Hierarchical Models:
Intrinsic Separability in High Dimensions

Wen-Yan Lin
daniellin@smu.edu.sg

Abstract

It has long been noticed that high dimension data exhibits strange patterns. This has been variously interpreted as either a “blessing” or a “curse”, causing uncomfortable inconsistencies in the literature. We propose that these patterns arise from an intrinsically hierarchical generative process. Modeling the process creates a web of constraints that reconcile many different theories and results. The model also implies high dimensional data possesses an innate separability that can be exploited for machine learning. We demonstrate how this permits the open-set learning problem to be defined mathematically, leading to qualitative and quantitative improvements in performance.

1. Introduction

Despite the widespread use of high dimensional data in computer vision and pattern recognition, understanding of high dimensions has proven elusive. In the dominant mathematical paradigm, high dimensions are “cursed”. The “curse” is rooted in a stochastic model where all instances are generated from some distribution-of-everything. This causes almost all instances to be equi-distant to any test point, making them indistinguishable from each other and any machine learning impossible [1, 2, 14].

Yet, effective high dimensional machine learning does exist. Further, it utilizes the same equations used to derive the “curse”. However, by assuming multiple generative distributions [34, 25, 42], the “curse” turns into a “blessing” that disentangles distribution instances. Both “curse” and “blessing” papers appear mathematically sound but their conclusions are contradictory and their assumptions are not mutually exclusive. This creates a puzzle.

We believe “curse” and “blessing” both stem from patterns arising from an intrinsically hierarchical data generation process. This would permit both their assumptions to be simultaneously true and may also explain why hierarchies are a recurring theme in language [38, 40], data-structures [43] and clustering algorithms [41]. To understand this phenomenon, we develop a high dimensional hierarchical-model. The model unifies “curse” and “blessing” into an elegant web of constraints that links all generative distributions to each another and ultimately to a distribution-of-everything, forming an over-arching framework that reconciles many theories and results.

Beyond explanatory power, the hierarchical-model also predicts almost-all instances of a distribution and its sub-distributions, lie on a distinctive-shell that other instances almost-never enter. Such distinctive-shells can be estimated from instances of the distribution. This relates current observations to all other potential observations, creating a mathematical formulation for open-set learning!

Open-set learning is one of the most difficult machine learning problems. In it, a classifier trained on a small number of labeled classes must deal effectively with a much larger set of unlabeled classes [31, 45, 5]. Open-set learning is critical to handling unforeseen circumstances gracefully. However, it is relatively unstudied because, before the hierarchical-model, it appeared difficult (if not impossible) to make reasonable a priori assumptions on the properties of all possible unseen classes.

We formulate the classic open-set problem of one-class
learning in terms of the hierarchical-model. This creates a shell-learner that is algorithmically similar to one-class SVM [9]. However, unlike traditional one-class learning, it can approximate the absolute probability of a class given an image.

Such absolute scores are essential to operating in unknown environments. It also means scores of independently trained shell-learners can be combined coherently without re-training. This is especially important in life-long learning [22] where constant re-training can cause potentially catastrophic forgetting and interference problems [32]. An example is illustrated in Fig. 1.

In summary, this paper:
• Proposes a hierarchical-model that unifies “curse” and “blessing” approaches in high dimensions;
• Uses the hierarchical-model to predict the existence of distinctive-shells that make open-set problems amenable to mathematical analysis;
• Demonstrates a one-class learner which estimates the probability of a class given an image. Previously one-class learners only provided relative scores.

1.1. Related Works

This paper lies at the intersection of many research fields. Its formulation follows the long tradition of statistical machine learning. Examples include Expectation Maximation [13, 28, 12], Latent Dirichlet Allocation (LDA) [6, 35, 39], and Bayesian inference [7, 18, 17]. However, data density vanishes as dimensions increase, making probability density estimation ill-conditioned [36]. Thus, ever, data density vanishes as dimensions increase, making probability density estimation ill-conditioned [36].

While high dimensions are often mathematically intractable, computer vision has successfully utilized them in deep-learning. Examples include deep-learned image features [3, 21, 37], Generative Adversarial Networks [15, 20] and Variational Autoencoders [23]. Results are undoubtedly good but interpretation has proven difficult. We hope hierarchical-models make high dimensions and (indirectly) deep-learners more interpretable.

Finally, the hierarchical-model provides a statistical interpretation of high dimension Euclidean distances. As many data projection algorithms [33, 26, 27]) seek to minimize or preserve Euclidean distance, this has implications on the interpretation of their results as discussed in Sec. 7.2 of the supplementary.

2. Distributions in High Dimensions

Our approach is based on the analysis of high dimensional distributions. Following the definition in [25], a high dimensional distribution is one whose random vectors are quasi-ideal, i.e. there are a large number of dimensions, most of whom are independent. This is elaborated below, with notations adapted from [25].

Definition 1.

• \(d^2(\cdot)\) denotes an operator for normalized squared \(\ell_2\) norm, such that for \(x \in \mathbb{R}^k\), \(d^2(x) = \frac{|x|^2}{\mu^2}\). If \(x\) is the difference between two vectors, we refer to \(d^2(x)\) as normalized squared difference (NSD);
• \(d(\cdot)\) is the normalized \(\ell_2\) norm operator, \(d(x) = \sqrt{d^2(\cdot)}\);
• \(S(\mu, r)\) denotes a thin shell centered at \(\mu\), with radius \(r\).

Let \(Z = [Z[1], Z[2], \ldots, Z[k]]^T\) denote a \(k\) dimensional random vector where \(Z[i]\) is a random variable,

• \(Z\) is high dimensional if and only if \(Z\) is quasi-ideal [25], i.e., as dimension \(k \to \infty\), each dimension \(Z[i]\) has finite fourth moment and a finite number of pairwise dependencies.
• \(d^2(\cdot)\) operator can be applied to random vectors. \(d^2(Z) = \frac{1}{k^2} \sum_{i=1}^k Z[i]^2\) is a random variable formed by averaging \(z\)’s squared elements;
• \(\mu_Z = E(Z) = [E(Z[1]), \ldots, E(Z[k])]^T\) is a vector of each dimension’s expectation;
• \(\nu_Z = \sum_{i=1}^k \text{var}(Z[i])\) is the average variance;
• \(\nu_Z \equiv \text{a.s.}\) denotes almost-surely-equal. Thus, the relation \(P(d(X-c) = t) \to 1\) as \(k \to \infty\), is written \(d^2(X-c) \equiv \text{a.s.} t\).

Unit-vector-normalization. As dimension \(k \to \infty\), unit-vector-normalization causes individual entries to tend to 0. For unit-vector-normalized data, the definitions of \(d^2(\cdot)\) and \(\nu_Z\) are modified to avoid dividing by \(k\):
• \(d^2(\cdot)\) is a squared \(\ell_2\) norm, such that \(d^2(Z) = \|Z\|^2\);
• \(\nu_Z = \sum_{i=1}^k \text{var}(Z[i])\), is the total variance.

Let \(A\) and \(B\) be two independent, high dimensional random vectors, with respective mean and average variances \(\mu_A, \mu_B\) and \(\nu_A, \nu_B\). Due to the law of large numbers, the normalized squared difference between instances of \(A\) and \(B\) almost-surely depend, only on their mean and average variance.

\[
d^2(A - B) \equiv \nu_A + \nu_B + d^2(\mu_A - \mu_B). \quad (1)
\]

This is key to understanding high dimensions.

