Learning Hierarchical Graph Neural Networks for Image Clustering

Yifan Xing* Tong He* Tianjun Xiao Yongxin Wang Yuanjun Xiong
Wei Xia David Wipf Zheng Zhang Stefano Soatto

Amazon Web Services
{yifax, htong, tianjux, yongxinw, yuanjx, wxia, daviwipf, zhaz, soattos}@amazon.com

Abstract

We propose a hierarchical graph neural network (GNN) model that learns how to cluster a set of images into an unknown number of identities using a training set of images annotated with labels belonging to a disjoint set of identities. Our hierarchical GNN uses a novel approach to merge connected components predicted at each level of the hierarchy to form a new graph at the next level. Unlike fully unsupervised hierarchical clustering, the choice of grouping and complexity criteria stems naturally from supervision in the training set. The resulting method, Hi-LANDER, achieves an average of 49% improvement in F-score and 7% increase in Normalized Mutual Information (NMI) relative to current GNN-based clustering algorithms. Additionally, state-of-the-art GNN-based methods rely on separate models to predict linkage probabilities and node densities as intermediate steps of the clustering process. In contrast, our unified framework achieves a three-fold decrease in computational cost. Our training and inference code are released 1.

1. Introduction

Clustering is a pillar of unsupervised learning. It consists of grouping data points according to a manually specified criterion. Without any supervision, the problem is self-referential, with the outcome being defined by the choice of grouping criterion. Different criteria yield different solutions, with no independent validation mechanism. Even within a given criterion, clustering typically yields multiple solutions depending on a complexity measure, and a separate model selection criterion is introduced to arrive at a unique solution. A large branch of unsupervised clustering methods follow the hierarchical/agglomerative framework [41, 42, 44], which gives a tree of cluster partitions with varying granularity of the data, but they still require a model selection criterion for the final single grouping. Rather than engineering the complexity and grouping criteria, we wish to learn them from data.2 Clearly, this is not the data we wish to cluster, for we do not have any annotations for them. Instead, it is a different set of training data, the meta-training set, for which cluster labels are given, corresponding to identities that are disjoint from those expected in the test set. For example, the test set might be an untagged collection of photos by a particular user, for which there exists a true set of discrete identities that we wish to discover, say their family members. While those family members have never been seen before, the system has access to different photo collections, tagged with different identities, during training. Our goal is to leverage the latter labeled training set to learn how to cluster different test sets with unknown numbers of different identities. This is closely related to “open-set” or “open universe” classification [40, 26].

We present the first hierarchical/agglomerative clustering method using Graph Neural Networks (GNNs). GNNs are a natural tool for learning how to cluster [51, 57, 56], as they provide a way of predicting the connectivity of a graph using training data. In our case, the graph describes the connectivity among test data, with connected components ultimately determining the clusters.

Our hierarchical GNN uses a novel approach to merge connected components predicted at each level of the hierarchy to form a new graph at the next level. We employ a GNN to predict connectivity at each level, and iterate until convergence. While in unsupervised agglomerative clustering convergence occurs when all clusters are merged to a single node [42, 44], or when an arbitrary threshold of an arbitrary model complexity criterion is reached, in our case convergence is driven by the training set, and occurs when no more edges are added to the graph by the GNN. There is no need to define an arbitrary model selection criterion. Instead, the “natural granularity” of the clustering

---

*Indicates equal contribution.

1https://github.com/dmlc/dgl/tree/master/examples/pytorch/hilander

2Of course, every unsupervised inference method requires inductive biases. Ours stems naturally from supervision in the meta-training set and density in the inferred clusters.
process is determined inductively, by the ground truth in the training set. Unlike prior clustering work using GNNs [51, 57, 56], we perform full-graph inference to jointly predict two attributes: linkage probabilities at the edges, and densities at the nodes, defined as the proportion of similar vertices that share the same label within a node’s neighborhood [14, 3, 56]. The densities establish a relative order between nodes [3, 56], which is then used to guide the connectivity. Nodes at the boundary between two ground-truth clusters, or nodes having a majority of their neighbors belonging to different classes, tend to have a low density, and accordingly a low expectation of linkage probability to their neighbors. Prior methods predict the edge connectivity as a node attribute on numerous sampled sub-graphs [51, 56]; ours directly infers the full graph and predicts connectivity as an attribute of the edges. Also, prior methods require separate models for the two attributes of linkage probabilities and node densities, whereas ours infers them jointly. This is beneficial as there is strong correlation between the two attributes, defined by the ground truth. A joint model also achieves superior efficiency, which enables hierarchical inference that would otherwise be intractable. Compared to the two separate models, we achieve a speedup from 256s to 36s as shown in Table 1.

