UNSUPERVISED VISUAL KNOWLEDGE DISCOVERY AND ACCUMULATION IN DYNAMIC ENVIRONMENTS

A Dissertation

Submitted to the Faculty

of

Purdue University

by

Ziyin Wang

In Partial Fulfillment of the

Requirements for the Degree

of

Doctor of Philosophy

December 2019

Purdue University

West Lafayette, Indiana

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF DISSERTATION APPROVAL

Dr. Gavriil Tsechpenakis, Chair

School of Science

Dr. Voicu S. Popescu

School of Science

Dr. Murat Dundar

School of Science

Dr. Jean F. Honorio Carrillo

School of Science

Approved by:

Dr. Clifton W. Bingham

Head of the School Graduate Program

ACKNOWLEDGMENTS

I would like to express my deepest appreciation to my advisor, Dr. Gavriil Tsechpenakis, who has continuously supported my Ph.D. study and research, for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and writing of this thesis. I could not have imagined having a better advisor and mentor for my Ph.D. study.

I would like to thank the rest of my committee: Dr. Voicu S. Popescu, Dr. Murat Dundar, and Dr. Jean F. Honorio Carrillo, for their encouragement, insightful comments, and hard questions. I would also like to thank Dr. Dimitra Panagou from the University of Michigan, for providing valuable aerial video data and the experimental environment.

My sincere thanks also go to Miss. Yicheng Cheng, who has inspired me countless ideas in my research and philosophical thought beyond scientific research. I would like to thank Mr. Sepehr Farhand, who has brought me constructive suggestions on my research and writing.

Last but not least, I would like to thank my mother, Yuying Mao, for giving birth to me, and for supporting me, financially and spiritually, throughout my Ph.D.

TABLE OF CONTENTS

Page

Page

LIST OF TABLES

LIST OF FIGURES

Figure Page

ABSTRACT

Ziyin Wang Ph.D., Purdue University, December 2019. Unsupervised Visual Knowledge Discovery and Accumulation in Dynamic Environments. Major Professor: John Q. Professor.

Developing unsupervised vision systems in Dynamic Environments is one of the next challenges in Computer Vision. In Dynamic Environments, we usually lack the complete domain knowledge of the applied environments before deployment, and computation is also limited due to the need for prompt reaction and on-board computational capacity. This thesis studies a series of key Computer Vision problems in Dynamic Environments.

First, we propose a stream clustering algorithm and a number of variants for unsupervised feature learning and object discovery, which possess several crucial characteristics required by applications in Dynamic Environments, e.g. fully progressive, arbitrary similarity measure, matching object while the feature space is increasing, etc. We give strong provable guarantees of the clustering accuracy in statistic view. Based on the above the approaches, we tackle the problem of discovering aerial objects on-the-fly, where we assume all of the objects are unknown at the beginning of the deployment. The vision system is required to discover from the low-level features to salient objects on-the-fly without any supervision. We propose a number of approaches with respect to object proposal, tracking, recognition, and localization to achieve real-time performance. Extensive experiments on prevalent aerial video datasets showed that the approaches efficiently and accurately discover salient ground objects.

To explore complex and deep architectures in Dynamic Environments, we propose Unsupervised Deep Encoding which unifies traditional Visual Encoding and Convolutional Neural Networks. We found strong relationships between single-layer Neural Networks and Clustering and thus performed unsupervised feature learning at each layer from the

feature maps of the previous layer. We replaced the dot product inside each neuron with a similarity measure, which is also used in unsupervised feature learning. The weight vectors of our network are initialized by cluster centers. Therefore, one feature map is a visual encoding of its previous feature map. We applied this mechanism to pre-training Convolutional Neural Networks for image classification. It has been found by extensive experiments that pre-training benefits the network more reliable learning dynamics (e.g.fast convergence without Batch Normalization) and better classification accuracy.

1. INTRODUCTION AND RELATED WORKS

1.1 Dynamic Environments: Definition and Motivation

Vision demands intensive computation and supervision due to its complexity, diversity, and dimensionality of images and videos. Visual tasks also require images to be densely labeled, e.g. tight bounding boxes with class labels for Object Detection [1] [2] [3] [4] [5] [6], pixel-level class labels for Semantic Segmentation [4] [7] [8], which makes training data usually expensive and sometimes limits the improvement of more advanced Computer Vision models. Such circumstances also bound vision to heavy computational platforms with a long period of training-deploying cycles. Recent advances in modern Computer Vision powered by Deep Convolutional Neural Networks (CNN) have pushed the state-of-the-art into a satisfactory level. However, apart from expensive labeling and training processes, the drawbacks are as follows.

First of all, a well trained deep CNN is not able to discover new visual knowledge, neither from feature level nor the scene level. All detection and recognition are limited to classes of the training set. Since the architecture of a CNN model is fixed through training to deploying, the prediction will always be encoded into one of the K classes in the training set. At the same time, we can hardly extract information indicating unknown classes or even similarities between different samples.

Secondly, most paradigms train a large model with 1000 or more classes to generalize features of each level [9] [10] [11] [12], therefore most computation will be wasted and sometimes it biases for a specific application. Many classes may never appear when the model is deployed to a specific application. If we collected data of previously unseen classes after training, because of Stochastic Gradient Descend, it is inevitable to go back and retrain the entire model along with all the dataset collected before. Current transfer learning assumes the low-level patterns are consistent between training data and transferred

application. Even if sometimes low-level features can be transferred to the updated models, the last few levels still cannot avoid retraining.

Thirdly large models require heavy computation in the testing/deploying stage. Recent advances in Image Classification, Object Detection, and Semantic Segmentation rely on the growth of GPU power. Therefore, applications are bound to heavy computation platforms or systems with a stable network connection. For those slim devices, especially small mobile devices, with limited computational power, models are still limited to traditional methods.

Regardless of the above drawbacks, deep architecture is still heavily studied in Computer Vision and significant progress has been achieved in recent years. The general trend of Convolutional Neural Networks is that models are becoming deeper and larger [13] [14]. The philosophy behind this trend is partially due to the complexity of visual information where clear and closed-form theory is currently unavailable. Therefore, encoding more information usually results in a model with a bigger capacity. Though larger models are emerging along with fast-improving modern computational platforms, current state-of-the-art has not met the applicational bar of many real-world tasks. We suggest this difficulty roots in the attempt of building a vision for the general environments, where the vision system is expected to recognize thousands or more categories of objects.

In this thesis, instead of considering the above heavyweight application scenarios, we target at "**Dynamic Environments**" which is featured by one or more of the following situations:

- Computation power is limited to mobile-level CPUs or small GPUs, and network connection is expensive. Learning is expected to be performed on-board within a reasonably short period of time.

- We lack strong domain knowledge of the deployed environment, hence training data is insufficient or unavailable. Thus we expect the system to discover, learn and accumulate emerging visual knowledge.

- The application scenario is specific, and there is a moderate number of object categories. Only task-related visual knowledge is expected to be learned.

- Images or videos possess special characteristics such that fast computation is tractable, e.g. aerial images or videos.

Many application environments are dynamic, e.g. Unmanned Aerial Vehicles (UAVs) flying over a harsh, dangerous and hostile environment, AI Apps in mobile devices for security and education, embedded systems of intelligent devices, smart sensors in advance manufacturing, a vision system for agents launched into alien planets, etc. In this thesis, we focus on two application environments: (1) unsupervised aerial object discovery and detection on-the-fly; and (2) Light-weight Convolutional Neural Networks for image recognition. It worth noting that our method is general to the Computer Vision and Machine Learning domain, and can be easily navigated to other above applications. We select these two application environments as our key focus because (will be explained in detail in following sections) (1) there are relatively sufficient public datasets available (compare to other sub-domains); (2) public attention and previous works on this domain are insufficient; and (3) valuable real-life applications can be directly implemented from methods that solve these problems.

1.2 Fully Stream Clustering for Unsupervised Feature Learning and Object Discovery in Dynamic Environments

Clustering is regarded as the core of unsupervised learning. The clustering problem assumes data possesses grouping patterns that related to real-world semantics and the ideal algorithm should be able to discover such patterns automatically without human intervention. Vision tasks in Dynamic Environments require efficient Stream Clustering for unsupervised learning. It is necessary for a brief review of Visual Encoding to illustrate the reason why a new stream clustering is necessary.

The most common methodology for Visual Encoding (also considered in this thesis) is known as Bag-of-Visual-Words (BoVW) [15] [16] [17]. In BoVW, we first extract small

image patches (usually follows some transformation, e.g. SIFT [18], HoG [19]) from the entire dataset of images in a sliding-window fashion. The collected local image patches are then fed into a clustering algorithm to build a bag of visual words. These cluster centers are known as a dictionary, each of which (known as a feature or word) represents a low-level visual pattern that appears frequently across the dataset. Given an image or a Region of Interest (RoI), each local patch is assigned to its closest feature and thus we can use a histogram to express the image or region. Such histograms are known as descriptors. We can then consider the descriptors of all the images or regions as the standard input data to standard machine learning models for supervised or unsupervised learning. The above process may also be applied to the Gaussian Pyramid of each image to learn features from different scales.

In an algorithm view of Visual Encoding, one underlying mechanism that affects online learning is that the bins of the histogram are ordered. The cluster centers given by the algorithm are usually stored in an array-like container whose order is usually random for each run of the clustering algorithm. Therefore, encoding cannot start unless all the visual words are learned and stored in an array. In the stream scenario, images arrive as a sequence and the dictionary changes along with the stream. Therefore, the clustering algorithm has to be able to organize the dictionary order such that the similarity between RoI descriptors encoded from different dictionaries can be computed in linear time. The most natural way, which is considered by this thesis and illustrated by Fig.1.1 and 1.3, is to maintain the cluster centers chronologically according to the time of the feature appeared in the stream.

According to the above discussion, as well as those of previous subsections, the key properties of the Stream Clustering strictly required by the above two Dynamic Environments are:

- Ordered: Clustering centers must be kept in chronological order.
- Adaptive: It has to be able to find the number of clusters automatically and accurately.
- Fast: The data set is very large and can only be passed once.
- Light: No data can be stored with size proportional to the entire dataset.

- Prompt: Clustering results must be updated immediately along the stream.
- Universal: The algorithm should be suitable for arbitrary similarity measure.
- Theoretical: Strong provable accuracy and top-tier performance.
- Denoising: The algorithm be able to eliminate noise along the stream.
- Sublinear: The clustering result has to provide an optional data structure such that an image patch can be encoded efficiently without searching through all of the clustering centers when the number of clusters is very large.

We note that though numerous clustering algorithms have been proposed in the last few decades, to the best of our knowledge, there is no algorithm that satisfies all the above properties. In most hash environments, the learning system requires all clustering properties above. Therefore, it is necessary to design a new clustering method to solve problems in Dynamic Environments. Obviously, it is challenging and hence another key contribution of this thesis. Table1.1 summarizes popular and recent state-of-the-art stream clustering algorithms. In this table, we only include modern stream clustering proposed since 2010, except BIRCH (which regarded as a speed benchmark), where earlier works have been repeatedly reported to be outperformed by modern methods.

Building perfect clustering centers is still a challenging problem due to several reasons, as detailed in [27] [29]. Many methods produce satisfactory results in terms of accuracy, however, there are cases where efficiency is also crucial. For instance, in unsupervised object detection [28], where deep learning [29] cannot be applied due to the inherent lack of prior knowledge, clustering is usually employed to detect similar salient features across images. In such problems, computational complexity may not be an obvious issue, since the process can take place off-line. However, if the same application is used in robotics, e.g., on agents where images are continually captured, efficient clustering on-the-fly becomes a bottleneck.

Affinity Propagation (AP) [30] and Density Peaks (DP) [31] are representative examples of the state-of-the-art and have been proven to be among the most accurate approaches

Table 1.1.
Summary of modern stream clustering.

	Ord.	Adap.	Fast	Light	Prompt	Univ.	Theor.	Denoi.	Subl.
ours 2016	Y	Y	Y	Y	Y	Y	Y	Y	Y
StrKM 2011 [20]	N	N	Y	N	N	N	Y	Y	N
BICO 2012 [21]	N	N	Y	Y	Y	N	Y	N	Y
BIRCH 1998 [22]	N	Y	Y	Y	Y	N	N	N	Y
Kobren 2017 [23]	N	N	Y	Y	Y	N	Y	N	Y
Sculley 2010 [24]	N	N	Y	Y	Y	N	Y	N	N
Liberty 2016 [25]	N	N	Y	Y	N	N	Y	N	N
Bachem 2016 [26]	N	N	Y	Y	N	N	Y	N	Y

for clustering real datasets, yet they require quadratic complexity in time and space. To improve complexity, the K-means [32] principle has been a popular foundation, with some variants, such as K-means++ [33], yielding better computational times. Similar algorithms, especially EM clustering [34], have also been very popular due to their acceptable accuracy and relatively low complexity. In many cases though, such as the unsupervised object detection paradigm described above, the number of clusters is a priori unknown. AP and DP are able to detect the number of clusters accurately, but they can only be applied to smaller datasets. On the other hand, the Mean-shift clustering algorithm [35] provides a nice balance between accuracy and speed, it has been applied to computer vision problems, such as image segmentation, yet it has some accuracy limitations primarily due to initialization issues.

1.2.1 Related Works

The data stream clustering problem gave rise to a number of seminal works [22] [36] [37] [38]. The recent state-of-the-art focuses more on approximating well-known algorithms [39] [24] [20] [21] [40] [26]. [24] describes a heuristic method for online K-means but without approximation boundary guarantees. [20] presents a sampling method along with Map-Reduce to construct a weighted subset (coreset), such that running K-means++ on coreset can be a proven approximation of the original algorithm. The coreset idea was also adopted in [21] and [40]. A recent work in [25] presents an online approximation of K-means that instantiates a new cluster when an incoming data point is far away from existing clusters. In [26], Markov Chain Monte Carlo sampling is used as a seeding approximation of K-means++ algorithm. [23] describes a hierarchical clustering method along with tree rotation to approximate K-means with a tractable solution when the number of clusters is large.

A key attribute of data streams is that the data size can potentially be very large, which calls for designing scalable methods, such as [41]. [42] describes a parallel method to scale the K-means++ initialization process. In [43], a Map-Reduce model is used to minimize

dependency between iterations in K-means for efficient scaling. [44] shows a scalable solution in generating coreset for K-means and K-median problems.

Some hierarchy-based clustering algorithms, like BIRCH (balanced iterative reducing and clustering using hierarchies) [22], are reported to provide increased efficiency. BIRCH builds a CF (clustering feature) tree and uses an agglomerative clustering algorithm to merge leaves towards a specific number of clusters. Since agglomerative clustering is very expensive with $O(N^2 lgN)$ time complexity, the efficiency is guaranteed only if the user chooses the correct parameters to generate a reasonable amount of leaves. Although it can be fast with careful parameter selection, it does not provide natural clusters and its performance is usually sensitive to the permutation of the data [22].

Most algorithms that approximate foundational non-stream methods usually provide limited clustering quality, as explained above. Some other accurate solutions that have been approximated in streaming problems have higher complexity, and their streaming counterparts are more focused on maintaining clustering quality than achieving the best possible speed. For instance, [45] that uses a Dirichlet Process mixture model is a popular Bayesian nonparametric model for small, low-dimensional data, with $O(iNd^3)$ complexity, where i is the number of Gibbs Sampling iterations, N is the data size and d is the data dimension. [46] are faster approximations that reduce the cubic complexity and overcome the bottleneck of Gibbs Sampling. [30] maintains three $N \times N$ matrices to discover clusters in small datasets, with time and space complexities $O(iN^2)$ and $O(N^2)$ respectively. [31] maps the original data set into a 2-D density map with complexity $O(N^2)$ in time and space, such that few density peaks can be manually selected. Nowadays, even moderate-sized datasets are far beyond the capacity of most of these methods.

The greedily iterative paradigm is still very popular in Computer Vision, despite its drawbacks, due to its simplicity and its acceptable efficiency for some applications. Specifically, despite the advances of deep neural networks in image classification, building visual vocabularies [30, 7, 16, 23] can still provide significant benefits for various tasks, including unsupervised object detection in image collections [28] or in streaming data where new categories may emerge. Coates et al. [47] use simple K-means clustering and a triangle metric

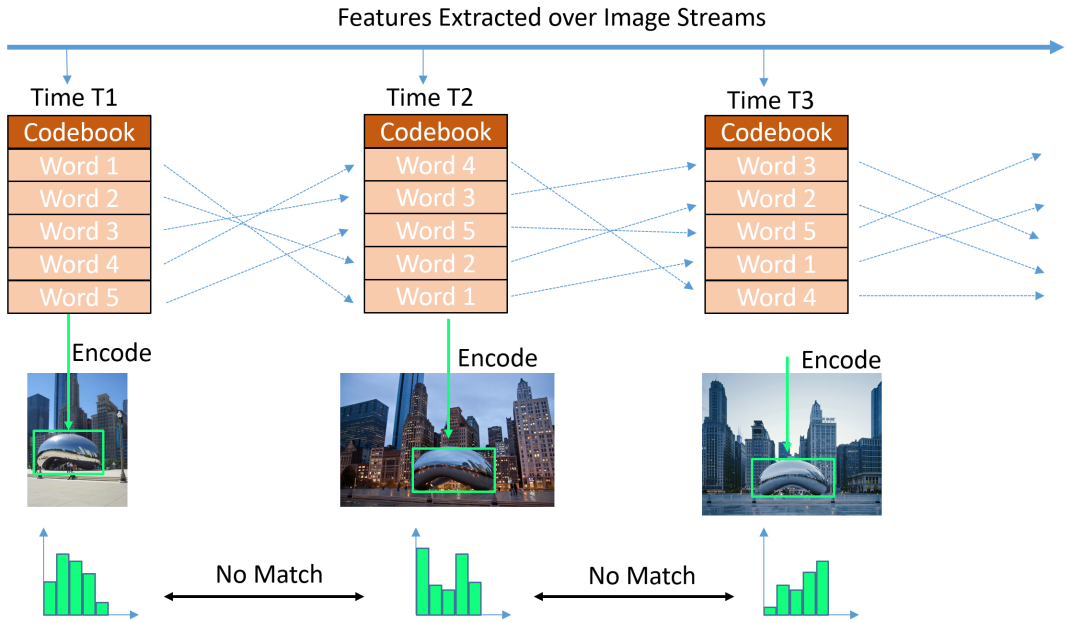

Figure 1.1. If the stream clustering is not fully progressive, the cluster centers are stored in random order in a list at each time of obtaining the latest clustering results. Such disorders disable matching previously encoded objects directly. A quadratic order matching algorithm must be performed to reorder the new center list (Codebook), which is unacceptable for dynamic scenarios. Moreover, this category of stream clustering algorithm requires pre-knowledge of the number of clusters. Therefore, it is not able to discover emerging features or objects along with the image stream.

to learn small image blocks and use these learned features to encode an image. In [16] and [22, 23], K-means and EM clustering are used to encode SIFT [18] features detected from an image, known as VLAD (Vector of Locally Aggregated Descriptors) and Fisher Vector Encoding, respectively. For object retrieval, [48] uses randomized k-d forest when matching between centers and points to boost the speed of simple K-means, and reports better results than the vocabulary tree method in [49].

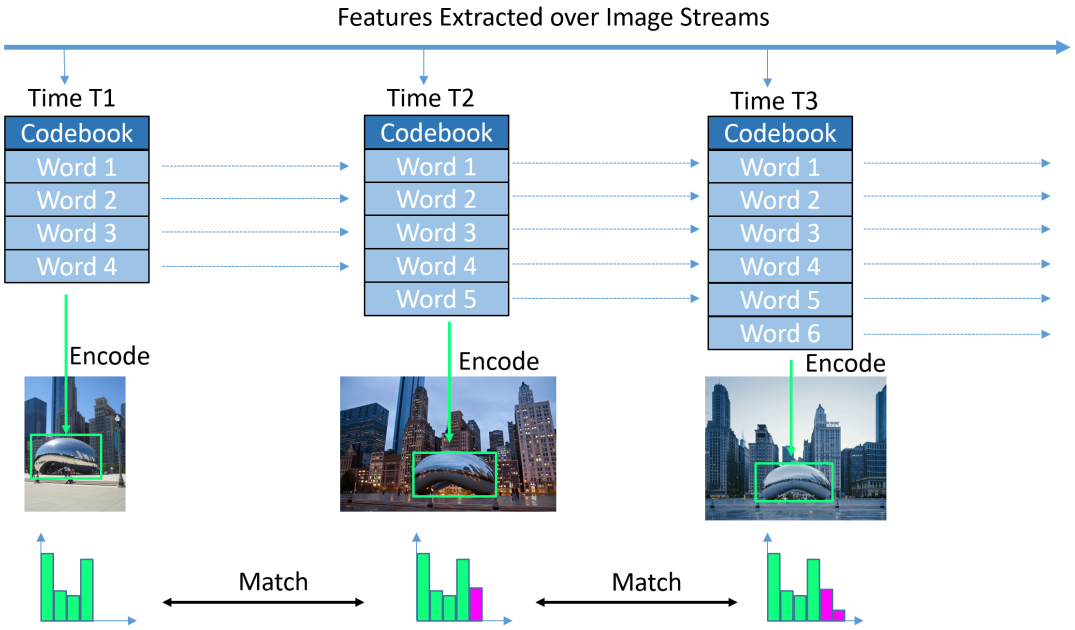

Figure 1.2. The stream clustering algorithm is fully progressive. The cluster centers are ordered chronologically and the feature space increments by adding emerging cluster centers (features) at the end of the Codebook. Thus we can still match encoded objects though the feature space is in different sizes. Note that this is one of the properties that benefit the system. There are several other properties boost the entire system in different aspects and will be discussed in the following chapters.

