2016-03-01

INCREMENT - Interactive Cluster Refinement

Logan Adam Mitchell
Brigham Young University - Provo

INCREMENT - Interactive Cluster Refinement

Logan Adam Mitchell

A thesis submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Master of Science

Tony Martinez, Chair
Christophe Giraud-Carrier
Daniel Zappala

Department of Computer Science

Brigham Young University

March 2016

ABSTRACT

INCREMENT - Interactive Cluster Refinement

Logan Adam Mitchell
Department of Computer Science, BYU
Master of Science

We present INCREMENT, a cluster refinement algorithm which utilizes user feedback to refine clusterings. INCREMENT is capable of improving clusterings produced by arbitrary clustering algorithms. The initial clustering provided is first sub-clustered to improve query efficiency. A small set of select instances from each of these sub-clusters are presented to a user for labelling. Utilizing the user feedback, INCREMENT trains a feature embedder to map the input features to a new feature space. This space is learned such that spatial distance is inversely correlated with semantic similarity, determined from the user feedback. A final clustering is then formed in the embedded space. INCREMENT is tested on 9 datasets initially clustered with 4 distinct clustering algorithms. INCREMENT improved the accuracy of 71% of the initial clusterings with respect to a target clustering. For all the experiments the median percent improvement is 27.3% for V-Measure and is 6.08% for accuracy.

Keywords: Clustering, Cluster Refinement, Active Learning, User Feedback, Human in the Loop

Table of Contents

1 Introduction 1

2 Related Work 5

3 INCREMENT 8

3.1 Sub-Clustering 8

3.2 User Queries 10

3.3 Feature Embedder 11

4 Experiments 15

4.1 Datasets 15

4.2 Clustering Algorithms 17

4.3 Measures 19

4.4 Setup 20

5 Results 21

5.1 Documents 21

5.2 Faces 25

5.3 Active 28

5.4 Fine-Tuned CNN 30

6 Discussion 31

7 Conclusion and Future Work 33

References **37**

1 Appendix: Active 41

2 Appendix: Kmeans 43

3 Appendix: HAC - Complete Link 45
4 Appendix: Spectral 47
5 Appendix: None 49

1 Introduction

In unsupervised learning, or clustering, it is often assumed that each instance of data has some latent, but unknown, class. Clustering, therefore, attempts to discover these latent variables through the grouping of data into clusters. However, since the latent class is unknown, additional assumptions must be made in order to determine which clusterings are reasonable. These assumptions, or biases, are often reflected in the feature representation and the clustering algorithm used.

The basic kmeans algorithm [20], for example, maintains the bias that instances of the same class are spatially close in the feature space and instances of different classes are distant in the same space. Due to this, kmeans prefers to form spherical clusters while using a distance metric such as Euclidean distance. OPTICS [1] operates under the assumption that instances of the same class, not only should be nearby to each other, but should form a consistent density. Under this bias, OPTICS can cluster arbitrary shapes, so long as the shape maintains the same spatial density of instances within the feature space. The bias of hierarchical agglomerative clustering, or HAC, is influenced by the metric used for cluster distance, such as single or complete link. HAC merges the closest clusters together at each hierarchy until a final clustering is reached.

Because each clustering algorithm operates according to its own bias, no single algorithm will work well for all tasks [32]. They do however, work well for any task matching their respective bias. Also, because the latent class variable is unknown, clustering becomes ambiguous. In order to achieve the desired clustering, a user must carefully select both a feature representation and a clustering algorithm with biases that correctly capture the desired latent classes.

To illustrate, consider the images in Figure 1. There are multiple clusterings that could be considered correct. One user could require clusters based on individuals. Another user might need two clusters based on gender. A third might desire clusters on whether

Figure 1: These images from the CMU Face dataset [21] depict the ambiguity of clustering. Many valid cluster could be formed with these images. These clusters could be based on individual, gender, the presence of sunglasses, etc.

or not the person is wearing sunglasses. Each of these users would have to select a feature representation which captures and highlights faces, gender, or sunglasses respectively. The user would then need to use a clustering algorithm, with an appropriate bias, to recover their desired clusters. This process is often non-trivial.

We present **In**teractive **C**luster **Re**fine**ment** (INCREMENT) to help solve this problem. INCREMENT utilizes a minimal amount of user feedback to refine the feature space. In this embedded feature space, semantically similar instances are learned to be spatially close to each other while semantically dissimilar instances are learned to be spatially distant. Semantic similarity is determined by the user feedback. Thus, the mapping is learned according to the bias the user desires. Furthermore, the new space is learned according to a known bias, which enables us to more accurately form final clusters in the new feature space.

Figure 2 depicts a general overview of INCREMENT. INCREMENT is designed to take in any arbitrary clustering as input, i.e. the initial clustering can come from any clustering algorithm. Figure 2.1 shows an initial clustering provided to INCREMENT with two clusters, circles and triangles. In Figure 2.2 INCREMENT begins by sub-clustering each initial cluster. The sub-clusterings can be formed by any clustering algorithm. In this paper, INCREMENT uses OPTICS [1] to create the sub-clusterings. Ideally, each sub-cluster only

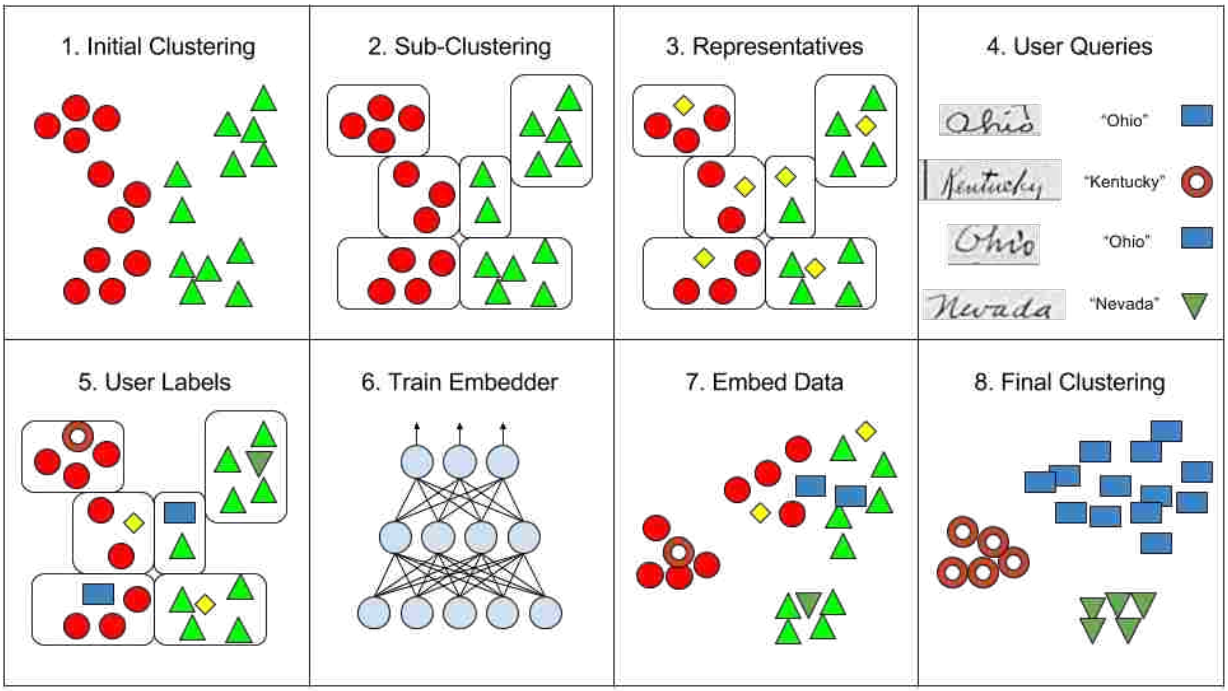


Figure 2: An outline of the steps of INCREMENT. INCREMENT is initially provided with and initial clustering, processes the data, queries a user, maps the data to a new, learned embedded space, and returns a refined clustering.

contain instances which the user would consider similar to each other. The sub-clusters are formed to reduce the amount of user feedback required by INCREMENT. To reduce the feedback required, one representative is selected from each sub-cluster. This is shown in Figure 2.3. A subset of these representatives are presented to a user for labelling on behalf of their respective sub-clusters.

For each user query, the user provides a label for a single representative. Figure 2.4 shows four separate queries. The labels assigned to each representative is temporarily applied to its entire sub-cluster. In Figure 2.6, a supervised feature embedding algorithm is trained to map instances from the input feature space to a new embedded feature space. While any model could be used as the embedder, in this paper INCREMENT uses a "Siamese" Network [9] trained using Contrastive Loss [14]. This model learns to map semantically similar instances to the same spacial location in the embedded space while separating dissimilar instances.

Once the embedder has been trained, every instance is mapped into the new embedded space, shown in Figure 2.7. This embedded space is learned so as to place semantically similar instances near each other. This enables INCREMENT to learn to separate instances which were initially clustered together, but appear semantically dissimilar. Figure 2.8 shows the final clustering which is performed by kmeans [20] in the embedded space. The number of clusters is determined according to the number of distinct labels that the user provided.

INCREMENT is designed to be able to refine clusterings of dense data representations, such as images. While there do exist some cluster refinement algorithms in the sparse text document domain [19, 23], to the best of our knowledge, no other cluster refinement algorithm exists for domains with dense feature representations.

INCREMENT is validated by showing how well INCREMENT is able to improve the quality of clusterings produced on 9 datasets using 4 clustering algorithms. Due to the availability of some of datasets, the 3 document image datasets are only clustered using one of the clustering algorithms, CONFIRM. Furthermore, since CONFIRM is designed

specifically to cluster document images, it is only used to cluster the 3 document datasets. This provides 21 initial clusterings used in the experiments. The experiments in this paper make use of labeled data in order to define a target clustering and provide user feedback. Thus, INCREMENT is validated using supervised measures. The only labels INCREMENT utilizes are the ones provided to it by the user queries.

INCREMENT improved the accuracy of 71% of the initial clusterings. Over all the experiments, the average percent improvement is 37.0% with a standard deviation of 62.5% in accuracy, 292.0% with a standard deviation of 698.0% in V-Measure [25], and 77.5% with a standard deviation of 99.3% in JCC [7] with respect to a target clustering. However, due to the high deviation, we also present the median percent improvement. The median percent improvement over the initial clustering is 6.08% in accuracy, 27.3% in V-Measure, and 17.38% in JCC. The measures are described in Section 4.3.

Section 2 reviews work relating to INCREMENT. Each component of INCREMENT is explained in Section 3. The datasets, clustering algorithms, measures, and experiments used in validation are described in Section 4. Representative results and discussion are presented in Section 5 and Section 6 while the complete set of results are found in the Appendix. Finally, the conclusion and future work are described in Section 7.

2 Related Work

Liu et al. [19] apply cluster refinement to clusters of text documents. Using bag-of-words assumption, Liu et al. propose an iterative cluster refinement algorithm based on feature voting. In each iteration of the refinement process, a set of discriminative features are identified and selected. The clustering is then updated by having each discriminative feature vote for a single cluster assignment. While discriminative feature voting works well under the bag-of-words assumption, the voting process becomes non-intuitive with continuous feature values or when all instances share the same feature set. Park et al. [23] also propose a cluster refinement algorithm using a metric they call coherence of cluster which is based on the

cosine similarity of all documents within a cluster. This approach relies only upon the bias of cosine similarity. This approach is also limited to text document clustering

In the interactive clustering domain, Balcan and Blum [4] proposed a model using user feedback to improve the quality of clusters. Users provide feedback by specifying clusters that either need to be split or merged. Applying this model to concept classes, Balcan and Blum present proofs on the bounds for user requests needed to achieve a target clustering. This approach is solely based on user feedback.

Awasthi et al. [3] expand on the work of Balcan and Blum by relaxing some of the constraints required by Balcan and Blum. Their proposed model builds a global average linkage HAC tree to determine where identified clusters should be split and merged. Awasthi et al. also provide proofs for various applications showing theoretical bounds on the maximum number of queries required to reconstruct the target clustering.

One limitation of the models proposed by Balcan and Blum as well as the initial model proposed by Awasthi et al. is that the user is expected to be able to observe and comprehend the entire clustering. Often, the number of data instances is simply too large for a user to be expected to search through each cluster. Awasthi and Zadeh [2] mitigate this by presenting small, random subsets of points to the user. However, they then present large theoretical bounds for the number of queries required to achieve the exact target clustering.