In terms of accuracy, our method achieves an average 49% improvement in F-score, from 0.390 to 0.585, and an average 7% increase in NMI, from 0.778 to 0.836 compared to state-of-art GNN based clustering methods [56, 51] over the face and species clustering benchmarks as shown in Table 3. Furthermore, the pseudo-labels generated by our clustering of unlabeled data can be used as a regularization mechanism to reduce face verification error by 14%, as shown in Table 4, from 0.187 to 0.159 in compared to state-of-art clustering methods, allowing us to approach the performance of fully supervised training at 0.136.

In the next section, we summarize our contributions in the context of prior related work. In Section 3 we introduce the technical innovations of our paper, and in Section 4 we detail our experiment results. We discuss failure modes and limitations of our method in Section 5.

2. Related Work and Contributions

Unsupervised Visual Clustering Traditional unsupervised clustering algorithms utilize the notion of similarity between objects, such as K-means [27] and hierarchical agglomerative methods [32, 41, 37]. [5] extends Hierarchical Agglomerative Clustering (HAC) [41] with a distance based on node pair sampling probability. Approaches based on persistent-homology [61] and singular perturbation theory [33] deal with the scale-selection issue. [14, 3, 8] utilize a notion of density defined as the proportion of similar nodes within a neighborhood. Spectral clustering methods [33, 17, 47] approximate graph-cuts with a low-dimensional embedding of the affinity matrix via eigen-decomposition. Graclus [13] provides an alternative to spectral clustering with multi-level weighted graph cuts. H-DBSCAN [8] removes the distance threshold tuning in [14]. FINCH [39] proposes a first neighbor heuristic and generates a hierarchy of clusters. More recent unsupervised methods [23, 24] utilize deep CNN features. [60] proposes a Rank-Order distance measurement. Our hierarchical design relates the most to [39], however, instead of the heuristic to link the first-neighbor of each node for edge selection, which is prone to error and has limited capability in dealing with large-scale complex cluster structures, we use a learnable GNN model.

Supervised Visual Clustering Supervised graph neural network-based approaches [51, 57, 59, 57, 56] perform clustering on a k-NN graph. In contrast to these methods that produce only a single partition, our method generates a hierarchy of cluster partitions and deals with unseen complex cluster structures with a learnt convergence criterion from the natural granularity of the “meta-training” set. In contrast to [56] which requires two separate models to perform edge connectivity and node density estimation, our method jointly predicts these two quantities with a single model of higher accuracy and efficiency (Table 1). Furthermore, [51, 56] estimate linkage as a node attribute on sub-sampled graphs, whereas we estimate it as an edge attribute with natural parallelization through full-graph inference and significantly reduce runtime (Table 5). [1] uses a two-step process that first refines visual embeddings with a GNN and then runs a top-down divisive clustering, with testing limited to small datasets. In contrast, our method performs clustering as a graph edge selection procedure.

Hierarchical Representation Hierarchical structures have also been extensively studied in many visual recognition tasks [34, 21, 28, 53, 29, 15, 31, 22]. In this paper, our hierarchy is formed by multiple k-NN graphs recurrently built with clustering and node aggregation, which are learnt from the meta-training set. Hierarchical representation has also been explored in the graph representation learning literature [58, 9, 4, 19, 18, 25]. There, the focus is to learn a stronger feature representation to classify graph [58] or input nodes [18] into a closed set of class labels. Whereas, our goal is to “learn” to cluster from a meta-training set whose classes are disjoint to those of test-time.

Graph Neural Networks in Visual Understanding The expressive power of GNNs in dealing with complex graph structures is shown to benefit many visual learning tasks [20, 16, 10, 50, 45, 12, 54, 55, 11, 6, 52]. [16] samples and aggregates embeddings of neighboring nodes. [45] further advances [16] with additive attention. [10] uses a batch training scheme based on [16] to reduce computational cost. [50] performs node classification with edge convolution and feature aggregation through max-pooling. Our method dif-
fers from [50] in that we use a unified model that jointly learns node densities and edge linkages with two supervision signals. Furthermore, our GNN learns the edge selection and convergence criteria for a hierarchical agglomerative process.

**Contributions** We propose the first hierarchical structure in GNN-based clustering. Our method, partly inspired by [39], refines the graph into super-nodes formed by sub-clusters and recurrently runs the clustering on the superno- node graphs, but differs in that we use a learnt GNN to predict sub-clusters at each recurrent step instead of an arbitrary manual grouping criterion. At convergence, we trace back the predicted cluster labels on the super-nodes from the top-level graph to the original data points to obtain the final cluster.