Replacing \(B\) in Eq. (1) by a distribution of mean \(c\) and zero variance, yields

\[
d^2(A - c) \equiv \nu_A + d^2(\mu_A - c), \quad \forall c \in \mathbb{R}^k. \quad (2)
\]

Let the distribution-of-everything represent a hypothetical, generative distribution that fully explains the creation of all natural images. If \(A\) represents the distribution-of-everything, Eq. (2) means almost-all instances are equidistant to any point \(c\). This has been used to argue that high-dimensions are “cursed” as it supposedly makes instances indistinguishable from each other [2]. If true, the
“curse” implies no machine learning algorithm can work in high dimensions. However, if we consider a more complex hierarchical generative model, the conclusion changes.

Our hierarchical-model is based on three assumptions:
1) Images are instances of some generative distribution; 2) Generative distributions are high dimensional; 3) Except for the distribution-of-everything, each generative distribution is a sub-distribution of some parent.

The “curse” does not directly apply to hierarchical-models as it cannot account for pairwise distances between instances of the same sub-distribution. This is because Eq. (2)’s almost-sure-equality permits exceptional events of infinitesimally small probability. Thus if each sub-distribution accounts for only an infinitesimal fraction of A’s instances, the pairwise distance between its instances could be exceptional.

While specific sub-distributions may violate Eq. (2), in aggregate, they must conform to Eq. (2). This creates a web of constraints linking all generative distributions (and images) to each other and to the distribution-of-everything.

3. The Hierarchical-Model

We begin by defining the relation between a distribution and its sub-distributions. The definition is then applied recursively to create the hierarchical-model.

Let \( A \) be a high-dimensional random vector with probability density function \( f_A(a) \), which represents a distribution-of-everything. \( f_A(a) \) is a compound distribution, i.e., it is distributed according to some sub-distributions, with these distributions’ parameters themselves being random variables. \( \Theta = \{V_\Theta, M_\Theta, \ldots \} \) is a set of random vectors representing sub-distribution parameters. \( V_\Theta \) and \( M_\Theta \) represent the average variance and mean parameters respectively. We write \( A = A_\Theta \) to denote that parent distribution \( A \) is explained by its sub-distribution parameters \( \Theta \) such that:

\[
f_A(a) = f_{A_\Theta}(a) = \int_{\Theta} f_\Theta(\theta) f_{A_\Theta}(a|\theta) d\theta. \tag{3}
\]

where each sub-distribution is itself high-dimensional.\(^1\)

Let \( a_\Theta \) be an instance of \( A_\Theta \), it is generated by a two-step process: 1) Generate an instance of sub-distribution parameters \( \theta \) from \( \Theta \). The random vector associated with \( \theta \) is \( A_\Theta \); 2) Generate instance \( a_\Theta \) from \( A_\Theta \).

We recursively define sub-distributions of sub-distributions through a hierarchical random process

\[
A_{\Theta^n} = (((((A_{\Theta^{[0]}}))_{\Theta^{[1]}})_{\Theta^{[2]}})_{\Theta^{[3]}} \ldots)_{\Theta^{[n]}} = A,
\]

where operator \( \Theta^{[i]} \) represents the random sampling of a sub-distribution, which in turn forms the parent distribution for \( \Theta^{[i+1]} \). \( A_{\Theta^n} \) denotes an n level hierarchical-process, with \( A_{\Theta^{[0]}} = A \).

If \( A_{\Theta^n} \) and \( A_{\Theta^m} \) denote independent runs of a hierarchical-process, starting from \( A \), from Eq. (1),

\[
d^2(A_{\Theta^n} - A_{\Theta^m}) \overset{a.s.}{=} 2v_A, \quad m, n \in \mathbb{Z}^+_0. \tag{4}
\]

The instantiated sub-distribution parameters of a hierarchical-process are denoted as

\[
\theta^n = \theta^{[0]} \theta^{[1]} \theta^{[2]} \ldots \theta^{[n]},
\]

where \( \theta^{[0]} \) is the parameter of \( A \). \( \theta^n \) records the distribution parameters’ values at every level, until the final distribution \( A_{\Theta^n} \). Since the final random vector is completely characterized by the final distribution parameters, \( A_{\Theta^n} = A_{\Theta^{[n]}} \).

A hierarchical-model is a series of hierarchical processes, where at any \( i^{th} \) stage, the \( A_{\Theta^n} \) random vector can be a parent of multiple, new hierarchical processes.

In the hierarchical-model, any two data-points are instantiations of two independent hierarchical-processes starting from some, most recent common ancestor. Let \( A_{\Theta^n} \) denote their common ancestor and \( A_{\Theta^{[0]}}, A_{\Theta^{[1]}, \ldots} \) their distributions. From Eq. (4), the normalized squared difference (NSD) between any two instances is:

\[
d^2(A_{\Theta^n} - A_{\Theta^{[n]}}) \overset{a.s.}{=} 2v_{\Theta^{[n]}}, \quad \forall i, j, n \in \mathbb{Z}^+_0. \tag{5}
\]

This is summarized in Theorem 1.

**Theorem 1. (Pairwise Difference)** The hierarchical-model predicts that normalized difference between any two data-points is almost-surely \( \sqrt{2v} \), where \( v \) is the average variance of their most recent common ancestor distribution.

Figure 2 illustrates a numerical example of the pairwise NSD. Similar to the “curse” \([2]\), this formulation begins with a distribution-of-everything. However, the NSD is not a constant. Instead, there is a defined pattern of distances which algorithms can exploit as a “blessing”. The distance patterns also ensure Euclidean distance remains meaningful in high dimensions, as elaborated in the supplementary.

In practice, NSD is very often equal to \( 2v_A \) as the most recent common ancestor is often the distribution-of-everything. This creates a paradox in which algorithms working directly with the affinity matrix experience a “curse” but those dealing with nearest-neighbor distance do not.

3.1. Parameter Constraints

This subsection relates distribution parameters across different hierarchies of the hierarchical-model, creating the key constraints exploited in later sections.

Thus far, a sub-distribution with identical parameters to its parent is considered valid. Without loss of generality, we
That is, if \( \theta^{[i]} \) and \( \theta^{[j]} \) are distribution parameters from the same non-trivial hierarchical-process, almost-surely,

\[
v_{\theta^{[i]}} > v_{\theta^{[j]}}, \quad \forall i < j, \ i,j \in Z_0^+.
\]

Proof. This follows from Theorem 2 and Eq. (6). \( \square \)

Manipulating Eq. (2), Theorem 2 and Corollary 1 leads to distinctive-shells, the paper’s main result.

3.2. Distinctive-Shells

Consider a non-trivial hierarchical-process that starts from the distribution-of-everything, \( A \) and terminates at \( A_{\theta^n} \). Instances of \( A_{\theta^n} \) and all its sub-distributions, \( A_{\theta^{m|\Theta}} \), are labeled \( \alpha \).

From Eq. (2), \( \forall c \in \mathbb{R}^k \)

\[
d^2(A_{\theta^{m|\Theta}} - c) \overset{a.s.}{=} d^2(A_{\theta^n} - c) \overset{a.s.}{=} v_{\theta[n]} + d^2(\mu_{\theta[n]} - c). \tag{10}
\]

Thus instances of \( \alpha \) are almost-always members of the set of shells, \( \{S(c, \sqrt{v_{\theta[n]}} + d^2(\mu_{\theta[n]} - c)) \mid c \in \mathbb{R}^k \} \). The minimum radius shell, \( S(\mu_{\theta[n]}, \sqrt{v_{\theta[n]}}) \), is termed the distinctive-shell and is of unique significance.