Dubey et al. [12] proposed an interactive clustering model where the user is able to either move an instance from one cluster to another, or shift the centroid of a cluster to better encapsulate domain knowledge from the user. Under this model, the user is also required to search through the entire clustering in order to provide feedback to the model.

Desjardins et al. [11] present a GUI for use in interactive clustering. A user provides feedback by inspecting a plot of clustered points. This approach, however, requires the entire clustering to be visualized in a 2 or 3 dimensional plot. The work in interactive clustering has typically focused on using user feedback to guide the clustering model. Often, the models

do not attempt to learn from the feedback, and thus require the user to continue to provide feedback until the clustering is finished.

Research in active clustering typically focuses on constrained clustering [24]. Grira et al. [13] apply active clustering to image categorization. They query the user to obtain must-link and cannot-link pairwise constraints to perform Active Fuzzy Constrained Clustering. Biswas and Jacobs [8] also make use of pairwise constraints. Biswas and Jacobs present Active-HACC, a constraint based version of HAC, and use expected model change to compute which query to present. Biswas [7] also introduces the concept of sub-clustering. Biswas performed sub-clustering in order to improve the efficiency of Active-HACC. He uses a bottom-up approach to form sub-clusters of small sizes. Whereas we implement a top-down approach to form sub-clusters that are as large as possible. Thus, INCREMENT utilizes fewer sub-clusters, which improves query performance, but the sub-clusters are potentially more noisy.

Basu et al. [5] developed an active learning scheme for selecting queries known as the farthest-first query traversal scheme. The pairwise query scheme first selects a point at random adding it to an empty set of points. The scheme then finds the point outside of the set furthest from the points in the set. The query consists of the far point and the point closest to it within the set of points. After the query is presented the far point is added to the set and the process is repeated. Basu et al. show that this is an efficient approximation to the k-centers problem [15].

Active learning, and by extension active clustering, generally attempts to select which set of data will induce the model with the best performance [28]. The contribution of INCREMENT is to improve clusterings by learning a new feature representation according to user feedback, not to select the best subset of points with which to query the user. Applying more powerful active learning techniques to INCREMENT, however, could potentially improve its performance and is a subject of future work.

Deep Convolutional neural networks have recently been used to obtain impressive results in the visual domain. Part of their strength comes from their ability to automatically extract usable, high level features. For example, Schroff et al. [27] learn a feature embedding using CNNs to achieve state of the art results on facial verification, recognition, and clustering. The authors use Triplet Loss to compute an error gradient during training. Triplet Loss is similar to Contrastive Loss function, which is utilized in INCREMENT. The main difference between the two losses is that Triplet Loss uses three points per instance; an anchor, a point of the same class, and a point of a different class. Contrastive Loss, described in Section 3.3, only requires pairs of points and their binary similarity. When a user provides a label to a point, the user effectively states that the point is dissimilar to all other points with a different label. Thus, we feel Contrastive Loss is more suited to learning from user feedback.

3 INCREMENT

3.1 Sub-Clustering

INCREMENT begins by being given an initial clustering. Any algorithm may be used to form this clustering. Intuitively, the initial clustering is simply a coarse partitioning of the data. Each sub-clustering then splits each coarse partition based on finer and finer details. The purpose of sub-clustering is to obtain sub-clusters which are as homogeneous as possible while reducing the number of instances that need to be managed. This enables INCREMENT to represent an entire sub-cluster using a single representative, gaining efficiency both in terms of computation and queries.

OPTICS [1] is used to implement the sub-clustering algorithm. OPTICS is a density based clustering algorithm which can cluster arbitrary shapes, requiring a single hyper-parameter called *minPts*. The hyper-parameter specifies how many points are required, at a minimum, to form a cluster. The output of OPTICS is an ordered list of points, each with a reachability score.

The reachability score is computed based on the distance from a point to each of its *minPts* nearest neighbors. A reachability distance is defined between pairs of points as $reachability_{minPts}(u, v) = max(dist(u, v), dist_{minPts}(u))$ where u, v are two points and $dist_{minPts}(u)$ is the distance from u to u's *minPts* nearest neighbor. The reachability score is the lowest reachability distance connected to each point. The order of the reachability plot is ordered such that the plot begins with an arbitrary point and then iteratively appends to the plot the point with the lowest reachability distance to any point already in the plot.

Figure 3 shows an example of the reachability plot produced by OPTICS. The color and shape of each point on the left represents the cluster the point is assigned to by OPTICS. The graph on the right is the reachability plot formed by OPTICS. Due to the construction of the reachability scores, clusters are typically formed in the "valley" of these plots. Simple approaches perform a hard threshold of this plot to determine the final clusters. For these experiments, spike detection is performed on the reachability plot in order to detect as many sub-clusters as possible. Then, OPTICS is recursively applied to each of these sub-clusters, halving *minPts* while $minPts >= 2$. This is done to extract as many pure sub-clusters as possible. In the experiments, an initial $minPts = 5$ is used as it produced reasonably pure results.

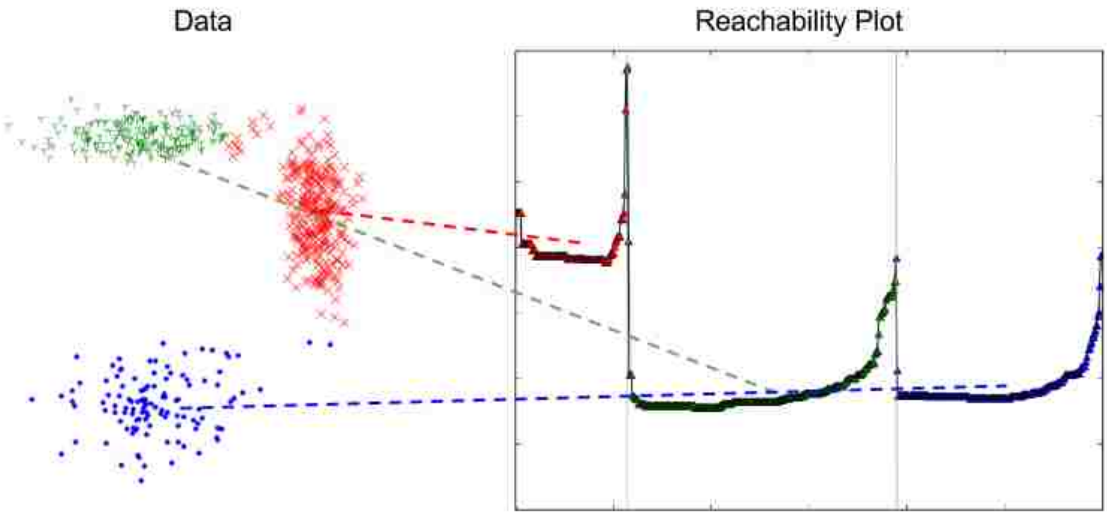


Figure 3: A depiction of a clustering formed by OPTICS. On the left is a clustering of synthetic data created by OPTICS. The color and shape of the points represent the cluster to which OPTICS assigned each point. On the right is the reachability plot created by OPTICS to cluster the data. The lines connect the clusters on the left with the same cluster in the reachability plot.

3.2 User Queries

Sub-cluster representatives are used in queries to obtain user feedback. Because representatives are used in the query, it should be as similar to every point within its sub-cluster as possible. Centroids are easy to compute and, by construction, are in the center of the sub-cluster. Medoids are more expensive to compute, but they are constrained to be actual points. Using medoids prevents a noisy sub-cluster from creating a noisy representative. Even though the medoid from a noisy sub-cluster might not be as similar to the sub-cluster, it is still a valid training instance which can be used to learn the feature mapping. Thus, the medoid of each sub-cluster is selected as a representative.

The user provides a label for each instance presented. By providing a label in feedback, INCREMENT extracts more pairwise constraints than simply performing a pairwise query. Specifically, with n label queries, $n * (n-1)/2$ pairwise constraints are generated. The farthest-first traversal scheme [5] determines the order to present points to the user. The first point is selected at random and added to a set. Each subsequent point selected is the

current point furthest from everypoint in this set. Once a point is selected, it is also added to the set. This enables the number of clusters in the final clustering to be approximated because it is an efficient approximation of the k-centers problem [15].

3.3 Feature Embedder

INCREMENT uses a feature embedder to map the input features into a new embedded feature space that better captures semantic similarity. Using the user feedback to construct this new space ensures that INCREMENT preserves the semantic similarity as defined by the user. Thus, in this new space, clusters are formed based on semantic similarity. While any supervised feature extractor could be used in INCREMENT, it currently uses a "Siamese" network [9] trained with Contrastive Loss [14] to create the feature embedder.

Conceptually, a Siamese network is a neural network consisting of two identical sub-networks which share weights, as shown in Figure 4. The loss function, Contrastive Loss, requires a pair of vectors, the embedded features, and a binary similarity score. The binary similarity score is determined from the labels that the user provides. Contrastive Loss generates an error gradient which guides the output vectors such that the distance between similar points is small and the distance between dissimilar points is large. Finally, after the full network has been induced, only a single sub-network is used to produce the feature embeddings. This sub-network is the feature embedder used by INCREMENT.

As discussed earlier, the embedded space is learned such that distances between similar points is small while distances between dissimilar points is large. Formally, the network learns to approximate a function $N : \mathbb{R}^d \rightarrow \mathbb{R}^f$ such that for some $\boldsymbol{x}, \boldsymbol{y} \in \mathbb{R}^d$, if $sim(\boldsymbol{x}, \boldsymbol{y}) = 1$ then $Dist(N(\boldsymbol{x}), N(\boldsymbol{y})) = 0$ and if $sim(\boldsymbol{x}, \boldsymbol{y}) = 0$ then $Dist(N(\boldsymbol{x}), N(\boldsymbol{y})) >= m$ where d and f are the input and embedding dimensions respectively, m is a margin parameter, and $Dist$ is euclidean distance.

To train the network N, stochastic gradient descent is used to minimize Contrastive Loss. The loss is computed using the euclidean distance between two vectors and their

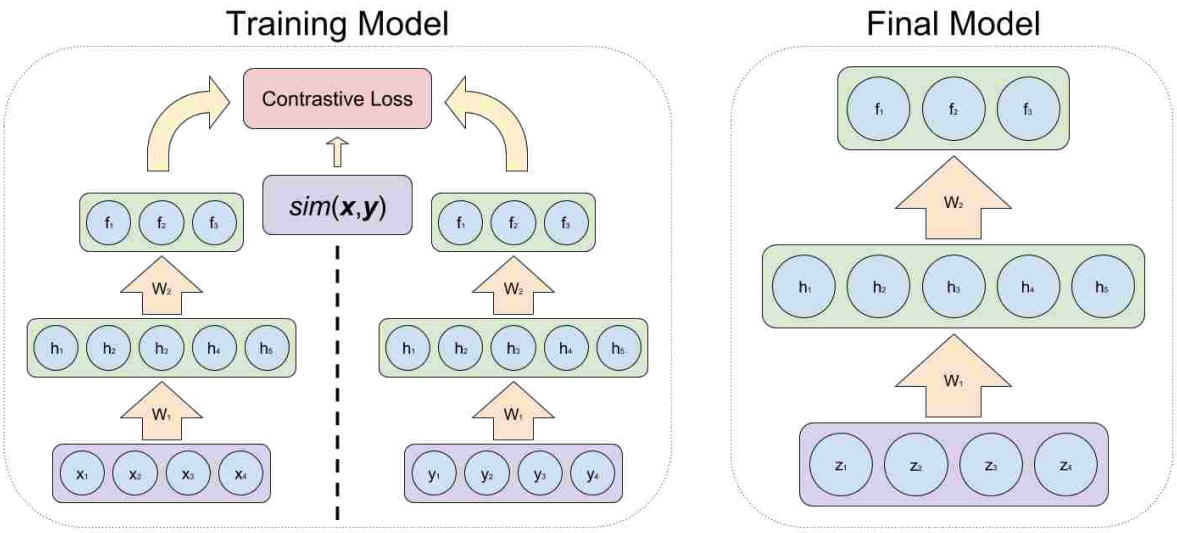


Figure 4: An illustration of a Siamese network with Contrastive Loss. On the left is depicted the structure of a Siamese network as it trains on a pair of inputs $\boldsymbol{x}, \boldsymbol{y}$. Notice that the two sub-networks share weights. The Contrastive Loss layer will generate error gradients to adjust the weights according to $sim(\boldsymbol{x}, \boldsymbol{y})$, the binary similarity between $\boldsymbol{x}$ and $\boldsymbol{y}$. On the right is the final network used as the feature embedder. This is the model used to compute the new feature vector $\boldsymbol{f}$ from an input vector $\boldsymbol{z}$. Notice, the final model only uses one side of the initial network.

similarity. A margin hyper-parameter, m, states the distance by which dissimilar points should be separated. Thus, the equation for the loss function of the network, which is computed at the loss layer, is

$$L_N(\boldsymbol{x},\boldsymbol{y}) = \frac{1}{2}\left[sim(\boldsymbol{x},\boldsymbol{y}) * Dist(N(\boldsymbol{x}),N(\boldsymbol{y}))^2 + (1 - sim(\boldsymbol{x},\boldsymbol{y})) * max(0, m - D(N(\boldsymbol{x}),N(\boldsymbol{y})))^2\right]$$

The first term in the equation generates loss when similar instances are distant. The second term generates loss only when the distance between dissimilar instances is less than the margin m. Thus, as Contrastive Loss is minimized during stochastic gradient descent, similar instances are brought close together and dissimilar instances are separated by at least m.