Our method converges to a cluster based on the level of granularity established by ground truth labels in the training set. Although the identities are different from the test set, they are sufficient to implicitly define a complexity criterion for the clustering at inference time, without the need for a separate model selection criterion.

To run multiple iterations of the GNN model efficiently and effectively, we design a base model that approximates label-aware linkage probabilities and densities of similar nodes that share the same label. The densities are useful for additional regularization and refining edge selection. We refer to this base model as our **Link Approximation and Density Estimation Refinement** (LANDER) module. Finally, we denote our hierarchical clustering method Hi-LANDER and Figure 1 illustrates its structure.

The key innovation of our method is two-fold: 1) we produce a hierarchy of cluster partitions instead of a single flat partition of [57, 51, 56]; 2) we perform full-graph inference to jointly predict attributes of both nodes and edges, whereas prior GNN methods used sub-graph inference and separate models for node and edge attribute prediction. These innovations are collectively responsible for improving the clustering performance by an average of 49% F-score and 7% NMI over existing GNN-based methods.

3. Methodology

3.1. Clustering with a k-NN Graph

Formally, given a set of $N$ images $D = \{I_i\}_{i=1}^N$ and their corresponding visual embeddings $F = \{f_i\}_{i=1}^N$, we first construct an affinity graph $G = \{V, E\}$, where $|V| = N$, via $k$-nearest neighbors determined w.r.t. cosine similarity, i.e., the inner-product of the normalized visual embeddings. Each image (for example one face crop) entails one object to cluster and represents a node in the graph, with the node feature being its visual embedding $f_i$. The edges connect each node to its $k$ neighbors. Per the clustering paradigms in [14, 8, 3, 39, 51, 56, 35], a function $\phi$ takes as input the affinity graph $G$ and the node features $F$, and produces an edge subset $E' \subset E$, i.e. $E' = \phi(G, F)$. The resulting graph $G' = \{V, E'\}$ is then split into connected components, with each corresponding to a cluster of nodes. Our method is built upon this $k$-NN graph based clustering paradigm.

3.2. Hierarchical Generalization to Hi-LANDER

In order to model the natural level of granularity of clusters in a dataset, we propose a hierarchical generalization to the above single-level $k$-NN based clustering paradigm. Given a set of initial visual embeddings $F$ and a small fixed value of $k$, we iteratively generate a sequence of graphs $G_i = \{V_i, E_i\}$ and the corresponding node features $H_i = \{h_i\}$, where $i = 1 \ldots |V|$ and $l = 1 \ldots$, using a fixed inner-product of the normalized visual embeddings.
base cluster function $\phi$ and an aggregation function $\psi$. Algorithm 1 summarizes the proposed hierarchical generalization process.

To start, we define $G_1$ as the $G$ in Section 3.1 and $H_1 = \{ f_i \}$. The function $\phi$ performs the following operation

$$E'_l = \phi(G_l, H_l),$$

(1)

-taking as input the node features and $k$-NN graph at level $l$ and producing the selected edge subsets $E'_l$. As a result, the graph $G'_l = \{ V_l, E'_l \}$ is split into multiple connected components. We define the set of connected components in $G'_l$ as $\{ c^{(l)}_i | i = 1 \rightarrow |V'_l| \}$, where $c^{(l)}_i$ is the $i$-th element.

In order to generate $G_{l+1}$, we obtain $V_{l+1}$, $H_{l+1}$ and $E_{l+1}$ as follows. First, we define the $i$-th node in $G_{l+1}$, $v_i^{(l+1)}$, as an entity representing the connected component $c^{(l)}_i$. Next, we generate the new node feature vectors through an aggregation function $\psi$, which performs

$$H_{l+1} = \psi(H_l, G'_l),$$

(2)

It aggregates the node features in each connected component $c^{(l)}_i$ into a single feature vector respectively. Finally, we obtain $E_{l+1}$ by searching for $k$-nearest-neighbors on $H_{l+1}$ and connecting each node to its $k$ neighbors.

The generation converges when no more new edges are added, i.e., $E'_l = \emptyset$. We define $L$ to be the length of the converged sequence. For the final cluster assignment, starting with $G_L$, we assign cluster identity (ID) $i$ to the connected component $c^{(L)}_i$, which propagates the ID $i$ to all its nodes $\{ v_j^{(L)} | v_j^{(L)} \in c^{(L)}_i \}$. Then, each $v^{(L)}$ propagates its label to the corresponding connected component $c^{(L-1)}$ of the previous iteration. This ID propagation process eventually assigns a cluster ID to every node in $V_i$, and this assignment is used as the final predicted clustering.