We represent the ancestors of \( A_{\theta^n} \) as \( A_{\theta^m} \), \( m < n \). All possible non-\( \alpha \) instances can be represented by \( A_{\theta^{m|\Theta}} \). Equation (2) relates all such instances to \( \mu_{\theta[n]} \) through

\[
d^2(A_{\theta^{m|\Theta}} - \mu_{\theta^n}) \overset{a.s.}{=} d^2(A_{\theta^m} - \mu_{\theta^n}) \overset{a.s.}{=} v_{\theta[m]} + d^2(\mu_{\theta[m]} - \mu_{\theta[n]}). \tag{11}
\]

From Theorem 2, almost-surely,

\[
v_{\theta[n]} + d^2(\mu_{\theta[m]} - \mu_{\theta[n]}) = v_{\theta[m]}, \quad \forall m < n, \ m,n \in Z_0^+. \tag{12}
\]

Combining Eq. (12) and (11), almost-surely,

\[
d^2(A_{\theta^{m|\Theta}} - \mu_{\theta[n]}) \overset{a.s.}{=} 2v_{\theta[m]} - v_{\theta[n]}, \quad \forall m < n, \ m,n \in Z_0^+. \tag{13}
\]

Thus for each \( A_{\theta^n} \) ancestor of \( A_{\theta^n} \), Corollary 1 almost-surely implies

\[
d^2(A_{\theta^{m|\Theta}} - \mu_{\theta[n]}) \overset{a.s.}{>} d^2(A_{\theta^{m|\Theta}} - \mu_{\theta[n]}) \overset{a.s.}{>} \ldots \overset{a.s.}{>} d^2(A_{\theta^{m|\Theta}} - \mu_{\theta[n]}) \overset{a.s.}{=} v_{\theta[n]}.
\]

 Recall that \( A_{\theta^{m|\Theta}} \) represents all possible non-\( \alpha \) instances. Thus Eq. (14) guarantees non-\( \alpha \) instances almost-surely fall outside the distinctive-shell. Further, Eq. (10) enables us to estimate the distinctive-shell by fitting the tightest possible shell to instances of \( \alpha \). This makes it possible to algorithmically relate current observations with all other potential observations!
4. Shell Based Learning

This section applies the theory of distinctive-shells to practical open-set problems, where a concept detector trained on limited data must generalize to unknown environments.

4.1. Formulation

Unit-Vector-Normalization

Despite the complexity of the image formation process, generative distributions of digital images cannot be modeled as high dimensional. This is because the brightness of all pixels in an image is correlated by a common exposure setting, making dimensions dependent. Exposure induced scaling can be removed by unit-vector-normalization that divides each data point by its \( \ell_2 \) norm, converting all data to unit vectors.

Suppose \( \mathbf{A} \) is a hierarchical-model’s distribution-of-everything. If each \( a_i \) instance, is perturbed by a random scalar such that \( a_i = s_i a_i \), after unit-vector-normalization, \( \hat{a}_i = \frac{a_i}{\|a_i\|} \), which is identical to unit vector normalization on the un-perturbed instances. Thus, the effect of normalization of perturbed instances can be understood by analyzing the normalization of un-perturbed instances.

Setting \( c \) to a zero vector, \( \mathbf{0} \), in Eq. (2) yields

\[
d^2(\mathbf{A}_{\theta^+}) \atop a.s. \Rightarrow \mathbf{d}^2(\mathbf{A}) \atop a.s. \Rightarrow \mathbf{v}_\mathbf{A} + d^2(\mu_\mathbf{A}) = \lambda_\mathbf{A},
\]

where \( \lambda_\mathbf{A} \) is a constant. Thus, unit-vector-normalization of hierarchical-model data, is almost-surely equivalent to scaling the entire hierarchical-model by a constant factor.

From the definition of \( d^2(\cdot) \) and Eq. (15),

\[
d^2(\mathbf{A}) = \frac{\|\hat{\mathbf{A}}\|^2}{k} \overset{a.s.}{=} \lambda_\mathbf{A} \Rightarrow \|\mathbf{A}\| \overset{a.s.}{=} \lambda_\mathbf{A} \sqrt{k},
\]

where \( k \) is the number of dimensions. If \( \hat{\mathbf{A}} \) be the normalized version of \( \mathbf{A} \), after normalization, \( \|\hat{\mathbf{A}}\|^2 \overset{a.s.}{=} \frac{\|\mathbf{A}\|^2}{\lambda_\mathbf{A} k} \).

Since \( k \) is already part of this normalization, for post unit-vector-normalized data, we modify the definition of \( d^2(\cdot) \) and \( v \) operators, detailed in Sec. 2, to avoid another division by \( k \). This extends all previous high dimensional analysis to post unit-vector-normalized data.

Statistical Framework

Let \( x \in \mathbb{R}^k \) be a data point and \( y \in \mathbb{Z}_+^l \) some label. The goal of learning is to estimate \( p(y|x) \) for all \( y \) and \( x \).

If instances of a \( \mathbf{A}_{\theta^+} \) correspond to label \( \alpha \), from Eq. (14), \( \alpha \) instances almost-surely lie on distinctive-shell \( S(\mu_{\theta^{\alpha}}, \sqrt{\sigma_{\theta^{\alpha}}}) \), which exclude almost-all other instances. Thus, if \( x \) is modified to be the distance from the distinctive-shell, we have the following:

\[
\begin{align*}
p(y = \alpha|x = 0) = 1, & \quad p(x = 0|y = \alpha) = 1 \\
p(y \neq \alpha|x = 0) = 0, & \quad p(x = 0|y \neq \alpha) = 0 \\
p(y = \alpha|x \neq 0) = 0, & \quad p(x \neq 0|y = \alpha) = 0 \\
p(y \neq \alpha|x \neq 0) = 1, & \quad p(x \neq 0|y \neq \alpha) = 1
\end{align*}
\]

This can be summarized as \( p(y = \alpha|x) = p(x|y = \alpha) \). In practice, we use:

\[
p(y = \alpha|x) \approx p(x|y = \alpha).
\]

While Eq. (18) does not represent all perturbations, it is still a remarkable approximation as it frees us from assuming a prior (that most statistical formulations require). This is key to permitting open-set formulations for unknown operating environments.

Shell Fitting

Given \( l \) instances of class \( \alpha \), \( \{f_0, f_1, \ldots, f_l\} \), from Eq. (14), the distinctive-shell \( S(\mu_{\theta^{\alpha}}, \sqrt{\sigma_{\theta^{\alpha}}}) \) is estimated by minimizing

\[
\arg\min_{\{\mu, v\}} \frac{1}{l} \sum_{i=1}^l ||f_i - \mu||^2 - v^2 + \lambda||v||^2.
\]

\( \mu, v \) are the respective estimates of \( \mu_{\theta^{\alpha}}, v_{\theta^{\alpha}} \) and \( \lambda \) is a regularizer that encourages small shells. Let \( x_i = ||f_i - \mu||^2 \), \( p(x|\alpha) \) is estimated by applying a parzen window to \( x_i \) instances. This approximates \( p(\alpha|x) \) in Eq. (18).

Note: One-class SVM libraries [9] can also estimate \( \mu \) but minimizing Eq. (19) is better in extreme cases, as shown in supplementary Sec. 8.3.