Figure 5 shows an example of an embedded space learned on the CMU Face dataset using Contrastive Loss. The graph depicts how Contrastive Loss forms the embedded space. Faces of the same individual are, for the most part, near other images of the person. While the embedding of Figure 5 is created using only 2 dimensions, in the experiments below the embedding is created using 100 dimensions.

Intuitively, it is easy to think of the Siamese network as a pair of networks, depicted in Figure 4, that feed into the Contrastive Loss. During training, this network is often trained using mini-batches. Mini-batches are small groups of training instances over which the error gradient is averaged. This leads to a more stable gradient. However, since the structure and weights of the two sides of the network are identical, the performance is improved by reducing the model to a single network. To achieve this, every pair of points within a mini batch are compared and passed to the Contrastive Loss layer. This simplification reduces the size of the model and prevents generating pairs of points before training, which can be slow. Thus, the model actually trained only uses a single side, but Contrastive Loss is used to bring similar points together and push dissimilar points apart within a mini batch. The Contrastive Loss layer implemented in caffe [16] was extended to achieve the desired functionality.

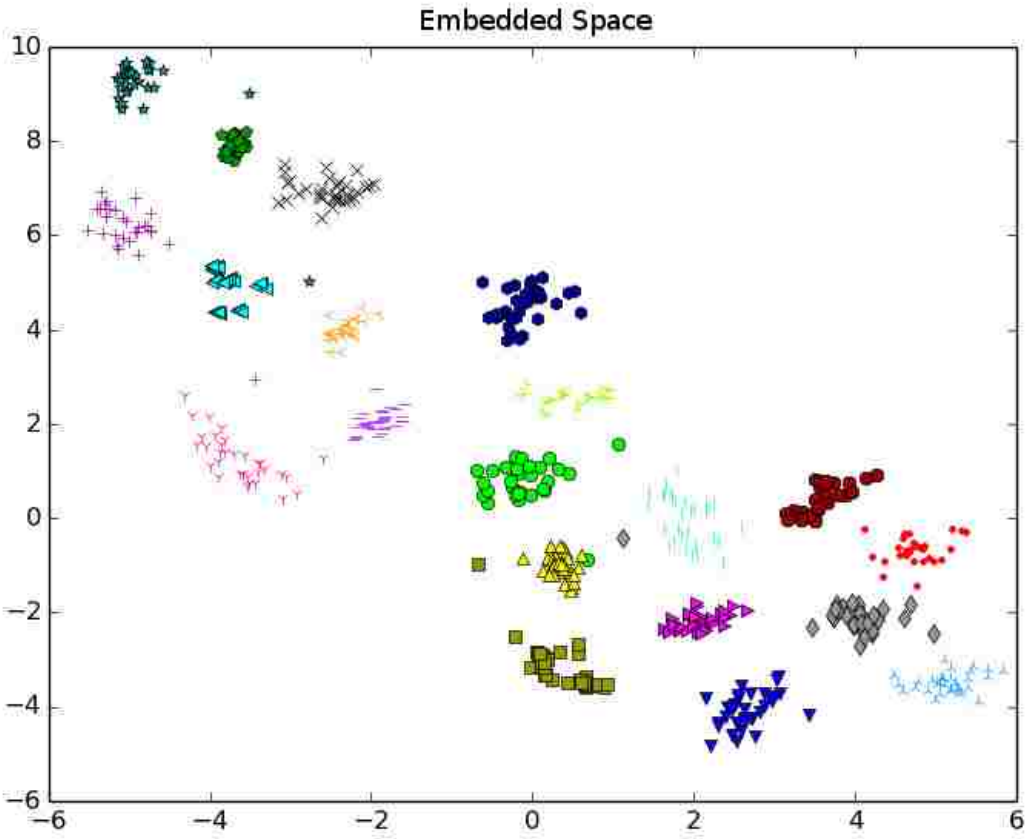


Figure 5: The embedded space learned using Contrastive Loss on the CMU face dataset. Each point is the embedded coordinates of a different facial image into 2 dimensions. The shape and color of each point represents the identity of the individual in the image. Notice how images of the same individual tend to form small clusters. After learning this embedded space, kmeans produces the final clusterings.

To obtain the data used for training, the label a user assigned to a representative is temporarily assigned to the representative's sub-cluster. Only the sub-clusters which are assigned a label are used for training. The data is shuffled and the embedder is trained using pairwise similarities within mini-batches. The implementation of Contrastive Loss computes the binary similarity at train time using the labels provided by the user. Once the embedder has been induced, all instances are mapped into the new space using the trained embedder.

Kmeans forms the final clustering in the embedded space. Kmeans operates under the bias that points in the same cluster should be spatially close, while points of separate clusters should be spatially separated. Recall that this matches the bias imposed on the embedded feature space. The number of clusters is determined by counting the number of distinct labels provided by the user. The final clustering is the clustering created by kmeans using all instances mapped into the learned embedded space.

4 Experiments

4.1 Datasets

Labeled datasets are used in the experiments to simulate user feedback using the labels. In actual usage, each user effectively defines an implicit, usually unknown, label set according to which the user desires clusters to be formed. The labels are only used within INCREMENT to simulate user feedback and to validate the clusterings. The datasets used in our experiments are described below.

- **CMU Faces:** This dataset contains 624 gray scale images of faces which are 128x120 pixels. This dataset is labelled with the following four types of labels. This dataset enables us to demonstrate how well INCREMENT is capable of forming clustering with respect to different user feedback [21].

- **Names:** This label is a unique identifier of the person in the image. Clustering according to this label would be equivalent to clustering by identity. There are images of 20 distinct individuals in the dataset.
- **Poses:** In each image, the individual is facing one of four directions. This label specifies which direction the person is facing. The four labels are left, right, straight and up.
- **Eyes:** Half of the images contain individuals wearing sunglasses. This label specifies whether or not the person is wearing them.
- **Expressions:** This label specifies which emotion the person is expressing in the image. There are four possibilities; neutral, sad, happy, and angry. This is perhaps the most challenging label to cluster due to the small variation between facial expressions.

- **Document Images:** Ancestry.com has provided some datasets consisting of document images. The labels associated with the documents specify what type of document it is, i.e. birth record, death record, marriage record, census record, passenger record, etc. Thus, any information on a document that does not differentiate between document types is effectively noise. We use these datasets with CONFIRM, a state of the art clustering algorithm specifically designed to extract features for and to cluster documents images [30].
  - **Washington Passenger:** The Washington Passenger dataset is a simple dataset consisting of 2000 passenger manifest images for both airlines and vessels. The label is binary and the two document types are quite different.
  - **PA Deaths:** Death records from Pennsylvania. The dataset contains 4974 documents distributed across 5 document types. The differences in the types are due to the records changing over time as different information is requested.

- **England Wales Census:** This dataset is a collection 23,056 images of 22 different document types from the 1911 England Wales Census. Some document types are extremely similar to each other, often differing by only a small column of data.

- **Leeds Butterfly:** This dataset contains 832 butterfly images from 10 different species of butterflies. This dataset was originally designed to classify each butterfly image based on a text description of each species. While this dataset is neither intended for classification nor clustering, each image is labelled and thus defines the target clustering used for validation. The dataset also provides segmentation masks for each image. The text descriptions and the segmentation masks are ignored in these experiments [31].

- **Leafsnap:** The Leafsnap dataset is a large dataset containing images of leaves. The dataset is separated into two smaller datasets. The first is the lab dataset which consists of more than 20,000 images taken in a lab. The other dataset consists of 7719 images taken in the field using mobile devices such as iPhones. A subset of 1000 images are randomly selected from 60 classes in the field images. The difficulty of this dataset comes from small variations between leaf types distributed across a large number of classes [18].

4.2 Clustering Algorithms

INCREMENT is validated by showing improvement in clusterings through refining the clusterings produced from various algorithms. Whenever an algorithm, such as kmeans, requires the number of clusters to be specified, it is set equal to the number of target clusters. We feel this yields a stronger initial clustering for INCREMENT to refine.

- **CONFIRM:** CONFIRM is a state of the art clustering algorithm designed for clustering images of documents. CONFIRM extracts relevant features from the documents such as vertical lines, horizontal lines and text lines. These features are used to create a codebook. The codebook is used to generate consistent feature vectors. CONFIRM

then performs feature selection and learns a distance metric which is used to efficiently cluster the document images. The output of CONFIRM is a new feature vector and a cluster assignment for each document image [30].

- **Kmeans:** Kmeans is one of the most widely used clustering algorithms. It works by iteratively creating cluster assignments and computing new centroid values. When based on euclidean distance, kmeans forms spherical clusters. It does not take into account density, standard deviation, nor any other statistic of the cluster. Even so, it performs quite well on numerous tasks. Kmeans forms the final clusterings of INCREMENT within the embedded space. [20]
- **HAC:** Hierarchical Agglomerative Clustering (HAC) initially treats each instance as its own cluster. HAC then iteratively merges the two closest clusters until clustering is finished. For the distance metric between clusters, complete link distances are used. Complete link distance is defined by $Complete(C_1, C_2) = \max_{\boldsymbol{x} \in C_1, \boldsymbol{y} \in C_2}(Dist(\boldsymbol{x}, \boldsymbol{y}))$ [10].
- **Spectral:** Using the eigenvalues of the affinity matrix, spectral clustering embeds the inputs into a low dimensional space. Kmeans performs the final clustering in the embedded space [22]. For two clusters, this is equivalent to solving the normalised cuts problem on the similarity matrix [29].
- **Active:** Using this scheme, each instance of the dataset is placed into its own cluster. This effectively skips the sub-clustering of optics and reduces INCREMENT to resemble an active clustering framework. This scheme loses efficiency in terms of user queries.
- **None:** This approach simply places every instance into a single cluster. Thus, the sub-clustering algorithm must find the pure sub-clusters without any previous algorithm performing the coarse initial separation.

4.3 Measures

Because the datasets are labeled, the performance of INCREMENT is measured using supervised metrics. In actual usage, the user defines an implicit target labelling which is sampled by querying the user. The measures take into account the label assigned to each cluster, how many clusters were formed, and how homogeneous each cluster is. Each of these measures portrays the quality of a clustering, thus the percent improvement between the initial clustering and the final refined clustering is presented. This shows how much improvement INCREMENT is able to make by refining the initial clustering.

- **Accuracy:** Accuracy compares the label assigned to the cluster and the true label of each instance in the cluster. The label of a cluster is determined to be the most common ground truth label within the cluster. Thus, accuracy is effectively a measure of how pure each cluster is. A drawback to accuracy is that it is biased towards over clustering. Clustering every point in its own cluster yields 100% accuracy, yet is a terrible clustering.
- **Homogeneity:** Homogeneity is an entropy based measure similar to accuracy. Homogeneity measures how many instances within each cluster have the same true label. The main difference between Homogeneity and accuracy is that Homogeneity, being entropy based, factors in how many instances of each label are included in the cluster. Accuracy only considers whether an instance's label is the same as the cluster label. However, Homogeneity is also biased towards over clustering and will yield a 1.0 when each cluster contains only a single instance [25].
- **Completeness:** Completeness is another entropy based measure that measures how well instances of the same ground truth label are clustered together. Thus, Completeness only scores 1.0 when every point with the same ground truth label is in a single cluster. Unlike Homogeneity, Completeness is biased toward under clustering. This measure will yield a 1.0 whenever there is only a single cluster containing all instance [25].

