploitative approaches. We find that granularity is one major shortfall of both exploitative approaches, which under-split the datasets: they discover 8.6–9.8 active clusters on MNIST and Fashion-MNIST, while 7.4–9.4 on USPS. This is obviously lower than 10.1, 11.2 and 10.2 by our converge-to-surprise scheme.

Table 3: Differentiable (exploitative) versions of the surprise score against the converge-to-surprise scheme (%, mean ± std over five runs). All three rows share the identical data pipeline, network architecture and augmentation hyper-parameters. They differ only in the training objective and optimizer. The best mean per column is in **bold**.

	MNIST			Fashion-MNIST			USPS		
Approach	NMI	ARI	ACC	NMI	ARI	ACC	NMI	ARI	ACC
Plain softmax	$91.3_{\pm 3.7}$	$81.0_{\pm 9.0}$	$82.8_{\pm 8.4}$	$60.6_{\pm 3.5}$	$46.5_{\pm 4.4}$	$58.5_{\pm 6.2}$	$80.4_{\pm 4.3}$	$70.6_{\pm 9.0}$	$72.1_{\pm 8.1}$
Gumbel softmax	$93.3_{\pm 2.0}$	$87.6_{\pm 6.0}$	$88.2_{\pm 7.4}$	$\mathbf{64.3_{\pm 2.3}}$	$50.2_{\pm 3.9}$	$60.5_{\pm 4.2}$	$84.4_{\pm 5.3}$	$77.0_{\pm 9.7}$	$82.7_{\pm 8.3}$
conv.-to-surprise	$\mathbf{95.6_{\pm 0.7}}$	$\mathbf{96.1_{\pm 1.0}}$	$\mathbf{97.9_{\pm 1.3}}$	$64.0_{\pm 1.7}$	$\mathbf{51.9_{\pm 2.2}}$	$\mathbf{64.1_{\pm 1.8}}$	$\mathbf{89.4_{\pm 1.7}}$	$\mathbf{89.1_{\pm 3.0}}$	$\mathbf{93.4_{\pm 3.1}}$

Comparing to the plain softmax approach, the Gumbel-softmax is better everywhere, and most clearly on USPS ($+10.6$ ACC). Here is our explanation: the sampling noise of Gumbel-softmax perturbs the assignments at every step, keeping the model from settling early into a coarse partition. We regard this as a weak, undirected substitute for mutation-selection.

Both exploitative approaches are also less stable, with ACC standard deviations of 4–8 points, against 1.3–3.1 from converge-to-surprise scheme. In general, the explorative approach beats exploitative approaches on both performance and stability.

Finally, we find that in each independent run on each dataset, the differentiable surprise score stops growing in the last 20% epochs. Also, the experimental scores produced by model checkpoints from 80% epochs show no statistical difference comparing to those produced by fully trained models. Hence, 300 epochs on MNIST and Fashion-MNIST as well as 900 epochs on USPS are enough for the training of both exploitative approaches.

Although not a strict proof, experimental results in this appendix section are supportive to Hypothesis 1. That is, exploitative approaches may optimize a surprise score to some extent, but not fully. A full optimization on the surprise score likely requires explorative approaches, or the mutation-selection mechanism.

D Further discussions

The main paper presents converge-to-surprise as a novel self-supervised learning framework. Here, we step back and discuss the broader picture behind it. We first evaluate the surprise score through the lens of alignment and uniformity, a pair of properties shared by self-supervised objectives in general (Section D.1). Then, we argue that the null hypothesis $\mathcal{H}_0$ need not stay fixed: learning can be viewed as a repeated cycle in which the model rejects the current null hypothesis, folds what it discovered into an updated one, and searches again (Section D.2). Finally, we further evaluate the surprise score from an information theory point of view, showing that learning means to build order out of chaos (Section D.3).

These discussions are conceptual and supplemental. They are not required for the results in the main paper.

D.1 Alignment and uniformity

(Wang & Isola, 2020) identified two properties that a contrastive objective such as SimCLR (Chen et al., 2020) optimizes at once: *alignment* pulls the two views of one image together in the representation space, and *uniformity* spreads over the representations of different images as much as possible. Both are necessary, since alignment alone is satisfied by a constant encoder — that is, by representation collapse. Recently, (Mikulasch & Zenke, 2026) extended this reading to self-supervised approaches in general.

Our surprise score admits the same reading, with the two properties appearing as the two arguments of the binary Kullback–Leibler divergence in Eq. 8. The observed view-matching rate $\hat{q}_k$ is the alignment term, growing only when both views of one image fall into cluster k. The marginal clustering rate p_k is the uniformity term, where the distribution $\{p_k\}_{k=1}^{K}$ needs to spread over all

K dimensions in order to minimize the null view-matching rate $q_k = {p_k}^2$ for as many clusters as possible. The per-cluster surprise score $D(\hat{q}_k \,\|\, q_k)$ will increase if and only if both the alignment and uniformity properties are satisfied.

As we mentioned in Section 3.2, our surprise score $\mathcal{S}(\theta)$ naturally prevents the output representation from collapsing. From another perspective, this indicates that neither alignment nor uniformity itself can guarantee effective learning. If the model assigns every image to a single cluster (high alignment alone), then $p_1 = \hat{q}_1 = q_1 = 1$ and hence $\mathcal{S}(\theta) = 0$. On the other hand, if the clustering of both views are totally independent (high uniformity alone, which also means that the null hypothesis is really true), then there is no significant view-matching at all and hence $\mathcal{S}(\theta) \approx 0$.

D.2 Learning as iterative rejection of the null hypothesis

In the main paper, without any prior knowledge, we invoked the Principle of Maximum Entropy and assumed the data to be pure random noise. In our case, this means that every pixel is i.i.d. This gives us the null hypothesis $\mathcal{H}_0$. The purpose of our learning framework is then to discover *non-randomness*, or *surprise*, that rejects $\mathcal{H}_0$. However, this is only the starting point of a more general picture.