4.2. Re-normalization

This section shows the impact of coordinate frame choice on learning. We assume data is unit-vector-normalized.

Similar to Sec. 3.2, let \( \mathbf{A}_{\theta^+} \) be some ancestor distribution of \( \mathbf{A}_{\theta^+} \). From Eq. (13), the gap in distinctive-shell distance between instances of \( \mathbf{A}_{\theta^+} \) and \( \alpha \) instances of \( \mathbf{A}_{\theta^-} \) is:

\[
G_{mn} = d^2(\mathbf{A}_{\theta^+}, \alpha - \mu_{\theta^{\alpha}}) - d^2(\mathbf{A}_{\theta^-}, \alpha - \mu_{\theta^{\alpha}}) \overset{a.s.}{=} 2(v_{\theta^{\alpha}} - v_{\theta^{\alpha}}), \quad m \leq n.
\]

For convenience, we shorten the expression to

\[
g_{mn} = 2(v_{\theta^{\alpha}} - v_{\theta^{\alpha}}), \quad m \leq n.
\]

In theory, a finite gap suffices for separability. However, due to noise, ensuring the largest possible gap is important. Gaps can be altered through re-normalization, where each dimension is translated by some value and resultant vectors unit-normalized again to magnitude 1.

Given the mean of the distribution-of-everything, \( \mu_{\mathbf{A}} \), we can re-normalize by subtracting \( \mu_{\mathbf{A}} \) from all instances and dividing by their new magnitude. From Eq. (2),

\[
d^2(\mathbf{A}_{\theta^+} - \mu_{\mathbf{A}}) \overset{a.s.}{=} \mathbf{v}_{\mathbf{A}}.
\]

Thus this step is equivalent to re-centering almost-all instances and dividing them by a scalar \( s_{\mathbf{A}} = \sqrt{\mathbf{v}_{\mathbf{A}}} \). Let \( \hat{\mathbf{A}}_{\theta^+}, \hat{\mathbf{A}}_{\theta^-} \) be the re-normalized sub-distributions. The gap becomes:

\[
g_{mn} = \frac{2(v_{\theta^{\alpha}} - v_{\theta^{\alpha}})}{s_{\mathbf{A}}} = \frac{2(v_{\theta^{\alpha}} - v_{\theta^{\alpha}})}{\mathbf{v}_{\mathbf{A}}}, \quad m \leq n.
\]
As data has been unit-vector-normalized, \( d^2(\mathbf{A} - 0) \) is bounded. \( v_A \) is any \( v_A \) such that \( v_A + d^2(\mu_A - 0) = 1 \) and \( v_A \leq 1 \). Thus, re-normalizing with \( \mu_A \) is guaranteed to not reduce the gap. If \( \mu_A \) is far from zero, \( v_A \) can be very small, causing a corresponding huge improvement in the gap for almost-all pairs of distributions \( m, n \). Re-normalization with \( \mu_A \) is identical to standard normalization procedures in machine learning and helps explain the importance of this step in general machine-learning.

The gap can be further manipulated if we have knowledge of the ancestors of \( A_{\alpha} \). This is summarized as follows:

**Corollary 2.** (Re-Normalization) Let \( A_{\alpha^l} \) be some ancestor of \( A_{\alpha^n} \), i.e. \( l \in \mathbb{Z}_0, l \leq n \). If we re-normalize data with \( \mu_{\alpha^l} \), almost-surely,

\[
\bar{g}_{mn} = \begin{cases} 
\frac{2(v_{\alpha^l m} - v_{\alpha^l n})}{v_{\alpha^l l}}, & \text{if } 0 \leq l \leq m \leq n, \\
\frac{2(v_{\alpha^n l} - v_{\alpha^n m})}{v_{\alpha^n n}}, & \text{if } 0 \leq m \leq l \leq n.
\end{cases}
\]

**Proof.** Proof is in Sec. 7.1 of the supplementary.

From Corollary 1, \( 1 \geq v_A > v_{\alpha^1} > v_{\alpha^2} > v_{\alpha^3} \ldots > v_{\alpha^n} \).

Thus, the first case of Eq. (23) implies for instances of \( A_{\alpha^l m} = \alpha^n \), where \( m \) is between \( l \) and \( n \), re-normalizing with \( \mu_{\alpha^l} \), increases separability from instances of \( A_{\alpha^n m} \), which correspond to class \( \alpha \). The second case of Eq. (23) shows this comes at the expense of separability when \( m \) is less than \( l \). This is equivalent to stretching contrast (separation gap) of \( A_{\alpha^l} \) descendants at the expense of compressing contrast of its ancestors. In the extreme case, where \( l = n \), the gap between \( A_{\alpha^n} \) and all its ancestors vanishes. Thus, standard normalization with one-class learning causes disastrously bad results, as shown in supplementary Sec. 8.2.

These trade-offs motivate Sec. 4.4’s training of a stack of one-class learners by re-normalizing data with different ancestor means. This enables the exceptionally fine retrieval demonstrated in Fig. 4.

### 4.3. Magic of Deep Learned Features

Using deep-learned features is known to “magically” improves learning results. This phenomenon can be studied in the context of hierarchical-models.

Part of the improvement may arise from an innately better representation (that we cannot explain), which reduces the ratio of within-class variance to between-class variance, essentially amplifying the gap in Eq. (23).

However, Sec. 4.2 shows coordinate frame choice is also important to learning. In particular, it is ideal if the mean of the distribution-of-everything is at zero. Deep-learned do indeed seem to have made this choice and changing their coordinate frame significantly reduces performance, as shown in Sec. 8.4 of the supplementary. We also leverage this property in our one-class learning algorithm in Sec. 4.4.

### 4.4. Implementation

We term our overall algorithm Shell-Stacked (SS). Details are as follows.

**Training (Algorithm 1):** Given training images from a target class \( \alpha \), we store their unit-vector-normalized ResNet features [21] as \( F \). Optionally, we are given a set of ancestor distribution means or “external knowledge”, denoted as \( \mathbf{M} = [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_K] \). If \( \mathbf{M} \) is not given, we estimate it by crawling \( K - 1 \) semantic concepts related to the target class \( \alpha \) from the Internet. We first estimate the mean, \( \mathbf{m} \) of training data \( \mathbf{F} \). For crawled concepts, we compute the mean of their image features and rank them from nearest to farthest from \( \mathbf{m} \), creating \( [\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_{K-1}] \). By taking a weighted average of \( \mathbf{m} \) with \( \mathbf{m}_1 \), we estimate \( \mathbf{m}_1 \), an approximation of a parent distribution mean. This is given by \( \mathbf{m}_i = \frac{m_i + \sum_{i=1}^{m_i} m_i}{i}, i \geq 1, m_K = \mathbf{0} \). The final zero vector corresponds to assuming ResNet feature’s distribution-of-everything has mean, \( \mathbf{0} \), as described in Sec. 4.3.

If internet crawling is unavailable, set \( \mathbf{M} = [\mathbf{0}] \). This is termed Shell-One (SO) and is a special case of SS.

We re-normalize all input data with each \( \mathbf{m}_i \) and fit a tight shell to the re-normalized data, using Eq. (19), to estimate shell centers, \( \mu_i \) and probability density function \( p_i(x_i = x_i|\alpha) \). The output is a list parameters \( \mathbf{m}_i, \mu_i \) and \( p_i \).