- **V-Measure:** V-Measure is the harmonic mean of Homogeneity and Completeness. Because V-measure is the harmonic mean, it strikes a balance between measuring both over clusters and under clusterings. The measure is bounded between 0.0 and 1.0. It can only yield a 1.0 when there is a perfect clustering and a 0.0 with the worst possible clustering [25]. Becker [6] has shown V-Measure to be equivalent to Normalized Mutual Information (NMI) when normalized by the sum of the class entropies.

- **Relative Jacquard Coefficient (JCC):** The Relative Jacquard Coefficient is a statistical measure created by Biswas to measure the similarity between two clusterings. JCC is analogous to the Jacquard Similarity Coefficient which measures the similarity between two sets. The Jacquard Similarity computes its measure by dividing the size of the intersection by the size of the union. JCC computes the similarity using the following as

$$JCC(C_1, C_2) = \frac{SS}{SS + SD + DS}$$

where SS is the total number of pairs that are in the same cluster within both clusterings C_1 and C_2. The term SD is the number of pairs of points which are placed in the same cluster in clustering C_1 but placed in different clusters in C_2. Likewise, DS is the number of pairs of points placed in the same cluster in C_2 but placed in different clusters in C_1. The JCC measure is computed with respect to the target clustering. Thus, JCC measures how similar the final and target clusterings are to each other [7].

4.4 Setup

While the base model for the embedder could use any supervised feature extractor, the embedder uses a conventional neural network with two hidden layers unless otherwise stated. The network is trained as if it were a "Siamese" network using Contrastive Loss. Section 5.4 shows results using a more powerful deep convolutional neural network. For the conventional network the first layer contains 500 hidden sigmoid units and the second contains 250 units.

The output embedding is a 100 dimensional vector. A mini-batch size of 10 was found to work well for training. Unless otherwise stated, the base learning rate of 0.01, the momentum of 0.9, and the maximum number of mini-batches set to 10000 were used in training. Number of mini-batch updates was chosen to be relatively low to prevent overfitting the data. Weight decay was not used. No parameter optimization was performed to obtain these settings, but they tended to work well in these experiments.

5 Results

Afew representative results which demonstrate the performance of INCREMENT are presented here. The entire set of results is found in the appendix.

5.1 Documents

Table 1 shows the results of running INCREMENT on the document image datasets which were initially clustered using CONFIRM. CONFIRM provides a strong baseline, providing highly accurate initial clusterings. While CONFIRM is fairly accurate, it tends to over cluster the documents, which is reflected in the Completeness score.

The results for the England Wales data set show that INCREMENT failed to discover 2 document labels through user queries. The classes that were not discovered contained 16 and 12 images respectively, while the majority of other labels contain above 200 images each. These classes were not presented to the user because they were buried within other clusters and were not selected as representatives using the farthest-first query scheme. However, the improvement in the other measures show that INCREMENT was still able to improve the overall quality of the clustering.

Due to the trade-off between Homogeneity and Completeness, V-Measure can only be improved by improving the over all clustering. Simply reducing the number of clusters would raise Completeness, but unless clusters were merged intelligently, Homogeneity would

		England	PA Deaths	Washington Passenger
		CONFIRM		
Accuracy	Initial	86.11	82.63	97.95
	Final	91.35	88.60	99.95
	Improvement	6.08 %	7.23 %	2.04 %
Homogeneity	Initial	0.87	0.58	0.92
	Final	0.91	0.69	0.99
	Improvement	3.87 %	18.43 %	8.07 %
Completeness	Initial	0.65	0.41	0.54
	Final	0.88	0.55	0.99
	Improvement	35.52 %	34.40 %	85.60 %
V-Measure	Initial	0.75	0.48	0.68
	Final	0.89	0.61	0.99
	Improvement	19.94 %	27.30 %	46.83 %
JCC	Initial	0.27	0.37	0.67
	Final	0.75	0.60	1.00
	Improvement	175.44 %	64.73 %	49.10 %
Clusters	Initial	47	6	4
	Final	20	5	2
	Target	22	5	2
Queries		812	126	74

Table 1: The results of INCREMENT refining clusters generated by CONFIRM from document images. One query was presented for each sub-cluster representative.

drop. Thus, because INCREMENT is able to raise both Homogeneity and Completeness, the clustering is both more pure and more concise.

The results presented in table 1 were computed after all sub-cluster representatives were presented as queries to the simulated user. INCREMENT uses a single query per sub-cluster. Hence, the maximum number of queries is presented. For the England Wales dataset, which contains 23,056 images, a maximum of 812 of the documents were labeled. Figure 6 shows that INCREMENT obtains similar results with 300 queries. The graphs show that INCREMENT usually surpasses the quality of the initial clustering early on. The improvement slows as the embedder struggles to learn additional information from newer queries. The improvement is also limited by noisy training labels resulting from impure sub-clusters.

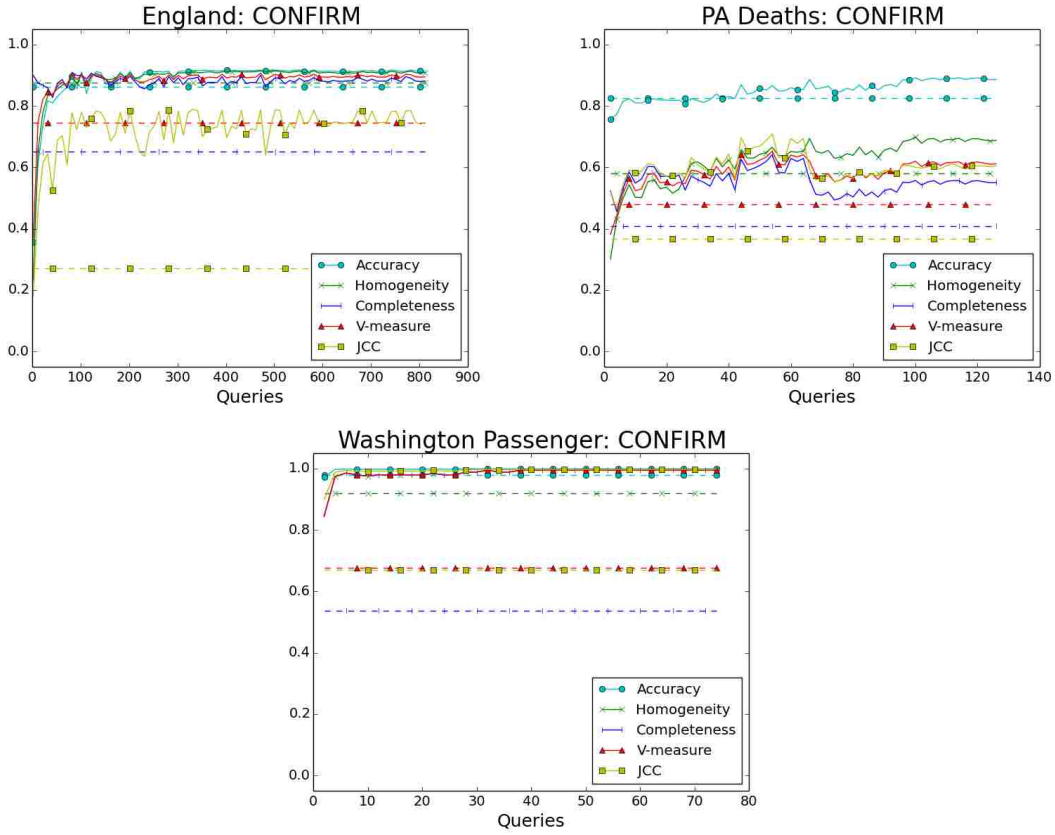


Figure 6: The results of running INCREMENT using CONFIRM on the document datasets with respect to the number of queries. The dotted lines represent the results of the initial clustering formed by CONFIRM.

		England	PA Deaths	Washington Passenger
		CONFIRM		
Accuracy	Initial	86.11	82.63	97.95
	Final	89.62	88.64	99.7
	Improvement	4.08 %	7.27 %	1.79 %
Homogeneity	Initial	0.87	0.58	0.92
	Final	0.90	0.69	0.97
	Improvement	3.45 %	18.43 %	5.43 %
Completeness	Initial	0.65	0.41	0.54
	Final	0.91	0.56	0.97
	Improvement	40.00 %	36.59 %	79.63 %
V-Measure	Initial	0.75	0.48	0.68
	Final	0.91	0.62	0.97
	Improvement	21.33 %	29.17 %	47.76 %
JCC	Initial	0.27	0.37	0.67
	Final	0.78	0.59	0.99
	Improvement	188.89 %	59.46 %	47.76 %
Clusters	Initial	47	6	4
	Final	20	5	2
	Target	22	5	2
Queries		82	66	4

Table 2: The results of INCREMENT refining clusters generated by CONFIRM from document images. The results presented in this table were taken from the knee of the curve before all representatives were presented as queries.

Table 2 shows the results of the above experiment using fewer queries. Since the user decides how many queries to answer, it is important that the initial queries yield the most improvement. Notice that the results are comparable to using the full set of queries. Even on the large England Wales dataset, INCREMENT makes significant improvements using only 82 queries. On the simpler Washington Passenger dataset, INCREMENT only requires 4 queries to achieve near perfect results. Thus, a user is able to label only a small subset of representatives while still improving the clustering. This enables the user to determine the trade-off between the number of user queries and the amount of improvement made. The more queries that the user decides to respond to, the better the final clustering.

5.2 Faces

The CMU Faces dataset show how INCREMENT adapts to forming different target clusterings by simulating different user feedback. Figure 7 Shows the results of INCREMENT as the user responds to more queries. The dotted line shows the results of the initial clustering. The figures depict how well and how quickly INCREMENT is able to begin improving on an initial clustering.

Once the sub-cluster representatives were selected, INCREMENT began to iteratively query the user. For each set of user queries, a new feature embedder was induced which mapped the input data into the embedding space and the results were measured. Due to the random initialization and stochastic nature of networks, the results displayed on the graph are not smooth. Even so,the general trend of the graph as queries are presented to the user is still visible.

When the user responds to only a few queries, the embedder only receives a small set of points to train on. Thus, the embedder tends to overfit to the instances it trains on. This impedes generalization performance, resulting in low scores across all measures. For the Names and Poses label sets, INCREMENT approaches baseline and begins to improve upon it in some measures with only 10 queries.

While it does not decrease the perfomance, INCREMENT does struggle with the Faces Expressions dataset. The Expressions label is a difficult problem due to subtle variations which distinguish between labels. The small 2 hidden layer neural network is not capable of extracting the subtle nuances of facial expressions. Shown in Table 3, both kmeans and INCREMENT achieve only about 25% accuracy, which is equivalent to a random clustering.

The number of clusters do not change because kmeans is given the correct number of clusters and INCREMENT is able to accurately recover it using the queries. The faces datasets usually generate around 74 sub-clusters. INCREMENT used a single representative per query and presented each representative.

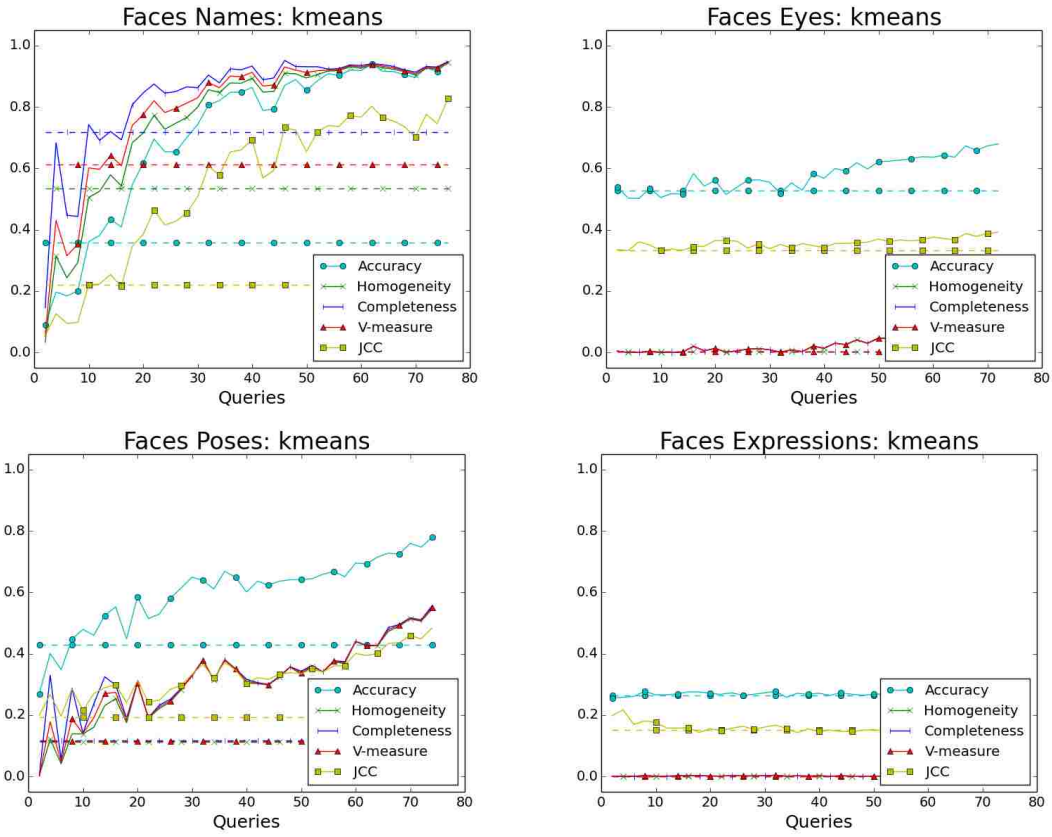


Figure 7: The results of INCREMENT over the Faces dataset using different target labels with respect to the number of queries. The initial clustering algorithm is kmeans (dotted line).