Once the model has discovered some non-randomness, we may fold that structure back into our description of the data. Concretely, we update the data distribution to account for what is newly found, obtaining an improved hypothesis $\mathcal{H}_1$ that is no longer pure noise: it already encodes the regularities discovered so far. We may now treat $\mathcal{H}_1$ as the new null hypothesis and repeat the process: further evolving and training the network to discover new surprise that rejects $\mathcal{H}_1$. If we succeed, we update the hypothesis again to $\mathcal{H}_2$, take it as the newer null hypothesis, and continue:

$$\mathcal{H}_0 \;\to\; \mathcal{H}_1 \;\to\; \mathcal{H}_2 \;\to\; \cdots \tag{86}$$

Each round explains more of the data, and each new null hypothesis is harder to reject than the last. The process terminates when no further surprise can be found, no matter what output representation we use or how we observe it. At that point, we accept the current hypothesis and stop the learning process.

In general, this refers to the meaning of 'learning', which we summarize as three principles:

1. **Maximum-entropy prior.** Without any prior knowledge, we must assume the data distribution to be totally random noise, following the Principle of Maximum Entropy. This is our initial null hypothesis.
2. **Learning as discovering surprise.** To learn is to discover *non-randomness*, or *surprise*, from the data distribution in order to reject the null hypothesis.
3. **Iterative updating.** We update the null hypothesis to incorporate what we have discovered. Then, we seek new non-randomness (new surprise) that can reject the updated hypothesis. If, after enough updates, no representation and no way of observing can reveal further surprise, we accept the current hypothesis and stop learning.

The converge-to-surprise framework in the main paper realizes a single round of this process. Fully iterating through this process is a natural extension of our framework.

D.3 Orderliness: building order out of chaos

Now, we evaluate a surprise score from an information theory point of view. Recall that in Section 3.2, we obtain the two cluster sequences from a batch of N images: $\mathrm{seq}^{(i)} = (\hat{y}_1^{(i)}, \ldots, \hat{y}_N^{(i)})$ and $\mathrm{seq}^{(j)} = (\hat{y}_1^{(j)}, \ldots, \hat{y}_N^{(j)})$. Pairing the two views of each image gives the cluster pair sequence

$$\Pi \;=\; \big((\hat{y}_1^{(i)}, \hat{y}_1^{(j)}), \ldots, (\hat{y}_N^{(i)}, \hat{y}_N^{(j)})\big), \tag{87}$$

whose observed distribution over the $K \times K$ cluster pair space is

$$\pi_{a,b} \;=\; \frac{1}{N} \sum_{n=1}^{N} \mathbf{1}\Big[\hat{y}_n^{(i)} = a \text{ and } \hat{y}_n^{(j)} = b\Big]. \tag{88}$$

Then, we define the *observed pair entropy* as

$$\hat{H}_{\Pi} \;=\; -\sum_{a,b:\,\pi_{a,b}>0} \pi_{a,b}\,\log \pi_{a,b}. \tag{89}$$

Suppose $n_a^{(i)}$ is the number of times cluster a appears in $\mathrm{seq}^{(i)}$, and $n_b^{(j)}$ is the number of times cluster b appears in $\mathrm{seq}^{(j)}$. Then, it is easy to see that

$$p_a^{(i)} = \sum_b \pi_{a,b} = n_a^{(i)} \,/\, N, \quad p_b^{(j)} = \sum_a \pi_{a,b} = n_b^{(j)} \,/\, N. \tag{90}$$

That is, unlike Section 3.2, we observe the marginal probability distribution of a cluster within each sequence.

Then, consider an inner-view shuffling: we can permute the entries within $\mathrm{seq}^{(i)}$ and within $\mathrm{seq}^{(j)}$ independently, as many times as we like. But $\mathrm{seq}^{(i)}$ and $\mathrm{seq}^{(j)}$ do not exchange entries. Such permutation leaves $\{p_k^{(i)}\}_{k=0}^{K-1}$ and $\{p_k^{(j)}\}_{k=0}^{K-1}$ unchanged, but scrambles the pairing and hence the cluster pair sequence Π.

Given the original Π, suppose all pair sequences that we can possibly obtain after inner-view shuffling form the set $\mathcal{R}_{\Pi}$. Then, we define the *shuffled-maximal entropy* as the largest attainable value in the form:

$$H_{\Pi}^{\star} = \max_{\tilde{\Pi}\in\mathcal{R}_{\Pi}} \hat{H}_{\tilde{\Pi}}. \tag{91}$$

That is, we calculate the observed pair entropy $\hat{H}_{\tilde{\Pi}}$ for each pair sequence $\tilde{\Pi} \in \mathcal{R}_{\Pi}$, and then take the maximum as $H_{\Pi}^{\star}$.

It is easy to see that $\hat{H}_{\Pi} \le H_{\Pi}^{\star}$ always holds true. Also, by a routine analysis, we can get that

$$H_{\Pi}^{\star} = H\Big(p^{(i)}\Big) + H\Big(p^{(j)}\Big), \;\; H(p^{(\cdot)}) = -\sum_k p_k^{(\cdot)} \log p_k^{(\cdot)}; \tag{92}$$

and

$$H_{\Pi}^{\star} - \hat{H}_{\Pi} \;=\; I\Big(\hat{y}^{(i)}; \hat{y}^{(j)}\Big) \;\ge\; 0. \tag{93}$$

Here, $I\big(\hat{y}^{(i)}; \hat{y}^{(j)}\big)$ is the mutual information between the original, un-shuffled $\mathrm{seq}^{(i)}$ and $\mathrm{seq}^{(j)}$, as described in Section 3.2.

Finally, we define the *orderliness* (or *neg-entropy*):

$$\Omega \;=\; \frac{H_{\Pi}^{\star} + \lambda}{\hat{H}_{\Pi} + \lambda}, \tag{94}$$

where $\lambda > 0$ is a small constant that keeps the ratio well-defined in degenerated cases. Since $\hat{H}_{\Pi} \le H_{\Pi}^{\star}$, we always have $\Omega \ge 1$, with larger values indicating more order.