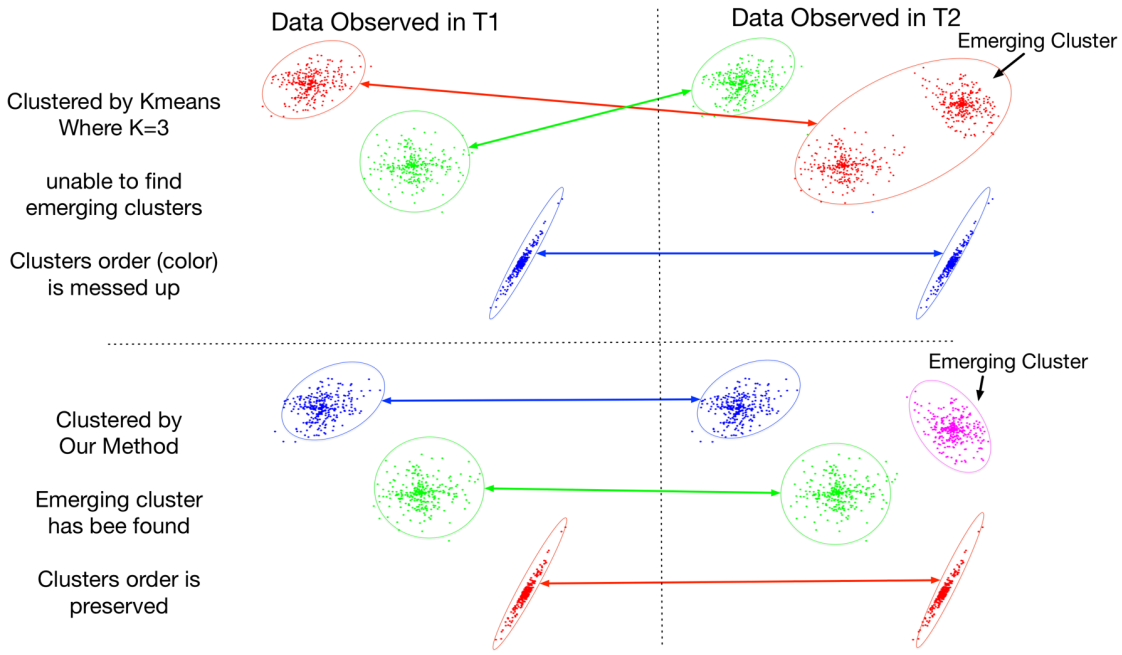

Figure 1.3. Illustration of fully progressive clustering.

1.2.2 Challenge

The stream clustering algorithms studied in this thesis are more demanding than general stream clustering algorithms. Unlike some recent state-of-the-art tackling "pseudo stream" setting where only part of the stream clustering properties is satisfied, our method satisfies the full definition of clustering data streams. For example, though claimed to be stream, [20] [21] [24] [26] do not discover the number of cluster along the stream, neither adapt to content drift. To the best of our knowledge, the clustering algorithm follows the full definition of Data Stream with provable accuracy is rarely proposed in previous works.

At the mean time, the algorithms have to carefully organize the clustering results to benefit the efficiency of the higher level of learning. As a concrete example, in the task of Unsupervised Object Discovery, we need the cluster center to be sorted chronologically. In most coreset-based stream clustering algorithms, the cluster centers are obtained by running the K-means algorithm on the coreset. As illustrated in Fig. 1.2 and 1.3, a cluster center appeared in i^{th} position of the center list in time t_1 will appear in a random position in time t_2. Therefore, a quadratic algorithm has to be performed due to the randomness of different executions of K-means over time. This will be unacceptable for our problem setting. Also, because of the unstableness of K-means, a valid cluster center in time t_1 may disappear in another execution of K-means, which makes even a quadratic complexity algorithm impossible (Fig.1.1). Moreover, the clustering center cannot be simply organized by a list where each query is in linear complexity. We need to embed the chronological property into a tree-like data structure (we propose Memory Tree in this thesis) to reduce query complexity to the logarithm.

1.2.3 Contribution

We propose a fully progressive stream clustering algorithm that satisfies the complete definition of Data Stream. This algorithm can be applied to arbitrary data types with arbitrary similarity measures. The algorithm provides several properties that are specifically designed for online unsupervised object discovery to eliminate inefficiency caused by

multi-level feature representation. The algorithm can be further developed into a hierarchical algorithm whose process is building a tree-like data structure called Memory Tree. This data structure increases the feature query, which is the most time-consuming part of Visual Encoding, from linear time to logarithmic time. We provide solid theoretical proof of accuracy guarantees in a sufficiently general setting. Our algorithm generally outperformed not only the known recent state-of-the-art stream algorithms but also some state-of-the-art batch algorithms. The systematic experiments in Visual Encoding demonstrate that our clustering framework is a very promising candidate for online unsupervised object discovery. The clustering framework is repeatedly used (with multiple variants) in the challenging vision tasks in this thesis.

1.3 Unsupervised Deep Encoding: a Bridge Between Visual Encoding and Convolutional Neural Networks

Deep Convolutional Neural Networks (CNNs) fundamentally improved the Computer Vision field since 2012 [9] and strongly diluted the attention on Visual Encoding [15–17,47] as in the latter, visual features are not trainable through supervisory signals. Though outstanding improvements have been achieved so far, CNN architectures are not fully explained in the Computer Vision domain and studying new architectures is still oriented by massive computations with trial and error. In this paper, we propose a multi-level unsupervised feature learning and encoding method (Fig.1.4), which is seamlessly embedded into CNNs and pre-trains the learnable weights of the network.

The first argument we address is the relationship between a single-layer Neural Network and Clustering. We found that when computation inside a node is a similarity measure between an input and its weight, the weight vector of that node can be regarded as a clustering center. Thus, the output vector of a single-layer Neural Network can be thought of as unnormalized soft assignments to a set of cluster centers. As shown in figure 1.4, each convolution measures the similarity between the input patch and the corresponding filter weight. Therefore the resulted output vector (blue shadowed volume in $F_i^{(l+1)}$ in

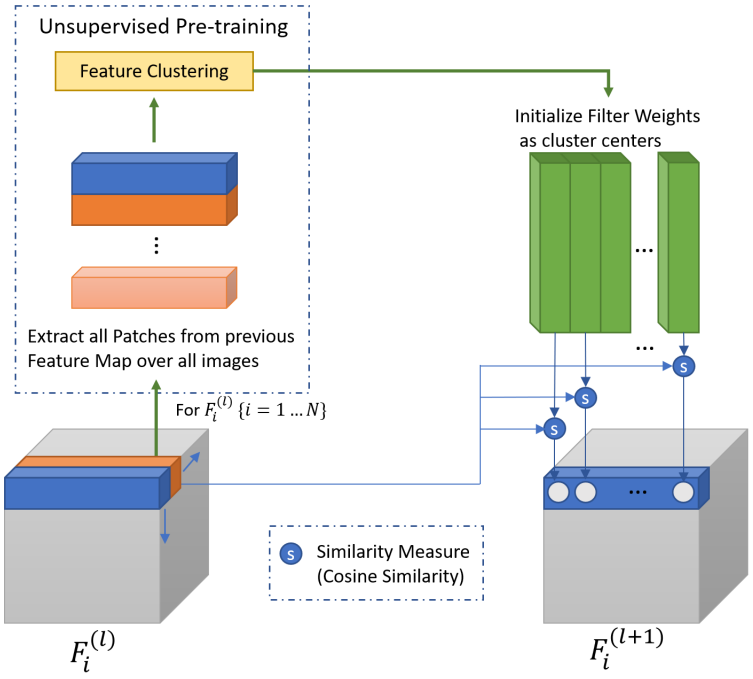

Figure 1.4. Similarity Net. with pre-training. Given $F_i^{(l)}$ as the Feature Map of i^{th} image in layer l, we extract Feature Map patches and initialize the filter weights of layer $l+1$ through feature clustering. Convolution Operation is a Similarity Measure that computes a Similarity Score between the input Feature Map Patch with the corresponding filter. This Pre-training process can be applied layer by layer to initialize the weights of all layers.

fig.1.4) is an unnormalized soft assignment. In this paper, we consider Cosine similarity instead of the dot product, similar to [50] but we provide a different interpretation. Since each convolution computes the similarity between the input patch and the corresponding filter, the output vector corresponding to an input patch convolved with multiple filters will become a vector of similarities. Such encoding can be applied to feature maps at every layer, and hence follow the general architecture of Convolutional Neural Networks.

The above mechanism inspires us to pre-train filter weights by feature clustering, where each filter abstracts a group of patches, whose visual patterns frequently appear in the previous layer. [47] found Unsupervised Feature Learning by clustering with triangle assignment outperformed single-layer Neural Networks, which indicates those unsupervisedly learned features are already good representations of visual information. In this paper, we propose to further apply the pre-training method layer by layer to learn deep features, similar to Auto-Encoder [51, 52]. Unlike traditional Visual Encoding, all the unsupervisedly learned features in our architecture can be fine-tuned by supervisory signals and trained end-to-end. We demonstrate that, when filter weights are initialized by our proposed unsupervised feature learning method, they are more related to the label information at the early stages of gradient descend.

The second argument is that the representative ability of feature maps is obscure. It has been found that well-trained convolutional filters respond to relevant visual patterns [53, 54], and such property has been regarded as "features" learned from images. However, since each filter is a linear classifier wrapped with some non-linearity, a high response only indicates that the input is far away from the classification hyper-plane and this high response value does not necessarily indicate similarity between the weight and the input vectors. In Visual Encoding, on the contrary, each feature is a kernel clustered from image patches and a high response to this feature indicates a strong similarity.

1.3.1 Related Works

Unsupervised representation learning for Neural Networks has been paid sustained effort. Most unsupervised training method aimed at forcing the network to learn a generative representation level by level. Autoencoder [51] is a landmark of modern deep learning, where the network is trained by reconstructing the input throught a hidden layer. It has been argued and generally accepted that this training fashion will force the hidden layer to learn a good representation of the original data [55] [56]. Its direct successor, Stacked Denoising Autoencoder (SDE) [52], placing a series of the encoders in a bottleneck fashion to force the bottleneck layer to learn a sufficiently compact map from the data space. Because of the nature of autoencoders, SDE can be initialized by training each layer locally, which is argued that it finds a better initial state for gradient descend [55] [56] [57]. Autoencoders can also be used for unsupervisedly learning image representation and high-level features from large image datasets [58]. Unfortunately, SDE fashion was not successfully applied to CNN-based Computer Vision models because of the architecture inconsistency between fully connected layer and convolutional layer, which limits SDE to images with low resolution. For this reason, it was not been incorporated SDE into advanced Computer Vision tasks such as object detection and semantic segmentation.

Another branch of effort in unsupervised deep learning is adversarial learning. [59] introduced adversarial learning with a gaming system between discriminator and generator. Adversarial Autoencoder (AAE) [60] considers the bottleneck layer as the generating space. The discriminator of AAE classifies the real sample from a given distribution and the mapped data. By adding ”style neurons” in the bottleneck layer, AAE can perform clustering along with representative learning, which achieved remarkably increased accuracy in several benchmarks. Unlike early works in clustering using Self Organizing Map (SOM) [61], AAE architecture learns non-linear representative features for clustering and is regarded as an advanced clustering framework. Some works take advantage of adversarial learning into Convolutional Neural Networks [62] [63] and Claimed improved performance.

The idea of Adversarial Learning also benefits unsupervised Computer Vision tasks. [63] proposed a Bidirectional architecture that can be an auxiliary feature learning for classification. [64] proposed the Stacked Generative Adversarial Nets that are trained to learn hierarchical features. [65] proposed a framework to learn high-level features unsupervisedly by enforcing a CNN solving the problem of predicting the relative position of two given patches. Similarly, [66] conducted high-level feature learning by enforcing CNN solving the problem of inpainting missed patch of images. And [67] follows a similar approach by training CNN to solve jigsaw puzzles. [68] used a large number of unlabeled videos as a training resource where tracked patches are regarded as the same class.

Another aspect of Neural Networks which has been heavily studied is the choice of pre-activation functions. Batch Normalization [69] is most commonly considered to reduce Internal Covariate Shift. Layer Normalization [70] considers normalizing the output vector into zero mean and unit variance, instead of using batch statistics. Weight Normalization [71] was introduced to take L2 normalized weight vector to improve learning dynamics. Similarly, [50] normalizes both the weight vectors and the input vectors thus the pre-activation becomes cosine similarity. [72] introduced ZCA whitening for each batch and gave concise gradient computation.

From the optimization point of view, initialization is crucial to Stochastic Gradient Descend (SGD) [73] since training a multi-layer Neural Networks is a non-convex problem. In contrast to the attention to architecture designing, fewer works pushed the boundary of this area. [52] [57] proposed initialize each layer by local training. For each layer, we can fix all previous layers and unsupervisedly train it as an autoencoder. It has been argued that this method will initialize the network close to better local optimum [74]. Unfortunately, such unsupervised pre-training is not well accepted in major CNN models. [55] proposed to initialize the weights in each layer with the same level of standard deviation, such that information passes through the entire network healthily. Resnet [10] indeed initializes the deeper net as an identity map since convolutional layers are usually initialized to a very small value. Until now, most CNN systems use [55] to initialize all filters randomly. The pre-training of CNN is rarely studied.

1.3.2 Contribution

In this paper, we establish a strong connection between a single-layer Neural Network and clustering. Such finding results in a model that unifies Visual Encoding and CNN as well as an unsupervised pre-training method that initializes the weights more related to label information at the early stages of gradient descend. We explain Visual Encoding as a special case of a single-layer CNN, and a CNN can be explained as deep Visual Encoding that can be trained end-to-end. Such property enables our networks to be trained fast without Batch Normalization, where a better final optimum with the same or less number of parameters is achieved. We conduct extensive experiments to demonstrate the effectiveness of our pre-training method.

1.4 Unsupervised Aerial Objects Discovery and Detection on-the-fly

Unmanned Aerial Vehicles (UAVs) equipped with high-resolution cameras provide a very different way of acquiring visual information and many attractive tasks have been studied and applied in recent years. UAVs are able to fly over a harsh, dangerous and hostile environment, which ideally fits tasks of military scouting, security surveillance, post-disaster rescue, and wildlife discovery. And because of the unfriendly environment, real-time response is crucial to these applications. Existing works on object detection from on-board aerial videos followed the supervised paradigms, where intensive training data and expensive annotation is necessary. Such framework limits the detection reservoir into well-known targets such as vehicles and pedestrian, and also make the problem itself trivial. The more valuable and challenging task on aerial videos is to discover suspicious, valuable and most likely unknown targets, e.g. arsenals of enemies, illegal drag factories, strongholds of terrorists, wreckage or survivors of an air crash, endangered species on East Africa Savanna, as well as evidence of life from low-orbit satellite images of alien planets etc. Meanwhile, we expect these targets can be discovered immediately only based on on-board computational resources since their values drastically diminish along with time passing. Collecting sufficient training data for these targets is extremely expensive, not

only for annotating cost but also due to the unfriendly environment and the rareness of the targets themselves, which makes even unlabeled images precious. At the same time, those targets are very likely to be intentionally camouflaged, making themselves quite different from those in training data.

Motivated by the above real-world tasks and many other similar missions, we study the problem of real-time, fully unsupervised visual knowledge building from aerial videos. This problem requires conducting unsupervised learning from feature level to object-level simultaneously when UAVs are flying over the targeted territory. We consider each object as a distinct instance and do not attempt to infer categorical information, namely we do not regard arsenals and factories as a category of ”building” (which naturally fit the real-world applications mentioned at the beginning). Since the system is supposed to learn visual knowledge within a dedicated area, object diversity is not as intensive as general vision applications. The major challenges in this problem are (1) there is absolutely no pre-training at all since we never know the environment before deploying and (2) learning is on-the-fly since the immediate reaction is necessary, especially when capturing criminals and rescuing survivors.

Unsupervised object discovery arises sustained effort on fully unsupervised approaches. Some works directly discover distinct and meaningful patches from a static image or video dataset. This category of approaches is not demanding in computational resources but usually does not provide a deep and non-linear map tackling visual diversity. Namely, matched patches are only similar to each other concerning appearance instead of semantic meaning. Quite a few others learn a deep map to extract high-level features with semantic meaning and perform weakly supervised or unsupervised object discovery based on the trained model. However, all these two categories of approaches can not be borrowed to the problem we study in this paper because (1) all of the existing works are iterative approaches and require the entire dataset ready before learning starts; (2) all of the existing works are computation expensive for real-time response in harsh deploying environment; and (3) no existing work addresses on aerial images or videos.

As shown in fig.4.1, we proposed a fully unsupervised, stream and real-time visual discovery system that discover knowledge from feature-level to object-level simultaneously. The system keeps discovering and accumulating emerging features, parts and objects in the video. Like [68], we treat tracked objects in the video as rich resources for visual diversity: robustly tracked objects should have the same semantic label. However, we do not use massive video for pre-training.

Among previous works on unsupervised object discovery, general object proposal methods provide increased efficiency. Many successful object proposal methods have been developed and evaluated on non-aerial images, such as the PASCAL VOC dataset. Objects in aerial images are quite different from ordinary images (indoor or outdoor images). First, objects in aerial images are sparse, i.e. most areas of an aerial image are background (grass, trees or other textures), especially when UAVs are flying over the wilderness or ocean. Also, objects are usually very small and in low resolution, which made the assumption invalid of several unsupervised object proposal methods. Moreover, object proposal is inefficient for real-time object discovery. Though extremely fast object proposals do exists, it generates two much negative bounding boxes, which makes encoding and recognition taking much longer time. Due to the above applications of aerial videos, objects usually posses more corner-like features than the background. According to this intuition, we propose a simple, fast but surprisingly accurate object proposal method for aerial images, and drastically benefits our on-line learning framework.

It worth noting that, previous approaches to unsupervised object discovery are in a ”passive paradigm”: the learning system takes the input images/videos and process them in a static pre-engineered pipeline, which is not able to determine by itself whether the discovery task is complete or not without external evaluation. Such capacity is crucial for aerial applications such as autonomous scouting, where the UAVs have to use ”active feedback” from the visual system to make decisions of the flying routes, e.g. how long should an agent rover around to ensure a specific territory is completely discovered and understood. In our approach proposed in this paper, with certain realistic and controllable

assumptions, we provide provable guarantees for learning quality that can be expressed by the system itself, without any external evaluation.

Clustering is the fundamental theory of unsupervised learning. Recent works on clustering data streams provide strong theoretical guarantees approximating popular algorithms. However, the majority of these works have not successfully addressed the fully progressive problem, where emerging clusters are accumulatively discovered over time and listed chronologically by the time of emerging. We propose a clustering algorithm for data streams addressing the above problem and provide strong provable guarantees.

1.4.1 Related Works

The majority of existing works on object detection from aerial videos address the problem of supervised ground targets detection, tracking and off-line action/event recognition. [75] proposed an efficient method detecting moving targets by background stabilization and classify detected targets into vehicle and person. Similar works also studied the problem of moving vehicle detection from aerial videos in low frame rate [76] and aerial videos in urban areas [77], as well as further development of detection and segmentation [78] for moving targets. [79] proposed Track Before Detection (TBD) method efficiently detects and tracks moving vehicles. For more complex surveillance tasks, [80] proposed an approach to persistent tracking under a strongly occluded environment. [81] addressed the problem of event recognition from aerial videos. All the above existing works on aerial follow the supervised framework and the detected object classes are restricted to known moving targets such as vehicles and people. Real-time detection for unknown multi-class targets from on-board aerial images caught rare attention in the literature.

Fully unsupervised learning for visual knowledge building arises an effort recently due to expensive annotation costs. One branch of works construct visual knowledge for unknown object detection directly. [82] mines distinct mid-level patches from a collection of ”discovery set” with help of ”natural set”. By switching the roles of discovery set and natural set, the proposed approach successfully find patches are both representative and dis-

criminative. [83] fit the object detection and localization into Multiple Instances Learning Problem to find salient patches across images. [28] uses Hough Matching to find frequently matched image regions, where object proposals that are frequently matched to other patches are voted for localizing as foreground. [84] combines Hough Matching and tracking to detect and localize objects from videos. [85] proposed an appearance-based clustering method for unsupervised object detection for autonomous driving. All the above works do not involve massive training processes nor the large scale of training images or videos. Most of the training resides inside the moderate size of the training set.

Another branch of effort on unsupervised visual knowledge learning arises in the Deep Learning domain with a focus on learning high-level visual representation in an unsupervised fashion. Generative Adversarial Nets [86] and its successors [63] [64] conduct a paradigm for learning highly representative features that is able to generate artificial data to "fool" a classifier. [63] propose a Bidirectional architecture that can be an auxiliary feature learning for classification. [64] propose the Stacked Generative Adversarial Nets that are trained to learn hierarchical features. [65] propose a framework to learn high-level features unsupervisedly by enforcing a CNN solving the problem of predicting the relative position of two given patches. Similarly, [66] conducted high-level feature learning by enforcing CNN solving the problem of inpainting missed patch of images. And [67] follows a similar approach by training CNN to solve jigsaw puzzles. [68] use a large number of unlabeled videos as a training resource where tracked patches are regarded as the same class.

All the above approaches either involve massive training (taking days or weeks running on most advanced GPU) or massive training samples, or both. However, neither the above two situations satisfy the problem we solve, where on-board computation is very limited and immediate learning is crucial. Further, all the works above study and are evaluated by ordinary images or videos.