		Expressions	Eyes	Names	Poses
		Kmeans			
Accuracy	Initial	26.28	52.72	35.90	42.95
	Final	26.76	67.95	94.71	77.89
	Improvement	1.83 %	28.88 %	163.84 %	81.34 %
Homogeneity	Initial	~ 0.0	~ 0.0	0.53	0.11
	Final	~ 0.0	0.10	0.94	0.55
	Improvement	N/A	N/A	76.76 %	386.46 %
Completeness	Initial	~ 0.0	~ 0.0	0.72	0.12
	Final	~ 0.0	0.10	0.95	0.56
	Improvement	N/A	N/A	31.93 %	374.63 %
V-Measure	Initial	~ 0.0	~ 0.0	0.61	0.11
	Final	~ 0.0	0.10	0.95	0.55
	Improvement	N/A	N/A	54.40 %	380.60 %
JCC	Initial	0.15	0.33	0.22	0.19
	Final	0.14	0.39	0.83	0.48
	Improvement	-4.16 %	17.38 %	276.72 %	150.49 %
Clusters	Initial	4	2	20	4
	Final	4	2	20	4
	Target	4	2	20	4
Queries		72	72	76	74

Table 3: The results of running INCREMENT on the CMU Faces dataset with different labels using kmeans as a base. Note some scores were so low that they were close to 0. The percent improvement for these measures were not included to prevent division by exceptionally low numbers.

5.3 Active

Previously, the results from refining clusterings created with kmeans were presented. In these experiments, results from INCREMENT without using any initial clustering are presented. As described in the Active section above, each instance is placed into its own cluster for refinement. The results are shown in Figure 8.

For the Faces datasets, running INCREMENT in an active setting provides better results than from refining kmeans in terms of accuracy, V-Measure, and JCC. However, it requires significantly more queries to obtain the results. This is because INCREMENT can leverage the sub-clusters to train the embedder with more data, even if it is noisy. Whereas in this active setting, the embedder only trains on the instances presented to the user.

The results never obtain perfect accuracy during training, even when every point is presented to the user. The feature embedder only trains for 10000 iterations. The embedder is not allowed to overfit the data because the embedder should be capable of handling some noise in its training data. Even so, these results show that the embedder is able to better generalize by training on more homogeneous data, at the cost of user queries.

In each of these graphs, the improvement begins to taper as queries progress. This is due both to the embedder learning all that it can given its architecture and limited training. For example, the Faces Names dataset stops improving around 100-150 queries. After this point, additional user queries do not provide any additional information that the embedder can take advantage of.

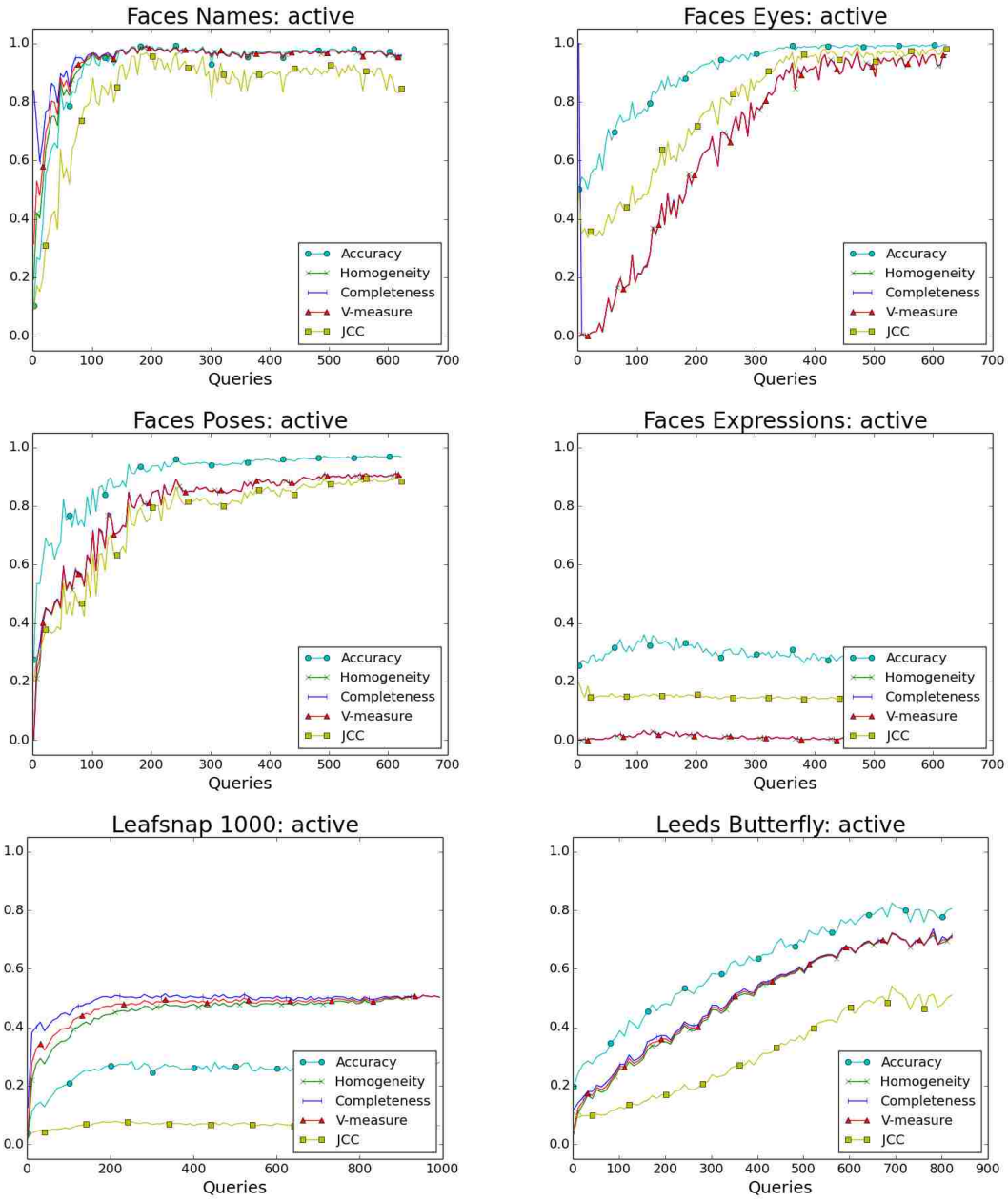


Figure 8: The results of INCREMENT without an initial clustering with respect to the number of queries.

5.4 Fine-Tuned CNN

Deep convolutional neural networks have dramatically improved the performance of various tasks within the visual domain. However due to their complexity, CNNs usually require much more data to train than can easily be provided by user queries alone. In the experiment, the embedder takes advantage of supervised pre-training, using network originally trained on similar, but different labeled data. The embedder then fine-tunes the CNN using user queries to speed convergence and improve performance. The more closely that the statistics of the pre-training data match the fine-tuning data, the better the final model.

For these experiments, the number of iterations was set to 8000 and the $l2$ weight decay was set as 0.0005. All other parameters were the same as previous experiments. The architecture of the fine-tuned network is the Alexnet model [17]. Alexnet was originally trained on the ILSVRC10 dataset [26]. ILSVRC10 is a large image dataset containing 1.2 million color images of many different sizes and scales. Alexnet resized these to be 227x227. Alexnet trained to classify images into 1000 different categories. Thus, the images are resized to be 227x227 to work with the initial Alexnet weights.

Table 4 shows the results of fine-tuning on a few of the harder datasets. In these experiments, the active base clustering was used and the number of queries was set to be 10% of the total data, similar to the number of representatives selected in previous experiments. These results are compared to the results from the active clustering using the previous conventional neural network and an equivalent number of queries.

The Faces Expressions, however, is the only dataset which performed slightly worse than its equivalent, non-Convolutional network. It still performed better than the refined kmeans experiment from above. The CMU Faces images are greyscale and smaller than the 227x227 that alexnet requires. To be able to fine-tune the network using this data, the images are upscaled and treated as color images. Thus, the statistics of the CMU Faces dataset are quite different than the statistics of ILSVRC10. This combined with the difficulty of the

		Expressions	Leafsnap	Butterfly
		Active		
Accuracy	Original	31.57 %	20.80 %	34.62 %
	CNN	28.37 %	31.90 %	77.76 %
	Improvement	-10.14 %	53.37 %	124.42 %
Homogeneity	Original	0.02	0.36	0.21
	CNN	0.01	0.51	0.74
	Improvement	-50.0 %	41.67 %	252.38 %
Completeness	Original	0.02	0.44	0.23
	CNN	0.01	0.64	0.75
	Improvement	-50.0 %	45.45 %	221.74 %
V-Measure	Original	0.02	0.40	0.22
	CNN	0.01	0.57	0.74
	Improvement	-50.0 %	42.50 %	236.36 %
JCC	Original	0.15	0.05	0.11
	CNN	0.15	0.11	0.51
	Improvement	$\sim$ 0.0 %	120.0 %	363.64 %
Queries		62	100	83

Table 4: The differences between using a 2 hidden layer neural network (Original) and a fine-tuned CNN. The number of queries allowed was set to be 10% of the data.

labels and the relatively few training examples prevented the CNN from generalizing to the Faces Expressions dataset well.

The Leafsnap 1000 and Leeds Butterfly datasets performed much better. The statistics of these datasets align much more closely to that of ILSVRC10 because they are both datasets contain color images. In each measure, the more powerful CNN improved INCREMENTs performance substantially. On the Leeds butterfly dataset especially, the simple neural network was unable to achieve such high performance on all measures until almost 700 queries were presented, about 84% of the data. The increase in JCC shows that the final clustering is much more similar to the target clustering than without fine-tuning, while the improvement in V-Measure shows that the clusterings are more concise and pure.

6 Discussion

INCREMENT is able to refine and improve clusterings from various different algorithms according to user feedback. By incorporating the user feedback into the creation of the

embedded feature space, INCREMENT is able to create different clusterings according to the user's target clustering. The CMU Faces dataset shows how well INCREMENT can adapt to different target clusterings implicitly defined by a user. The median percent improvement was 6.08% in accuracy, 27.3% in V-Measure, and 17.38% in JCC. Recall that in order to improve V-Measure, the resulting clustering must be more homogeneous and more concise.

The results found in the appendix show that the initial clustering algorithm does not drastically impact the performance of INCREMENT. With the exception of the Active and None schemes, the final results of refining the initial clusterings were fairly similar. The Active scheme, while it consistently performed better, required significantly more queries to obtain similar results. Refining the None scheme scored comparably with the other initial clusterings in all measures except accuracy, which scored slightly lower. Experiments using the Active and None schemes are not included in the presented averages because the initial measures are heavily skewed. The skewed values do not correctly reflect any improvement made by INCREMENT.

Overall, INCREMENT improved the accuracy of 15 out of 21 initial clusterings. The 6 initial clusterings that INCREMENT was not able to improve were formed using the Faces Expressions, the Leafsnap 1000, and the Leeds Butterfly datasets. Using the conventional neural network as the feature embedder, INCREMENT was unable to learn an appropriate embedding. This is likely because the simple network is unable to fully model these tasks. However, using the CNN as the embedder greatly improved the final clusterings. The results presented in Section 5.4 are not included in the average percent improvement because the initial clustering was formed with the Active scheme, which would skew the results.