**Testing (Algorithm 2):** \( y \) is a unit-normalized ResNet feature. Compute \( \tilde{y}_i \), by re-normalizing \( y \) with \( \mathbf{m}_i \). The score of \( y \) is \( \frac{1}{K} \sum_{i=1}^{K} p_i(d^2(\mu_i - \tilde{y}_i)|\alpha) \).

### 4.5. Errors in Shell Learning

In theory, shell fitting results should be perfect. In practice, there are three major error sources:

Firstly, computationally tractable features do not have infinite dimensions, with even deep-learned features having only a few hundred linearly independent dimensions. Thus data typically form cloudy approximates of the ideal shell;

Secondly, there can be semantic gaps between labels and distributions. An extreme example would be the label apple mapping to both the iPhone and the fruit. This causes an excessively large common shell that encompasses many unrelated instances. The problem can be alleviated by clustering. However, that is beyond this paper’s scope;

Thirdly, while Eq. (19)’s regularization provides robustness its estimated shells are too small, causing some true instances to fall outside the shell.

### 5. Experiments

This section focuses on checking the hierarchical-model’s predictions and quantitative analysis of shell based learning.
Table 1: AUROC score of various one-class learners. Inputs of hierarchical-models In predictive and explanatory power. The usefulness of such theories depend on their omnipresence of the internet, such assumptions are not restrictive. While comparing algorithm scores does not represent fair competition, it creates a baseline for assessing justifications for its effectiveness.

5.2. Shell Based Learning

SO and SS refer to Sec. 4.4’s Shell-One and Shell-Stacked algorithms. By strict definitions, SO and SS are not one-class learners as feature choice and normalization embedded assumptions on external distributions. In particular, SS treats the means of other training data as external knowledge for estimating parent distribution means. However, given the omnipresence of the internet, such assumptions are not restrictive. While comparing algorithm scores does not represent fair competition, it creates a baseline for assessing hierarchical-model’s relevance in the real-world.

Data-sets are described by, name; [citation]; (number of training images per class). They are: Fashion-MNIST [44]
|        | airplane (ship) | bird (cat) | car (truck) | cat (dog) | deer (horse) | dog (cat) | horse (dog) | monkey | ship (truck) | truck (car) | Average |
|--------|----------------|------------|-------------|-----------|--------------|-----------|-------------|--------|-------------|-------------|---------|
| SO-Ours | 0.886          | 0.970      | 0.842       | 0.713     | 0.884        | 0.746     | 0.701       | 0.929  | 0.897       | 0.619       | 0.819   |
| SS-Ours | 0.972          | 0.994      | 0.933       | 0.826     | 0.954        | 0.927     | 0.914       | 0.969  | 0.942       | 0.784       | 0.922   |

|        | abbey (alley) | airport terminal (airport terminal) | alley (airport terminal) | amusement park (airport terminal) | aquarium (amusement park) | Average |
|--------|---------------|-----------------------------------|--------------------------|-----------------------------------|--------------------------|---------|
| SO-Ours | 0.829         | 0.909                             | 0.895                    | 0.703                             | 0.824                    | 0.832   |
| SS-Ours | 0.965         | 0.959                             | 0.981                    | 0.938                             | 0.954                    | 0.959   |

Table 2: AUROC on the most difficult class pairs of Internet STL-10 and MIT-Places. Shells-Stacked’s (SS) gain over Shells-One (SO) is much more noticeable.

5.3. Shell Learners as Classifiers

Multiple independently trained shell-learners can be used to perform multiclass classification by assigning each instance to the highest scoring learner. On this task, SS’s average precision over all data-sets is 0.862. This is only slightly below discriminative classifiers like linear-SVM at 0.899 and much higher than radial-basis-kernel SVM at 0.437, whose density based formulation may be inappropriate for high dimensions where density vanishes.

These are remarkable results that, to our knowledge, have never been demonstrated by one-class learning formulations [22]. Figure 3 shows utilizing one-class SVM on the same task creates a vertical zig-zag precision-recall line, corresponding to each learner creating its own unique range of scores. The results are still more remarkable when we recall SS is a generative formulation, which trades some discriminative performance [29] for open-set capability demonstrated in Fig. 4.

6. Conclusion

This paper suggests many data generation processes can be explained by a hierarchical-model. This makes it possible to mathematically formulate open-set problems that often seem impossible to analyze rigorously. We demonstrate this with a one-class learning formulation that adds new classes to a model without needing retraining. This creates the exciting prospect of lifelong learners that can expand their understanding indefinitely [22].
References

[1] https://en.wikipedia.org/wiki/ Curse_of_dimensionality. Accessed: 2019-11-15.

[2] Charu C Aggarwal, Alexander Hinneburg, and Daniel A Keim. On the surprising behavior of distance metrics in high dimensional space. In International conference on database theory, pages 420–434. Springer, 2001. 1, 2, 3, 7

[3] Relja Arandjelovic, Petar Gronat, Akihiko Torii, Tomas Pajdla, and Josef Sivic. Netvlad: Can architecture for weakly supervised place recognition. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5297–5307, 2016. 2, 7

[4] Relja Arandjelovic and Andrew Zisserman. All about vlad. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1578–1585, 2013. 7

[5] Abhijit Bendale and Terrance E Boult. Towards open set deep networks. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1563–1572, 2016. 1

[6] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. In Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 601–608, 2002. 2

[7] George EP Box and George C Tiao. Multiparameter problems from a bayesian point of view. The Annals of Mathematical Statistics, 36(5):1468–1482, 1965. 2

[8] Raghavendra Chalapathy, Aditya Krishna Menon, and Sanjay Chawla. Anomaly detection using one-class neural networks. arXiv preprint arXiv:1802.06360, 2018. 7, 8

[9] Yunqiang Chen, Xiang Sean Zhou, and Thomas S Huang. One-class svm for learning in image retrieval. In Proceedings of IEEE International Conference on Image Processing (ICIP), pages 34–37. Citeseer, 2001. 1, 2, 5, 7, 8, 13, 15, 16

[10] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In Proceedings of the fourteenth international conference on artificial intelligence and statistics, pages 215–223, 2011. 8, 13, 14, 15

[11] Lucas Deecke, Robert Vandermeulen, Lukas Ruff, Stephan Mandt, and Marius Kloft. Anomaly detection with generative adversarial networks. https://openreview.net/forum?id=S1EfyiZ0Z, 2018. 7, 8

[12] Frank Dellaert. The expectation maximization algorithm. Technical report, Georgia Institute of Technology, 2002. 2

[13] 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. 2

[14] Pedro M Domingos. A few useful things to know about machine learning. Commun. acm, 55(10):78–87, 2012. 1

[15] Alexey Dosovitskiy and Thomas Brox. Generating images with perceptual similarity metrics based on deep networks. In Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 658–666, 2016. 2

[16] Jeremy Elson, John (JD) Douceur, Jon Howell, and Jared Saul. Asirra: A captcha that exploits interest-aligned manual image categorization. In Proceedings of 14th ACM Conference on Computer and Communications Security (CCS). Association for Computing Machinery, Inc., October 2007. 8, 14, 15

[17] Nir Friedman, Dan Geiger, and Moises Goldszmidt. Bayesian network classifiers. Machine learning, 29(2-3):131–163, 1997. 2

[18] Andrew Gelman, John B Carlin, Hal S Stern, David B Dunson, Aki Vehtari, and Donald B Rubin. Bayesian data analysis, Chapman and Hall/CRC, 2013. 2

[19] Izhak Golan and Ran El-Yaniv. Deep anomaly detection using geometric transformations. In Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 9758–9769, 2018. 7, 8, 15