Then, we describe two special cases:

1. **Fresh initialization:** Right after initializing the model, $\mathrm{seq}^{(i)}$ and $\mathrm{seq}^{(j)}$ are essentially independent. So, the observed pairing is already close to its most disordered form. Thus, we have $\hat{H}_{\Pi} \approx H_{\Pi}^{\star}$ and $\Omega \approx 1$. This matches $\mathcal{S}(\theta) \approx 0$: nothing surprising has been found.
2. **Collapse:** If the network collapses so that every view is mapped to a single cluster k_0, then only the pair (k_0, k_0) is ever observed. In this case, no shuffling can change the cluster pair sequence Π. Hence, $H_{\Pi}^{\star} = \hat{H}_{\Pi} = 0$ and $\Omega = \lambda/\lambda = 1$ always hold true. In other words, collapse is not learning.

Thus, genuine learning drives $\Omega \gg 1$, which by Eq. 94 requires the observed pair entropy $\hat{H}_{\Pi}$ to be much smaller than the shuffled-maximal entropy $H_{\Pi}^{\star}$. The latter demands diverse marginals (many clusters being observed), i.e. high potential disorder or high uniformity; the former demands the actual pairing to be highly dependent (both views of an image tend to agree), i.e. strong order or

strong alignment. In other words, to learn is to realize uniformity and alignment at the same time, or to keep the marginal diversity while imposing order on how these diversities are organized — to *build order out of chaos*.

Combining this with Section D.2, we arrive at an intuitive picture of the whole process: learning repeatedly discovers order out of chaos and folds what being discovered into an ever-stronger hypothesis, and continues until there is no more order left to discover.

E THE ALGORITHM OF CONVERGE-TO-SURPRISE SCHEME

Algorithm 1 shows the complete converge-to-surprise scheme:

Algorithm 1 Converge to Surprise

Require: images $\mathcal{X}$; population m; noise σ; learning rate η; weight decay λ; warm-up T_0; training periods $T_{1,a}$ and $T_{1,b}$; threshold τ.

initialize θ
for epoch $e = 1, 2, \ldots$ **do**
  sample N images; build complementary masked views and augment each independently $\rightarrow \tilde{\mathbf{X}}^{(i)}, \tilde{\mathbf{X}}^{(j)}$
  // ES outer step (explore)
  for $p = 1$ to $m/2$ **do**
    draw $\epsilon_p \sim \mathcal{N}(0, \sigma^2 I)$
    score $\theta \pm \epsilon_p$ using $\mathcal{S}(\cdot)$ (Eq. 8)
  end for
  shape the m scores into centered ranks r_p
  $\theta \leftarrow (1 - \eta\lambda)\,\theta + \frac{\eta}{m\sigma} \sum_p r_p\, \epsilon_p$ ▷ Eq. 10
  // gradient-descent training (consolidate)
  if $e > T_0$ and $e \,\%\, T_{1,b} = 0$ **then**
    on un-augmented views, take argmax labels and keep contributing positions $\mathcal{C}$ (Eq. 11)
    median-balance $\mathcal{C}$ across clusters into $\mathcal{B}$
    for epoch $e = 1, 2, \ldots, T_{1,a}$ **do**
      re-augment both views; update θ by minimizing $\mathcal{L}_{\text{ft}}(\theta)$ (cross-entropy to the surrogate labels)
    end for
  end if
end for
return θ

In summary, when the optimization begins, we do not have a clear training loss in each step to exploit. So, we just let the model explore and evolve toward the ultimate goal (the surprise score). After enough epochs, we may see some promising output representations, which in our case are the clusters with surprising view-matching results. Then, we train the network using these clusters as surrogate labels to further strengthen, or reinforce, the discovered promising representations.

F IMPLEMENTATION DETAILS

Network: Our model f_θ is a ResNet-9 (He et al., 2016): two convolutional stems (Aloysius & Geetha, 2017), two residual blocks (He et al., 2016), an adaptive average-pooling layer, and a single linear head that returns a logit vector with dimension $K = 64$. The cluster assignment of a view is the argmax over these logits. The output is ℓ_2-normalized before the argmax. We set $K = 64$, far above the ten ground-truth classes of every dataset. Since the model is never told the number of true classes, K only acts as an upper bound. The number of active clusters is discovered during training. The same architecture is used for all datasets, with a single input channel for an image view.

Datasets: We use three handwritten-digit and fashion-item benchmarks, all with ten ground-truth classes. **MNIST** (Deng, 2012) contains 70k images of handwritten digits at 28×28 resolution (10k for testing). **Fashion-MNIST** (Xiao et al., 2017) matches MNIST in size and resolution but depicts

ten clothing categories, and is substantially harder because several classes (e.g. pullover/coat/shirt) differ only in fine texture. **USPS** (Hull, 2002) contains 7,291 training and 2,007 test handwritten digits at a native 16×16 resolution, and is notably class-imbalanced.

Data augmentation: After the chessboard masking of Section 3.1, the two complementary views of each image are augmented independently, preserving the zero-mutual-information property between two views under $\mathcal{H}_0$. For MNIST and Fashion-MNIST, augmentations include random rotation, random cropping, and brightness/contrast jitter. Fashion-MNIST additionally applies a random horizontal flip with probability 0.5. Digits are not flipped, as a mirrored digit is not the same class.