By learning an embedded feature representation, INCREMENT can utilize both low level feature representations, such as the image pixels and carefully crafted high level features such as those provided by CONFIRM. CONFIRM learned a feature representation based on the horizontal, vertical, and text lines. INCREMENT is able to incorporate both user

feedback and feature information into the embedded space. In this new embedded space, the final clusters formed tend to be more similar to the target clustering.

While INCREMENT can be used more as an active clustering tool, refining a base clustering is more efficient in terms of user queries. INCREMENT uses the initial clustering to form sub-clusters. These sub-clusters enable us to provide more data to train the emedder, even being noisy. Thus, the more pure each sub-cluster, the better the results. In the active setting, each sub-cluster is just a single instance, completely pure. However, the larger each sub-cluster, the more data the embedder can train on, reducing the number of queries. Ideally, each sub-cluster would be large and pure with respect to the target clustering.

As long as INCREMENT is able to present a representative with each label, INCREMENT will compute the exact number of target clusters from the user feedback. This is because the final number of clusters is computed directly from the labels provided by the user. Thus, INCREMENT cannot overcluster the data. While the results in table 1 show INCREMENT can under cluster the data, this only happens when a member of some class is not presented to the user.

No attempt was made to ensure that INCREMENT will yield perfect results when all queries are presented because INCREMENT needs to be robust to noisy training data. None of INCREMENT's results are perfect, even after every point has been queried. This both allows the embedder to train on noisy data, as well as be more robust to the occasional erroneous query. To avoid overfit, only 10,000 mini-batch updates were used. Furthermore, if a user is willing to label every instance, then they can simply use those labels and do not need INCREMENT.

7 Conclusion and Future Work

INCREMENT is a promising tool by which users are able to refine clusterings based on their responses to queries. The results of the experiments show that INCREMENT is capable of forming refining clusterings provided by various algorithms on different datasets based on

user feedback, using a minimal amount of queries. INCREMENT is also capable of learning from different types of dense features such as pixel values or specially crafted representations.

INCREMENT improved the accuracy of 71% of the initial clusterings. On average accuracy improved by 37.0% with a standard deviation of 62.5, V-Measure improved by 292.0% standard deviation of 698.0, and JCC improved by 77.5% standard deviation of 99.3. The median improvement was 6.08% in accuracy, 27.30% in V-Measure, and 17.38% in JCC. Furthermore, INCREMENT yields the most improvement early, enabling users to respond to a minimal amount of queries and still greatly improve the quality of the clustering.

Future work includes better incorporating information contained in the base clustering. This would improve the quality of clusterings when there have not been enough queries posed to the user. A possible approach could be to seed the similarity of sub-cluster representatives with other representatives from the same original cluster. This would train the embedder to mimic the original clustering when a user did not respond to enough queries.

The quality of the training data could be improved by incorporating a cluster validity metric into INCREMENT. This could then be used to only query and train with sub-clusters above some quality threshold or to filter out the least confident points from the sub-clusters before training. Removing noise from the embedder's training data would enable it to learn quicker and more accurately as previously discussed.

Even with clean training data, not all pairs of points are equally useful for training. For example, two similar points mapped to approximately the same location will generate a small error gradient, while two highly separated points of the same class would adjust the embedder's weights more quickly. Schroff et al. [27] proposed a method for online selection of triplet points to speed convergence for Triplet Loss. A similar selection process could also speed convergence for Contrastive Loss.

The queries themselves could also be improved. INCREMENT currently uses a label based query. This type of query simply presents an instance for labelling. Since Contrastive Loss only requires similarities between pairs of points, it lends itself naturally to other forms

of queries as well. One such query could allow a user to hand cluster a small set of points. This would be more analogous to clustering.

The final clustering is created using kmeans. Kmeans clusters according to the bias that Contrastive Loss places on the embedded feature space. However, kmeans requires that the number of clusters be specified. While, the farthest-first traversal scheme is a reasonable approximation of the k centers problem, the approximation happens in the original feature space, not the learned embedded space. While kmeans works well, a different approach could improve the final clusterings and could include additional information such as the original cluster assignment.

The occasional under clustering is caused from not presenting an example of a class to the user. This usually happens because no example of the class is selected as a representative. Our current approach is to simply select the medoid of the sub-cluster as the representative. Other points might be better suited to representing a sub-cluster or we might select multiple representatives per sub-cluster.

References

[1] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod Record*, volume 28, pages 49–60. ACM, 1999.

[2] Pranjal Awasthi and Reza B. Zadeh. Supervised clustering. In J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 91–99. Curran Associates, Inc., 2010. URL `http://papers.nips.cc/paper/4115-supervised-clustering.pdf`.

[3] Pranjal Awasthi, Maria-Florina Balcan, and Konstantin Voevodski. Local algorithms for interactive clustering. *arXiv preprint arXiv:1312.6724*, 2013.

[4] Maria-Florina Balcan and Avrim Blum. Clustering with interactive feedback. In Yoav Freund, Lszl Gyrfi, Gyrgy Turn, and Thomas Zeugmann, editors, *Algorithmic Learning Theory*, volume 5254 of *Lecture Notes in Computer Science*, pages 316–328. Springer Berlin Heidelberg, 2008. ISBN 978-3-540-87986-2. doi: 10.1007/978-3-540-87987-9_27. URL `http://dx.doi.org/10.1007/978-3-540-87987-9_27`.

[5] Sugato Basu, Arindam Banerjee, and Raymond J Mooney. Active semi-supervision for pairwise constrained clustering. In *SDM*, volume 4, pages 333–344. SIAM, 2004.

[6] Hila Becker. *Identification and characterization of events in social media*. PhD thesis, Columbia University, 2011.

[7] Arijit Biswas. Semi-supervised and active image clustering with pairwise constraints from humans. 2014.

[8] Arijit Biswas and David Jacobs. Active image clustering with pairwise constraints from humans. *International Journal of Computer Vision*, 108(1-2):133–147, 2014.

[9] Jane Bromley, James W Bentz, Léon Bottou, Isabelle Guyon, Yann LeCun, Cliff Moore, Eduard Säckinger, and Roopak Shah. Signature verification using a siamese time delay neural network. *International Journal of Pattern Recognition and Artificial Intelligence*, 7(04):669–688, 1993.

[10] Daniel Defays. An efficient algorithm for a complete link method. *The Computer Journal*, 20(4):364–366, 1977.

[11] Marie Desjardins, James MacGlashan, and Julia Ferraioli. Interactive visual clustering. In *Proceedings of the 12th international conference on Intelligent user interfaces*, pages 361–364. ACM, 2007.

[12] Avinava Dubey, Indrajit Bhattacharya, and Shantanu Godbole. A cluster-level semi-supervision model for interactive clustering. In *Machine Learning and Knowledge Discovery in Databases*, pages 409–424. Springer, 2010.

[13] Nizar Grira, Michel Crucianu, and Nozha Boujemaa. Active semi-supervised fuzzy clustering for image database categorization. In *Proceedings of the 7th ACM SIGMM international workshop on Multimedia information retrieval*, pages 9–16. ACM, 2005.

[14] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *Computer vision and pattern recognition, 2006 IEEE computer society conference on*, volume 2, pages 1735–1742. IEEE, 2006.

[15] Dorit S Hochbaum and David B Shmoys. A best possible heuristic for the k-center problem. *Mathematics of operations research*, 10(2):180–184, 1985.

[16] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, and Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[17] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[18] Neeraj Kumar, Peter N. Belhumeur, Arijit Biswas, David W. Jacobs, W. John Kress, Ida Lopez, and Joo V. B. Soares. Leafsnap: A computer vision system for automatic plant species identification. In *The 12th European Conference on Computer Vision (ECCV)*, October 2012.

[19] Xin Liu, Yihong Gong, Wei Xu, and Shenghuo Zhu. Document clustering with cluster refinement and model selection capabilities. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 191–198. ACM, 2002.

[20] James MacQueen et al. Some methods for classification and analysis of multivariate observations. In *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability*, volume 1, pages 281–297. Oakland, CA, USA., 1967.

[21] Tom M Mitchell. Machine learning. wcb, 1997.

[22] Andrew Y Ng, Michael I Jordan, Yair Weiss, et al. On spectral clustering: Analysis and an algorithm. *Advances in neural information processing systems*, 2:849–856, 2002.

[23] Sun Park, Dong Un An, ByungRea Char, and Chul-Won Kim. Document clustering with cluster refinement and non-negative matrix factorization. In *Neural Information Processing*, pages 281–288. Springer, 2009.

[24] Ruggero Pensa, Cline Robardet, and Jean-Francois Boulicaut. *Constrained Clustering: Advances in Algorithms*, chapter Constraint-driven Co-Clustering of 0/1 Data, pages 123–148. Data Mining and Knowledge Discovery Series. Chapman & Hall/CRC Press, June 2008. URL `http://liris.cnrs.fr/publis/?id=3262`.

[25] Andrew Rosenberg and Julia Hirschberg. V-measure: A conditional entropy-based external cluster evaluation measure. In *EMNLP-CoNLL*, volume 7, pages 410–420, 2007.

[26] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 115(3):211–252, 2015. doi: 10.1007/s11263-015-0816-y.

[27] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. *arXiv preprint arXiv:1503.03832*, 2015.

[28] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52 (55-66):11, 2010.

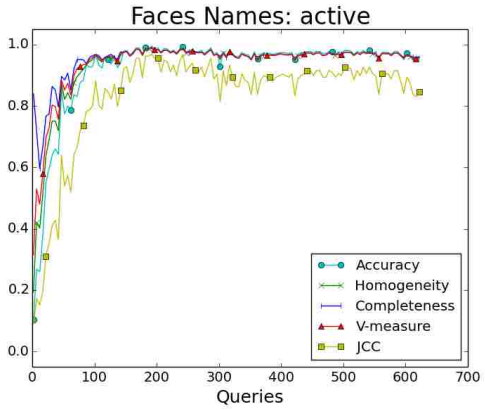
[29] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. Technical report, 2000.

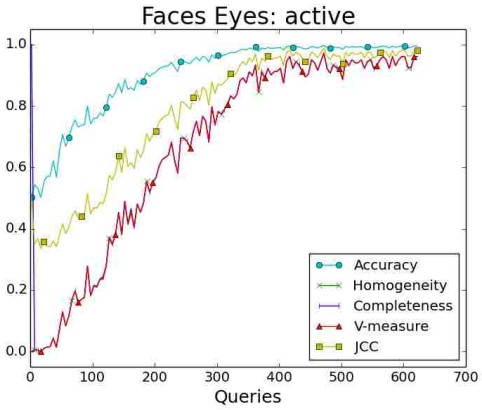
[30] Christopher Tensmeyer and Tony Martinez. Confirm - clustering of noisy form images using robust metrics. In *Family History Technology Workshop*, 2015.

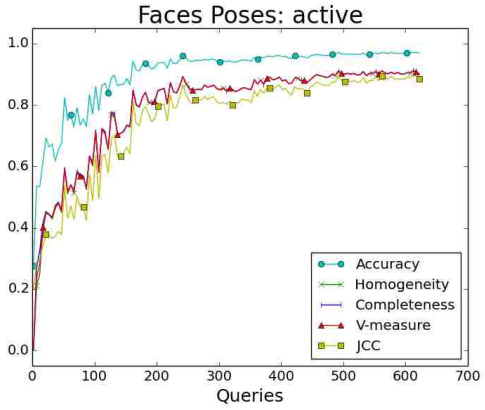
[31] Josiah Wang, Katja Markert, and Mark Everingham. Learning models for object recognition from natural language descriptions. In *Proceedings of the British Machine Vision Conference*, 2009.

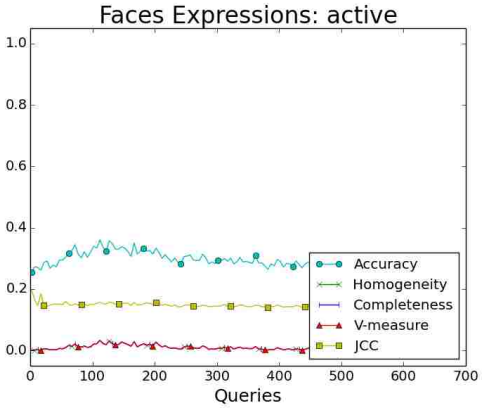
[32] David H Wolpert and William G Macready. No free lunch theorems for search. Technical report, Technical Report SFI-TR-95-02-010, Santa Fe Institute, 1995.