In the following sections, we describe the design of the base cluster function $\phi$, the aggregation function $\psi$ and how we learn the overall Hi-LANDER model with a meta-training set. We refer LANDER to our underlying single-level model, akin to a single iteration of Hi-LANDER.

### 3.3. Realizing the Cluster Function $\phi$

To achieve high accuracy, we design $\phi$ as a learnable GNN model for clustering in a supervised setting to deal with complex cluster structures, where each node $v_i$ in $V$ comes with a cluster label $y_i$, but only in the meta-training set. Unlike unsupervised clustering methods, we do not engineer an explicit grouping criterion but learn it from data. State-of-art supervised clustering methods [51, 56] show that density and linkage information are effective supervision signal to learn the GNN model and we use both of them. However, unlike prior work, to improve both efficiency and accuracy, we jointly predict these two quantities using the embeddings produced by a single graph encoder. The linkage and density estimates are then passed through a graph decoding step for determining edge connectivity and thus cluster prediction. Below details our LANDER design.

**Graph Encoding** For each node $v_i$ with corresponding input feature $h_i$, a stack of Graph Attention Network (GAT) [45] layers encode each $h_i$ as the new feature or embedding $h'_i$. In general though, we found that alternative encoders (e.g., vanilla graph convolutional network layers), produce similar performance (see supplementary).

**Joint prediction for density and linkage** For each edge $(v_i, v_j)$ in $E$, we concatenate the source and destination node features obtained from the encoder as $[h'_i, h'_j]$, where $[\cdot, \cdot]$ is the concatenation operator. Then, we feed it into a Multi Layer Perceptron (MLP) layer followed by a softmax transformation to produce the linkage probabilities $p_{ij} = P(y_i = y_j)$, i.e., an estimate of the probability that this edge is linking two nodes sharing the same label. We also use this value to predict a node pseudo-density estimate $\hat{d}_i$, which measures the similarity-weighted proportion of same-class nodes in its neighborhood. $^4$

For this purpose, we first quantify the similarity $a_{ij}$ between nodes $v_i$ and $v_j$ as the inner product of their respective node features, i.e., $a_{ij} = \langle h_i, h_j \rangle$. Subsequently, we compute corresponding edge coefficients as $\hat{e}_{ij}$ as

$$\hat{e}_{ij} = P(y_i = y_j) - P(y_i \neq y_j)$$

(3)

where $j$ indexes the $k$ nearest neighbors of $v_i$. We may then define $\hat{d}_i$ as

$$\hat{d}_i = \frac{1}{k} \sum_{j=1}^{k} \hat{e}_{ij} \cdot a_{ij}.$$  

(4)

This estimator is designed to approximate the ground-truth pseudo-density $d_i$, which is obtained by simply replacing $\hat{e}_{ij}$ in Eq. 4 with $e_{ij} = \mathbb{1}(y_i = y_j) - \mathbb{1}(y_i \neq y_j)$ using $^4$Note that $\hat{d}_i$ is only a density proxy, not a strict non-negative density that sums to one.

---

**Algorithm 1: Hi-LANDER Clustering**

**Input** $N, F, k$

$l \leftarrow 1$

$H_1 \leftarrow F$

**while not converged do**

$G_l \leftarrow k$-nearest-neighbor($H_l, k$);

$E'_l \leftarrow \phi(G_l, H_l)$;

$G'_l \leftarrow$ connected-components($E'_l$);

$H_{l+1} \leftarrow \psi(H_l, G'_l)$;

$l \leftarrow l + 1$

**end**

**Return** ID
the ground-truth class labels, where $1$ is the indicator function. By construction, $d_i$ is large whenever the most similar neighbors have shared labels; otherwise, it is small. And importantly, by approximating $d_i$ in terms of $\hat{e}_{ij}$ via $p_{ij}$, the resulting joint prediction mechanism reduces parameters for the prediction head during training (see Section 3.5 below), allowing the two tasks to benefit from one another.

**Graph Decoding** Once we obtain the linkage probabilities and node density estimates, we convert them into final clusters via the following decoding process. Prior methods rely on an analogous decoding step [3, 56]; however, herein we tailor this process to incorporate our joint density and linkage estimates. Initially we start with $E' = \emptyset$. Given $\hat{e}_{ij}$, $\hat{d}_i$, $p_{ij}$ and an edge connection threshold $p_r$, we first define a candidate edge set $E(i)$ for node $v_i$ as

$$E(i) = \{ j \mid (v_i, v_j) \in E \text{ and } \hat{d}_i \leq \hat{d}_j \text{ and } p_{ij} \geq p_r \}. \quad (5)$$

For any $i$, if $E(i)$ is not empty, we pick

$$j = \arg \max_{j \in E(i)} \hat{e}_{ij} \quad (6)$$

and add $(v_i, v_j)$ to $E'$. We emphasize that the selection of the edge connection threshold $p_r$ is a hyper-parameter tuning process only on the validation set split from the meta-training set. It stays fixed after meta-training. This is different from the arbitrary parameter selection in unsupervised agglomerative clustering where the selection criteria will likely need to change across different test sets.