For USPS, we add a single-axis zoom-out after cropping, which pads one randomly chosen axis with a black border before resizing back, producing an anisotropic (tall/short, wide/narrow) deformation. This is motivated by the imbalance of the USPS dataset: the digit-0 class is dominant and exhibits far more aspect-ratio variation than its MNIST counterpart. The anisotropic augmentation de-correlates this global shape so that same digits are less likely to be further split based on width or height. To leave room for the anisotropic deformation, the remaining augmentations on USPS images are kept milder than on MNIST/Fashion-MNIST images. Besides, USPS characters occupy a relatively larger fraction of the frame. So, we mask at the native 16×16 resolution, upsample each view independently to 32×32, and crop only down to a 30×30 window (i.e., a milder cropping range: $30 \sim 32$). Table 4 lists the exact per-dataset augmentation parameters referenced in Section 4.1.

Table 4: Per-dataset augmentation parameters, applied independently to each view after chessboard masking. Rotation angle is drawn uniformly from $[-r, r]$, and being applied with a rotation probability. Brightness/contrast are multiplicative jitters drawn from $[1 - b,\ 1 + b]$. The square cropping size (cropped edge length) is sampled uniformly from the listed range and resized back to $S \times S$. Also, edge length S is among the listed range of cropping. This means that although the cropping probability is set to 1, there is actually a chance of sampling S itself and implementing no cropping. Since USPS digits are relatively larger, the listed range of cropping is narrower. Also, for USPS, we first obtain the masked views from the original 16×16 image. Then, we upsample per view to 32×32. Additionally, we pad one random axis with 2–6 black pixels (zoom-out) independently on each view, which is only applied to USPS. A '–' denotes a disabled augmentation.

Parameter	MNIST	FMNIST	USPS
Side S	28	28	32
Rotation r (deg)	20	20	10
Rotation prob	1.0	1.0	0.5
Bright/contrast b	0.3	0.3	0.15
Crop edge	$\{24..28\}$	$\{24..28\}$	$\{30..32\}$
Crop prob	1.0	1.0	1.0
Flip prob	–	0.5	–
Zoom-out prob	–	–	1.0
Zoom-out range	–	–	$\{2..6\}$

Optimization and protocol: We optimize $\mathcal{S}(\theta)$ with the ES outer loop under a population size $m = 32$, interleaved with periodic gradient-descent training using the discovered surprising clusters. When identifying contributing positions according to formula 11, we choose $\tau = 0.005$. For each dataset, we run five independent experiments from scratch and report the mean $\pm$ standard deviation. At test time, only view-i (without augmentation) from chessboard masking on each test image is passed through the network, and assigned to its argmax cluster. The clustering results are then used in evaluations.

Training schedule: On MNIST and Fashion-MNIST, we use a two-stage training schedule: 2000 epochs of pure ES, followed by epochs 2000–3000 in which 4 gradient-descent training epochs are run every 25 ES epochs. On USPS we use a three-stage schedule: 4000 epochs of pure ES; then a weak stage over epochs 4000–8000, with 2 gradient-descent training epochs every 500 ES epochs; and finally a strong stage over epochs 8000–9000, with 4 gradient-descent training epochs every 25 ES epochs (identical to the other datasets). USPS is run for many more epochs because it has far fewer training images ($7{,}291$ vs. $70{,}000$), so the model needs more epochs to evolve.

Batch size: The surprise score is calculated over a batch of N images at each ES step. We use $N = 3000$ for MNIST/Fashion-MNIST, and $N = 3650$ for USPS. The latter is chosen so that the 7,291 USPS training images are split into two nearly equal-sized steps per epoch (3650 and 3641). This matters because each ES updating direction is estimated from the statistics of its batch alone. Then, a last step with only a few images gives unreliable statistics, steering the evolution into a poorly-estimated direction. So, we choose N carefully to avoid this.

Metrics and baselines: Following the non-parametric clustering protocol, we report three standard metrics: clustering accuracy (ACC) under the Kuhn-Munkres (Hungarian) algorithm (Kuhn, 1955), Normalized Mutual Information (NMI) (Kvålseth, 2017), and the Adjusted Rand Index (ARI) (Santos & Embrechts, 2009). For all three, higher is better, with more details introduced in (Ronen et al., 2022). We additionally report the inferred number of active clusters $\hat{K}$ (ground truth $K^\star = 10$). As for baselines, we use the scores reported by DeepDPM (Ronen et al., 2022) — including the performances of DeepDPM itself, performances of the classical non-parametric clusterers DBSCAN (Ester et al., 1996), performances of a memorized variational DPM model called moVB (Hughes & Sudderth, 2013), and performances of a DPM sampler. According to DeepDPM (Ronen et al., 2022), all these models are based on pre-trained autoencoder features. Also, performances of the three UNSEEN variants (Leiber et al., 2024) (UNSEEN+DCN (Yang et al., 2017), UNSEEN+DEC (Xie et al., 2016), and UNSEEN+DKM (Fard et al., 2020)), each of which is also based on a pre-trained autoencoder backbone, are used as baselines. In contrast to all these, our method is trained from scratch, with no pre-trained or separately-learned feature extractor.

Computation and hardware: All our experiments are conducted using one NVIDIA H200 GPU. We apply parallel computing within a single GPU: Given the ES population size $m = 32$, we duplicate the network parameters for 16 times, perturb each of them individually, and implement them on 16 duplicated batches of image views. So, one ES step requires $32/16 = 2$ iterations, and in each iteration there are $16/2 = 8$ mirrored pairs. It takes around 5 hours to fully optimize a ResNet-9 model on MNIST or FashionMNIST, and around 2 hours on USPS.

G Inferred number of clusters

As we mentioned in the main paper, our clustering results are obtained without telling the model the number of true classes. The number of active clusters $\hat{K}$ our method discovers stays close to the ground-truth value $K^\star = 10$: $10.1_{\pm 0.3}$ on MNIST, $10.2_{\pm 0.8}$ on USPS, and $11.2_{\pm 1.0}$ on Fashion-MNIST. Table 5 shows the inferred number of clusters by all baseline models.

Table 5: Inferred number of clusters $\hat{K}$ (mean $\pm$ std) on the three benchmarks, with the ground-truth value to be $K^\star = 10$. Baseline values are as reported by (Ronen et al., 2022). With no prior knowledge, our discovered number of active clusters are close to the ground truth.