Recent object proposal methods provide a efficient tool for speeding up object localization [87] [88] [89] [90] [91] [92] [93] [94] [95]. [87] proposed an unsupervised object proposal method by investigating complete contours inside each sliding window. [95] uses the gradient norm to localize object boundaries and results in a very fast object proposal

method. All the above existing methods have been designed for and tested by ground images such as PASCAL VOC [96]. Bounding boxes usually intensively overlap with each other and a piece of the region may be proposed quite a few times. Thus, final localization from proposed boxes must involve strong computational burden [83] [28] [84]. The general object proposal for aerial images is rarely studied. [79] proposed a track-before-detection method for fast localizing ground moving vehicles, but unable to propose static, interesting objects such as buildings and factories. In this paper, we propose an efficient object proposal method and object realization approach for aerial images to propose, recognize, discover and localize objects simultaneously on-the-fly.

1.4.2 Challenge

There are two major challenges in this problem (1) Unsupervised Object Discovery and (2) reliable real-time performance. Each of the above tasks itself is already a general challenge in Computer Vision and Machine Learning. Concurrently tackling these two challenges drastically increases the difficulty to a higher level of challenge, which is rarely studied in the literature.

Unlike unsupervised learning in general Machine Learning that classifies collected samples without training labels, the problem studied in this thesis requires accurate localization with a bounding box indicating the position and area of the discovered object. Moreover, our problem is more challenging to general unsupervised object detection: there is no training data before deployment. Vision usually requires multi-level feature learning to represent the image such that objects can be classified or clustered in the feature space. Most training computation works on learning feature representations. Our system requires not only efficient online feature learning but also stably matching objects while feature representation drifts along with emerging objects.

Since we need all the computation performed on-board to adapt to extreme situations, we need sufficiently efficient algorithms and data structures to optimize the real-time performance, in both time and space aspect. For instance, unsupervised feature learning can

not only be a stream clustering algorithm but should provide a series of desirable properties including intermediate data structures to speed up object encoding. Object proposal has no computational resources to reach high recalls but relies on object realization to patch up the whole piece of objects. Namely, it has to be much more exquisite than a pipeline-fashion solution, and each level of the system has to be woven together to eliminate the last bit of inefficiency.

1.4.3 Contribution

We propose a fully unsupervised visual system that is able to learn different levels of visual information accumulatively in real-time for aerial applications. We provide an efficient object/part proposal method for aerial images and a bottom-up approach to discover part-object relationships. We propose a fully progressive clustering framework for data streams with strong provable guarantees to support our real-time learning. We conduct experiments of our system on the UCF action dataset, UCLA Aerial Action Dataset and Okutama Action Dataset with the most related works on unsupervised object discovery. As a result, we obtained sufficient purity and outperformed other compared approaches in coverage. To the best of our knowledge, this is the first study focus on real-time unsupervised visual knowledge discovery on aerial videos.

2. FULLY STREAM CLUSTERING FOR UNSUPERVISED FEATURE LEARNING AND OBJECT DISCOVERY IN DYNAMIC ENVIRONMENTS

Clustering is usually regarded as a paradigm of unsupervised learning. As discussed previously, stream clustering is the major problem setting we encounter in Dynamic Environments. In this chapter, we present a series of fully incremental clustering algorithms for Data Streams that will be repeatedly used for the rest of this thesis. The algorithm is able to detect emerging clusters along the Data Stream by dynamically locating kernels on the most promising area and perform a Stochastic Mean Shift to find clustering centers in a single pass. We develop a density estimation method for dynamic initialization by regarding a sub-stream following an ”emerging data” as a sample set and applying Hypothesis Testing (p-value Approach) to estimate its local density. The sub-stream size and the p-value is determined such that a strong accuracy guarantee can be proved. All the above theoretical analysis is concisely and compactly designed into a single pass clustering algorithm for Data Streams. We validate our performance by compare state-of-art clustering algorithms on realistic and complex datasets. As a result, our method is not only able to drastically outperformed state-of-art stream algorithms (major jump in most datasets) but also outperformed more complex (non-stream) algorithms that many stream algorithms attempt to approximate.

As we can see later, this stream clustering algorithm uniquely fits vision problems studied in this thesis and is crucial to solving vision problems in Dynamic Environments. This method, and several of its variants will be repeatedly used by almost all the following sections, include fast visual feature encoding, aerial object proposal, discovery, matching and realizing objects on-the-fly, and unsupervised multi-stage feature learning for Convolutional Neural Networks.

In this chapter, we tackle the Density Peak Clustering formulation: given a neighborhood size θ and a minimum density ratio $0 < p_f < 0.5$, find all local density peaks $C = \{c_1, c_2, ..., c_k\}$ such that, $p^{\theta}_{c_i} = \frac{N_i}{N} \geq p_f$ for all $i = 1, ..., k$, where $p^{\theta}_{c_i}$ is the density ratio of θ-neighbourhood around c_i, where N_i is the number of data points covered by θ-neighbourhood around c_i and N is the data size. We consider the above problem formulation for Data Streams because (a) the number of clusters is probably unknown for Data Streams and new clusters are likely to emerge and (b) a density peak represents its neighborhood naturally. In our method, neighborhood θ and density threshold p_f are the only two user-specified parameters, and no other prior information is required.

We further decomposite the above problem into two sub-problems:

- If we found a "promising" location for each cluster, how can we detect the correct clustering center (closest valid density peak) with provable guarantees.
- How can we detect at least one "promising" location for every cluster with provable guarantees.

Obviously, the above two sub-problems have to be solved within a single pass of the stream. The detailed solution is presented in the following sections of this chapter.

2.1 Mean Shift in Single Pass

Following the Density Peaks [31] and Mean-shift [35] paradigms, we assume that each clustering center is located at the density peak of a local neighborhood. In every step, Mean-shift computes the mean of data inside a kernel and moves the kernel to the mean location. If we take a random sample from data inside the kernel, its probabilistic expectation happen to be the mean location, namely $\sum_i p(i)x_i = \frac{1}{N}\sum_i x_i$, where $p(i)$ is the probability that x_i is chosen. This property allows us to seek local density peaks within a single pass.

Consider a toy dataset, in Fig. 2.1, generated by a Gaussian distribution. Suppose we initially place a kernel with bandwidth θ at Point A. We randomly sample a data point $\boldsymbol{x_i}$ from this dataset (without replacement), and compute its distance to the center. If it is smaller than θ we call it a match, and we update the center as,

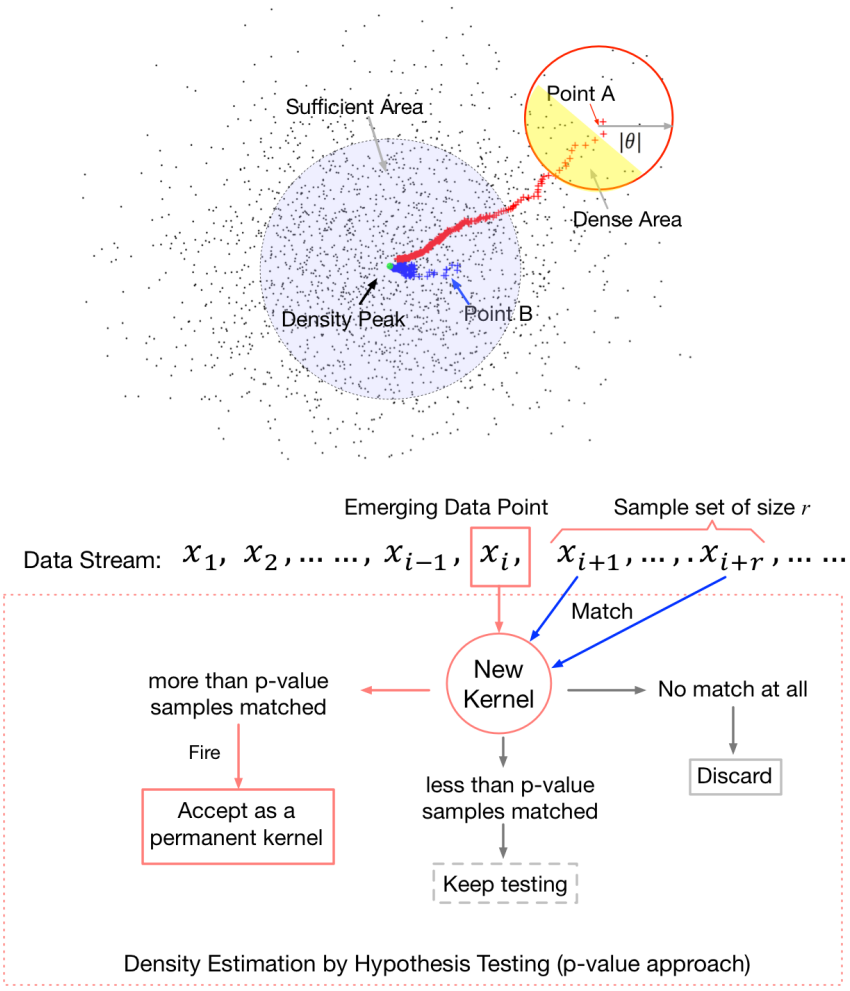

Figure 2.1. Updating a center: two candidate cluster initializations, at Point A and Point B. The paths shown in red and blue colored cross points indicate convergence towards the density peak of the natural cluster during the 'sampling, matching, and updating' process. The denser region (in yellow) inside the θ-defined kernel denotes where the next matched sample is more likely to be. The blue shadow shows a "sufficient area" where any kernel inside this area is sufficiently dense. The entire process requires a single pass through the dataset, with no iterative operation involved.

$$\boldsymbol{c_k}^{(new)} = \frac{n_k \boldsymbol{c}_k + \boldsymbol{x}_i}{n_k + 1} \tag{2.1}$$

where $\boldsymbol{c_k}$ is the center of the kernel and n_k is the number of samples it has been matched with so far. Note that inside the kernel, the matched sample is more likely to come from a dense dataset region (yellow-highlighted in Fig. 2.1). Therefore, each update will problematically move the center towards a denser location. If we apply this "sampling, matching, and updating" process throughout the entire dataset, the center will keep moving towards the nearest density peak, as shown with the red cross path in Fig.2.1. The resulting path is quite similar to the Mean-shift procedure, however, the entire process here is stochastic and merely requires a single pass.

Note that n_k is free to be reset into a small value if the kernel stopped moving since the kernel is always moving towards the density peak. However, this case is unlikely to happen given an appropriate similarity value θ or an initial point close to the density peak (point B).

2.2 Dynamic Initialization: Density Estimation

To initialize the kernels, with respect to their number (*how many?*) and locations (*where?*), our objective is, for any density peak, find at least one kernel that can be shifted to it. We can achieve this objective if we manage to estimate the local density around an emerging data point and initialize a new kernel there if it is dense, where an emerging data point is a data point cannot fall into any existing kernel. Recall the Density Peak problem setting, we define a kernel as ***sufficient*** if its density ratio is greater than p_f, and ***insufficient*** otherwise.

It is worth noting that, for any valid density peak, there exists a sufficient area such that any kernel inside this area is sufficient (as illustrated by the blue circle in Fig.2.1). Therefore, as long as we correctly estimated the density of at least one emerging data from the sufficient area, it will be guaranteed to find the density peak via Stochastic Mean Shift. And even if the kernel is outside the sufficient area (such as point A), we are still able to

robustly find the nearest density peak. Therefore, our dynamic kernel generation principle is:

(i) every data point that cannot be matched to any current kernel is temporarily set as a location of a new kernel;

(ii) for every new kernel, we perform an efficient density test (estimate its density by p-value approach), and discard it as noise if it cannot pass the test; and

(iii) we use the kernels that pass the density test to seek local density peaks.

We regard each data point on the stream as a random sample from the entire dataset. Given an emerging data, its subsequent r data points construct a sample set with size r, which can be used for density estimation following p-value approach. Specifically, for a statistical confidence α (in this thesis $\alpha = 1\%$) the following two conditions must be satisfied:

Condition A:

If a kernel is known to be sufficient, we have $1 - \alpha$ confidence that it will not fail density test.

Condition B:

If a kernel passes the density test, we have $1 - \alpha$ confidence that it is indeed sufficient.

Proposition 1 (Satisfying Condition A): For any r random samples from the dataset, with

$$r = \frac{ln\alpha}{ln(1 - p_f)}, \tag{2.2}$$

the probability that there are no data points covered by a sufficient kernel k is less than α.

Proof: The probability that a single sample is not covered by kernel k is $1 - p_k$; therefore, the probability that none of the r samples are covered by the kernel will be,

$$\begin{aligned}(1 - p_k)^r = (1 - p_k)^{\frac{\ln \alpha}{\ln(1-p_f)}} &= (1 - p_k)^{\log_{1-p_f} \alpha} \\ &\leq (1 - p_f)^{\log_{1-p_f} \alpha} = \alpha\end{aligned} \tag{2.3}$$

□

Proposition 2 (Satisfying Condition B): Consider any r samples from the entire dataset and a sufficient kernel k. Let the stochastic variable X denote the number of data points that fall into kernel k among these r samples. Let

$$X_f = rp_f + \Phi^{-1}(\alpha)\sqrt{rp_f\Big(1-p_f\Big)}, \tag{2.4}$$

we have

$$P\{X > X_f\} \geq 1-\alpha, \tag{2.5}$$

where $\Phi^{-1}(.)$ is the inverse Cumulative Density Function of Standard Normal distribution. Namely, we have $1-\alpha$ confidence that if the kernel passes the density test, it is indeed sufficient.

Proof: X follows Binomial Distribution $X \sim B(r, p_k)$. According to the Central Limit Theorem, when r is large and $rp_k(1-p_k) \geq 10$, we can approximate binomial distribution $B(r, p_k)$ with normal distribution $\mathcal{N}(rp_k, \sqrt{rp_k(1-p_k)})$. If we consider the statistical variable, $Y = \frac{X-rp_k}{\sqrt{rp_k(1-p_k)}} \sim \mathcal{N}(0,1)$,

$$\begin{aligned} P\{X > X_f\} &\geq P\left\{\frac{X-rp_k}{\sqrt{rp_k(1-p_k)}} > \frac{X_f-rp_k}{\sqrt{rp_k(1-p_k)}}\right\} \\ &\geq P\left\{Y > \frac{X_f-rp_f}{\sqrt{rp_f(1-p_f)}}\right\} \\ &= P\{Y > \Phi^{-1}(\alpha)\} = 1-\alpha, \end{aligned} \tag{2.6}$$

given $p_k \geq p_f$, and $p_f \leq 0.5$ (more than one density peak). □

As shown in Fig.2.1, the sufficient area covers a large number of data points and we are able to detect the density peak of only at least one emerging data from this area passes density test. Suppose we totally test L emerging data points from a sufficient area. According to Proposition 1, the probability of a single failure is $\alpha = 1\%$. Then failure of all L tests will be α^L, which will be a extremely small probability of failure to detecting a density peak. And according to Proposition 2, if it passes, we have $1-\alpha$ confidence that it is indeed from sufficient area.

Correctness Guarantees. Our algorithm takes two assumptions:

(1) Monotonic Density Assumption: probabilistic density function that generates the data

is monotonically increasing when approaching a valid nearest density peak inside the corresponding sufficient area.
(2) Separation Assumption: the nearest distance between any two valid density peaks are larger than kernel size θ.

Satisfying the above two assumptions guarantees that a valid density peak will be discovered as long as we initial a kernel inside its sufficient area. And Proposition 1 and Proposition 2 provide guarantees discovering sufficient area for every density peak. Therefore, our algorithm optimally solved the clustering objective in this paper for datasets that satisfy Monotonic Density Assumption and Separation Assumption.

2.3 The Single-Pass Algorithm

Algorithm 1 describes our single-pass clustering algorithm. As shown in fig.2.2 We introduce two data structures, *Dictionary* that stores permanently kernels, and *Memory* where every newly emerged kernel is examined according to a density test and is transferred to Dictionary if it passes, or is discarded as noise if it fails. Both Dictionary and Memory are empty at the beginning and then enriched while parsing the data.

Proposition 1 and 2 can be concisely implemented by maintaining a variable *activity* of every kernel in Memory. Activity is set to r when a new kernel is created and decrease by 1 after processing each data point on the stream. When a Memory kernel is matched, its activity value is increased by r; when this value exceeds a threshold ϕ, the corresponding kernel is transferred to the Dictionary as a permanent kernel. Therefore, a new Memory kernel encounters at least r data point after initialized or matched, which forms a sample set of size r. A Memory kernel is removed if the situation in Proposition 1 happens. To follow Proposition 2, the upper threshold ϕ is

$$\phi = rX_f, \tag{2.7}$$

The other aspect of the activity is that sufficient kernel must be matched frequently along the Data Stream while insufficient kernel must be matched rarely because of their

Algorithm 1: Stochastic_Mean_Shift(D, θ, p_f)

```
1  Dictionary, Memory ← ∅, r←eq.(2.2), φ ← eq.(2.7)
2  for each x ∈ D do
3      k ← nearest kernel to x in Dictionary
4      if similarity (x, k) ≥ θ then
5          update Dictionary kernel k by eq.(2.1)
6      else
7          k ← nearest kernel to x in Memory
8          if similarity (x, k) ≥ θ then
9              update Memory kernel k by eq.(2.1)
10             k.activity = k.activity + r
11             if k.activity ≥ φ then
12                 trasfer k to Dictionary
13         else
14             initial a kernel k_new at x
15             k_new.activity = r
16             Memory.add(k_new)
17     Decrease activity of each kernel in Memory by 1
18     Remove kernels with negative activity
19 return Dictioanry
```

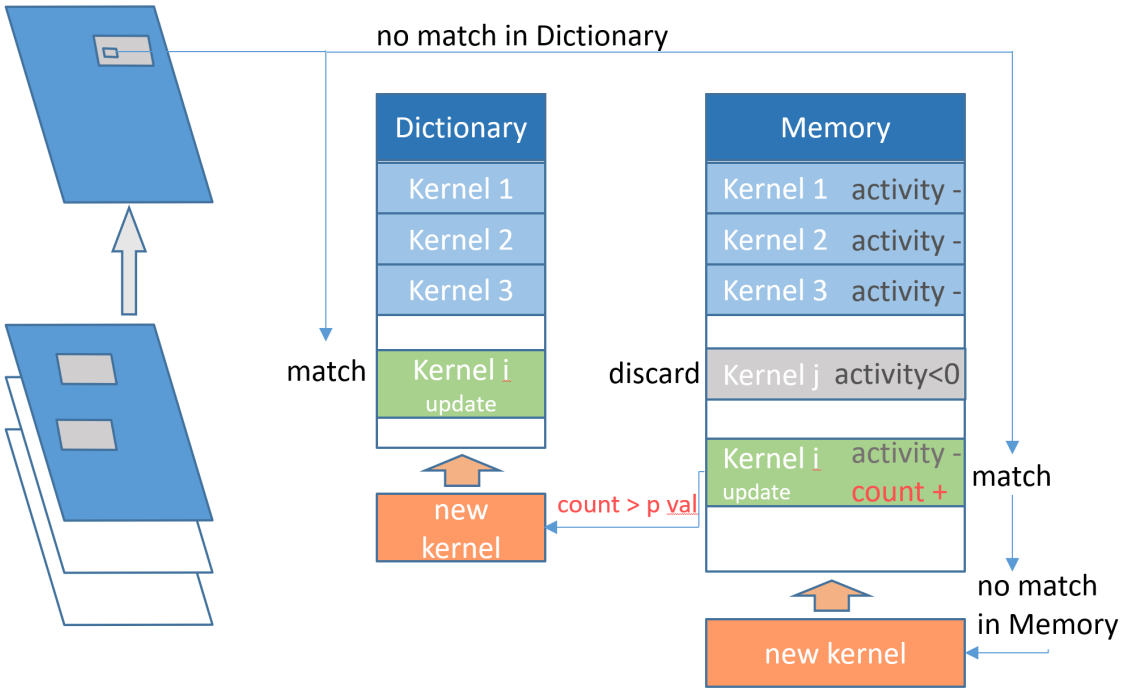

Figure 2.2. Illustration of the single-pass algorithm. Data fed into the algorithm is arbitrary (small image patch from the video stream in this case.), as long as match and update are well defined.

density. In the previous section, we assumed each data point is independent of each other. Indecency assumption assumes each cluster is evenly distributed along the stream, which makes the densest r sub-stream most sparse. It can be easily proved that, if the data is distributed unevenly, there must exist a r sub-stream that is denser than average density. Therefore, our assumption is indeed tackling the worst case, which makes Proposition 1 and 2 stand in general Data Streams.

2.3.1 Subset Size for Successful Clustering

Consider a sub-stream of size N^* in the Data Stream formed by the first N^* data points. We hope that by processing only N^* samples, our algorithm will find a sufficient kernel for every valid density peak so that the Dictionary will have enough subsequent data to update itself to the density peaks.

Given a sufficient kernel k in a dense area covers N_k data points, with $\frac{N_k}{N} = p_k > p_f$. Let N_k^* be the number of instances, among the first N^* random samples, covered by kernel k. Then N_k^* follows Binomial Distribution $N_k^* \sim B(N^*, p_k)$. Using again the Central Limit Theorem, let $M = (N_k^* - N^* p_k)/\sqrt{N^* p_k q_k}$, where $q_k = 1 - p_k$ and M follows Standard Normal distribution, $M \sim \mathcal{N}(0, 1)$. We hope that in the first N^* random samples, the instances covered by kernel k are more than 0.9 times of the expectation, $N_k^* > 0.9 N^* p_k$; the probability is,

$$\begin{aligned} & P\left\{N_k^* > 0.9 N^* p_k\right\} \\ & = P\left\{\frac{N_k^* - N^* p_k}{\sqrt{N^* p_k q_k}} > \frac{0.9 N^* p_k - N^* p_k}{\sqrt{N^* p_k q_k}}\right\} \\ & = P\left\{M > -\frac{0.1 N^* p_k}{\sqrt{N^* p_k q_k}}\right\} = \Phi\left(\frac{0.1 N^* p_k}{\sqrt{N^* p_k q_k}}\right), \end{aligned} \tag{2.8}$$

We expect $P\left\{N_k^* > 0.9 N^* p_k\right\} > 1 - \alpha$; if $\alpha = 0.01$, then from eq. (4.15), we have[1],

$$N^* > 497.3 \frac{1 - p_k}{p_k} > 497.3 \frac{1 - p_f}{p_f} \tag{2.9}$$

This equation indicates that our method does not prefer small datasets, especially when the

[1] If $\Phi(x) = .99$ then $x = 2.23$

cluster population is very small. However, simple post-processing, such as data duplication or interpolation, can resolve this issue.