[20] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 2672–2680, 2014. 2

[21] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 770–778, 2016. 2, 6, 7, 13, 16

[22] Navid Kordan and Kenneth O Stanley. Fitted learning: Models with awareness of their limits. arXiv preprint arXiv:1609.02226, 2016. 1, 2, 8

[23] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In Proceedings of International Conference on Learning Representations (ICLR), 2014. 2

[24] Yin-Hsi Kuo, Hsuan-Tien Lin, Wen-Huang Cheng, Yi-Hsuan Yang, and Winston H Hsu. Unsupervised auxiliary visual words discovery for large-scale image object retrieval. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 905–912, 2011. 8

[25] Wen-Yan Lin, Siying Liu, Jian-Huang Lai, and Yasuyuki Matsushita. Dimensionality’s blessing: Clustering images by underlying distribution. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 5784–5793, 2018. 1, 2

[26] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. Journal of machine learning research, 9(Nov):2579–2605, 2008. 2

[27] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. arXiv preprint arXiv:1802.03426, 2018. 2

[28] Todd K Moon. The expectation-maximization algorithm. IEEE Signal processing magazine, 13(6):47–60, 1996. 2

[29] Andrew Y Ng and Michael I Jordan. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. In Proceedings of Advances in Neural Information Processing Systems (NIPS), pages 841–848, 2002. 8
Anh Nguyen, Jason Yosinski, and Jeff Clune. Deep neural networks are easily fooled: High confidence predictions for unrecognizable images. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 427–436, 2015.

Poojan Oza and Vishal M Patel. C2ae: Class conditioned auto-encoder for open-set recognition. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019.

German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. Neural Networks, 2019.

Karl Pearson. Liii. on lines and planes of closest fit to systems of points in space. The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science, 2(11):559–572, 1901.

Miloš Radovanović, Alexandros Nanopoulos, and Mirjana Ivanović. Reverse nearest neighbors in unsupervised distance-based outlier detection. IEEE transactions on knowledge and data engineering, 27(5):1369–1382, 2015.

Bryan C Russell, William T Freeman, Alexei A Efros, Josef Sivic, and Andrew Zisserman. Using multiple segmentations to discover objects and their extent in image collections. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), volume 2, pages 1605–1614, 2006.

David W Scott. The curse of dimensionality and dimension reduction. Multivariate Density Estimation: Theory, Practice, and Visualization, pages 195–217, 2008.

Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

George Gaylord Simpson. Principles of animal taxonomy. 1961.

Josef Sivic, Bryan C Russell, Andrew Zisserman, William T Freeman, and Alexei A Efros. Unsupervised discovery of visual object class hierarchies. In Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1–8, 2008.

Tod F Stuessy. Plant taxonomy: the systematic evaluation of comparative data. Columbia University Press, 2009.

Gabor J Szekely and Maria L Rizzo. Hierarchical clustering via joint-between-within distances: Extending ward’s minimum variance method. Journal of classification, 22(2):151–183, 2005.

Nenad Tomasev, Milos Radovanovic, Dunja Mladenic, and Mirjana Ivanovic. The role of hubness in clustering high-dimensional data. IEEE Transactions on Knowledge and Data Engineering, 26(3):739–751, 2014.

Peter F Windley. Trees, forests and rearranging. The Computer Journal, 3(2):84–88, 1960.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. arXiv preprint arXiv:1708.07747, 2017.
7. Supplementary Material (Theoretical)

7.1. Proof for Re-Normalization Corollary

Before proving Corollary 2, we need to introduce a Lemma:

**Lemma 1.** A distribution’s mean, one of its sub-distribution’s mean and an arbitrarily point \( c \in \mathbb{R}^k \) almost surely form a right-angled triangle,

\[
d^2(M_{\theta^n,\Theta} - c) = d^2(\mu_{\theta^n} - c) + d^2(M_{\theta^n,\Theta} - \mu_{\theta^n})
\]

**Proof.** From Eq. (2) and Theorem 2,

\[
d^2(A_{\theta^n,\Theta} - c) = \tilde{d}(\theta^n - c) + d^2(M_{\theta^n,\Theta} - \mu_{\theta^n}) = \tilde{d}(\theta^n - c) + d^2(M_{\theta^n,\Theta} - c)
\]

Combining Eq. (25), Eq. (26), yields

\[
d^2(M_{\theta^n,\Theta} - c) = d^2(\mu_{\theta^n} - c) + d^2(M_{\theta^n,\Theta} - \mu_{\theta^n})
\]

Combining Eq. (2), Eq. (27), Eq. (29), almost-surely

\[
d^2(\tilde{A}_{\theta^n,\Theta} - \mu_{\theta^n}) = \tilde{d}(\theta^n - c) + d^2(\mu_{\theta^n} - \mu_{\theta^n}) = \tilde{d}(\theta^n - c) + d^2(\mu_{\theta^n} - c) = 2 - \tilde{d}(\theta^n - c)
\]

This proves the last case in Eq. (23),

\[
\tilde{g}_{mn} = 2(1 - \tilde{d}(\theta^n - c)) = \frac{2(\tilde{d}(\theta^n - c))}{\tilde{d}(\theta^n - c)}
\]

\[
\square
\]

7.2. Euclidean Distance in High Dimensions

In the hierarchical-model, the Euclidean distance between two high dimension instances is almost-surely a reflection of how recently they shared a common ancestor. This is summarized in the Euclidean distance corollary below:

**Corollary 3.** (Euclidean distance) In a non-trivial hierarchical-model, ranking data points by euclidean distance from \( a \), is almost surely, ranking based on how recently they shared an ancestor with \( a \).

**Proof.** Let \( a \) be an instance of \( A_{\theta^n} \), where

\[
\theta^n = \theta^0[1] \theta^1[2] \ldots \theta^n[m],
\]

is an instantiation of a non-trivial hierarchical process. The average variance corresponding to the parameters \( \theta^0[i] \) is \( \tilde{d}(\theta^n[i]) \). From corollary 1, almost surely,

\[
\tilde{d}(\theta^n[i]) = \tilde{d}(\theta^0[i]) = \tilde{d}(\theta^1[i]) = \ldots = \tilde{d}(\theta^n[i]).
\]

For any data-point \( p \) in the hierarchical-model, Theorem 1 implies, almost-surely, \( \tilde{d}(s - p) = 2\tilde{d}(\theta[i]) \), where \( i \) is the index of their most recent common ancestor. Thus, ranking based euclidean distance from \( a \) is almost-surely, ranking based on average variance of the most recent ancestor; is almost-surely, ranking based on how recently the points shared an ancestor.

\[
\square
\]

7.3. Predicted Distance Histograms

This section analyzes distances with deep-learned features to see if they follow the patterns predicted by the hierarchical-model. Results for real and simulated data are displayed in Fig. 5.

A) Statistical maximum pair-wise distance is \( \sqrt{2} \): Recall that for unit-vector-normalized data,

\[
d^2(A - 0) = d^2(M_{\theta^n,\Theta} - 0). \]

Thus, \( v_A \leq 1 \). From Fig. 2 and Corollary 3, the distance between any two instances is almost-surely less than or equal to \( \sqrt{2}v_A \) which is in turn less than or equal to \( \sqrt{2} \). This
creates a statistical maximum pairwise distance is $\sqrt{2}$ that is much less than the geometric maximum of 2.