Method	MNIST	Fashion-MNIST	USPS
DBSCAN (Ester et al., 1996)	$9.0_{\pm 0.0}$	$4.0_{\pm 0.0}$	$6.0_{\pm 0.0}$
DPM Sampler	$11.3_{\pm 0.8}$	$12.4_{\pm 1.0}$	$8.5_{\pm 0.9}$
moVB (Hughes & Sudderth, 2013)	$14.0_{\pm 1.0}$	$16.9_{\pm 2.3}$	$11.2_{\pm 1.1}$
DeepDPM (Ronen et al., 2022)	$10.0_{\pm 0.0}$	$10.2_{\pm 0.8}$	$9.2_{\pm 0.4}$
Ours	$10.1_{\pm 0.3}$	$11.2_{\pm 1.0}$	$10.2_{\pm 0.8}$

DeepDPM infers comparable counts on the same three benchmarks ($10.0_{\pm 0.0}$, $9.2_{\pm 0.4}$ and $10.2_{\pm 0.8}$ on MNIST, Fashion-MNIST and USPS, respectively. The following reports are in the same order), whereas the remaining non-parametric baselines reported by (Ronen et al., 2022) are markedly less accurate: the DPM Sampler gives $11.3_{\pm 0.8}$, $8.5_{\pm 0.9}$ and $12.4_{\pm 1.0}$; moVB over-estimates throughout, at $14.0_{\pm 1.0}$, $11.2_{\pm 1.1}$ and $16.9_{\pm 2.3}$; and DBSCAN under-estimates, collapsing Fashion-MNIST to only $4.0_{\pm 0.0}$ clusters (with $9.0_{\pm 0.0}$ on MNIST and $6.0_{\pm 0.0}$ on USPS).

Finally, Appendix H shows how the number of active clusters changes across epochs in our training process.

H NUMBER OF DISCOVERED CLUSTERS DURING TRAINING

Because the number of classes is never given to the model, it is instructive to watch how many clusters the model discovers as training proceeds. Recall that we introduced $D(\hat{q}_k \,\|\, q_k)$, the surprise score at cluster k, via formula 6 in Section 3.2. Then, we regard cluster k as *surprising* if $D(\hat{q}_k \,\|\, q_k) \geq \tau = 0.005$. Figures 2, 3 and 4 plot, for five independent runs per dataset, the number of surprising clusters after each ES epoch.

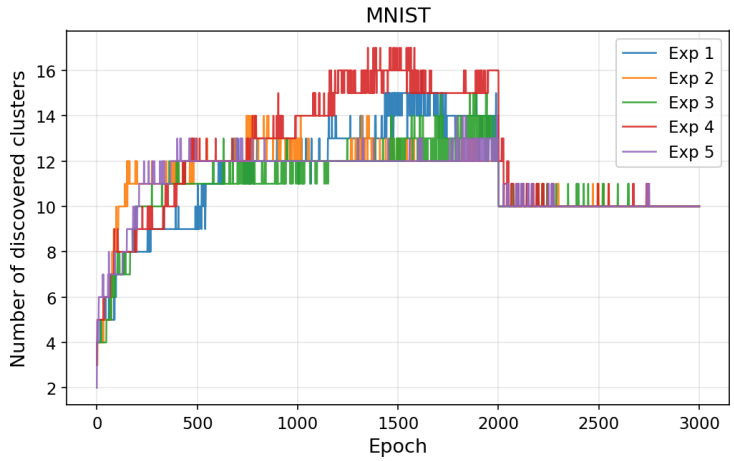


Figure 2: Number of discovered (surprising) clusters vs. training epoch on MNIST, for five independent runs. The model over-produces surprising clusters during pure ES stage. Then, the number of surprise clusters are consolidated back to 10 once gradient-descent training begins (two-stage schedule, starts from epoch 2000).

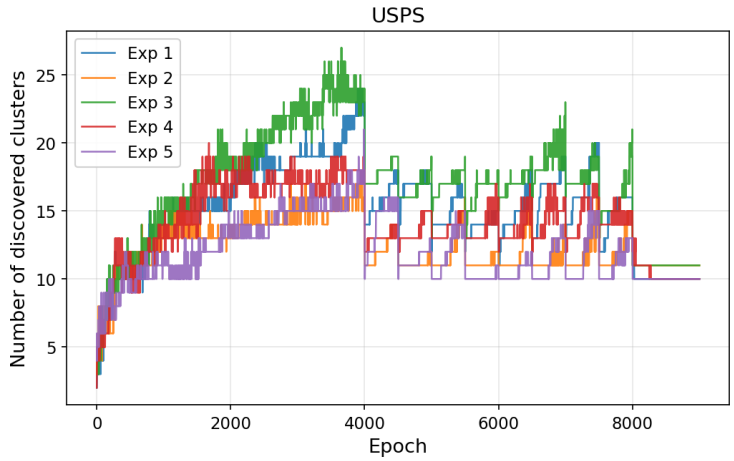


Figure 3: Number of discovered (surprising) clusters vs. training epoch on USPS, for five independent runs (three-stage schedule). Again, pure ES stage (0–4000) over-produces; weak training stage (4000–8000) yields the consolidation/re-exploration sawtooth; the final strong gradient-descent training stage settles the number of surprising clusters in each run to 10 or 11.