2.3.2 Performance for Stream Data

Our method inherently carries the idea of sequential processing: it parses the dataset once, and each data point requires constant computation. However, we shuffled the data sequence in order to distribute noise evenly among the considered subsets of size r and avoid noise accumulation. However, in stream mode, we are not able to shuffle the data sequence at the beginning or sample from the entire dataset along with parsing. In this case, we show another conditions:

Condition C: For any real cluster, there exists a "dense" subsequence so that we will miss it with low probability.

Proposition 3 (Satisfying Condition C): For any permutation of a binary set $\{\nu_i \mid \nu_i = 0 \; or \; \nu_i = 1\}_N$ and any positive integer $r \leq N$, there exists at least one consecutive subsequence $\boldsymbol{\nu}$ of length r, such that,

$$\frac{\parallel \{\nu_i \mid \nu_i = 1, \nu_i \in \boldsymbol{\nu}\} \parallel}{r} \geq \frac{\sum_{n=1}^{N} \nu_n}{N}, \tag{2.10}$$

where $\boldsymbol{\nu}$ can include tail-head permutations, i.e., $[\nu_i, \nu_{i+1}, \ldots, \nu_N, \nu_1, \nu_2, \ldots]$. (Note: This condition means that there is at least one consecutive subsequence of length r where the density of the cluster members is greater than or equal to the average density of the cluster members in the entire dataset.)

Proof. We prove this lemma by contradiction. Assume for any r-length consecutive subsequence of an arbitrary permutation $[\nu_1, \ldots, \nu_N]$,

$$\sum_{j=i}^{i+r-1} \nu_{mod(j,N)+1} < r\frac{\sum_{n=1}^{N} \nu_n}{N}, \;\; \forall i = 0, 1, \ldots, N-1 \tag{2.11}$$

We consider the modulo index $mod(j, N) + 1$ to account for tail-head permutations (see above). In total, there are N distinct consecutive subsequences. Then,

$$\sum_{i=0}^{N-1} \left(\sum_{j=i}^{i+r-1} \nu_{mod(j,N)+1} \right) = r \sum_{n=1}^{N} \nu_n, \tag{2.12}$$

i.e., each element in the dataset is added r times (consider an r-length window 'sliding' N times along the dataset/sequence). However, according to the assumption in eq. (2.11), we have,

$$\sum_{i=0}^{N-1}\left(\sum_{j=i}^{i+r-1}\nu_{mod(j,N)+1}\right) < \sum_{i=0}^{N-1}\left(r\frac{\sum_{n=1}^{N}\nu_n}{N}\right) = r\sum_{n=1}^{N}\nu_n, \tag{2.13}$$

which contradicts eq. (2.12). □

Proposition 3 shows that for a cluster who can support at least one sufficient kerel k, $p_k \geq p_f$, no matter what the data order is, there exists at least one consecutive subsequence of features $\mathbf{x}$, $\parallel \mathbf{x} \parallel = r$, such that,

$$\frac{\parallel \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \boldsymbol{\pi}_k, \boldsymbol{x}_i \sim k, \boldsymbol{x}_i \in \mathbf{x}\} \parallel}{r} \geq p_f \tag{2.14}$$

where $\boldsymbol{x}_i \sim k$ means $\boldsymbol{x}_i$ is covered by kernel k. According to Proposition 1 and 2, we have a probability of α to miss it. And in realistic situation, we have several more of such subsequence, the failure probability is exponentially lower.

A potential issue of this method is the order of data. If a few noises appeared adjacently along the sequence, they might be treated as a real cluster by mistake (though sometimes this is preferred). Therefore, in batch mode, we randomly shuffle the order to let every data instance appears evenly. As a result, every data point on the shuffled list will be a random sample from the entire dataset. In the stream mode, we put the earliest observations into a data pool. For each of the following observation, we first insert it into the data pool, and then randomly sample data from the data pool as the data instance in the current step. Alternatively, we can leave those noise in Dictionary first, then simply discard the entries with a small cluster population in the end.

2.4 The Memory Tree Clustering: Stochastic Mean Shift in Hierarchy

From eq.(2.2) it is obvious that r has high values when p_f becomes very small. This happens when we expect a large number of clusters (of smaller sizes), e.g., when building a local feature dictionary for a large and diverse image collection.

To account for such scenarios, we use a hierarchy as follows. At the first level, we choose a smaller distance threshold θ to generate a small number of large clusters. At the

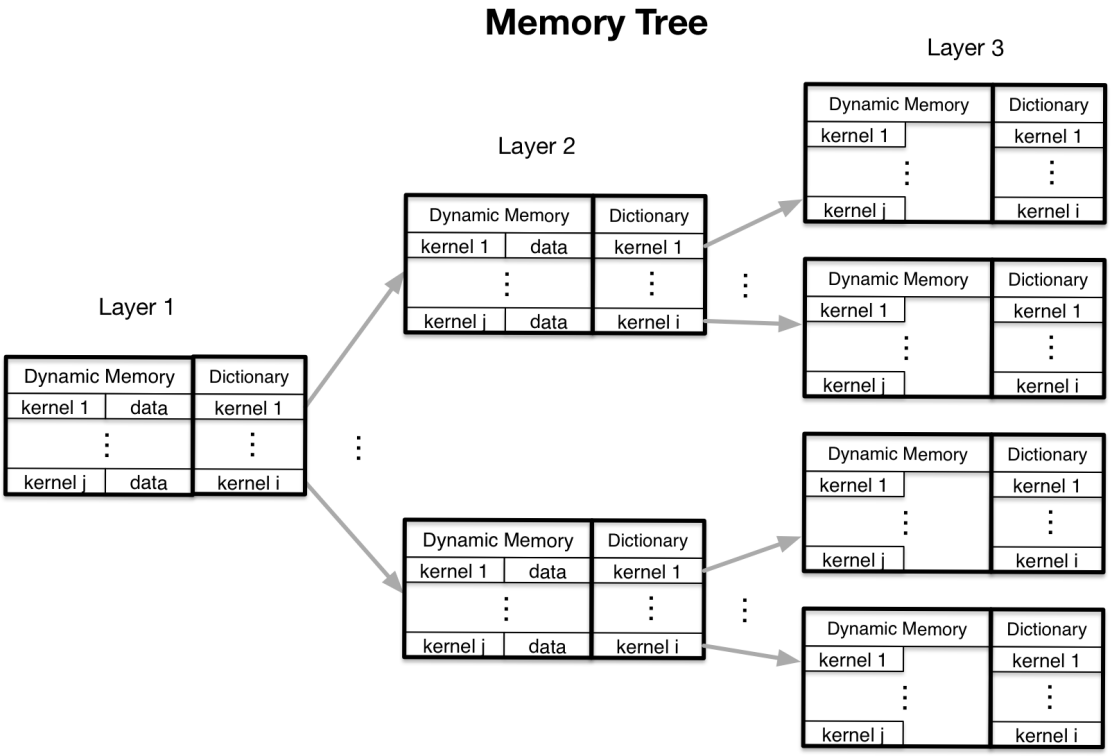

Figure 2.3. Memory Tree Clustering

Algorithm 2: Memory_Tree_clustering(d, root)

```
if root.depth = maxDepth then
    perform basic algorithm in root;
    Return;
index = match(d, root.Dictionary, root.θ);
if index > 0 then
    update root.Dictionary[index] according to eq. (2.1);
    root.Dictionary[index].number++;
    Memory_Tree_clustering(d, root.children[index]);
else
    index = match(d, root.Memory, root.θ);
    if index > 0 then
        update root.Memory[index] according to eq. (2.1);
        root.Memory[index].number++;
        root.Memory[index].ActiveValue+ = r;
        root.Memory[index].bucket.add(d);
        if Memory[index].ActiveValue > φ then
            root.Dictionary.add(Memory[index]);
            root.Memory.remove(index);
            root.children.append(Memory[index]);
            for p ∈ root.Memory.bucket do
                Memory_Tree_clustering(p, root.children[end]);
    else
        root.Memory.add(d);
        root.Memory.ActiveValue = r;
        root.Memory.bucket = d;
for j = 1 to root.Memory.size do
    root.Memory[j].ActiveValue − −;
    if root.Memory[j].ActiveValue < then
        root.Memory.remove(j);
```

next level, for each of these clusters, we increase the value of θ and generate smaller sub-clusters. We repeat this procedure until we reach a pre-defined maximum of the recursion depth. Our experiments show that practically three recursive steps are sufficient; we set $\theta[depth = 1]$, $\theta[depth = 3]$ for the first and last depth, and calculate $\theta[depth = 2]$ as geometric average $\sqrt{\theta[depth = 1]\theta[depth = 3]}$.

However, the method presented above is a batch method, it passes the dataset 3 times, though it is faster by pre-pruning unsuccessful matches. For stream scenario, we propose a data structure, Memory Tree, to perform Data Stream clustering. Fig. 2.4 shows a 3-layer Memory Tree. Every node of Memory Tree is a unit running the basic algorithm, it maintains a Memory and a Dictionary. Each entry in Dictionary links to a child of the current node. Namely, every non-root node is a kernel and the Memory Tree grows by running the basic algorithm in appropriate node.

To feed a data point into the Memory Tree, we start from the root and first try to match it to an existing kernel in Dictionary. If we found the match, we pass the data instance to the corresponding child to perform the same recursive operation. If no such match can be found, we will insert it into Memory in the current node as the basic algorithm. In Memory Tree, each Memory entry as a bucket stores data assigned to the corresponding kernel. And if a kernel in Memory passed Density Test and is transferred to Dictionary, all the data stored in Memory entry will be passed down recursively by running the basic algorithm on a deeper level. However, we do not store data in Memory entry in the leaf node, because it is unnecessary. The recursion will terminate if the dataset goes to Memory or reach the deepest layer.

One obvious concern is the bucket size. It is possible, though not likely, that many buckets could grow to a large size such that the main memory cannot fit the entire tree. We again play a simple but effective trick to solve this problem. We first set an acceptable max bucket size. If a bucket grows to the max size, we discard the oldest data instance $\boldsymbol{x_0}$ and update the kernel center by

$$\boldsymbol{c}^{new} = \frac{n_k \boldsymbol{c} - \boldsymbol{x_0}}{n_k - 1} \tag{2.15}$$

Table 2.1.
All the clustering algorithms in comparison

Algorithm	complexity	clusters	stream	#para
Our Method	$O(Nd)$	variant	yes	2
Density Peak [31]	$O(N^2d)$	variant	no	2
AP [30]	$O(iN^2d)$	variant	no	2
Infinite GMM [45]	$O(iNd^3)$	variant	no	5
K-means++ [33]	$O(iKNd)$	fixed	no	1
GMM [34]	$O(iKNd)$	fixed	no	1
Mean Shift [35]	$O(iKNd)$	variant	no	2
StreamKM++ [20]	$O(Nd + iKCd)$	fixed	yes	2
BICO [21]	$O(Nd + iKCd)$	fixed	yes	3
Pirch [23]	hierarchical	variant	yes	3
Birch [22]	hierarchical	variant	yes	5

This trick works because recall fig.2.1, the kernel always goes to a denser place in the probabilistic sense. The oldest data instance must be in a more sparse area. Therefore, the update of eq.2.15 will move the kernel closer to the density peak. In other words, a limited bucket size is even better than an unlimited bucket size.

2.5 Experiments

2.5.1 Basic Clustering on Real, Image-oriented Datasets

In this section, we compare our method against the state-of-the-art center-based clustering algorithms on realistic datasets. As discussed previously, our goal is to achieve top-level performance in both speed and accuracy. Therefore, we consider not only the state-of-the-art stream method but also the accuracy dedicated algorithms. All the clustering algorithm considered in this chapter is listed in Table 2.1 with basic information, where

Table 2.2.
Basic statistics of all datasets

dataset	clusters	points	dimension
Olivetti [97]	40	400	4096
Segmentation	8	2310	18
LaSat	6	6435	36
Letters	26	20K	16
MNIST [98]	10	70K	784
Covertype	7	581012	54
ALOI [99]	1000	108000	128
ILSVRC12 [100]	1000	1.3M	2048

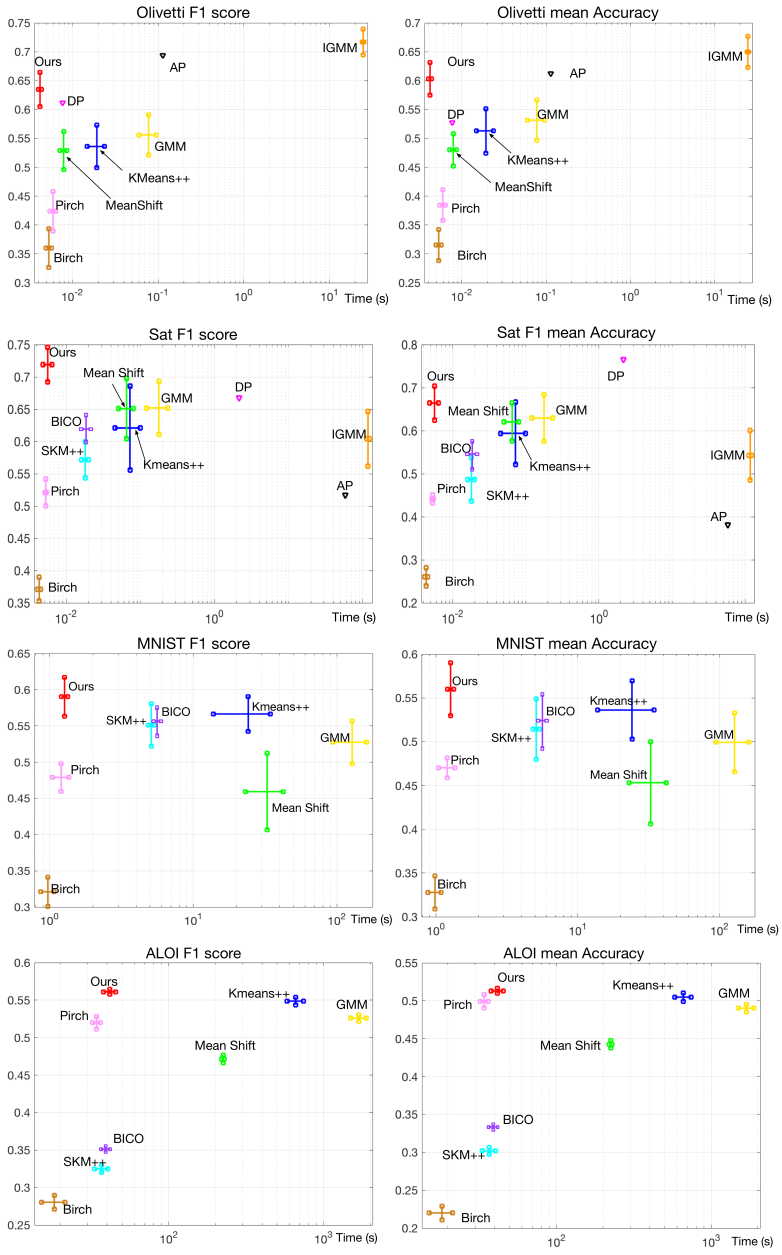

Figure 2.4. F1 scores and mean Accuracy of compared algorithms (part 1). Each method is denoted as the same color across all the datasets.

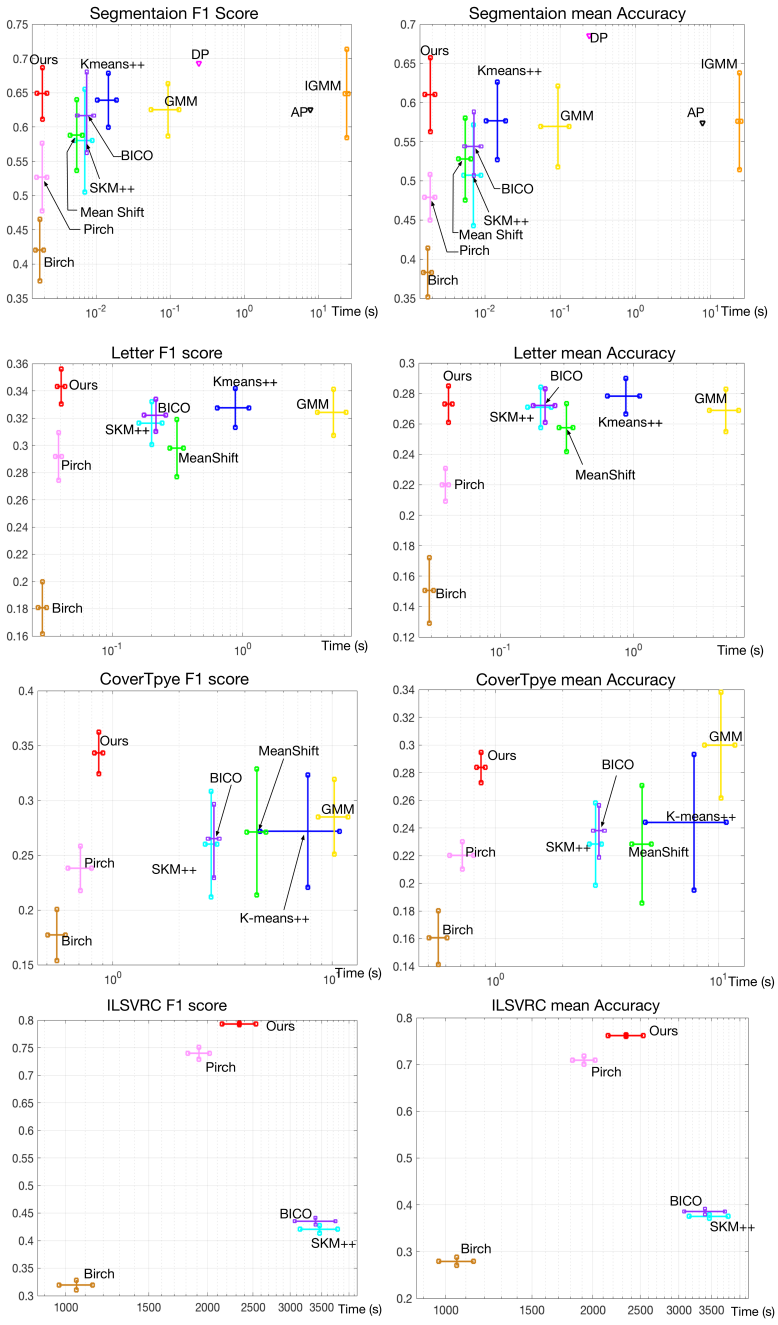

Figure 2.5. F1 scores and mean Accuracy of compared algorithms (part 2). Each method is denoted as the same color across all the datasets.

N is data size, d is data dimension, i is number of iterations (usually $30 < i \leq 100$ except IGMM, which usually takes hundreds of iteration of Gibbs Sampling), K is the number of clusters, and C is the size of Coreset size of StreamKM++ and BICO. We also list the ability to discover the number of clusters (indicated as "variant" in the "clusters" column) and the number of user-specified parameters ("#para"). Among these algorithms, Density Peak, Affinity Propagation and Infinite GMM are accuracy dedicated methods which only suitable for small datasets. K-means++, GMM and Mean Shift are popular algorithms applied in divisive domains and also regarded as approximating targets of many state-of-art clustering algorithms for Data Streams. StreamKM++ and BICO are two popular state-of-art approximation algorithms of K-means++ which are able to cluster very large data set with proven approximation accuracy. Pirch is a recent state-of-art carefully designed for Extreme Clustering [23], where the number of clusters is very large. Birch is an early work on clustering Data Streams providing a baseline comparison.

We evaluate the above algorithms on 8 datasets in different domains, dimensions, and sizes. General statistics of these datasets are shown in Table 2.2. Landsat satellite data, Letters, Segmentation and Covertype is provided as benchmark dataset by UCI Machine Learning Repository [101]. ILSVRC dataset consists of 1.3 Million of web images from 1000 categories, and we use the next-to-last layer of Inception Network [102] as the image description.

In this chapter, we consider two evaluation metrics for clustering accuracy: F1 score and mean accuracy. F1 score balances the impact between precision and recall. Mean accuracy only considered the number of points correctly assigned to the ground-truth cluster, which punishes too many numbers of clusters. We also report the running time of all the compared algorithms on a 2.7GHz Intel-i7 processor.

We run all the experiments 100 times except Affinity Propagation (AP) and Density Peak which are two deterministic algorithms. To efficiently visualize accuracy, running time and their standard deviations, we use a 2-D plot shown as Fig 2.4 and 2.5. The x-axis is the log-scale running time (since a few methods are very complex), and the y-axis is the clustering evaluation. We use two bars for each method. The horizontal bar represents its

running time and the vertical bar represents its clustering evaluation where the bar length represents the corresponding standard deviation. The cross-point of the two bars of each method represents the average evaluation value and running time. We always prefer a method resides on the left-top part of the plot.