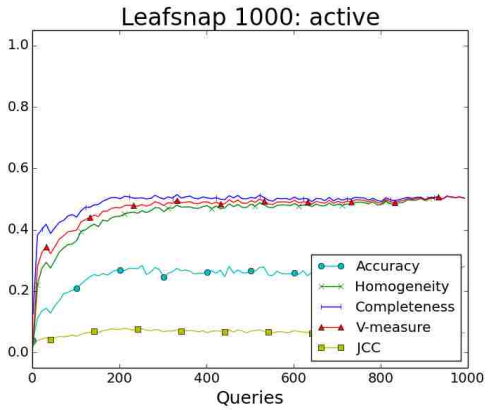
1 Appendix: Active

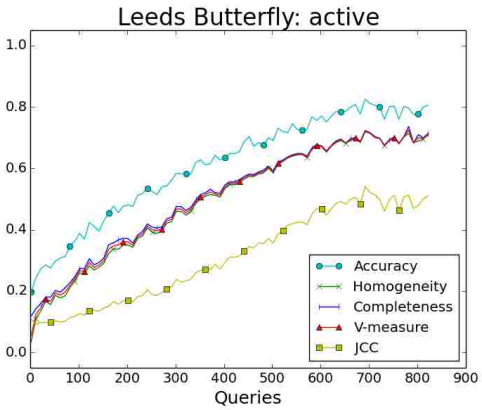






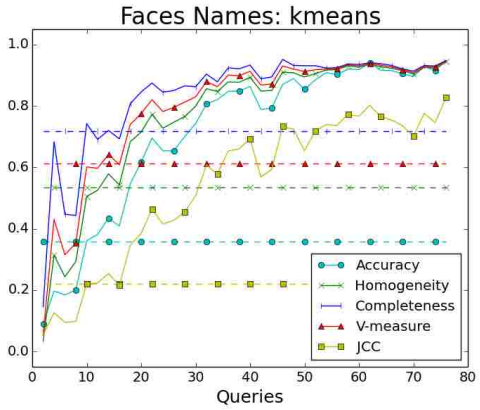


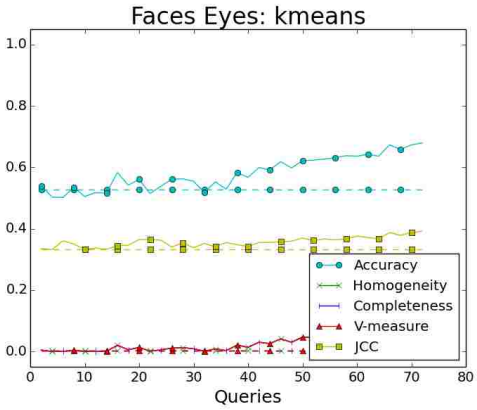


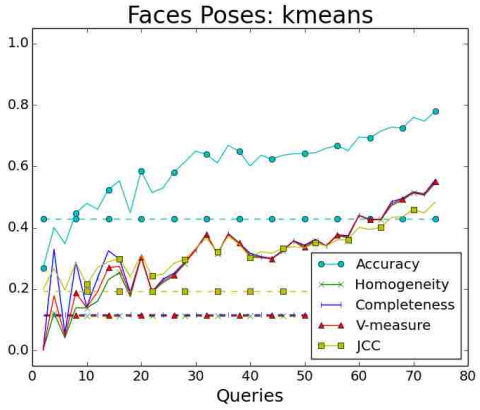


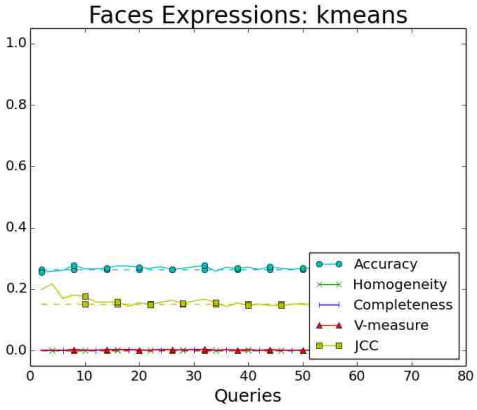
		Expressions	Eyes	Names	Poses	Leafsnap	Butterfly
		Active					
Accuracy	Initial	100.00	100.00	100.00	100.00	100.00	100.00
	Final	27.89	99.52	95.03	96.80	28.10	80.53
	Improvement	-72.11 %	-0.48 %	-4.97 %	-3.20 %	-71.90 %	-19.47 %
Homogeneity	Initial	1.00	1.00	1.00	1.00	1.00	1.00
	Final	~ 0.0	0.96	0.96	0.90	0.50	0.71
	Improvement	-99.69 %	-3.92 %	-4.43 %	-9.97 %	-49.68 %	-29.09 %
Completeness	Initial	0.22	0.11	0.47	0.22	0.58	0.34
	Final	~ 0.0	0.96	0.96	0.90	0.50	0.71
	Improvement	-98.53 %	792.20 %	106.46 %	318.15 %	-13.70 %	109.74 %
V-Measure	Initial	0.35	0.19	0.64	0.35	0.74	0.51
	Final	~ 0.0	0.96	0.96	0.90	0.50	0.71
	Improvement	-99.11 %	394.14 %	50.87 %	154.06 %	-31.67 %	40.09 %
JCC	Initial	~ 0.0	~ 0.0	~ 0.0	~ 0.0	~ 0.0	~ 0.0
	Final	0.14	0.98	0.85	0.88	0.06	0.51
	Improvement	N/A	N/A	N/A	N/A	N/A	N/A

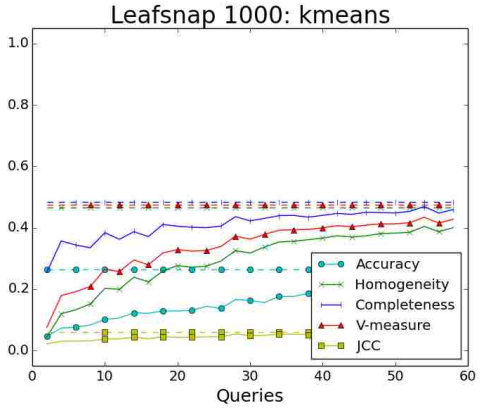
2 Appendix: Kmeans

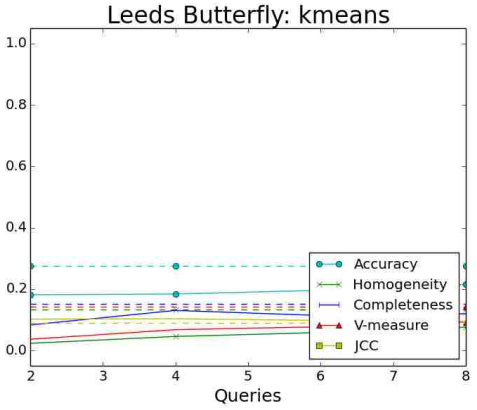






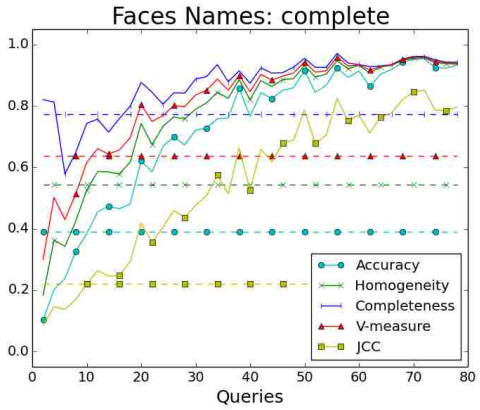


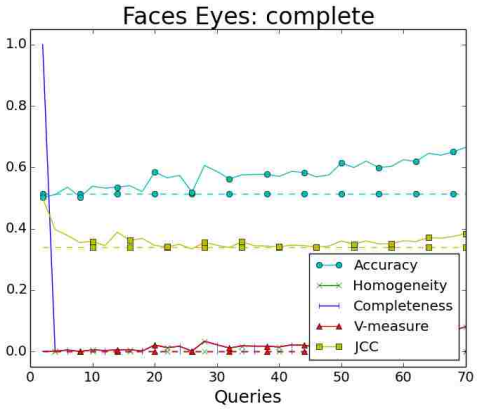


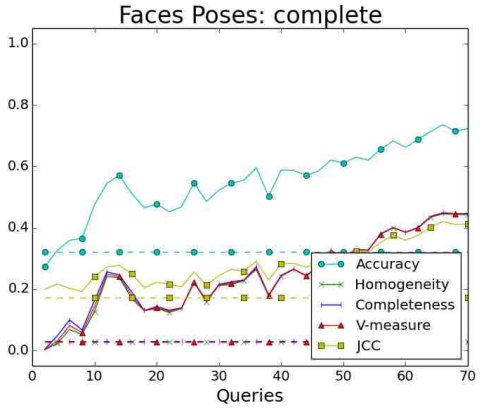


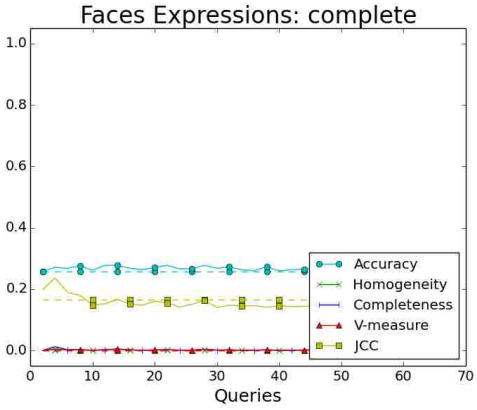
		Expressions	Eyes	Names	Poses	Leafsnap	Butterfly
		Kmeans					
Accuracy	Initial	26.28	52.72	35.90	42.95	26.30	27.52
	Final	26.76	67.95	94.71	77.89	20.70	21.51
	Improvement	1.83 %	28.88 %	163.84 %	81.34 %	-21.29 %	-21.84 %
Homogeneity	Initial	~ 0.0	~ 0.0	0.53	0.11	0.47	0.13
	Final	~ 0.0	0.10	0.94	0.55	0.40	0.08
	Improvement	N/A	N/A	76.76 %	386.46 %	-13.89 %	-43.68 %
Completeness	Initial	~ 0.0	~ 0.0	0.72	0.12	0.48	0.15
	Final	~ 0.0	0.10	0.95	0.56	0.46	0.12
	Improvement	N/A	N/A	31.93 %	374.63 %	-5.23 %	-21.12 %
V-Measure	Initial	~ 0.0	~ 0.0	0.61	0.11	0.47	0.14
	Final	~ 0.0	0.10	0.95	0.55	0.43	0.09
	Improvement	N/A	N/A	54.40 %	380.60 %	-9.85 %	-34.92 %
JCC	Initial	0.15	0.33	0.22	0.19	0.06	0.09
	Final	0.14	0.39	0.83	0.48	0.06	0.09
	Improvement	-4.16 %	17.38 %	276.72 %	150.49 %	-6.52 %	3.77 %