B) Distance to random unit-vector almost-surely constant:
From Eq. (2)

$$d^2(\mathbf{A}_\Theta^\ast - \mathbf{c}) \overset{a.s.}{=} v_\mathbf{A} + d(\mu_\mathbf{A} - \mathbf{c}) = \text{constant.} \quad (32)$$

This equation has often been mis-interpreted as a “curse”. While this is true, a different pattern emerges when considering pairwise distances;

C) Log pairwise histogram shows a variety of distances: From theorem 1 and Fig. 2, not all pairwise distances are constants. While many pairwise distances converge to $\sqrt{2} v_\mathbf{A}$, the log histogram displays a variety of distances.

8. Supplementary Material (Empirical)

8.1. Errors

Figure 6 shows distances of instances from a shell’s center. Labels are color coded. The shell is trained with instances of the red class. Observe that in practice, distances do not lie and a perfect constant but form a a cloud. Despite this, points of the red class are separable from the rest. Separability can be increased through re-normalization in Sec. 4.2.

8.2. Normalization in One-Class Learning

To date, there has been no clear guide on how to perform normalization in one-class learning. Table 3 shows traditional normalization which involves centering data and converting it to unit-vectors causes extremely poor results. This was predicted in 4.2. As a result one-class learning papers perform no normalization. Table 3.

8.3. Shell-Learning vs One-Class SVM

Given properly normalized data, shell-learning (SO) is similar to one-class SVM. However, there is a significant difference in more difficult conditions. Figure 7 shows the effect of shifting the mean used for normalization towards a target class, the degenerate case given in Eq. (23). Observe that while shell-learnings performance decreases, it remains much more stable than one-class SVM.

8.4. Magic of Deep Learning Features

Applying one-class SVM on deep-learned features creates magically good results. Much of this result is because deep-learned features use a coordinate frame in which zero corresponds to the mean of the distribution-of-everything. Shifting the coordinate frame causes a marked deterioration in the results as shown in Fig. 8.

8.5. Effect of Shells-Stacked (SS)

Figure 9 plots the one-class detection AUROC score on STL-10 dataset, with increasingly stacked shells. This shows the effectiveness of re-normalization in incorporating external knowledge into a one-class learning framework.

8.6. Mean-Variance Constraint Predicted by Hierarchical-Model

Sec. 3.1 shows that the parent and child distributions generated by a hierarchical process in the hierarchical-model must adhere to Theorem 2. Results in Table 4 shows that this relationship in real-world data follows the mean-variance constraint predicted by our hierarchical-model. Statistics for various fine-grained types of dogs and cats are collected from ImageNet data under Domestic Animal-(Domestic Dog, Domestic Cat) hierarchical synsets. Animals and transports data are from STL-10 dataset. Images are converted into ResNet features using pretrained ResNet50 and unit-normalized.
Figure 5: **Probe** is the histogram of the distance of all data-set instances to a single, unit vector probe. Data derived from a hierarchical-process, will have a histogram sharply peaked at some point. This is clearly not the case for both simulated and real-data. **Normalized-Probe** involves unit-vector-normalization of all data before computing their distance to the probe. Data derived from a hierarchical-process perturbed by some scaling functions will have histograms which peak sharply at $\sqrt{2} = 1.42$. This happens for both simulated and real-data. **Normalized-Pairwise** is the histogram of pair-wise distance between data-set points. Data derived from complex hierarchical-processes, have histograms that peak sharply at $1.42$ but the log histogram should show some instances that are less than $1.42$. $1.42$ is the statistical maximum distance which almost no distance should exceed. **Summary:** A hierarchical-process with scalar perturbations is an eerily accurate model of real-world data. This is reflected in histograms which exhibit the surprising characteristics predicted by the hierarchical-processes and in our ability to create simulations that mimic these characteristics.

| STL-10 [10] (ResNet Features [21]) | airplane | bird  | car   | cat   | deer  | dog   | horse | monkey | ship   | truck | Ave. |
|-----------------------------------|----------|-------|-------|-------|-------|-------|-------|--------|--------|-------|------|
| OC-SVM(no-normalization)          | 0.854    | 0.748 | 0.949 | 0.689 | 0.857 | 0.553 | 0.792 | 0.689  | 0.929  | 0.905 | 0.799|
| OC-SVM(traditional-normalization) | 0.550    | 0.458 | 0.670 | 0.468 | 0.492 | 0.351 | 0.565 | 0.471  | 0.722  | 0.472 | 0.522|

Table 3: AUROC score of OC-SVM [9] on STL-10. As predicted in Sec. 4.2, traditional normalization makes one-class learning very much worse. As such normalization is seldom performed.

Variances and means are calculated based on unit-normalized ResNet features. Low error ratio indicates the mean-variance relationship of parent and child distributions in real-world data is consistent with Theorem 2.

**8.7. Detailed Precision Scores for Each Class in Different Datasets**

We provide detailed numbers of multi-class recognition precision scores for each class in all the datasets we tested in Table 5.

![Figure 6: Distances to shell center for the “bird” class in STL-10 [10]. Colors represent different classes of STL-10 dataset. Red color represents the ‘bird’ class to which the shell is fitted.](image)

![Figure 7: Average AUROC score on STL-10 dataset, with increasingly poor normalization. Observe that shell-learning (SO) is much more robust than traditional one-class SVM [9] to poor normalization.](image)

**8.8. Detailed AUROC Scores for Each Class in Different Datasets**

We provide detailed numbers of one-class detection AUROC scores for each class in all the datasets we tested in Table 6.

**8.9. Clustering**

The *shell based learning* can also be applied to the problem of unsupervised clustering. The result is illustrated in Fig. 10, where we conduct unsupervised cluster-
### Table 4: Average variance of parent distribution, and predicted average variance of parent distribution using Theorem 2, and the respective error ratio in percentage.

| Parent Distribution        | Average Variance of Parent Distribution | Mean Average Variance of Sub-Distribution + NSD of Mean Vectors for all Sub-Distributions | Error Ratio % |
|---------------------------|----------------------------------------|---------------------------------------------------------------------------------------|--------------|
| Domestic Dog              | 0.000308                               | 0.000309                                                                              | 0.039%       |
| Poodle Dog                | 0.000260                               | 0.000264                                                                              | 1.794%       |
| Spitz                     | 0.000253                               | 0.000259                                                                              | 2.363%       |
| Shepherd Dog              | 0.000300                               | 0.000300                                                                              | 0.289%       |
| Sled Dog                  | 0.000235                               | 0.000245                                                                              | 4.113%       |
| Watch Dog                 | 0.000298                               | 0.000303                                                                              | 1.857%       |
| Sennenhunde               | 0.000229                               | 0.000229                                                                              | 0.041%       |
| Working Dog               | 0.000306                               | 0.000306                                                                              | 0.125%       |
| Domestic Cat              | 0.000269                               | 0.000268                                                                              | 0.119%       |
| Animals                   | 0.000337                               | 0.000337                                                                              | 0.000%       |
| Transports                | 0.000336                               | 0.000336                                                                              | 0.000%       |

### Table 5: Precision score of various multi-class classifiers. Integration of multiple SS as a multi-class classifier achieves recognition precision only slightly below Linear SVM in most cases.