In general, our algorithm always achieves top-tier accuracy across all the compared datasets with the fastest running time. Olivetti, Segmentation and Sat are small datasets, therefore we are able to run complex algorithms (AP, DP, and IGMM). We find every complex algorithm showed attractive performance (with respect to accuracy) in at least one dataset, however, no complex algorithm is able to perform well on all three small datasets. Our algorithm, though are not always the most accurate, shows a persistent clustering quality. More importantly, we achieved such clustering quality with the fastest running time. Since our method, AP, DP, and IGMM are all able to detect the natural number of clusters and require at least 2 user-specified parameters, our method can be a promising alternative for high-quality clustering. We still emphasis that these complex algorithms still have their preferred application areas where our method may not be able to always follow. And their theoretical elegance also plays a key role that inspiring emerging work on data clustering.

The letter, MNIST, Covertype, and ALOI are three medium-size datasets, where the superiority of our algorithm is clearly shown. Our method outperformed all other compared algorithms in these datasets when evaluated in the F1 score with the fastest speed. However, when evaluated by mean accuracy, K-means++ showed better results in the Letter dataset. The most likely reason might be that incorrectly assigned data points are not punished by mean accuracy, which is an indirect benefit from the known number of clusters for K-means. A similar phenomenon, but more severe, happened to GMM in the Covertype dataset, because of the unbalanced distribution of this dataset. Unlike these strange circumstances shown in K-means and GMM, our algorithm consistently generates top-tier quality regardless of data distribution, size and evaluation metric.

Finally, we evaluate our algorithm on complex datasets (ALOI and ILSVRC), which are in large size and contains a large number of clusters (1000 for each). Clustering on datasets that consists of a very large number of clusters is a recent challenge for clustering a very

large dataset [23]. We found that our algorithm drastically outperformed StreamKM++ and BICO. In the ALOI dataset, we have increased 61.43% of F1 score compared with BICO in ALOI, and 72.78% compared with StreamKM++. Similar results also showed when evaluated by mean accuracy. The reason for this drastic boost is each cluster in ALOI only contains 100 data points. If we maintain a coreset with 10% of the original dataset, each cluster in coreset contains merely 10 data points, which is too few to make K-means++ work. However, increasing the coreset size will sabotage its efficiency. As an extreme case of these two approximating algorithms, the K-means++ itself shows no better accuracy than our algorithm even costs 15.8 times longer running time. ILVRC is a very large dataset and we are only able to run stream algorithms. Again, our method provides a major jump on both F1 scores and mean accuracy for a similar reason. Specifically, we have increased the F1 score by 86.79% and 79.59% compared with StreamKM++ and BICO respectively. Evaluations by mean accuracy showed a similar pattern.

In general, we find StreamKM++ and BICO produce a well-balanced performance between speed and accuracy. Birch is always the fastest algorithm but with very limited accuracy (left-bottom in the plots). Pirch does not outperform StreamKM++ and BICO when the number of clusters is small but achieves major jump when the number of clusters is large and per-cluster population is small.

It can be easily recognized from Fig.2.4 and 2.5 that our method always locates at the upper-left corner of the plot (except in the very large dataset ILVRC), which indicates our algorithm provides best clustering quality and running speed. However, compared stream algorithms (StreamKM++, BICO, Birch and Pirch) usually locates at the right-bottom or mid-bottom area, which indicates these algorithms only provide a trade-off between speed and accuracy. We also notice that Our method maintains a small standard deviation across all the datasets compared with other non-deterministic clustering algorithms.

2.5.2 Building Visual Words on-the-fly

Visual Vocabulary [15] is a purely unsupervised image encoding framework, which is popular researched. Recent Deep Learning method [29] [103] successes in image classification and object recognition bring us a revolution in supervised learning in Computer Vision Domain. However, since labels are still expensive, purely unsupervised image encoding method is still motivated. Features from natural images are usually very cluttered, therefore there are no natural groups. Therefore, build a visual codebook is more of a data quantization than a clustering problem. In the most practical environments, K-means/EM is usually the best choice. This is because a more sophisticated method shows no significant superiority.

However, in some harsh environments, we sometimes have very limited resources. These situations could be a rover on Alain planets, drones discovering very dangerous places, where (a) the telecommunication is expensive or impossible, so all the computation must be on-the-fly; (b) the environment is totally strange so no existing visual codebook can be pre-installed; or (c) the environment is too harsh for a wait. Then the motivation of building visual worlds on-the-fly is raised. The task asks for an algorithm to

1. continuously discover emerging features

2. provide current visual codebook with absolutely no delay.

As a single-pass algorithm, we can provide the current visual vocabulary immediately at any time as well as discovering recently emerged features. Unlike a few approximation algorithms [20] [21], the current visual vocabulary can only be achieved by running K-means/EM on the reduced dataset. Therefore, the problem is, is our algorithm running in a harsh environment able to perform as satisfactorily as K-means/EM running in a gentle environment?

We apply our algorithm to building visual words and compared with other clustering methods. In this situation, AP and DP are not suitable for clustering a large collection of features. For VLAD and BoVW encoding, we run experiments on our algorithm, K-means, Stream KM++, BICO, Mean Shift and BIRCH. These algorithms typically provide

a hard assignment, which is suitable for VLAD and BoVW. We also compare our algorithm with EM on Fisher's Vector encoding, which usually prefers soft assignment. To get a soft assignment from our algorithm, we use the resulted clustering centers as initialization and perform 10 iterations of Naive EM algorithm.

We used three publicly available image collections: (a) Object Discovery 100 (**Obj. Disc**) consists of 3 categories, each containing 100 images; (b) Caltech 101 (**Caltech101**) consists of 102 categories, where each image contains a single object from the corresponding category; and (c) PASCAL VOC 2007 contains 20 categories of objects in increased clutter, occlusion, and varying viewpoints: we first used a subset of 6 chosen categories (**PASC(6)**) and then the entire collection (**PASC(all)**).

General Expeiment Setting. In Bag-of-Visual-Words, we compute the square root of each bin of histogram, and then L^2 normalize it before feeding into SVM. In VLAD and FV, we use signed square root for each bin of encoded image descriptor, and then L^2 normalize it before SVM.

We generate 8 by 8 patches in different scales with stride 4 and compute L^2 normed SIFT descriptors. These descriptors are whitened by PCA along with dimensionality reduction(reduced from 128 to 80 for FV and 100 for others). We also append the relative x-y spatial coordinate of each feature. The value of coordinate ranges from -0.5 to 0.5. In other words, the original coordinate of each image is at the center and the coordinate value is divided by image size for each axis.

We use the hierarchical version of our algorithm since the vocabulary size is usually larger than 100. We set the max number of layers to be 3, and choose the density ratio $p_f = 0.05$ for each layer. The distance threshold for the first layer, θ_1, is 1.2 for VLAD and BoVW and 1.0 for FV. We choose different distance thresholds in the third layer, θ_3, to generate a similar number of centers as its counterpart. We set the distance threshold in the middle layer by $\theta_2 = \sqrt{\theta_1 \cdot \theta_3}$.

Size of Clustering Data. From all the feature patches extracted, we sampled a subset to perform clustering. In the experiments that using K-means and Mean Shift, we randomly choose 0.5M for Object Discovery Dataset and 1M for others. Since our algorithm, as

well as other single-pass algorithms, are much efficient than the above iterative ones, it allows us to mine a larger dataset and still be much faster than those two. In the experiment applying the single-pass algorithm, for VLAD and BoVW, $N = 1M$ for Object Discovery, and $N = 5M$ for others. In FV, we let both our algorithms and EM mine 1M of feature for all datasets.

The experiment result is shown in table 2.3, 2.4 and 2.5. Parameters in compared algorithms are tuned to optimize their performance. For Mean Shift and BIRCH, we also tune the parameters to generate the number of clusters close to the value we set to K-means, StreamKM++, and BICO. From these experiments, we found that

1. our algorithm out-performed Stream KM++ and BICO in both speed and accuracy.
2. our algorithm is slightly slower than BIRCH but provides much better results.
3. our algorithm provides equally good results but much faster than K-means and EM.

Therefore, in general, our algorithm provides the best trade-off between all these compared clustering methods.

2.5.3 Matching SIFT Features on-the-fly

Finally, to test the performance of our approach in sequential data (clustering on the fly), we used videos captured by an onboard camera of a quadrotor during flight. Fig. 2.6 illustrates an indicative example of clustering detected SIFT features in eight non-consecutive 752×480 frames of a video (frame numbers are shown on the top left of each image).

For each frame, we show two emerging features and all the previously emerged features. For example, in frame 375, we show two features that appeared in frame 251-375 but not appeared in frame 1-250, marked as triangles. And the squares include two parts: (1) emerging features in frame 126-250, and emerging features in frame 1-125. In other words, we show all the emerging features we have shown previously. The color indicates feature clusters: matched features are shown in the same color, no matter what shape it is. For example, green triangles in frame 375 (indicating windows) are all matched to green

Table 2.3.

Results of Fisher's Vecotor

	Memory Tree			Naive EM		
data	K	mAP%	time(s)	K	mAP%	time(s)
Obj.Disc	103.0	95.41	80.46	100	95.06	63.37
	206.5	96.50	151.74	200	95.82	334.31
	385.5	**97.03**	**282.61**	400	**96.05**	**819.12**
	629.0	5.46	527.34	600	95.80	2012.20
Cal101	231.6	74.55	220.91	200	74.59	197.63
	414.3	76.26	423.07	400	75.96	879.18
	617.7	77.56	610.64	600	77.55	2146.14
	795.5	**77.62**	**677.01**	800	**77.08**	**3654.57**
voc(6)	224.0	72.65	233.46	200	72.46	197.36
	390.0	74.49	410.76	400	74.17	806.97
	627.0	75.85	521.74	600	75.43	1718.42
	813.0	**76.34**	**819.28**	800	**76.30**	**3636.91**
voc(all)	223.7	60.68	227.45	200	60.03	190.56
	419.4	63.08	512.09	400	62.26	782.23
	609.5	63.99	621.58	600	63.44	1551.14
	784.1	**64.68**	**801.59**	800	**63.82**	**3523.66**

Table 2.4.

Results of VLAD

	Memory Tree			Kmeans			StKM		BICO		MeanShift		BIRCH	
	K	mAP	time	K	mAP	time	mAP	time	mAP	time	mAP	time	mAP	time
Ob.Dis	131.8	94.60	6.2	100	95.57	441.3	94.59	15.8	94.21	14.7	94.31	405.4	94.28	13.3
	265.7	95.03	7.1	200	95.52	522.0	94.35	25.9	94.52	22.8	94.50	658.1	94.84	18.9
	417.9	**95.08**	**7.7**	400	**96.02**	**523.7**	94.41	39.6	94.51	33.5	94.10	841.1	93.62	19.6
	587.9	94.51	9.1	600	95.73	545.5	94.23	43.4	94.37	37.8	94.17	1028.0	94.01	20.3
Cal101	252.6	79.18	56.2	200	80.47	138.2	79.51	109.7	79.74	107.3	70.54	572.6	78.08	95.7
	420.0	80.85	90.4	400	80.57	331.6	80.02	143.8	80.08	132.9	71.32	1164.3	78.23	98.2
	671.7	81.02	139.6	600	80.82	456.4	80.29	158.6	80.14	144.1	70.28	1599.2	77.21	103.7
	839.6	**81.27**	**165.4**	800	**81.22**	**847.3**	80.59	222.1	80.27	173.2	73.93	2511.7	76.69	107.4
voc(6)	254.4	66.48	70.1	200	66.11	867.5	65.72	102.4	66.02	102.7	62.32	682.7	65.49	101.3
	393.5	68.36	77.1	400	69.37	897.2	68.65	147.2	68.48	127.1	62.57	1334.9	65.71	103.3
	567.0	70.23	86.3	600	70.21	1012.4	69.67	161.5	69.74	132.3	63.04	1508.2	64.07	104.4
	809.0	**71.55**	**104.6**	800	**70.42**	**1102.9**	69.92	218.7	69.89	181.5	63.21	1884.0	63.42	106.2
voc(all)	204.2	56.87	64.5	200	53.38	888.4	52.78	107.9	52.28	104.3	48.44	634.2	52.21	97.7
	394.4	55.82	77.1	400	56.26	908.2	56.18	143.5	56.46	133.2	50.43	1291.5	52.25	101.1
	654.9	57.51	101.3	600	58.08	992.1	56.92	152.9	57.04	152.0	50.15	1574.2	51.12	102.4
	782.9	**58.77**	**101.4**	800	**58.10**	**1092.0**	57.44	224.1	57.61	177.4	51.08	1717.2	51.04	104.5

Table 2.5.

Results of BoVW

	Memory Tree			Kmeans			StKM++		BICO		MeanShift		BIRCH	
	K	mAP	time	K	mAP	time	mAP	time	mAP	time	mAP	time	mAP	time
Ob.Disc	2666.7	95.23	26.9	2000	94.57	690.2	93.94	30.7	94.01	26.7	93.37	1431.2	93.51	13.8
	4227.8	95.21	38.5	4000	94.81	742.6	94.09	34.9	94.19	30.2	94.07	1724.9	94.02	16.8
	5977.2	**95.63**	**53.5**	6000	**94.76**	**814.4**	93.91	41.4	94.20	33.4	94.10	2039.1	94.17	17.1
Cal101	4188.8	75.50	152.5	4000	75.79	1798.5	74.80	665.9	75.07	477.6	66.64	3203.8	68.55	145.0
	6548.7	76.81	228.1	6000	76.81	1991.3	75.15	719.3	75.92	597.3	68.21	4887.3	70.89	165.7
	8207.4	77.03	286.6	8000	76.91	2068.9	75.62	1001.4	76.07	697.9	68.69	4056.2	71.46	169.2
	9939.8	**77.66**	**342.8**	10^4	**77.87**	**2132.7**	75.91	1120.6	76.29	797.1	68.84	4793.0	71.92	170.3
voc(6)	4154.2	62.49	128.3	4000	61.31	1804.5	60.67	572.9	61.02	475.3	56.87	3372.9	57.87	154.5
	6481.7	63.41	188.7	6000	63.03	2001.3	61.94	707.3	62.35	607.9	58.17	3979.2	58.21	161.0
	8016.5	63.88	233.6	8000	63.21	2054.9	62.37	873.1	62.47	724.4	58.61	4259.6	59.36	169.4
	9946.0	**64.95**	**290.3**	10^4	**63.45**	**2231.6**	62.64	1102.3	62.71	817.3	58.89	4966.5	59.42	172.8
voc(all)	4098.8	49.85	260.7	4000	48.43	1784.2	48.34	549.6	48.39	487.3	44.95	3237.6	45.67	159.2
	7037.3	50.85	406.5	6000	49.95	1997.7	49.20	735.3	49.22	594.6	45.01	3969.0	45.52	163.3
	8655.0	51.42	486.9	8000	50.20	2037.9	49.87	884.1	49.79	712.5	45.62	4101.5	46.09	167.4
	10958	**51.67**	**594.2**	10^4	**50.41**	**2257.4**	49.43	1021.9	49.51	823.1	45.88	4574.8	46.34	177.6

squares in frame 500, which are again windows. All the emerged features we selected are the most frequently matched features in previous frames.

The magenta-yellow arrows in frames $\#250$ and $\#375$ show indicative examples of newly emerged clusters (orange and green square categories), while the long double arrow indicates correspondence between features (features in the same cluster) across frames. In this experiment we used the basic version of our method (Algorithm 1), with $p_f = 0.005$ and distance threshold $\theta = 275$.

StreamKM++ and BICO are not suitable for detecting emerging features. Stream KM++ and BICO require the number of the cluster as a parameter and the clustering centers at a certain time are achieved by running K-means. Not only are they slow, but there is also no way to discover a new feature neither keep track of old features. BIRCH keeps splitting CF nodes, therefore, a cluster in time t_1 may not exist in $t2$, which faces the same problem with StreamKM++ and BICO.

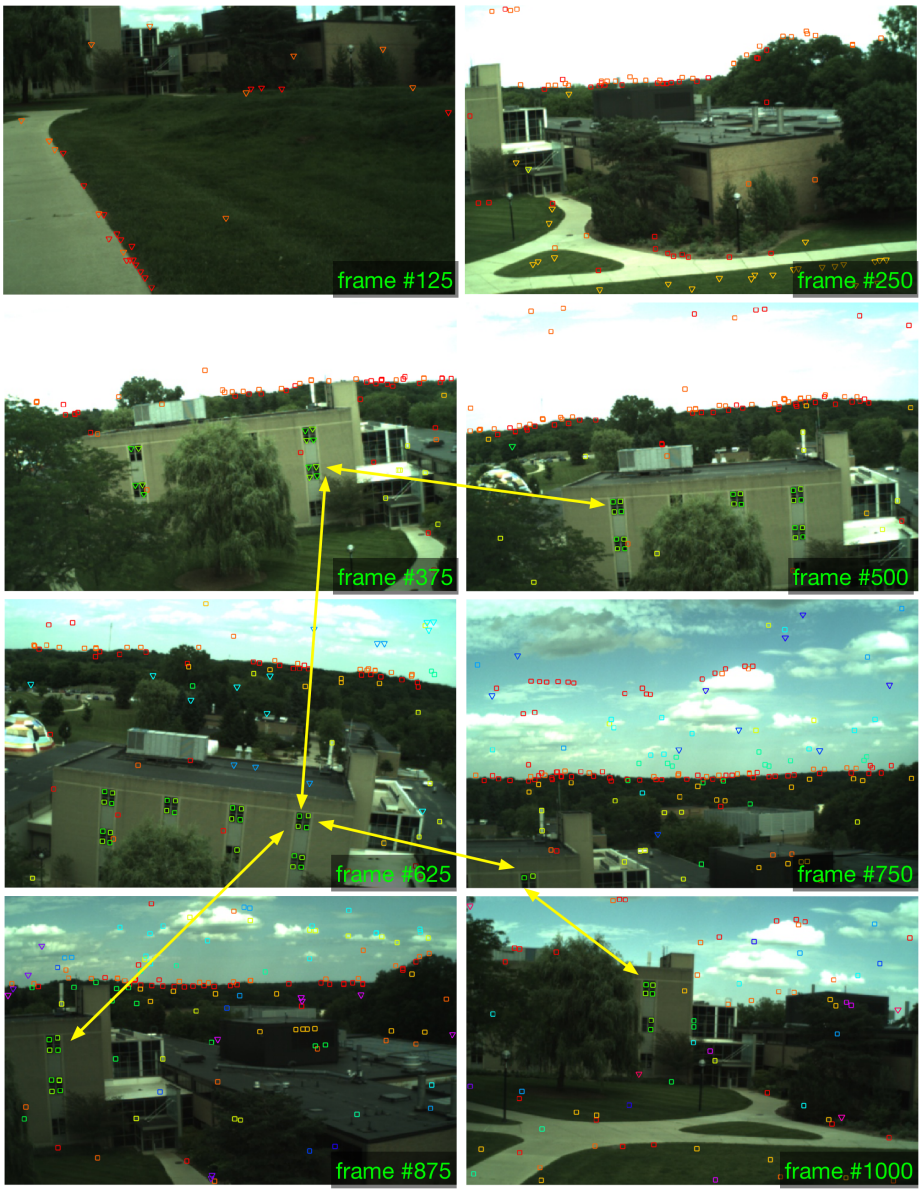

Figure 2.6. Building visual vocabularies on the fly: application of our approach in streaming videos. Colors and shapes indicate cluster assignments of the detected features, while the arrows in frames indicate correct matches.

2.6 Summary

We show in this chapter that accuracy is not necessarily sacrificed for faster speed and stream ability of a clustering algorithm. Samples of the original dataset express efficient

and enlightening information which can be used for saving computation. We regard a window of the Data Stream as a sample set to estimate the density of emerging data points such that the algorithm is able to detect promising regions to set up a kernel and then seek density peaks in a single pass. It worth noting that each data point itself contributes to all the existing kernels either explicitly serves to update or implicitly helps to estimate the density, because of the nature of randomness. Our extensive and divisive experiments demonstrated the clustering performance of our method.

We presented a center-based clustering algorithm that consists of three components: stochastic Mean Shift, dynamic initialization, and density test. These three components are compactly designed to carry the nature of randomness, such that high-quality single-pass clustering can be achieved. Our algorithm improves the trade-off between accuracy and efficiency when applied to Visual Encoding, and it can even outperform, in terms of accuracy, more complex iterative methods. Its statistical properties and the presented experiments show that it can be an excellent alternative for non-parametric clustering tasks and data quantization.

The desired properties will benefit the vision methods and applications in Dynamic Environments, as we will see in the following chapters.

3. UNSUPERVISED DEEP ENCODING: A BRIDGE BETWEEN CNN AND VISUAL ENCODING

In this chapter, we propose a Similarity-based Convolutional Neural Networks (CNN) that unifies Visual Encoding and CNNs. By finding equivalence between a single layer of Neural Network and Clustering, CNN can be interpreted as multi-level visual encoding that all the filters can be pre-trained by unsupervised feature learning. In our model, the pre-activation normalization is a similarity measure instead of the dot product. This mechanism forces the network to learn an explicit expression of features at each level. Pre-training considerably improves training speed and usually results in a better error rate because gradients are applied only to input that is sufficiently similar to the corresponding filter. Convolution by Cosine Similarity along with pre-training makes Batch Normalization unnecessary to our model since the pre-training method implicitly whitens the data at the early stage of gradient descend. We conduct a series of experiments on each of the above aspects of our model. Though Batch Normalization is not applied, Similarity CNN converges as fast as models with Batch Normalization. Compared with the state-of-the-art models with a similar number of parameters, our net shows improved parameter efficiency.