Additionally, the definition of $E(i)$ ensures that each node $v_i$ with a non-empty $E(i)$ adds exactly one edge to $E'$. On the other hand, each node with an empty $E(i)$ becomes a peak node with no outgoing edges. Meanwhile, the condition $\hat{d}_i < \hat{d}_j$ introduces an inductive bias in establishing connections. As nodes with low density tend to be those ones having a neighborhood that overlaps with other classes, or nodes on the boundary among multiple classes, connections to such nodes are often undesirable. After a full pass over every node, $E'$ forms a set of connected components $G'$, which serve as the designated clusters.

**3.4. Realizing the Aggregation Function $\psi$**

Recall that we denote $c_{i}^{(l)}$ to be the $i$-th connected component in $G_{l+1}$. To build $G_{l+1} = \{ V_{l+1}, E_{l+1} \}$, we first convert $c_{i}^{(l)}$ in $G_l$ to node $v_i^{(l+1)}$ in $V_{l+1}$. We define two node feature vectors for the new node, namely the identity feature $\tilde{h}_{i}^{(l+1)}$ and the average feature $\bar{h}_{i}^{(l+1)}$ as

$$\tilde{h}_{i}^{(l+1)} = \bar{h}_{m_i}^{(l)} \quad \text{and} \quad \bar{h}_{i}^{(l+1)} = \frac{1}{|c_{i}^{(l)}|} \sum_{j \in c_{i}^{(l)}} \tilde{h}_{j}^{(l)}, \quad (7)$$

where $m_i = \arg \max_{j \in c_{i}^{(l)}} d_{ij}^{(l)}$, represents the peak node index of the connected component $c_{i}^{(l)}$. Additionally, in the first level, $\tilde{h}_{i}^{(0)} = \bar{h}_{i}^{(0)} = f_i$, where $f_i$ is the visual embedding feature.

The next-level input feature for the base cluster function $\phi$ of node $v_i^{(l+1)}$ is the concatenation of the peak feature and average feature, i.e., $h_i^{(l+1)} = [\tilde{h}_{i}^{(l+1)}, \bar{h}_{i}^{(l+1)}]$. We empirically found that directly using one of the features produces similar performances as the concatenation on some validation sets and we left this as a hyper-parameter. The identity feature $\tilde{h}_{i}^{(l)}$ can be used to identify similar nodes across hierarchies, while the average feature $\bar{h}_{i}^{(l)}$ provides an overview of the information for all nodes in the cluster.

**3.5. Hi-LANDER Learning**

Because the merged features for super nodes, $\tilde{h}_{i}^{(l+1)}$ and $\bar{h}_{i}^{(l+1)}$, always lie within the same visual embedding space as the node features $h_i^{(l)}$ of the previous level, the same GNN model parameters can be shared across multiple levels of the hierarchy structure in learning the natural granularity of the cluster distribution of the meta-training set.

**Hierarchical Training Strategy** Given $k$ and the ground truth labels, we can determine the level $L$ at which the hierarchical agglomeration convergences. Thus, we build the sequence of graphs $\{G_l\}$ with respect to the algorithm depicted in Algorithm 1, the only difference being that we use the ground-truth edge connections $E_l^{gt}$ at all levels and thus ground-truth intermediate clusters $\{G_l^{gt}\}$ for graph constructions. We initialize LANDER, and train it on all intermediate graphs $\{G_l\}$. In one epoch, we loop through each $G_l$, perform a forward pass on graph $G_l$, compute the loss as will be defined next, and then update the model parameters with backpropagation.

**Training Loss** The Hi-LANDER model is trained using the composite loss function given by

$$\mathcal{L} = \mathcal{L}_{\text{conn}} + \mathcal{L}_{\text{den}}. \quad (8)$$

The first term $\mathcal{L}_{\text{conn}}$ provides supervision on pair-wise linkage prediction via the average per-edge connectivity loss

$$\mathcal{L}_{\text{conn}} = -\frac{1}{|E|} \sum_{(v_i, v_j) \in E} l_{ij}, \quad (9)$$

where $l_{ij}$ is the per-edge loss in the form

$$l_{ij} = \begin{cases} q_{ij} \log p_{ij} + (1 - q_{ij}) \log(1 - p_{ij}), & \text{if } d_i \leq d_j \text{ otherwise} \end{cases}. \quad (10)$$