We can see that during the pure-ES stage, the model keeps discovering new candidate clusters, exceeding the number of ground truth classes. We call this over-production. Once gradient-descent training begins, each training round consolidates redundant clusters, pulling the count back down. Then, the evolution strategy slightly rebuilds surprising clusters on the cleaner partition. But finally, the evolution strategy and gradient-descent training reach the balance. No new surprising cluster is

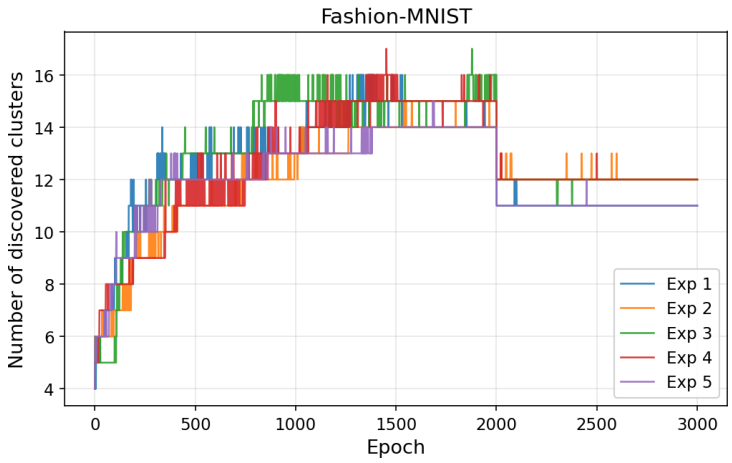


Figure 4: Number of discovered (surprising) clusters vs. training epoch on Fashion-MNIST, for five independent runs. The curves behave similarly as in the experiments on MNIST, whereas the final number of surprising clusters is slightly above 10. This is in line with the dataset's harder, texture-dominated structure.

discovered in the last 200 epochs. In general, results shown here coincide with our analysis in the ablation study 4.3.

On MNIST (Figure 2, two-stage schedule), the over-production peaks in the first 2000 epochs, and then collapses cleanly to 10 for all five runs. On USPS (Figure 3, three-stage schedule) the effect is more pronounced: the pure-ES stage (0–4000 epochs) over-produces up to around 25 clusters. In the weak stage (4000–8000 epochs, with 2 training epochs every 500 ES epochs), the ES optimization rebuilds back the number of surprising clusters after every gradient-descent training epoch, creating the sawtooth curves. Finally, the strong stage (8000–9000 epochs, with 4 training epochs every 25 ES epochs) settles the number of surprising clusters to 10 or 11. Fashion-MNIST (Figure 4) behaves similarly but stabilizes at a slightly larger count, reflecting its harder, texture-dominated data structure.

Once again, by converge-to-surprise, the model naturally discovers meaningful clusters from raw images without any prior knowledge.

I CLUSTERING VISUALIZATION

To qualitatively inspect what the network discovered, we visualize the test-set clustering of one trained model per dataset. For each active cluster k and each ground-truth class p, we draw a 3×3 grid of randomly selected test images that belong to cluster k and carry label p. We only draw such a grid when at least 50 such images exist (rare classes in each cluster are ignored). Each panel is captioned "Cluster k, digit p", or "Cluster k, item p".

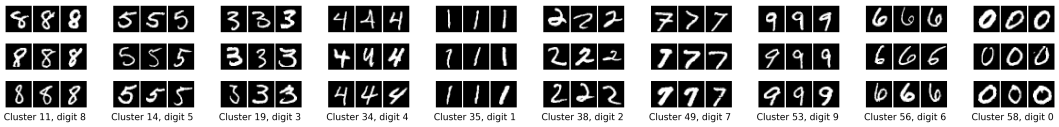


Figure 5: MNIST test-set clustering. Each panel shows nine random images of a single ground-truth class assigned to a given cluster. The ten active clusters map one-to-one to the ten digits, each at 97–99.7% purity.

MNIST: Figure 5 shows the MNIST clustering. The optimized network produces a clean one-to-one map: each active cluster corresponds to a single ground-truth class, and each class is captured by a

single cluster. This is confirmed by the cluster-purity report, where every active cluster is dominated by one digit class at 97% – 99.7% purity:

```
Cluster 35 | n=1134 | 99.0% (dom=cls1)
Cluster 38 | n=1043 | 98.6% (dom=cls2)
Cluster 19 | n=1036 | 97.0% (dom=cls3)
Cluster 49 | n=1031 | 98.5% (dom=cls7)
Cluster 53 | n= 997 | 97.1% (dom=cls9)
Cluster 58 | n= 980 | 99.4% (dom=cls0)
Cluster 56 | n= 968 | 98.3% (dom=cls6)
Cluster 34 | n= 968 | 98.9% (dom=cls4)
Cluster 11 | n= 958 | 99.7% (dom=cls8)
Cluster 14 | n= 885 | 99.0% (dom=cls5)
```

Each of the ten digits is represented by exactly one high-purity cluster, so Figure 5 contains exactly ten panels.

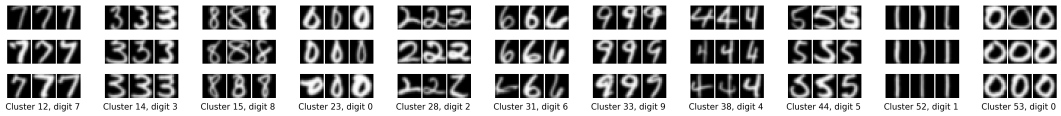


Figure 6: USPS test-set clustering for a run that inferred 11 clusters. Clusters 23 and 53 both capture the digit 0: cluster 53's zeros are round and plain, while cluster 23's are narrow, tall, and often carry an extra stroke — residual geometric variation the anisotropic augmentation did not fully absorb.

USPS: Next, Figure 6 shows a deliberately less-than-perfect USPS run, in which the model settled on 11 clusters instead of 10. Its per-cluster purity is still high, but two clusters share the same dominant class (digit 0), which is what pushes the count to 11:

```
Cluster 53 | n=313 | 97.1% (dom=cls0)
Cluster 52 | n=256 | 99.6% (dom=cls1)
Cluster 28 | n=199 | 95.0% (dom=cls2)
Cluster 38 | n=199 | 94.5% (dom=cls4)
Cluster 33 | n=188 | 93.1% (dom=cls9)
Cluster 15 | n=168 | 95.2% (dom=cls8)
Cluster 14 | n=166 | 97.6% (dom=cls3)
Cluster 31 | n=162 | 98.1% (dom=cls6)
Cluster 44 | n=157 | 95.5% (dom=cls5)
Cluster 12 | n=144 | 96.5% (dom=cls7)
Cluster 23 | n= 55 | 90.9% (dom=cls0)
```

Both cluster 23 and cluster 53 focus on the digit 0: the zeros in cluster 53 are relatively round and plain, whereas those in cluster 23 are narrow, tall, and often carry an extra stroke beyond the bare loop. This shows that the zoom-out (anisotropic) augmentation largely works as intended, but does not perfectly remove every unwanted topological and geometric variation of the dominant digit-0 class. This leads to further split of the digit-0 class.