3.0.1 Relationship between of Single-layer Neural Networks and Clustering

Consider a single layer of neural networks with ReLU activation function, $y = \sigma(W^T x + b)$, where $W = [w_1, ..., w_k] \in R^{d \times k}$ are the weight matrix, $\sigma(.)$ is the ReLU activation function, $x \in R^d$ and $y \in R^k$. Generally, each node can be regarded as a linear classifier defined by the separating hyperplane $z = w_i^T x + b_i$. If the input and the weights of each neuron are L2 normalized, namely, $||x||_2 = 1$ and $||w||_2 = 1$, then each element of z becomes the cosine similarity between x_i and w_i added by a bias. Therefore, if we regard (or let) each weight vector w_i as the clustering center, then the output y of this layer is the unnor-

malized soft assignment of x to k clustering centers. Such operation is equivalent to use k clustering centers to encode x from R^d input space into R^k feature space. Then the ReLU function becomes more meaningful than simply removing the saturation of other activation functions. If the output is positive, it means that the input x is similar to the corresponding feature w_i and the cosine similarity value defines the unnormalized membership strength. Namely, each node of a single layer of Neural Networks is a kernel and can be explicitly represented by its weight. The bias can be thought of as a similarity threshold to activate the correspondent kernel and also defines the "size" of the kernel.

Given the above observation and similar to [50], we consider cosine similarity as opposed to dot product inside the activation function, namely,

$$y = \sigma(\frac{w^T x}{|w| \cdot |x|} - b) \tag{3.1}$$

where $\sigma(.)$ is the activation function (usually ReLU Family [104] [105] [106] [107]). The bias b can be regarded as a similarity threshold. [50] considers cosine similarity as a normalization method. In this paper, we focus on how unsupervised feature learning benefits Convolutional Neural Nets. Cosine Similarity in our framework can be replaced by any similarity measure. Though the above analysis considers general feed-forward Neural networks, it can also be applied to Convolutional Neural Network. The only difference is that the input is a patch from the previous Feature Map, which is detailed as follows.

Compared with traditional clustering, the advantage of softly assigning data to cluster centers by a layer of a neural network is that we can fine-tune these centers end-to-end by Stochastic Gradient Descend, using label information. Therefore, we propose to pre-train the similarity-based Neural Network by clustering, where the cluster centers become the initialized values of kernel weights (detailed in 3.2). Similar to Autoencoders [51] and GAN [59], unsupervisedly pre-trained networks are representative. Unlike Autoencoders and GAN, these pre-trained weights are explicitly representative, because clustering centers directly express the pattern itself. For example, if the input data are images, then each filter weight vector is also an image.

Prevalent Deep CNN models use dot product instead. This design dates back to the paradigm of linear classifiers. In such a paradigm, neuron weights define the hyper-plane

that separates the input space. The positive output indicates that the input is above the classification plane and the negative value indicates it is under the classification plane. A single layer of neural networks with k neurons construct k hyper-planes and divide the input space into at least $k + 1$ sections (up to $2^k - 1$ sections). Since each section can be arbitrarily large, input data from the same hyper-section cannot be regarded as similar, especially after random initialization. Stacking another layer of neural networks can be considered as further dividing each hyper-sections into finer and more class-related areas. The current philosophy of convolutional layer considers any two image patches carry the same visual pattern if they both responded by the same filter, we argue that this is not the most efficient and explainable way to represent image features, as discussed following.

3.1 Similarity Convolution

Recall the framework of a typical Convolutional Layer with K filters of receptive size $d \times d$. The resulted Feature Maps is a $r \times c \times K$ volume, where r and c are the number of rows and columns of previous Feature Maps (given the previous Feature Map is padded). The computation of the convolution operation is

$$y = \sigma(w^T x + b) \tag{3.2}$$

where x is flattened local patch, w is the filter weights and $\sigma()$ is the ReLU function,

$$\sigma(z) = \begin{cases} 0 & \text{if} \quad z < 0 \\ z & \text{if} \quad z \geq 0 \end{cases} \tag{3.3}$$

The general idea behind Feature Maps is that similar visual patterns trigger common filter(s). Some works [53] [54] found that the output of the Feature Maps responses to particular patterns that ”look” similar. However, ”feature” considered in General CNN is different from that of traditional Visual Encoding. Prevalent CNN models use dot product for convolution operation followed by ReLU. This design dates back to the paradigm of linear classifiers. In such a paradigm, neuron weights define the hyper-plane that separates the input space. The positive output indicates that the input is above the classification

plane and the negative value indicates it is under the classification plane. A single layer of neural network with k neurons constructs k hyper-planes and divide the input space into at least $k+1$ sections (up to 2^k-1 sections). Since each section can be arbitrarily large, input data from the same hyper-section cannot be regarded as similar, especially after random initialization. We refer the above kind of features as ***Linear Feature***. Features in Visual Encoding, however, is a direct description of image patches whose pattern frequently appears. Namely, a feature vector defines a kernel, i.e. a close neighborhood of the feature vector with limited size. An input that has a similarity higher than the given threshold means it falls inside the kernel. We refer to such kind of features as ***Kernel Feature***. We propose to consider Kernel Features in Convolutional Neural Networks, where each convolution operation is a similarity measure, as shown in fig1.4. Specifically, given a filter with weights w and a Feature Maps F^l of layer l, we compute similarity $s(w,x)$ for x extracted at every location of F^l by sliding window. Thus the weights themselves explicitly express the corresponding Kernel Features.

To this end, we can regard such similarity-based CNN as special case of Visual Encoding. Since every location of the $F_i^{(l+1)}$ is the similarity value (or regarded as membership), then we indeed used this $F_i^{(l+1)}$ to encode features that are stored in filter weights of layer $l+1$. Unlike previous shallow encoding measures [15] [17] [16] [47], encoding by convolutional map can be performed layer by layer. And more crucially, unlike totally unsupervised feature learning, we can use Back Propagation to fine-tune these pre-learned features according to image labels. We also apply element-wise ReLU to each Feature Map. ReLu can be considered as sparsity discussed later in this section. We refer the above encoding method to Deep Kernel Feature Encoding.

In this chapter, we consider cosine similarity (eq.3.1) since it provides us elegant theoretical properties. When Cosine Similarity is considered, L2 normalized kernel weights become a group of basis spans on an input subspace. Each element of the output vector (blue bar in Feature Map F^{l+1}), is an unnormalized soft assignment of the corresponding Kernel Feature. At the same time, this output vector can be regarded as the projection on each basis defined by the Kernel Features. Therefore, we naturally expect such a group of

basis (1) to be orthogonal, and (2) to be representative to input patches. Orthogonal property indicates that given a group of Kernel Feature $\{f_{1...k}\}$, we have $f_i \cdot f_j = 0, \forall f_i, f_j, i \neq j$. Specifically, $\{f_{1...k}\}$ can be jointly optimized according to the following objective:

$$\begin{aligned} & argmin \sum_i \sum_{j \neq i} f_i \cdot f_j - \lambda \sum_n f_n^* \cdot x_n \\ & s.t. \\ & f_n^* \in \{f_{1...k}\}, \\ & f_n^* \cdot x_n \geq f_i \cdot x_n, \forall f_i \in \{f_{1...k}\} \end{aligned} \tag{3.4}$$

where $\{x_{1..N}\}$ is the set of all L2 normalized Feature Map patches of previous layer, $\{f_{1...k}\}$ is the set of L2 normalized Kernel Features (thus each dot product become cosine similarity), and f_n^* is the closest feature that x_n is assigned to. Though we can pre-train a layer according to the loss defined by eq.3.4, the above optimization problem actually is studied by Data Clustering where cosine similarity is considered as the similarity measure. The first term of the objective can be considered as inter-cluster dissimilarity and the second term can be considered as inner-cluster similarity. When training with millions of images, it is considerably more efficient to take advantage of recent advance of stream clustering [108], where similarity measure can be arbitrarily selected.

We can find the above encoding is related to whitening. Both of them attempt to find a group of basis that decorrelates the input dimension. We can expect that it benefits convergence since [72] found that batch level decorrelation by ZCA whitening improves training dynamics. The difference between Kernel Feature Encoding and PCA/ZCA is also clear. In PCA/ZCA whitening, the objective is to transfer the original data such that the projection on each axis has the same distribution (same mean and variance). The objective of Kernel Feature Encoding attempts to transfer original data such that the projection on each axis is "sparse". The sparsity indicates given a pre-trained basis, the projections of the most original data are small values except a group of data with similar patterns, whose projection is considerably large (e.g. close to 1), which we argue as following that it improves learning dynamics.

Consider two adjacent convolutional layer (say layer $l-1$ and layer l) in a deep architecture, with loss function L. Let's consider filter $f_i^{(l)}$ as the i^{th} filter in layer l. We have $s_i = \sigma(a)$, and $a = cos(f_i^{(l)}, x) + b$, where s_i is the output of convolution, $\sigma(.)$ is the ReLu activation function and b is the bias. The gradient of $f_i^{(l)}$ with respect to L is

$$\frac{\partial L}{\partial f_i^{(l)}} = \frac{\partial L}{\partial s_i}\frac{\partial s_i}{\partial a}\frac{\partial a}{\partial f_i^{(l)}} \tag{3.5}$$

and

$$\frac{\partial a}{\partial f_i^{(l)}} = \frac{1}{|w||x|}\cdot x - \frac{cos(f_i^{(l)}, x)}{|w|^2}\cdot w \tag{3.6}$$

We can see that $\frac{\partial L}{\partial f_i^{(l)}} = 0$ if $a \leq 0$ because of ReLU activation. It means that if the input feature map patch is not assigned to Kernel Feature $f_i^{(l)}$, it does not contribute to Gradient Descend. In other words, learning only take place when the input patch strongly related to the corresponding filter.

From eq.3.6, the gradient of $f_i^{(l)}$ is also proportional to the cosine similarity between the input patch x and the corresponding Kernel Filter $f_i^{(l)}$. Thus an input patch that is similar to the Kernel Feature contributes more to learning than unrelated inputs. Since filter responses are pre-trained to be sparse at the early stage of Gradient Descend, it helps to learn better targeted to most crucial parameters. We argue that this is the major reason that pre-training significantly benefits learning dynamics (see details in section 3.3.1).

Understand Pooling. Max Pooling and Average Pooling are the major two pooling methods used on CNN. Both will have a less obscure understanding under the framework of Similarity Net. In Max Pooling, we remove the negative responses because it is not similar to a certain Kernel Feature. Thus it is natural to select the location with the highest similarity to indicate the existence of a certain feature inside the pooled area. Now the bias of each filter becomes a similarity threshold for judging the existence of the corresponding feature. This naturally reminds us of traditional visual encoding models, we assign each local patch to a known feature and encode such information into a compact format (usually a histogram). As illustrated in fig.1.4, Similarity Net uses a sparse K dimensional vector to express the assignment of each local patch, those elements have 0 value (low similarity)

is screened out of the assignment, and thus removed from gradient descend. This encoding is also similar to the triangle metric [47] frequently used in unsupervised feature learning. Therefore, the entire map is a soft assignment to all local patches. Unlike the traditional encoding method typically computed a histogram, this representation reserves the spatial information of each local patch that can be further processed in deeper layers.

Average Pooling can be regarded as a weighted histogram of the pooled area. Namely, we are interested in how frequent a pattern appears in the pooled area. Therefore Average Pooling is more suitable for high-level, large area pooling works as a summary of a certain large acceptive field. Max Pooling, however, is more suitable for low-level, small acceptive field to remove the disturbance of patch unrelated to the corresponding Kernel Feature. As we can observe that recent successful models [10] [109] [110] [12] all take Max Pooling after the first Convolutional Layer and Average Pooling in higher level (usually right before the final fully connected layer).

3.2 Unsupervised Deep Encoding by Multi-level Unsupervised Feature Learning

Since each filter weight is the feature used in encoding, we can cluster a large number of sampled patches from previous Feature Maps, and cluster them to find K clustering centers. These centers will become the initial weights of the current layer. Since we use cosine similarity in the convolutional operation, we should also use cosine similarity in clustering. Also, due to a large number of patches generated from each layer, a stream clustering method is necessary. Note that the unsupervisedly pre-trained layer is already a good representation for a classification [47], hence fine-tuning corresponding parameters through Back Propagation could potentially achieve better performance and a faster convergence rate.

The process of unsupervised pre-training is as follows. Initially, there is 0 layers in the network. We extract patches from training images and cluster these patches using cosine similarity as the similarity measure. Thanks to recent advances in stream clustering [108], we are able to achieve high-quality cluster centers by a single pass through the image

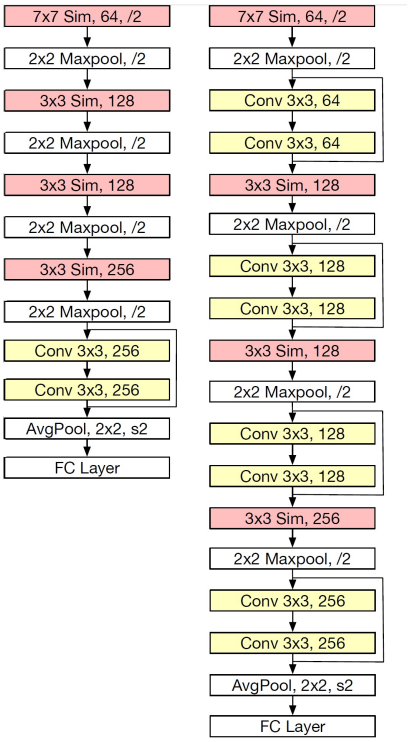

Figure 3.1. Architectures explored in this chapter. The left two architectures are tested for subsets of ILSVRC2018 to study the parameter efficiency of Similarity Layers. The third architecture is designed for ILSVRC 2018.

patches with an arbitrary similarity measure. After acquiring the cluster centers of extracted image patches, we initialize the first convolutional layer by assigning each cluster center to a convolutional filter. Next, we fix the parameters of the first convolutional layer and use eq.3.1 as the convolutional operation to compute the Feature Maps of the first layer. We then again extract patches from the computed Feature Maps and perform clustering to initialize the next convolutional layer, and so forth. When all the layers are established, we can start fine-tuning by Back Propagation. Since each convolutional operation computes a similarity score, we refer to our proposed model Similarity Convolutional Neural Networks.

3.3 Experiments and Analysis

Dataset. We consider 3 subsets of ILSVRC2018: randomly select 10, 50 and 100 classes from the entire dataset. ILSVRC dataset has 1000 class high-resolution images. Each class contains up to 1300 training samples and 50 validation samples. We train our models using all the images from the training set and report results in the validation set.

Architectures. In this section, we propose a hybrid architecture with our similarity-based conv layers and Resnet blocks, as shown in fig.5.1. We test three architectures for ILSVRC2018 [111], from shallow to deep. The first architecture stacks 4 similarity layers, each of which followed by a maxpooling layer. We then add a Resnet Block onto the last maxpooling layer. Standard average pooling and fully connected layer follow the Resnet Block. We find the final Resnet Block increases training speed. The second architecture is similar to the first with the difference that a Resnet block is inserted after each maxpooling layer, which results in 13 layers in total. For systematic comparison, layers used in Similarity Net must be a subset of its compared counterpart. For example, if a General CNN or a ResNet uses n convolutional layers with 64 3×3 filters, the compared Similarity Net can use at most n similarity layers with the same number of 3×3 filters.

Implementation.All the models are trained by Adam Optimizer [112] with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.001$. We do not apply Batch Normalization either in similarity conv layers nor in Resnet blocks of the Similarity Nets. We augment each image and its horizontal reflections by randomly resizing the longer edge into $[256, 480]$, and take random 224×224 crop, following [10]. To guarantee information flows majorly through pre-trained similarity layers in the early stages of training, we initialize the weights of Resnet blocks from a small interval $(-10^{-5}, 10^{-5})$). One advantage of this design is that we do not need auxiliary projection shortcuts used in [10].

3.3.1 Effective of Pre-training

In this section, we evaluate how pre-training benefits the network. We first train Similarity Networks of the same architectures with random initialization and apply the same

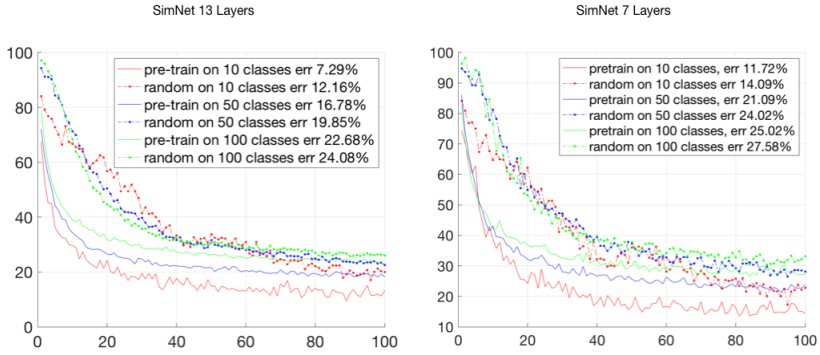

Figure 3.2. pre-train vs random initialization.

optimization method in the above 3 subsets. The learning curve is shown in Fig.3.2 and the final top 1 validation error is also presented in Table.3.1. We found that validation error decreases slower in the early stage of training. The whole learning process also takes more iterations to converge. The final error rates of randomly initialized models are higher than that of pre-trained models.

To explore deeper into the unsupervised feature learning, we conduct the following experiments. Given a pre-trained Similarity Net, we freeze all the similarity layers and let Gradient Descends applied only on the Fully Connected layer. We compare the result with randomly initialized Similarity Net, applying the same training process. Good features in Similarity Layer should result in better accuracy. We then perform a series of similar experiments, but let one more Similarity Layer unfrozen (from last Similarity Layer backward to the second). In this way, we can investigate the feature quality of each layer.

We consider 4 Similarity Layers architecture, similar to Sim-7, but remove the Resnet Block. We train all the configurations by Adam Optimizer for 30 epochs, with learning rate equals 0.001 for the first 20 epochs, and 0.0001 for the last 10 epochs. The experiment results are summarized in Fig.3.3. As we can see, through all the subsets of ILSVRC datasets, the pre-trained network achieves significantly better accuracy than its randomized counterparts. It demonstrates that features pre-trained by deep encoding are more related to semantic classes at each level.

3.3.2 Effect of Batch Normalization

It has been widely acknowledged that Batch Normalization plays a key role in modern Neural Networks. It speeds up convergence and usually improves the final accuracy. In this section, we further compare the effect of Batch Normalization on our model and traditional models. We use the same architectures tested in the previous section, but consider both situations that when BN is applied and not applied. All the datasets, augmentation and training approaches are the same as previous experiments. We plot on Fig.3.4 the learning curve of 2 situations for all 4 compared networks.

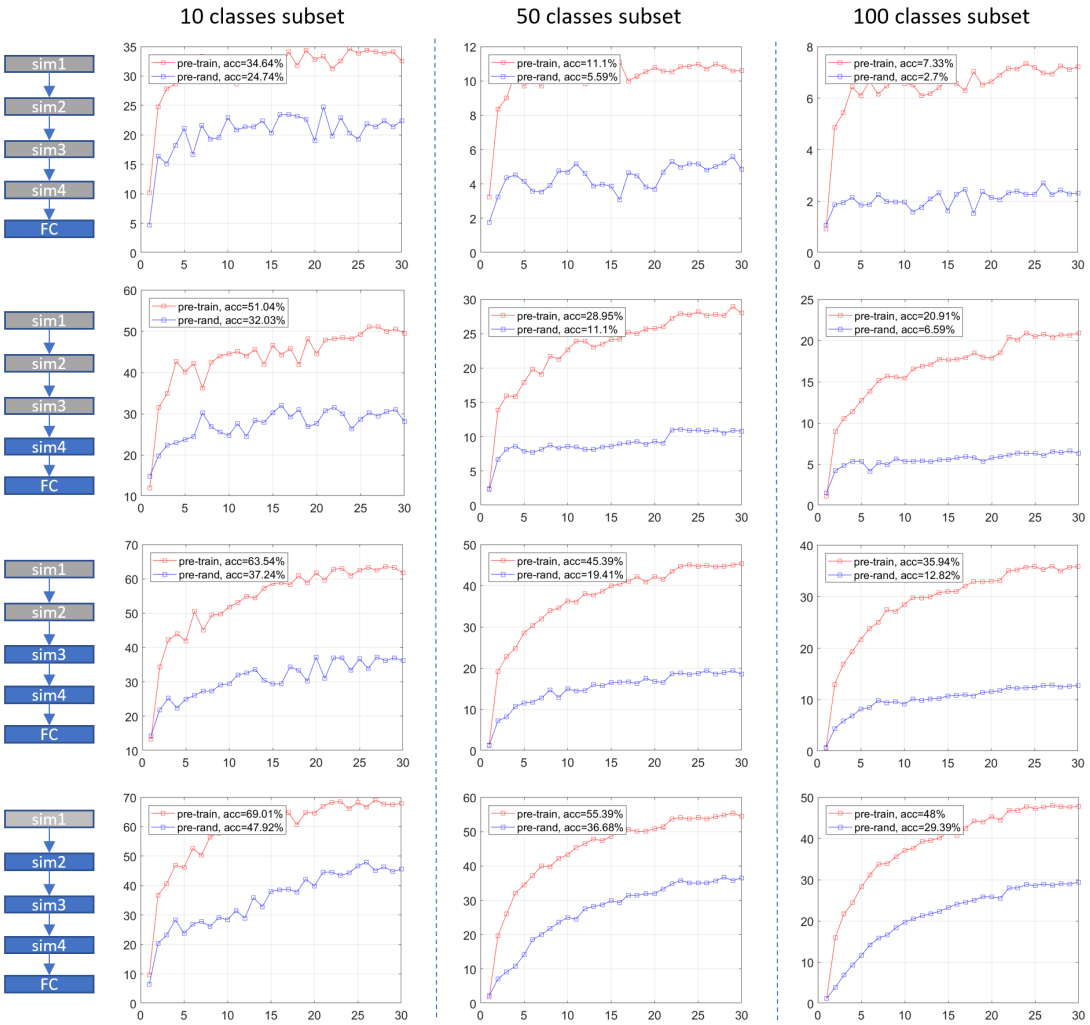

Figure 3.3. Significance of pre-training: validation accuracy when earlier layers are frozen. Each row of this figure indicates that we keep some of the Similarity Layer frozen and let SGD only apply on deeper layers. The architecture and frozen layers are shown on the left. The grey layers are frozen and the blue layers are trained. Each Similarity Layer is initialized either by pretraining (red curves) or in random (blue curves).