Here the ground truth label $q_{ij} = 1$ ($y_i = y_j$) indicates whether the two nodes connected by the edge belong to the same cluster, and can be computed across all levels as described previously (similarly for the ground-truth $d_i$ derived from the $q_{ij}$ values). Meanwhile, the second term $\mathcal{L}_{\text{den}}$ rep-
represents the neighborhood density average loss given by

\[ \mathcal{L}_{den} = \frac{1}{|V|} \sum_{i=1}^{|V|} ||d_i - \hat{d}_i||_2^2. \quad (11) \]

During training, both \( \mathcal{L}_{conn} \) and \( \mathcal{L}_{den} \) are averaged across data from all levels. Note that prior work has used conceptually related loss functions for training GNN-based encoders [56]; however, ours is the only end-to-end framework to do so in a composite manner without introducing a separate network or additional parameters.

4. Experimental Results

We evaluate Hi-LANDER across clustering benchmarks involving image faces, video faces, and natural species datasets. First, we show the sensitivity of our method to early-stopping and illustrate that it is only used to reduce complexity without affecting accuracy. We also illustrate ablation experiments over the model components. We then evaluate clustering performance under both settings of same train-test and unknown test-distributions. We further show the advantage of Hi-LANDER via a semi-supervised face recognition task with pseudo label training. Finally, we analyze the runtime cost.

We compare with the following baselines. The unsupervised methods include DB-SCAN [14], ARO [35], HAC [41], H-DBSCAN [8], Graclus [13] and FINCH [39], where the latter four are hierarchical baselines. The supervised baselines include L-GCN [51], GCN-V [56] and GCN-E [56]. Hyperparameters for the baselines are tuned to report their best performances respectively. For example, we tune the optimal MinPts parameter for H-DBSCAN. Supervised GNN baselines have their best parameters tuned with the validation sets (part of the meta-training set), e.g., we tune the optimal \( k \)-NN \( k \) and \( \tau \) parameters for GCN-V/E.

4.1. Evaluation Protocols

Datasets For face clustering, we use the large-scale image dataset TrillionPairs [2] and randomly choose one-tenth (660K faces) for training. For testing, we use IMDB (images, 1.2 million faces) [48] and Hannah (video-frames, 200K faces) [36]. Hannah has no overlapping person identities with the TrillionPairs training set, whereas IMDB has a small overlap (less than 2%). Features for all face datasets are extracted from a state-of-the-art embedding model [49] trained on TrillionPairs. The average cluster size of TrillionPairs, IMDB, and Hannah are 37, 25 and 800 respectively. For species clustering, we use iNaturalist2018 [43]. We follow the open-set train-test split for image retrieval as in [7], where the training (320K instances) and testing (130K instances) classes are disjoint. Both splits have similar cluster size distributions with an average of 56 instances per class. Features are extracted from a ResNet50 pretrained object retrieval model from [7]. Table 6 of the supplementary shows detailed statistics of all datasets. For all clustering training sets, we reserve 20% for validation and hyper-parameter tuning. When finalized, we re-train on the entire training split with fixed hyper-parameters. We use Deepglint and IMDB datasets for pseudo label training for face recognition and evaluate using the openset IJBC [30] benchmark.

Evaluation Metrics For clustering, we report the Normalized Mutual Information (NMI) [46] capturing both homogeneity and completeness. We also report the pairwise and bicubed F-score which are two types of harmonic mean of the precision and recall of clustering prediction, denoted by \( F_p \) and \( F_b \). We report the standard face recognition metrics, including False Non Match Rate (FNMR) @ various False Match Rate (FMR) for verification and False Negative Identification Rate (FNIR) @ different False Positive Identification Rate (FPIR) for identification.

4.2. Implementation Details

We use the validation sets to choose our optimal meta-training hyper-parameters. \( k \) is set to 10 for \( k \)-NN graph building and is fixed for inference for all settings and test sets. \( p_r \) is set to 0.9 for face clustering and 0.1 for species. Both face and species clustering use the identity feature aggregation (detailed in Section 3.4). All validation sets are part of the meta-training sets and we have no access to any test information during hyper-parameter tuning. Due to space limitations, sensitivity analysis to these hyperparameters and additional details are included in the supplementary.