Fashion-MNIST: Finally, Fashion-MNIST is far more challenging for non-parametric self-supervised image clustering, since several classes differ only in fine texture rather than global contour (Xiao et al., 2017). The per-cluster purity is correspondingly lower and some clusters mix classes:

```
Cluster 53 | n=1303 | 31.1% (dom=cls6)
Cluster 23 | n=1165 | 82.1% (dom=cls1)
Cluster 48 | n=1164 | 63.4% (dom=cls3)
Cluster  2 | n=1136 | 81.4% (dom=cls7)
Cluster  7 | n=1112 | 86.2% (dom=cls9)
Cluster 55 | n=1014 | 76.4% (dom=cls0)
```

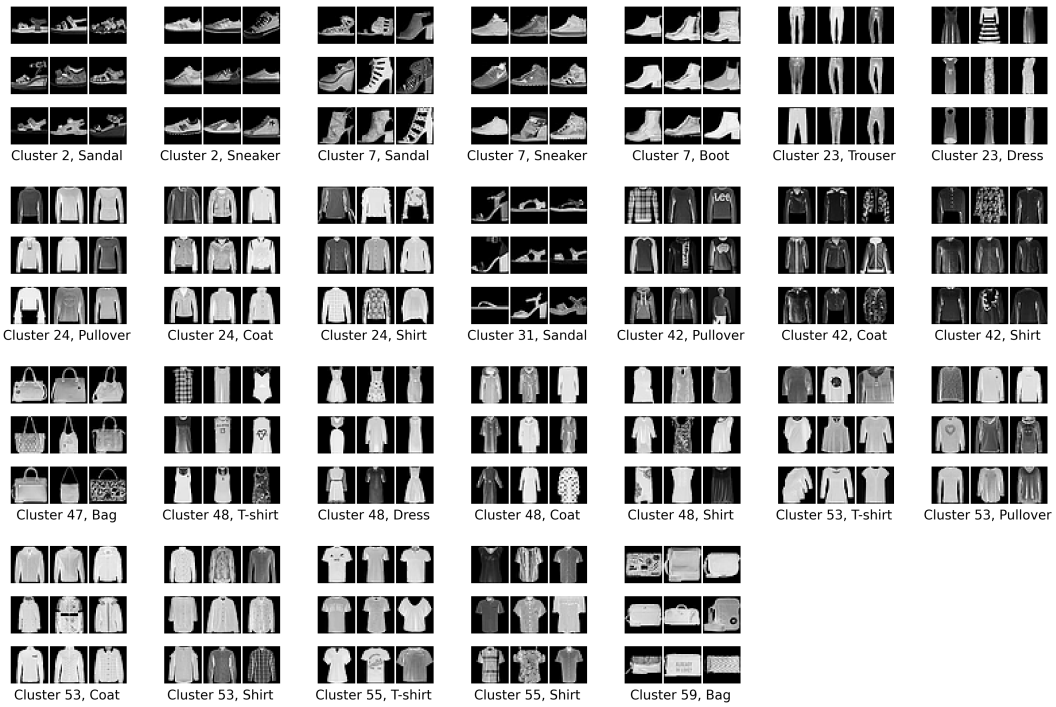


Figure 7: Fashion-MNIST test-set clustering. Beyond coarse categories, the network separates finer attributes: clusters 47 vs. 59 split bags by the presence of a handle/strap, and clusters 24 vs. 53 split upper-body garments by texture (patterned vs. plain) rather than contour.

```
Cluster 24 | n= 979 | 46.7% (dom=cls2)
Cluster 31 | n= 770 | 99.2% (dom=cls5)
Cluster 47 | n= 524 | 96.8% (dom=cls8)
Cluster 59 | n= 476 | 90.1% (dom=cls8)
Cluster 42 | n= 357 | 40.9% (dom=cls2)
```

Even so, the visualization in Figure 7 shows that the network discovers meaningful structure beyond the labels. Clusters 47 and 59 both isolate bags, but along a finer distinction: cluster 47 collects bags with a visible handle/strap, whereas cluster 59 collects bags with no (or only a very small, barely visible) handle. Clusters 24 and 53 both contain upper-body garments, but separate them by texture rather than contour: the garments in cluster 24 are more heavily textured/patterned, while those in cluster 53 are comparatively plain. In other words, despite using only a ResNet-9, converge-to-surprise optimization enables the network to distinguish both contour and texture.

Once again, although the per-cluster purity on Fashion-MNIST is lower than those on the other two datasets, we still achieve state-of-the-art performances on Fashion-MNIST.

J FAILURE MODE IN CIFAR10

Our scheme does not transfer successfully to CIFAR10 (Krizhevsky et al., 2009). We report this plainly, together with what we tried and what we believe the reasons are. The contrast between these failure modes and the successful results on digit and garment benchmarks is itself informative about when a surprise score aligns with semantics.

J.1 WHAT WAS TRIED

Every configuration in Table 6 uses the full converge-to-surprise scheme — an evolution strategy (ES) outer loop with periodic gradient-descent inner loop as described in Section 3.3 — with $K = 64$

and the same ResNet-9 backbone model (ResNet-34 is tried in one setting). But we make the batch size to be $N = 3125$ on CIFAR10, in which case the 50,000 training images can be equally divided into 16 batches each epoch. The major difference occurs in data augmentation: we use different methods to augment the colored views.