Table 3.1.
Error rate (single crop) in the subsets of ILSVRC2018. For Similarity Net, we apply the model with pre-training and random initialization. For each model, we also consider situations that Batch Normalization is applied and absent. Suffix "BN" stands for Batch Normalization.

	#para	10 classes	50 classes	100 classes
Sim 13 pre-train	2.37M	**7.29**	**16.78**	**22.68**
Sim 7 pre-train	1.71M	11.72	21.09	25.02
Sim 13 random	2.37M	12.16	19.85	24.08
Sim 7 random	1.71M	14.09	24.02	27.58
Sim 13 BN	2.37M	9.38	18.13	22.96
Sim 7 BN	1.71M	13.80	21.49	27.25

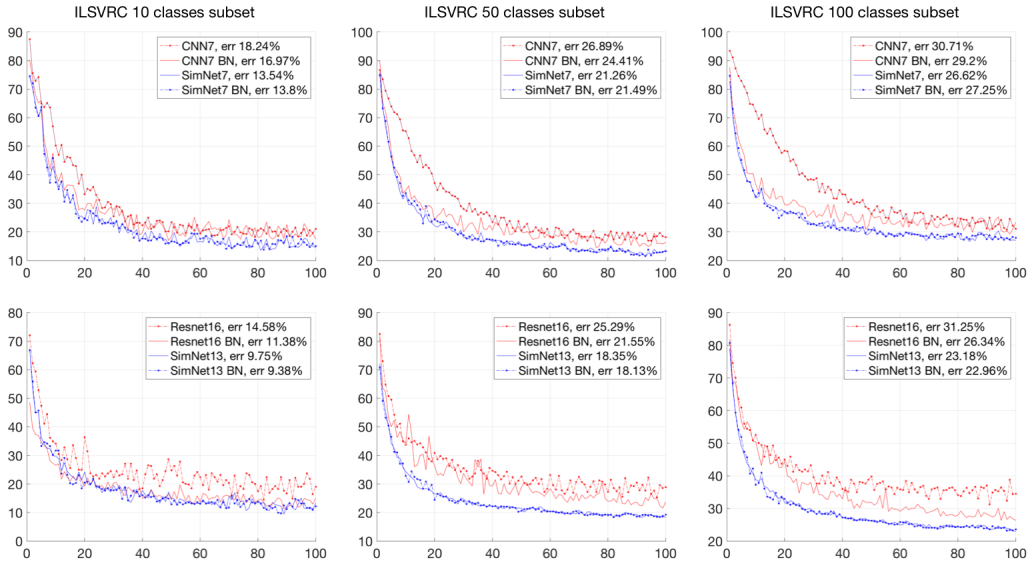

Figure 3.4. Learning Curve (validation error rate) of models with and without Batch Normalization. Each model is trained by Adam Optimizer for 100 epochs with a fixed learning rate (0.01).

As we can see, Batch Normalization speeds up the learning process of Resnets and CNNs and improves the final error rate as well. Moreover, Resnet without Batch Normalization does not converge unless we apply Gradient Normal Clipping [113]. By contrast, applying Batch Normalization has little on Similarity Net, neither on the learning speed nor the final accuracy. Compared models without BN converge slower than their BN enabled counterparts.

The above observation is consistent with the analysis in section 3.1. Batch Normalization aims at Reducing Internal Covariate Shift. Ideally, pre-activation normalization is expected to decorrelate the input from the previous layer at each step of Gradient Descend. Batch Normalization trade explicit decorrelation for lighter computational burden. Our approach is neither a complete decorrelation. We initialize the kernels to be possibly uncorrelated and representative, and then let Gradient Descend optimize the network without restriction. Since the output of Cosine Similarity followed by ReLU is always in $[0, 1]$, the Covariate Shift is not as heavy as dot product, which is another reason for fast convergence we suspect.

3.4 Summary

We proposed a framework that unifies Visual Encoding and Convolutional Neural Networks. The equivalence between a single layer of Neural Network and Clustering provides a way of pre-train a Convolutional Neural Network by unsupervised feature learning. Our experiments and analysis demonstrate that such a pre-training schema is crucial to Similarity Net, which also makes the model not rely on Batch Normalization. As a result, all the models showed advanced parameter efficiency compared to the models in similar parameter space. We argue that unsupervised feature learning initiates the Network that closes to a better local minimum, such that we are more likely to find better optimum by the same training method. Finally, unlike general CNN that performs linear classification at each filter, Similarity Net learns a number of kernels that represent the corresponding feature at

each level. For each input region, the convolution output is the similarity measure between input and the filter. This may bring us richer potential to reuse well-learned feature maps.

4. APPLICATION ON STREAM DATA: UNSUPERVISED OBJECT DISCOVERY ON-THE-FLY FROM AERIAL VIDEOS

Unmanned Aerial Vehicles (UAV) have been expected to fly autonomously and obtain videos in hash, hostile and dangerous environments to eliminate risks of life and surveillance workload. Therefore, equipping UAVs with the ability of unsupervised visual knowledge learning became crucial for real-life applications. In this chapter, we propose a real-time, unsupervised vision learning system that discovers interesting ground objects (both moving and stationary) without any pre-training. The system compactly and simultaneously accumulates visual knowledge from the low-level features to several mid-level patches and finally to the object-level detections. We provide an efficient unsupervised on-line learning scheme with provable accuracy guarantees under the Random Rover Assumption. Tracking is considered as a rich resource to learn visual diversity where stably tracked patches are regarded as the same class. For visual attentions, we propose an efficient object proposal method for aerial images, which works together with a bottom-up object realization approach to enable the system to group a few persistently adjacent object proposals into a semantically meaningful object. Unlike other ”passive” unsupervised object discovery approaches, our system provides provable learning confidence guarantees indicating whether a specific territory has been fully scouted and understood, giving valuable feedback to the control system for route optimization. We apply our system to unknown ground object discovery from public aerial video datasets and compare our approach to most related state-of-the-art unsupervised object discovery competitors. Our experiments found that the system proposed in this chapter is clearly more suitable for aerial videos than others with drastically improved efficiency and stream learning ability.

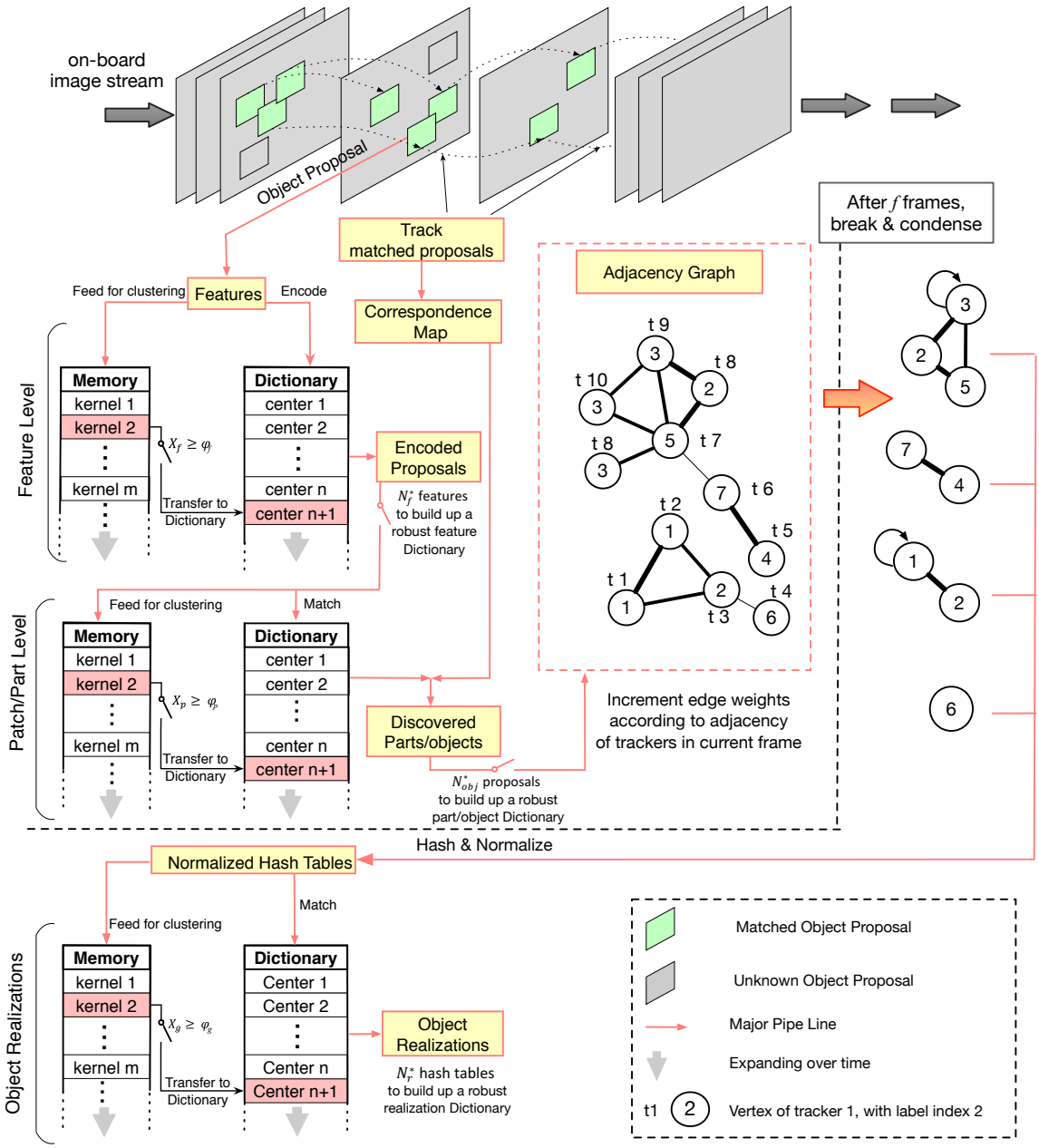

Figure 4.1. System Framework: unsupervised learning and accumulating multi-level visual knowledge on-the-fly

4.1 Framework Overview

As shown in fig.4.1, we perform 4 levels of unsupervised learning simultaneously in real-time: features; parts; patch correspondence under different views; and part-object relationship realization. For each frame, we extract features from (weak) object proposals and feed these features into our streams clustering algorithm for unsupervised feature learning to progressively learn a feature code-book on-the-fly. Patches are encoded into most recent feature space and we manage to learn distinct patch representations while the feature space is enlarging over time. We track each recognized patch and discover correspondence between different views of the same part or object. Finally, we propose a bottom-up approach to group frequently adjacent patches into an object.

Except for patch correspondence learning, all 3 levels of unsupervised learning include object proposal follows the stream clustering framework proposed in this thesis, which enables all levels of learning can be conducted on-the-fly simultaneously. In our algorithm, we provide a series of provable guarantees such that all levels of visual knowledge can be learned robustly within a certain number of frames (how long) from a particular area. Such guarantees provide informative feedback to the control system and ask UAVs to "stop and see" when necessary.

General Assumption. In this chapter, we assume the agent is randomly roving around a targeted area and each object appears randomly (following uniform distribution). Namely, if there are totally n objects in the area, the probability of an object appears in a frame is $\frac{1}{n}$.

4.2 Matching Proposals on-the-fly.

Recall that our stream clustering algorithm only performs two computational operations: match (check if a data point is inside a kernel) and update (eq.2.1 in our case). As long as we define these two operations, our algorithm will work on an arbitrary type of data, which enables us to match object proposals on-the-fly when the feature code-book is enlarging.

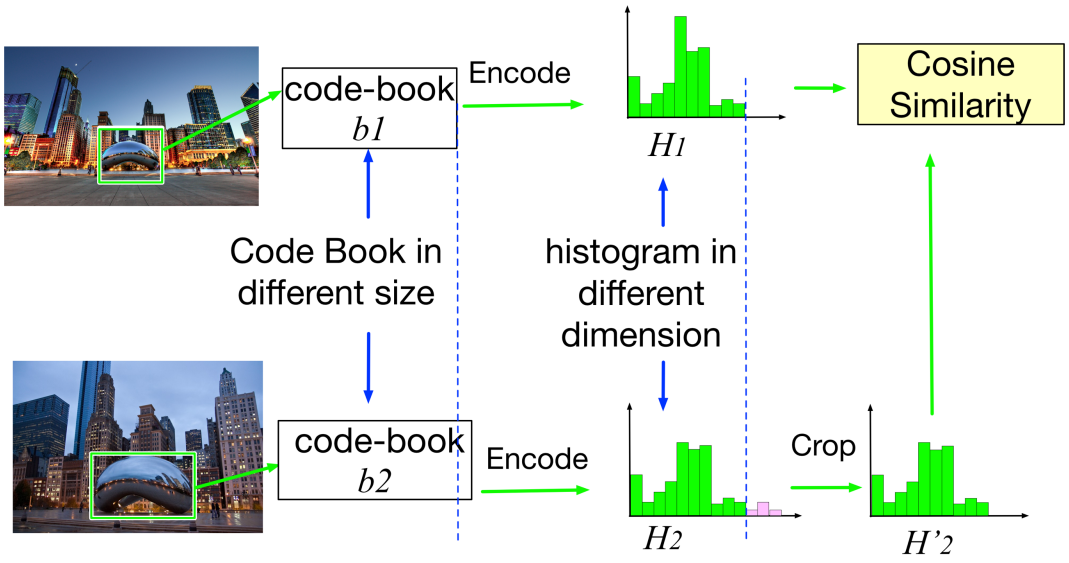

Figure 4.2. Match object proposals on-the-fly while the feature space is increasing over time.

As illustrated in Fig.4.2, let histogram h_i be an encoded proposal with dimension d_i. Since the feature space is enlarging over time, d_i is also enlarging along the data stream. Benefited from the fully progressive property of our clustering algorithm for feature learning, the first d dimension of any two given encoded histograms is always regarding the same features. Given a histogram kernel center c_k with dimension d_k, we update c_k with h_i by:

$$ {}^{(d)}c_k^{(new)} = \begin{cases} \frac{{}^{(d)}c_k n_{kd} + {}^{(d)}h_i}{n_{kd}+1} & \text{if } d \leq d_k \\ {}^{(d)}h_i & \text{if } d > d_k \end{cases} \tag{4.1} $$

where ${}^{(d)}c_k$ and ${}^{(d)}x$ are the d^{th} dimension of c_k and x, respectively. n_{kd} is the number of matched data point on d^{th} dimension of c_k. We set $n_{kd} = 1$ for $d > d_k$.

We define similarity measure as:

$$ S(c_k, h_i) = cos(c_k, h_{id_k}) \tag{4.2} $$

where $cos(.)$ is the cosine similarity and h_{id_k} is the first d_k dimension of h_i. According to eq.4.1, $d_k \leq d_i$ always holds.

By replacing the similarity measure and update method in eq.4.1 and eq.4.2, we can match and learn object proposals as the feature space is enlarging randomly. This is a crucial benefit from fully progressive clustering on feature learning, where learned features are accumulatively maintained and membership assignment is reserved over time. Note that, eq.4.1 degrades to eq.2.1 if $d_k \equiv d_i$.

4.3 Object Proposal and Object-Part Relationship Realization

Our object proposal approach for aerial images follows the observation that objects are more corner-like than background and usually in small size. As shown in fig.4.3, for each box filtered frame, we compute the Harris Map [114]. Those corner-like areas have much higher responses than plain background and texture areas. After thresholding the Harris Map, the highlighted pixels group in a pattern that naturally represents objectness. We cluster these pixels by our stream clustering algorithm proposed in this chapter since it is

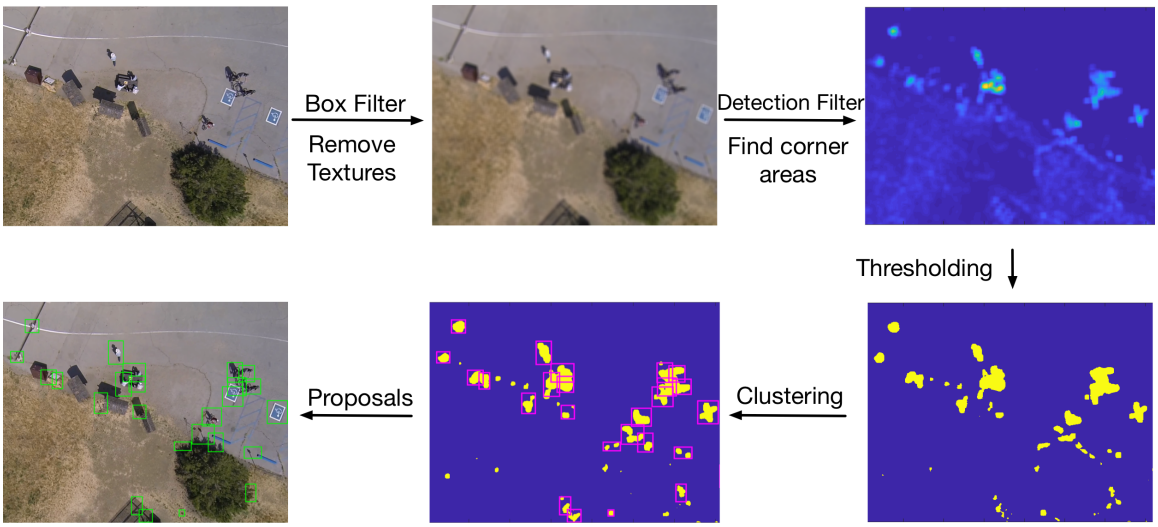

Figure 4.3. Real-time Object Proposal for Aerial Images

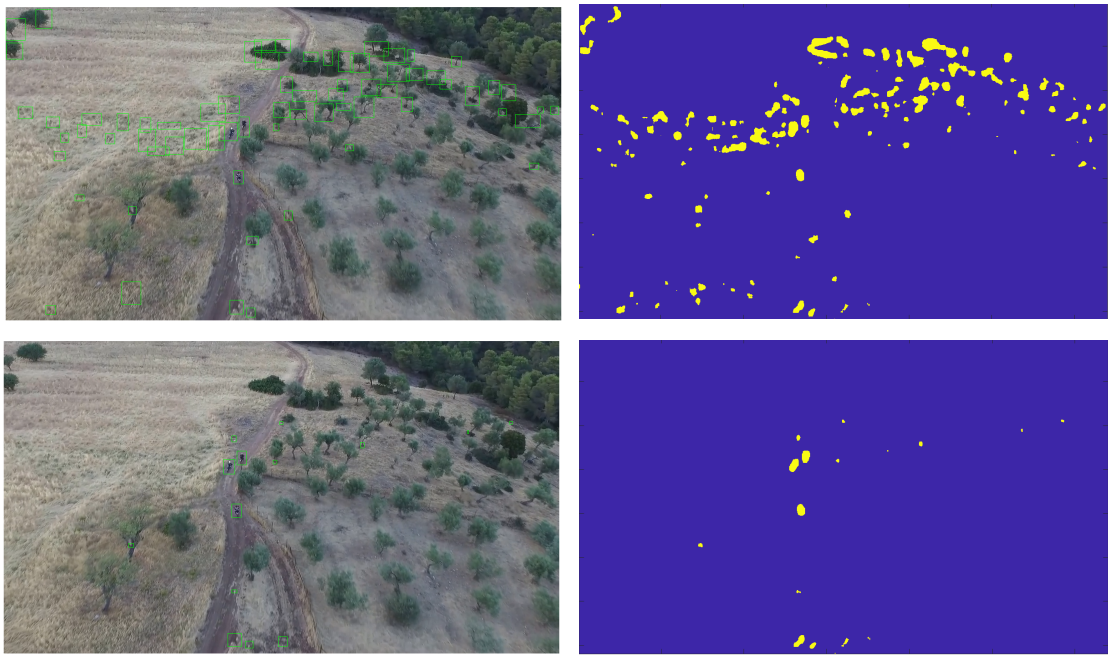

Figure 4.4. Removing known, unexpected green area to general more accurate object proposals. Images in the first line are the object proposals (green box) and thresholded Harris Map. Images in the second line are obtained after filter out tree-like areas.

fast and able to detect the natural number of clusters automatically. In our experiment, processing each image (1280 by 720) merely takes 0.05s on average. As discussed in section 4.1, we then match these proposals on-the-fly and learn a part level representation.