#### A) Fashion-MNIST (Raw Image)

|                    | t-shirt | trouser | pullover | dress | coat | sandal | shirt | sneaker | bag | boot | Ave. |
|--------------------|---------|---------|----------|--------|------|--------|-------|---------|-----|------|------|
| SS-Ours            | 0.802   | 0.996   | 0.565    | 0.718  | 0.508| 0.801  | 0.292 | 0.905   | 0.924| 0.913| 0.742|
| SVM(Linear)        | 0.812   | 0.968   | 0.713    | 0.799  | 0.951| 0.669  | 0.901 | 0.912   | 0.920| 0.834|
| SVM(RBF Kernel)    | 0.000   | 0.295   | 0.081    | 0.000  | 0.000| 0.000  | 0.197 | 0.468   | 0.000| 0.721| 0.176|

#### B) STL-10 [10] (ResNet-50 Features)

|                    | airplane | bird    | car     | cat     | deer  | dog    | horse  | monkey | ship | truck | Ave. |
|--------------------|----------|---------|---------|---------|-------|--------|--------|--------|------|-------|------|
| SS-Ours            | 0.922    | 0.969   | 0.961   | 0.779   | 0.891| 0.729  | 0.861  | 0.938   | 0.894| 0.803| 0.875|
| SVM(Linear)        | 0.935    | 0.954   | 0.951   | 0.885   | 0.895| 0.862  | 0.903  | 0.909   | 0.888| 0.909|
| SVM(RBF Kernel)    | 0.958    | 0.948   | 0.927   | 0.908   | 0.794| 0.659  | 0.950  | 0.909   | 0.805| 0.813| 0.867|

#### C) Internet STL-10 (ResNet-50 Features)

|                    | airplane | bird    | cat     | deer    | dog    | horse  | monkey | ship   | truck | Ave. |
|--------------------|----------|---------|---------|---------|--------|--------|--------|--------|-------|------|
| SS-Ours            | 0.646    | 0.626   | 0.941   | 0.910   | 0.706| 0.572  | 0.936  | 0.964   | 0.857| 0.870| 0.803|
| SVM(Linear)        | 0.780    | 0.816   | 0.873   | 0.877   | 0.687| 0.722  | 0.934  | 0.936   | 0.798| 0.850| 0.827|
| SVM(RBF Kernel)    | 0.000    | 0.100   | 0.000   | 0.000   | 0.000| 0.000  | 0.000  | 0.000   | 0.000| 0.000| 0.010|

#### D) MIT-Places [47]

|                    | abbey   | airport | alley   | amusement | park  | aquarium | Ave. | cat     | dog   | Ave. |
|--------------------|---------|---------|---------|-----------|-------|----------|------|---------|-------|------|
| SS-Ours            | 0.955   | 0.896   | 0.919   | 0.772     | 0.955| 0.899   | 0.995| 0.971   | 0.983|
| SVM(Linear)        | 0.936   | 0.941   | 0.946   | 0.911     | 0.962| 0.939   | 0.899| 0.990   | 0.990|
| SVM(RBF Kernel)    | 0.945   | 0.905   | 0.915   | 0.723     | 0.924| 0.882   | 0.000| 0.500   | 0.250|

#### E) Assira [16]

|                    | abbey   | airport | alley   | amusement | park  | Ave. |
|--------------------|---------|---------|---------|-----------|-------|------|
| SS-Ours            | 0.955   | 0.896   | 0.919   | 0.772     | 0.955| 0.899 |
| SVM(Linear)        | 0.936   | 0.941   | 0.946   | 0.911     | 0.962| 0.939 |
| SVM(RBF Kernel)    | 0.945   | 0.905   | 0.915   | 0.723     | 0.924| 0.882 |

Integration of multiple SS as a multi-class classifier achieves recognition precision only slightly below Linear SVM in most cases.

...
|                | airplane | bird | car | cat | deer | dog | horse | monkey | ship | truck | Ave. |
|----------------|----------|------|-----|-----|------|-----|-------|--------|------|-------|------|
| B) STL-10 [10] (ResNet-50 Features) |          |      |     |     |      |     |       |        |      |       |      |
| OC-SVM         | 0.854    | 0.748| 0.649| 0.689| 0.857| 0.553| 0.792 | 0.709  | 0.929| 0.905| 0.799|
| SO-Ours        | 0.974    | 0.965| 0.985| 0.894| 0.962| 0.928| 0.960 | 0.962  | 0.983| 0.969| 0.958|
| SS-Ours        | 0.993    | 0.994| 0.993| 0.976| 0.981| 0.978| 0.984 | 0.992  | 0.993| 0.987| 0.987|
| OC-NN          | 0.973    | 0.962| 0.969| 0.881| 0.955| 0.894| 0.952 | 0.966  | 0.972| 0.971| 0.949|
| Deep A.Det.    | 0.575    | 0.544| 0.801| 0.632| 0.819| 0.725| 0.841 | 0.858  | 0.797| 0.706| 0.730|
| DSEBM          | 0.677    | 0.590| 0.425| 0.630| 0.735| 0.569| 0.534 | 0.639  | 0.607| 0.308| 0.571|
| DAGMM          | 0.729    | 0.503| 0.613| 0.596| 0.647| 0.479| 0.591 | 0.519  | 0.235| 0.627| 0.554|
| AD-GAN         | 0.751    | 0.570| 0.480| 0.600| 0.722| 0.623| 0.508 | 0.582  | 0.671| 0.465| 0.602|
| C) Internet STL-10 (ResNet-50 Features) |          |      |     |     |      |     |       |        |      |       |      |
| OC-SVM         | 0.733    | 0.403| 0.895| 0.442| 0.464| 0.246| 0.419 | 0.275  | 0.860| 0.835| 0.557|
| SO-Ours        | 0.959    | 0.971| 0.977| 0.876| 0.965| 0.922| 0.946 | 0.960  | 0.980| 0.926| 0.948|
| SS-Ours        | 0.983    | 0.990| 0.989| 0.923| 0.982| 0.973| 0.976 | 0.987  | 0.990| 0.955| 0.975|
| OC-NN          | 0.939    | 0.955| 0.967| 0.851| 0.949| 0.891| 0.938 | 0.941  | 0.968| 0.921| 0.932|
| Deep A.Det.    | 0.528    | 0.616| 0.834| 0.697| 0.774| 0.695| 0.844 | 0.803  | 0.695| 0.686| 0.717|
| DSEBM          | 0.743    | 0.582| 0.272| 0.582| 0.750| 0.571| 0.548 | 0.633  | 0.633| 0.289| 0.560|
| DAGMM          | 0.607    | 0.500| 0.486| 0.481| 0.564| 0.530| 0.485 | 0.619  | 0.408| 0.489| 0.517|
| AD-GAN         | 0.743    | 0.533| 0.406| 0.550| 0.734| 0.558| 0.501 | 0.604  | 0.618| 0.306| 0.555|

Table 6: AUROC scores for one-class detection task on each class of various datasets using different models. SO and SS implemented with understanding of hierarchical-models is competitive with recent deep-learned models across a wide range of tasks and features, with SS being almost always the best.

our shell based learning on the clustering problem that is worth further exploring in the next paper.
Figure 8: Average AUROC score on STL-10 dataset using ResNet [21] and OC-SVM [9]. Observe that much of the magically good performance of ResNet [21] derives from its choice of coordinate frame. Shifting the mean from origin causes a steady deterioration in results.

Figure 9: AUROC score on STL-10 dataset for Shells-Stacked (SS) with increasing number of shells. This demonstrates its ability to leverage external information.
Figure 10: Shell based clustering result: x-axis and y-axis value in the plot represents the distance of a data point to the corresponding distinctive shell. From left to right: cat-vs-panda, dog-vs-cat and panda-vs-dog.