4.3. Ablation Experiments

Sensitivity to Early-stopping The proposed agglomeration process converges when there are no more new edges added. Though this convergence is reached without an explicit termination criterion, we observe that the process can be terminated early without affecting much the final clustering accuracy. Figure 2 shows the model sensitivity to early-stopping. The two dotted vertical yellow lines indicate the iterations at which the early-stopping and final convergence criteria are met. Clustering performance (Fp/Fb/NMI) plateaus after the iteration of early-stopping and there is no significant difference in accuracy and cluster numbers predicted compared to the final convergence. Therefore, simply for computational complexity considerations, we terminate the agglomeration early if computational cost is a concern. This choice is neither an arbitrary termination criterion nor a complexity / accuracy trade-off, rather, it is merely a computational expedient. Since there is no performance loss at early-stopping, we report performances with early-stopping in all subsequent sections.
Our early stopping criteria is based on the following observation. In the case where all clusters are \(k\)-ary trees, the number of new edges created at one level should be \(\leq 1/k\) of the number of edges created in the previous level. This matches the behavior in the early hierarchies when multiple intermediate clusters are merged. In the last couple of iterations, the model adds very few number of edges for several levels before an exact convergence. Therefore, if at any level the new edges created is more than \(1/k\) of the previous one, one can choose to early stop the agglomeration.

Figure 2. Sensitivity to early-stopping. The two dotted vertical yellow lines indicate the iterations at which the early-stopping and final convergence criteria are met. The left y-axis shows the accuracy of clustering prediction and right y-axis shows the predicted number of clusters (log-scale). The early-stopping is used to reduce further iterations after the model is close to convergence.

Table 1. Ablation experiments: 1) value of joint prediction compared to inference with two-separate models 2) value of hierarchy.

<table>
<thead>
<tr>
<th>Method</th>
<th>Hannah Sensitivity to Early Stopping</th>
<th>NMI</th>
<th>Fp</th>
<th>Fb</th>
<th>Runtime</th>
</tr>
</thead>
</table>
| GCN-V+E [56] | 0.062 0.224 0.640                   | 256.2
| LANDER       | 0.065 0.234 0.644                   | 44.9
| Hi-LANDER    | 0.714 0.677 0.797                   | 36.9

Table 2. Same train-test distribution clustering performance. First six rows show the unsupervised baselines (latter four are hierarchical based) and the last four rows show the supervised GNN based methods (including ours). Hi-LANDER outperforms both prior SOTA unsupervised and supervised GNN methods, with an average improvement of 35% and 8% in F-scores, respectively.

Table 3. Clustering with unseen test data distribution. Supervised methods for iNat2018-Test are trained on iNat2018-Train-DifferentDist. Hi-LANDER outperforms SOTA GNN supervised and unsupervised methods, with an average F-score boost of 49% and 47%. On Hannah, where the test-distribution is very different from that in meta-training, we improve the F-score from 0.224 to 0.677 and NMI from 0.640 to 0.797 via modeling the data granularity using a disjoint meta-training set with a learnt convergence.

4.4. Clustering Performance

Here, we compare Hi-LANDER with state-of-art unsupervised and supervised methods under the setting where the cluster size distributions of train and test data are similar. For face, we sample a subset of IMDB to match the training distribution of DeepGlint, and name this sub-sampled testset as IMDB-Test-SameDist. For species, we use iNat2018-Train and iNat2018-Test for training and testing since they follow the same cluster size distribution. Table 2 shows the

<table>
<thead>
<tr>
<th>Method</th>
<th>IMDB-Test-SameDist</th>
<th>iNat2018-Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fp</td>
<td>Fb</td>
</tr>
<tr>
<td>DBSCAN [14]</td>
<td>0.064 0.092 0.822</td>
<td>0.100 0.116 0.753</td>
</tr>
<tr>
<td>ARO [33]</td>
<td>0.012 0.079 0.821</td>
<td>0.007 0.062 0.747</td>
</tr>
<tr>
<td>HAC [41]</td>
<td>0.398 0.591 0.904</td>
<td>0.117 0.245 0.732</td>
</tr>
<tr>
<td>H-DBSCAN [8]</td>
<td>0.423 0.628 0.895</td>
<td>0.178 0.241 0.754</td>
</tr>
<tr>
<td>Graclus [13]</td>
<td>0.014 0.099 0.829</td>
<td>0.003 0.050 0.735</td>
</tr>
<tr>
<td>FINCH [39]</td>
<td>0.001 0.001 0.155</td>
<td>0.014 0.024 0.783</td>
</tr>
<tr>
<td>LGCN [51]</td>
<td>0.695 0.779 0.940</td>
<td>0.069 0.125 0.755</td>
</tr>
<tr>
<td>GCN-V [56]</td>
<td>0.722 0.753 0.936</td>
<td>0.300 0.360 0.719</td>
</tr>
<tr>
<td>GCN-V+E [56]</td>
<td>0.345 0.567 0.864</td>
<td>0.273 0.353 0.719</td>
</tr>
</tbody>
</table>