Table 6: Results (in percentage) on CIFAR10 by the converge-to-surprise scheme. Each row applies its unique data augmentation configuration or network backbone. $\hat{K}$ is the number of active clusters out of $K = 64$. The ground truth is $K^\star = 10$. Chance accuracy is 10%. We can see that every configuration lands in the same band — NMI 7–12, ACC 20–24 — regardless of how the color and texture cues are augmented or modified. Repeated runs of one configuration vary by about one NMI point. So, the differences between rows are of the same order as the noise.

View construction	input channels	NMI	ARI	ACC	$\hat{K}$
Per-view color tint, ResNet-34	3	9.1	5.7	23.0	6
Per-view color tint	3	8.8	5.5	22.5	6
Greyscale + smoothing, with rotation	1	6.9	4.1	20.0	6
Greyscale + smoothing, no rotation	1	6.7	3.4	19.6	9
Sobel, squashed encoding	2	11.4	5.6	20.5	12
Sobel, IIC-aligned jitter	2	**12.0**	**6.5**	**23.7**	10
Sobel, IIC-matched crop	2	10.3	5.8	21.4	6

In row 1 and 2 of Table 6, color tint gives each view its own additive color offset, so that absolute color cannot agree across views. Row 2 applies ResNet-34 instead of the standard ResNet-9 across our experiments. In row 3 and 4, greyscale removes color outright and fills the chessboard masked pixel positions with the mean of their visible neighbors. Row 4 further cancels rotation in each view. In row 5, 6 and 7, Sobel augmentation (Shorten & Khoshgoftaar, 2019) replaces each view by its two spatial-gradient channels. Following IIC (Ji et al., 2019), row 5 removes absolute brightness as well. Finally, row 6 and 7 additionally match IIC's photometric jitter and its crop aggressiveness.

Table 6 shows that none of this changes the outcome. Every configuration lands between 19.6% and 23.7% accuracy, against 10% of pure guessing and against the 98.3%, 64.5% and 94.9% our scheme reaches in Table 1. Different configurations move NMI by a few points, but never changes the character of the result. For reference, IIC reports 61.7% accuracy (ACC) on CIFAR10 with $K = 10$ given to the model.

In addition, the failure seems not to stem from the converge-to-surprise scheme itself. On the greyscale configuration, the surprise score reached 1.054 during pure ES, a value indistinguishable from the digit runs (1.21 on MNIST, 1.17 on Fashion-MNIST, 1.25 on USPS), while the mutual information with the ground-truth labels stayed near 0.07. That is, the model finds view-consistent features, which are surprising enough but unfortunately not semantic related.

J.2 Why it fails

The evolution strategy applied in this paper is preliminary, which is under-powered for natural images. Across all our experiments, each ES step estimates a parameter update direction from a population of $m = 32$ (with mirrored perturbation) in a parameter space of 107,320 dimensions. A zeroth-order estimate of this kind pays a cost in the ambient dimension that back-propagation does not, and what it can resolve is correspondingly coarse. On handwritten digits that suffices, because a digit essentially is its low-frequency shape, so the direction that improves the surprise score is also the direction that improves the clustering. On natural images, the features that separate a cat from a dog are far more complex than a 32-sample directional estimate can resolve. Thus, Table 6 shows that the parameter space search settles on whatever coarse statistic the two views share.

The obvious remedy is to enlarge the population substantially. However, to accelerate the optimization process, the entire population is evaluated in parallel in a single fused pass and its memory cost grows with m. With a single GPU (although the advanced NVIDIA H200), the memory consumption is unfeasible to us. On the other hand, if we remove the parallel computing and iterate on each mirrored perturbation pair, the total training time explodes. We regard this as a limitation of our approach. In the future, smarter evolution strategies will be explored.

The surprise score in this paper is preliminary, which provides insufficient distinguishing among views of different images. This is the deeper reason. Contrastive learning (Chen et al., 2020) pulls the two views of one image together and explicitly pushes views of different images apart. Self-distillation (Caron et al., 2021) achieves a comparable effect by centering and sharpening the teacher's output over a very large set of prototypes (output dimensions), so that views can hardly be predicted to the same output distribution.

Our surprise score does neither. It asks only this: the two views of the same image should agree more often than what the independence baseline $q_k = p_k^2$ predicts. Once any output cluster achieves any kind of surprising view-matching results, the objective is satisfied. Nothing else forces two different images to be set apart. With only $K = 64$ output dimensions and no such pressure, a partition that merely reflects a low-level cue shared by both views — global brightness layout, or edge-density layout after Sobel — scores as well as a semantic one. On digits or garment items this is harmless, since the easiest view-consistent feature happens to be the class. On natural images, there is a whole ladder of view-consistency. Non-semantic features occupy the low level and are much easier to capture. Hence, the evolution never manages to climb the ladder to reach at the semantic level.

Still, some perspective is due. Again, to the edge of our knowledge, our converge-to-surprise scheme is the first explorative SSL approach trained from scratch for self-supervised image clustering. Proposing a novel self-supervised learning framework is never easy, and making it scale to every dataset at the first attempt is even harder. We claim that the failure on CIFAR10 comes from two technical deficiencies — a surprise score that is too sparse, and an evolution strategy that is too naive — rather than from the converge-to-surprise principle itself.

Both issues have drawn our attention, and both point to concrete next steps. On the objective, we will study surprise scores over denser output representations: several heads at different granularities, or a factorized code whose number of states far exceeds $K = 64$. On the optimizer, we will replace the fixed perturbation scale by step-size and covariance adaptation (Wierstra et al., 2014), and add an acceptance test, so that one generation cannot move the parameters when every sampled candidate scores worse than the current ones.

Anyway, the new state-of-the-art performances on the digit and garment benchmarks show the effectiveness of the converge-to-surprise principle. Extending that effectiveness to natural images is our next step, rather than a refutation of the framework.