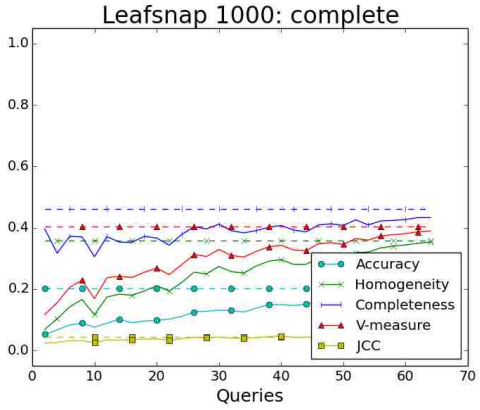
3 Appendix: HAC - Complete Link

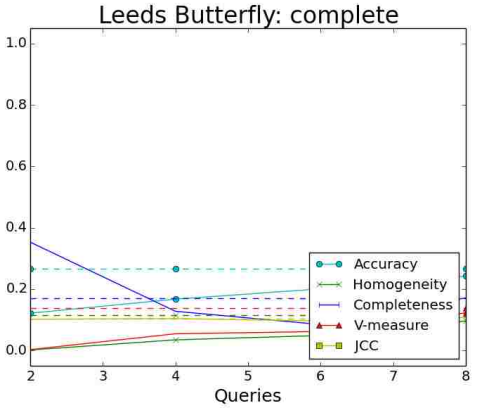






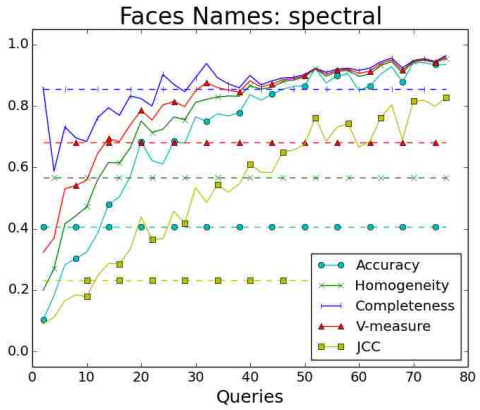


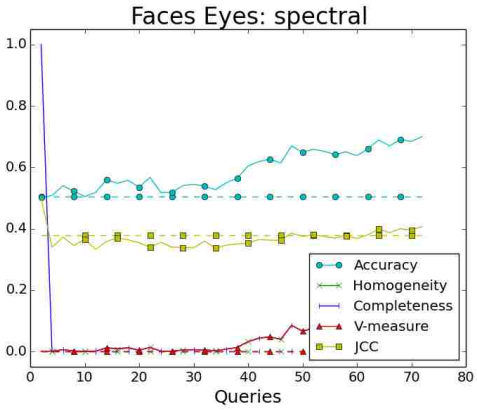


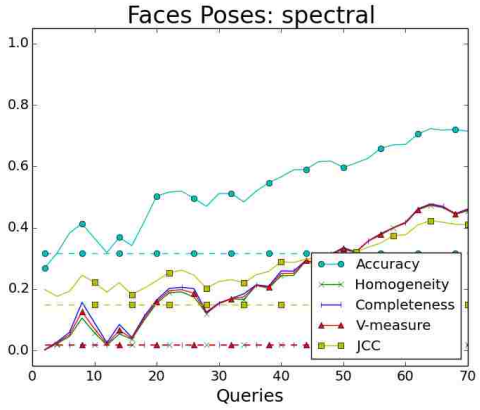


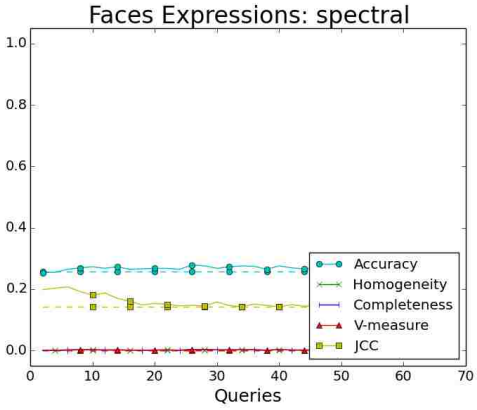
		Expressions	Eyes	Names	Poses	Leafsnap	Butterfly
		Complete					
Accuracy	Initial	25.80	51.28	38.94	32.05	20.20	26.56
	Final	26.60	66.51	93.27	72.28	17.10	24.28
	Improvement	3.11 %	29.69 %	139.51 %	125.50 %	-15.35 %	-8.59 %
Homogeneity	Initial	~ 0.0	~ 0.0	0.54	0.03	0.36	0.12
	Final	~ 0.0	0.08	0.94	0.44	0.35	0.10
	Improvement	N/A	N/A	72.28 %	1524.79 %	-1.06 %	-17.02 %
Completeness	Initial	~ 0.0	~ 0.0	0.77	0.03	0.46	0.17
	Final	~ 0.0	0.08	0.94	0.45	0.43	0.17
	Improvement	N/A	N/A	22.17 %	1330.62 %	-6.32 %	1.09 %
V-Measure	Initial	~ 0.0	~ 0.0	0.64	0.03	0.40	0.14
	Final	~ 0.0	0.08	0.94	0.44	0.39	0.12
	Improvement	N/A	N/A	47.31 %	1428.04 %	-3.43 %	-10.53 %
JCC	Initial	0.17	0.34	0.22	0.17	0.04	0.10
	Final	0.14	0.38	0.80	0.41	0.05	0.11
	Improvement	-14.79 %	12.97 %	261.77 %	139.99 %	15.52 %	3.67 %

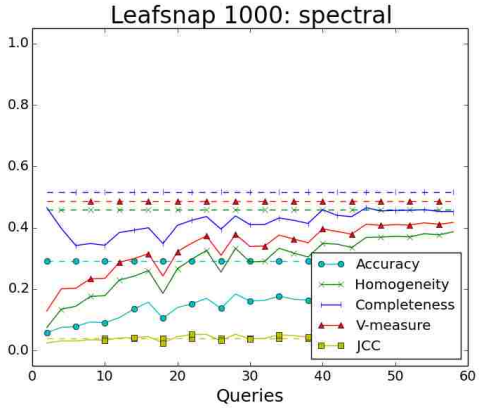
4 Appendix: Spectral

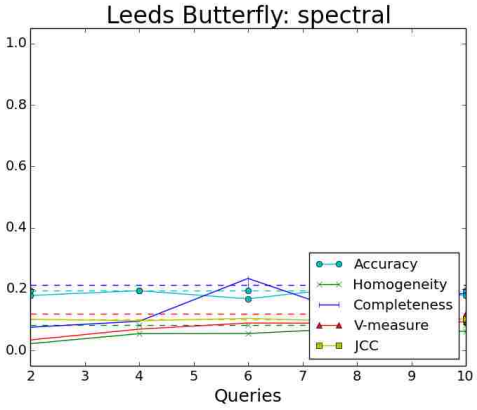






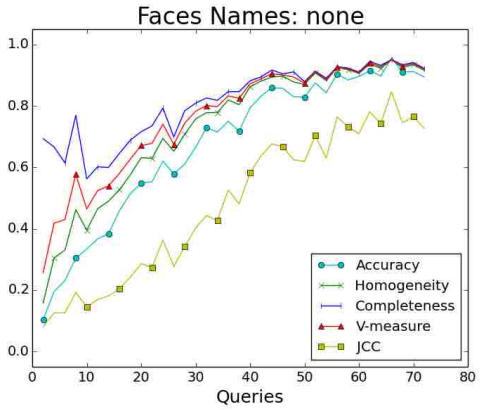


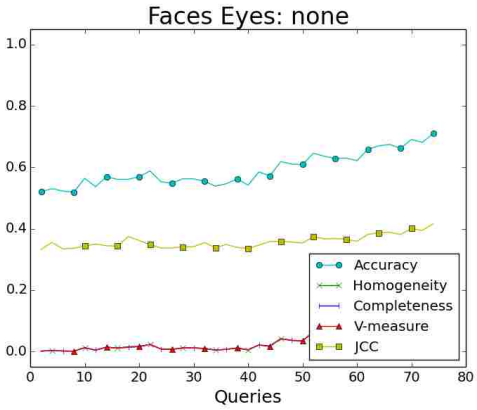


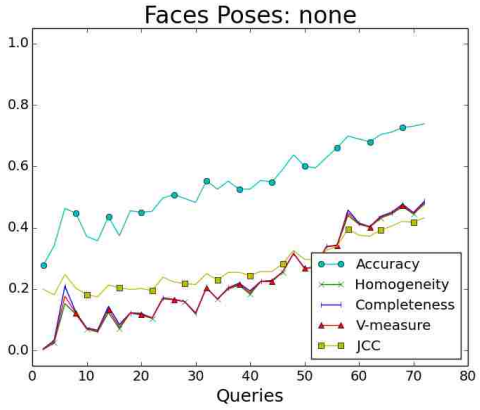


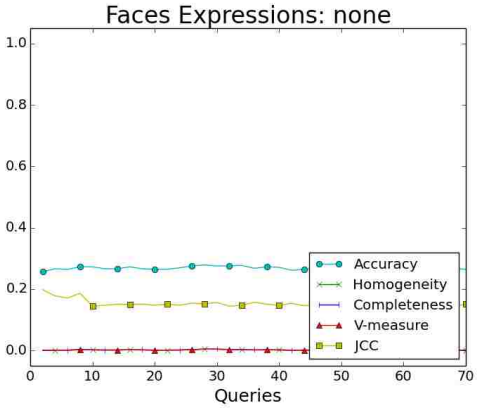
		Expressions	Eyes	Names	Poses	Leafsnap	Butterfly
		Spectral					
Accuracy	Initial	25.80	50.48	40.70	31.73	29.10	19.47
	Final	26.28	70.03	93.43	71.47	19.60	17.91
	Improvement	1.86 %	38.73 %	129.53 %	125.25 %	-32.65 %	-8.02 %
Homogeneity	Initial	~ 0.0	~ 0.0	0.57	0.02	0.46	0.08
	Final	~ 0.0	0.12	0.95	0.46	0.39	0.06
	Improvement	N/A	N/A	68.69 %	2429.07 %	-15.57 %	-26.13 %
Completeness	Initial	~ 0.0	~ 0.0	0.86	0.02	0.52	0.21
	Final	~ 0.0	0.12	0.96	0.46	0.45	0.19
	Improvement	N/A	N/A	12.68 %	2429.67 %	-12.41 %	-12.09 %
V-Measure	Initial	~ 0.0	~ 0.0	0.68	0.02	0.49	0.12
	Final	~ 0.0	0.12	0.96	0.46	0.42	0.09
	Improvement	N/A	N/A	40.83 %	2429.35 %	-14.11 %	-22.65 %
JCC	Initial	0.14	0.38	0.23	0.15	0.04	0.10
	Final	0.14	0.41	0.83	0.41	0.06	0.10
	Improvement	0.67 %	7.97 %	256.12 %	173.68 %	41.84 %	1.14 %

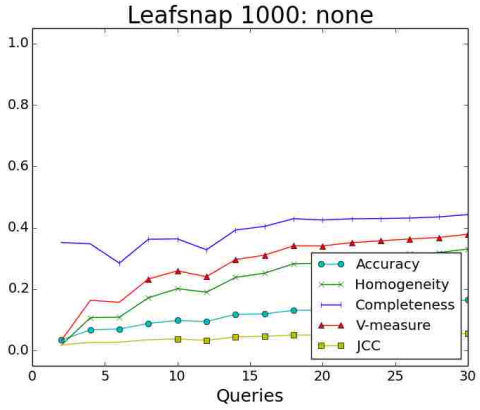
5 Appendix: None

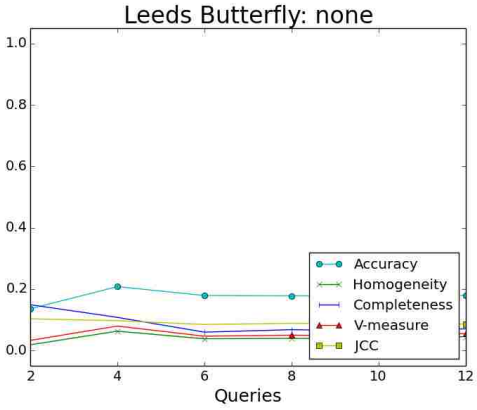












		Expressions	Eyes	Names	Poses	Leafsnap	Butterfly
		None					
Accuracy	Initial	25.32	50.16	5.13	25.16	3.10	12.02
	Final	26.44	70.99	89.42	73.88	16.50	17.91
	Improvement	4.43 %	41.54 %	1643.82 %	193.63 %	432.26 %	49.01 %
Homogeneity	Initial	~ 0.0	~ 0.0	~ 0.0	~ 0.0	~ 0.0	~ 0.0
	Final	~ 0.0	0.13	0.92	0.48	0.33	0.05
	Improvement	N/A	N/A	N/A	N/A	N/A	N/A
Completeness	Initial	1.00	1.00	1.00	1.00	1.00	1.00
	Final	~ 0.0	0.13	0.92	0.49	0.44	0.07
	Improvement	-99.91 %	-86.81 %	-7.76 %	-51.43 %	-55.73 %	-92.94 %
V-Measure	Initial	~ 0.0	~ 0.0	~ 0.0	~ 0.0	~ 0.0	~ 0.0
	Final	~ 0.0	0.13	0.92	0.48	0.38	0.06
	Improvement	N/A	N/A	N/A	N/A	N/A	N/A
JCC	Initial	0.25	0.50	0.05	0.25	0.02	0.10
	Final	0.15	0.42	0.73	0.43	0.06	0.09
	Improvement	-39.34 %	-16.68 %	1396.85 %	73.76 %	209.41 %	-15.23 %