4.5. Clustering with Unseen Test Data Distribution

We also report clustering performance under the setting where test-time distribution is unknown and different from that of meta-training. Namely, parameters (such as \(\tau\) and \(k\)-NN \(k\) in GCN-V/E and max cluster size in L-GCN) cannot be adjusted in advance using test-time information. For face clustering, we train with TrillionPairs-Train and test on Hannah and IMDB. For species, we sample a subset of the iNat2018-Train to attain a drastically different train-time cluster size distribution as iNat2018-Test, and name it iNat2018-Train-DifferentDist. Table 3 illustrates the results. Hi-LANDER outperforms prior supervised methods by a large margin over Hannah, where distribution of cluster sizes is long-tail (varying from 1-20K). This is due to Hi-LANDER’s ability to perform dynamic inference and adapt to clusters with largely varying sizes during test (as shown
in supplementary Figure 1). Some unsupervised baselines such as H-DBSCAN and HAC outperform the supervised ones over Hannah, showing better generalization capability. Despite being a supervised method, Hi-LANDER outperforms all unsupervised baselines, owing to the strong expressive power of our unified GNN LANDER model.

Table 4. Face recognition on IJBC[30]. Hi-LANDER outperforms all baselines and improves over the best result from prior-arts with a 14% error reduction. Models trained with pseudo labels generated by Hi-LANDER brings the performance (0.159) closer to the lower bound with fully supervised training (0.136).

4.6. Representation Learning with Pseudo-Labels

We follow a setting similar to that of [59, 38, 57] for face recognition with pseudo label training. Starting with an initial representation learned through some labeled datasets, we utilize the clustering methods to generate pseudo labels for unlabeled datasets and train with these pseudo labels to better learn a representation. The face recognition experiment involves the following steps: 1) Start with an initial face recognition model learnt on TrillionPairs. 2) Train a clustering model on the TrillionPairs or use an unsupervised clustering method with the initial face representations. 3) Generate pseudo label on IMDB (overlapping identities with TrillionPairs removed). 4) Train face recognition models on IMDB via the pseudo labels. 5) Evaluate the learned face representation on the open-set IJBC benchmark. Table 4 shows the results. We also report the lower bound from fully supervised training on IMDB with human labeled data. Hi-LANDER achieves a 14% error reduction compared to the best baseline. Interestingly, pseudo label training with Hi-LANDER brings the performance to 0.159 (verification FNMR@FPIR 1e-4), closer to the lower bound of fully supervised training at 0.136 than any of the baselines.

4.7. Runtime Analysis

We compare the runtime (seconds) of Hi-LANDER with all baselines (Table 5). The hardware and software specifications are included in the supplementary. The complexity numbers above are from Hi-LANDER with early-stopping. Our method is faster than most baselines and is comparable with GCN-V[56], FINCH[39] and Graculus[13]. The multiple hierarchies introduced do not bring in additional overhead since Hi-LANDER runs faster level by level, with fewer number of nodes remaining after each level.

5. Discussion

The proposed clustering method aims at providing a rich representation of unlabeled data using induction from an annotated training set. GNNs represent a natural tool, for they allow training from a disjoint dataset a model that outputs a graph structure. Since the clustering problem is intrinsically ill-posed, for there is no unique "true" cluster, we aim to provide a rich hierarchical representation that gives the user more control – in the spirit of agglomerative hierarchical clustering. To tackle the computational challenge in replicating the basic graph operations across levels of the hierarchy, we have proposed enhancements of current GNN-based methods that improve efficiency. Though the complexity of our method is \( O(kN) \), the same as the vanilla flat-version of GNN clustering, the full-graph inference is a natural parallelization and significantly reduces the runtime compared to prior GNNs with sub-graph inference.

Hi-LANDER is subject to the usual failure modes of all inductive methods, when the distribution of test data is extremely different from that in training. In addition, the current node feature aggregation takes the form of averaging while there might be more sophisticated methods such as learnable attention for more informative aggregation.

Even so, our goal is to reduce the number of arbitrary choices as much as possible and defer to the data the most critical design decisions. One is the choice of clustering criterion. This is inherited by the training set, through the simple classification loss. So is the level of granularity of the partition of the data. Though we use early stopping, we do so only after verifying that the method, when iterated to convergence, settles on a solution that is not substantially different from that obtained in earlier iteration. Therefore early stopping is not chosen as a design parameter or an inductive bias, but merely as a way to reduce computation.

Table 5. Runtime comparison on all benchmarks in Table 3 (secs).
References

[1] https://cs.nyu.edu/media/publications/choma_nicholas.pdf. 2