In the clustering algorithm, the similarity threshold θ is assumed to be the expected scale of interesting objects. The obvious issue of the above method is the expected object scale changes along with flying altitude. When scale becomes larger, the object proposal may break into pieces, as shown in the first row of Fig.4.3. And the proposal will be under-fitted if the scale becomes smaller, as shown in the second row of Fig.4.3. To detect scale increasing, we find closely located object proposals (distance smaller than θ) and merge them into a larger proposal. Then we try to match it with discovered objects after re-sizing it into the current scale. We accept the merge if a good match can be found. We keep

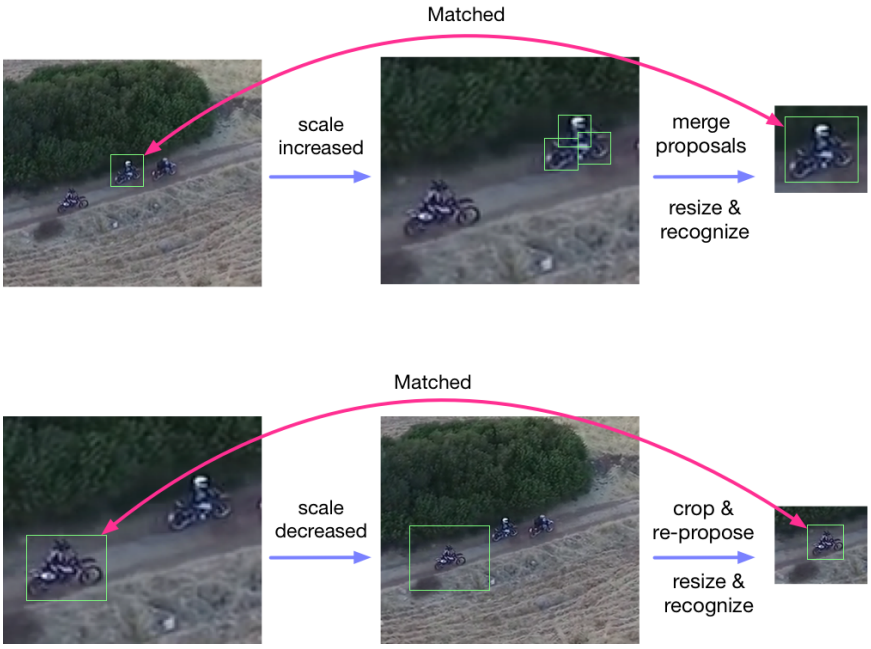

Figure 4.5. Adjusting scales automatically. The first row shows the situation that scale increases when the UAV flies lower. The second row shows that the scale decreases when the flying altitude becomes higher.

accumulating matched numbers until a sufficient number of merges is achieved so that we have strong confidence to increase the value of θ.

To detect scale decreasing, we inspect the difference between the cluster size and θ. We define the size of the cluster as the longer edge l_{max} of its bounding box. If such difference is salient, say $l_{max} \leq 0.5\theta$, we consider l_{max} as its tentative scale and re-encode the area of cluster bounding box. If we find a better match than the original object proposal, we accumulate the number of shrinks and reduce scale if a sufficient shrink number is achieved.

In some scenarios, we have prior knowledge of the deployed environment and we know that some objects or textures are unexpected. To remove those unwanted objects efficiently, such as trees, grass, or brick textures, we can apply a mask on the thresholded Harris Map. As shown in fig.4.4, we remove tree-like areas since trees and grass are not interested.

However, relying on the above intuition sometimes fails to detect the whole piece of large objects such as cars and buildings. In the following of this section, we propose an approach to mine part-object relationships on-the-fly, to enable the machine to "realize" several frequently adjacent patches are actually from the same object.

Parts from the same object usually persist certain adjacency patterns over frames, e.g. car wheels are usually adjacent to doors or windows. As shown in fig.4.6 given f consecutive frames, we construct a graph such that each vertex indicates a tracked object, i.e. a tracker. Initially, there is no edge in the graph. In each frame, if two trackers are adjacent, we increment their edge weight by 1. Thus after parsing f frames, frequently adjacent patches can be easily obtained by breaking edges with smaller weights (we set the threshold as $0.5f$), which returns l connected components $\{g\}_l$. To have a compact and well-generalized representation, we merge all the vertices with the same label index and convert edges between the same label index into a self index. However, confirming a part-object relationship as above is sensitive to f and accidentally neighboring parts. A robust part-object relationship should frequently appear over the image stream, which naturally formulates the clustering stream problem if we regard each g as a data point. We again apply the clustering framework proposed in this chapter by defining similarity measure and update method as follows.

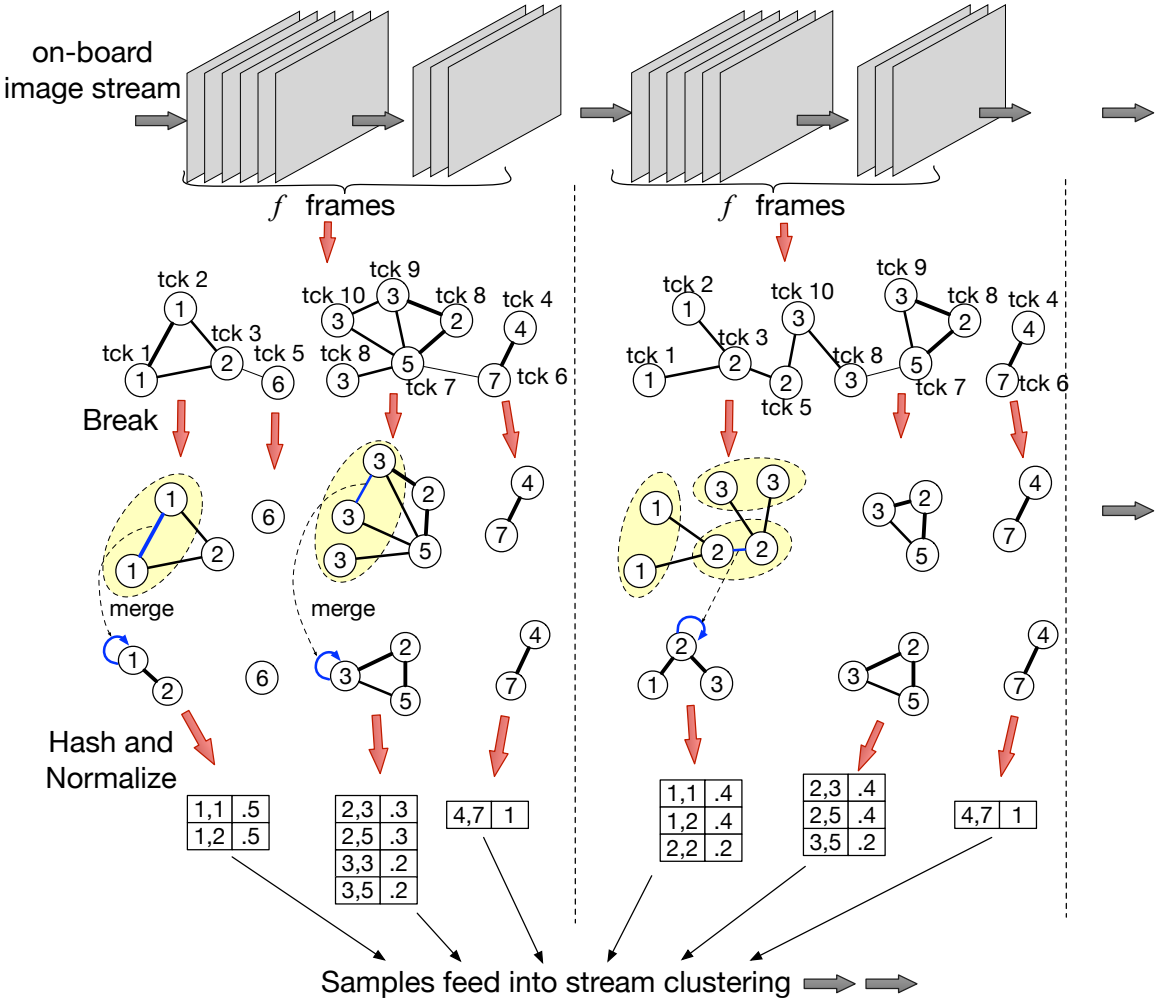

Figure 4.6. Object Realization Process

Since each vertex in g has a unique label index, every edge in g is uniquely defined by the two label indices. Therefore, we can hash all the edges if regard the label pair $\langle p, q\rangle$ of the two vertices as key, hence convert g into an edge hash table t. We normalize t such that

$$\sum_{\langle p,q\rangle \in K} t\langle p, q\rangle = 1 \tag{4.3}$$

where K is the key set and $t\langle p, q\rangle$ is the hash value returned by key $\langle p, q\rangle$, namely edge weight between label p and label q. Note that the hash value in t is always positive. We compute similarity between a normalized edge hash table t and an existing kernel center t_c by:

$$S(t_c, t) = \sum_{\langle p,q\rangle \in K_c \cap K} |t_c\langle p, q\rangle - t\langle p, q\rangle| + \sum_{\langle p,q\rangle \in \bar{K_c} \cap K} t\langle p, q\rangle + \sum_{\langle p,q\rangle \in K_c \cap \bar{K}} t_c\langle p, q\rangle \tag{4.4}$$

The update method is defined as

$$t_c\langle p, q\rangle^{(new)} = \begin{cases} \frac{n_k t_c\langle p,q\rangle + t\langle p,q\rangle}{n_k+1} & \text{if } \langle p, q\rangle \in K_c \cap K \\ \frac{1}{n_k+1} t\langle p, q\rangle & \text{if } \langle p, q\rangle \in \bar{K_c} \cap K \\ \frac{n_k}{n_k+1} t_c\langle p, q\rangle & \text{if } \langle p, q\rangle \notin K_c \cap \bar{K} \end{cases} \tag{4.5}$$

where K and K_c are the set of keys in t and t_c, respectively. n_k is number of tables has been matched to t_c so far.

Note that $0 \leq S(t_c, t) \leq 2$. $S(t_c, t) = 0$ when t_c and t are identical, and $S(t_c, t) = 2$ when $K_c \cap K = \emptyset$. Also, $t_c\langle p, q\rangle^{(new)}$ updated by eq.4.5 is also normalized, since

$$\begin{aligned} \sum_{\langle p,q\rangle \in K_c \cup K} t_c\langle p, q\rangle^{new} =& \frac{1}{n_k+1}\Bigg(\sum_{\langle p,q\rangle \in K_c \cap K} \Big(n_k t_c\langle p, q\rangle + t\langle p, q\rangle \Big) \\ &+ \sum_{\langle p,q\rangle \in \bar{K_c} \cap K} t\langle p, q\rangle + \sum_{\langle p,q\rangle \in K_c \cap \bar{K}} n_k t_c\langle p, q\rangle \Bigg) \\ =& \frac{1}{n_k+1}\Bigg(\sum_{\langle p,q\rangle \in \cap K} t\langle p, q\rangle + n_k \cdot \sum_{\langle p,q\rangle \in \cap K_c} t_c\langle p, q\rangle \Bigg) = 1 \end{aligned} \tag{4.6}$$

The above mapping from graphs to normalized hash tables indeed map the graph into a euclidean space with infinite dimension. The normalized hash table is a sparse representation such that similarity measure and update method of eq.4.4 and eq.4.5 are linear to

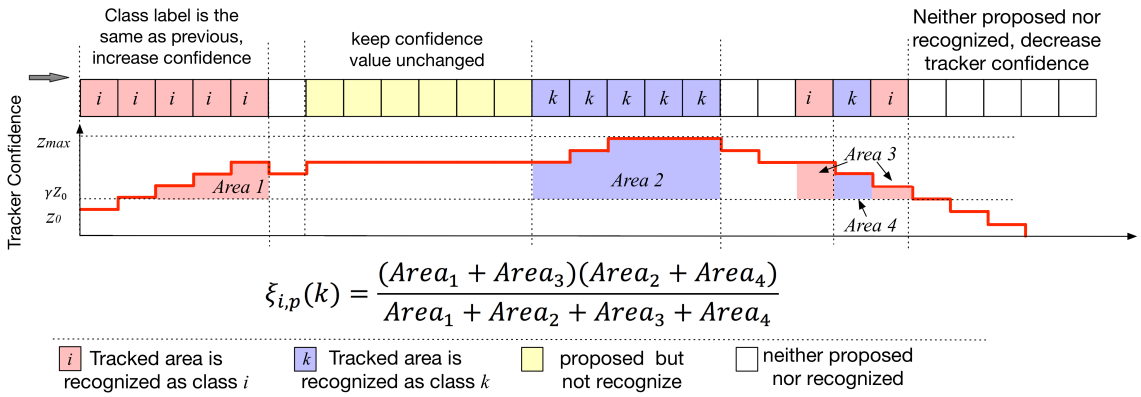

Figure 4.7. Integrate object discovery and tracking for object correspondence inferring.

graph size. Note that, this map is re-convertible, namely given a normalized hash table, there exists and only exists one corresponding graph, and vice versa. And both directions of mapping are linear to graph size and table size.

4.3.1 Correspondence Inferring by Tracking

Tracking provides valuable information for visual diversity, where confidently tracked patches can be regarded as the same class. In our framework, we avoid learning a complex map from feature space to semantic labels, because such a map is too complex to learn in real-time. Instead, we directly relate recognized objects or parts under the same track line thus progressively build a probabilistic model to describe semantic correlations between discovered object classes.

As shown in fig.4.3.1, given a tracker validly tracked for a number of frames. The tracked area is recognized as either one of the K object classes discovered so far, or non-object. A desired semantic correlation instance between class i and class j should be the case that both classes are stably tracked. And we can confirm such a pattern if there are sufficient instances of semantic correlation between class i and j. Therefore, we will consider the following two problems:

- How can we extract a robust correspondence instance from each tracker?
- How can we identify a strong correspondence pattern from all the tracker information collected so far?

Tracker Confidence Curve. To provide the solution to the first problem, we maintain a Tracker Confidence Curve for each tracker. We perform object discovery and recognition once every m frames (as keyframe) and track the recognized patches through these m frames without explicit encoding and recognition. The Tracker Confidence Curve has two values for each keyframe: recognized object class l_t and tracker confidence c_t. The recognized object class can be a null value if the tracked area is not recognized as a known class. The tracker confidence is initialized to z_0 and ranges from 0 to z_{max}. We stop tracking when the tracker confidence decreases down to 0. Given the value of tracker, confidence is initialized or updated at time t, its value is updated at time $t+m$ according to the following 4 situations.

(1) the tracked object is recognized as the same object in frame t. In this case, we increase the confidence value by 1 and re-initialize the tracker. If the confidence value increase to z_{max} we keep the value unchanged.

(2) the tracked object is proposed but not recognized. This circumstance happens usually because view changed while the track is still valid but the current appearance has not been learned yet. Therefore, potential object correspondence may exist along this track line. In this case, we keep tracking and wait for the tracked object proposal to be learned later.

(3) the tracked area is recognized as another object class. This case directly indicates a class correlation. However, any uncertainty may also result in a class transition. Therefore, we decrease the tracker confidence by 1. If the tracked area is recognized as this class again in the next keyframe, the tracker confidence will increase.

(4) the tracked object is neither proposed nor recognized. The tracking may not be valid in this case, but it is also possible that the object proposal method failed due to a change of illumination or local contrast. Therefore, we try to recognize the tracked area.

If the recognition succeeded, we go to either case (1) or (2). Otherwise, we decrease the confidence value of the tracker and keep tracking.

As shown in fig.4.3.1, we can use the area under the curve for each particular object class to describe the stability of tracking. The first operation we take on the Tracker Confidence Curve has translated the curve vertically by $-\gamma z_0$ for better robustness. Namely, we only take confident tracking into account. The coefficient γ can be understood as a sensitivity coefficient. Given a tracker initialized at time t_0 and diminished at time T, we define Object Confidence of class i as:

$$A_i = \sum_{t=t_0}^{T} \sigma\big(l_t, i\big) \cdot max\big(c_t - \gamma z_0, 0\big) \tag{4.7}$$

where $\sigma(l_t, i) = 1$ if $l_t = i$, and $\sigma(l_t, i) = 0$ otherwise, l_t and c_t is the recognized class label and the tracker confidence of time t, respectively. A_i is basically the area under the curve when tracked area is recognized as class i. Obviously, large value of A_i indicates that object class i is stably tracked. Thus we can describe the semantic correspondence between class i and k of tracker p as:

$$\xi_{i,p}(k) = \frac{A_i A_k}{\sum_{j=1}^{K} A_j} \tag{4.8}$$

where K is the total number of classes discovered so far. Eq.4.8 can be understand in two aspects. First, $\xi_{i,p}(k)$ will be a large value if A_i and A_k are both large. And second, $\xi_{i,p}(k)$ will be a large value if A_i and A_k both take large proportion of $\sum_{j=1}^{K} A_j$. Both indicates class i and k are stably tracked.

Modeling the global semantic correspondence over all trackers. Information from one single tracker is not reliable. Given an object class i, we model the semantic correspondence of all other discovered object class by a multinomial distribution, $C_i \sim M(\boldsymbol{\alpha_i})$, where $\boldsymbol{\alpha_i} = [\alpha_{i,1}, ..., \alpha_{i,j}, ..., \alpha_{i,K}]$ is an $K-1$ dimension vector (namely $j \neq i$). Each element represents the correspondence likelihood with respect to each discovered object class expect class i. We convert each tracker as a group of sample from the underlying distribution it follows. For example, $\xi_{i,p}(k) = 3$ is explained as 3 one-hot vectors that taking spikes at k^{th} element.

First we select an object class i as the dominant class such that $\frac{A_i}{\sum_{k=1}^{K} A_k}$ is maximized (Namely the most stably tracked class). The value $\xi_{i,p}(k)$ is regarded as the number of instances that class i and class k is correlated. Note that $\xi_{i,p}(k)$ is not necessarily an integer. We then collect these instances to learn (maximize) a Dirichlet posterior distribution where we consider a conjugate Dirchlet Prior $p(\boldsymbol{\alpha_i}|\boldsymbol{\beta}) = Dir(\boldsymbol{\beta})$:

$$\ln\left(p(\boldsymbol{\alpha_i}|\boldsymbol{D},\boldsymbol{\beta})\right) = \ln\left(C \cdot p(\boldsymbol{D}|\boldsymbol{\alpha_i})p(\boldsymbol{\alpha_i}|\boldsymbol{\beta})\right) \tag{4.9}$$

$$= \sum_{k=1,k\neq i}^{K} (\beta + m_{i,k} - 1)\ln(\alpha_{i,k}) + \ln(C) \tag{4.10}$$

and $\alpha_{i,k}$ subject to

$$\sum_{k=1,k\neq i}^{K} \alpha_{i,k} = 1 \tag{4.11}$$

where C is the normalization coefficient, $\boldsymbol{\beta} = \beta[1, ..., 1]$ (explained later), $p(\boldsymbol{D}|\boldsymbol{\alpha_i})$ is the likelihood, and $m_{i,k} = \sum_p \xi_{i,p}(k)$. The solution to eq.4.9 with restriction rq.4.11 is

$$\alpha_{i,k} = \frac{\beta + m_{i,k} - 1}{(K-1)(\beta-1) + \sum_{k=1,k\neq i}^{K} m_{i,k}} \tag{4.12}$$

For each discovered object class i, the semantic correspondence vector $\boldsymbol{\alpha_i}$ is updated along the stream. Each element $\alpha_{i,j}$ indicates the learned correspondence strength between i and j. Therefore we can use the entropy of $\boldsymbol{\alpha_i}$ to discover whether a correspondence pattern emerges. Entropy is defined as

$$E(\boldsymbol{\alpha_i}) = -\sum_{k=1,k\neq i}^{K} \alpha_{i,k}\ln(\alpha_{i,k}) \tag{4.13}$$

which is not normalized with respect to the value of K (number of object classes so far except class i). Therefore, we normalize it by the maximized entropy given the value of K, where each of the $K-1$ elements of $\boldsymbol{\alpha_i}$ take $\frac{1}{K-1}$, namely

$$E_{norm}(\boldsymbol{\alpha_i}) = \frac{E(\boldsymbol{\alpha_i})}{-\sum_{k=1,k\neq i}^{K} \frac{1}{K-1}\ln(\frac{1}{K-1})} = \frac{E(\boldsymbol{\alpha_i})}{\ln(K-1)} \tag{4.14}$$

$E_{norm}(\boldsymbol{\alpha_i})$ ranges from 0 to 1, where smaller value indicates that some elements are significantly larger than others. We select 0.5 as the threshold for discovering a correspondence pattern, which can be understood as a situation that there are $\sqrt{K-1}$ elements take

value $\frac{1}{\sqrt{K-1}}$ and the rest elements are infinitely small. Consider eq.4.12, $m_{i,k}$ is initially 0, thus $E_{norm}(\boldsymbol{\alpha_i}) = 1$. When $E_{norm}(\boldsymbol{\alpha_i})$ satisfies the threshold ($E_{norm}(\boldsymbol{\alpha_i}) = 0.5$), at least one element is significantly larger than β, which indicates that correspondence pattern is confidently discovered from previous trackers. We then perform non-maximum suppression over all the elements of $\boldsymbol{\alpha_i}$ and update the correspondence strength between class i and the class (say j) that the max element of $\boldsymbol{\alpha_i}$ represents. We then update the correspondence strength between class i and j on a undirected graph, where each vertex represents a discovered object class and each edge represents the semantic correspondence. This graph initially possesses no edge and grows along with the video stream. The final semantic correspondence is discovered by finding connected components of the graph.

4.4 Robustness Analysis

In this section, we show how our stream clustering algorithm provides guarantees for robust knowledge accumulation. Consider a series of on-board image streams when a UAV flying randomly around a targeted area. Our problem is, how long does it take to build robust visual knowledge to recognize all the objects in the current area and make sure that this territory is fully scouted and discovered. We assume that the agent is flying randomly and each object is evenly captured in the image stream.

We first consider, for a given minimum density ratio of ρ, how many data points it needs for building a dictionary that every density peak has at least one kernel inside its sufficient area. In such a situation, we can regard the clustering is "almost done". Consider a substream of size N^* in the data stream formed by the first N^* data points. We hope that by processing only N^* samples, our algorithm will find a sufficiently dense kernel for every valid density peak.

Given a kernel k in a dense area covers N_k data points, with $\frac{N_k}{N} = \rho_k > \rho$. Let N_k^* be the number of instances, among the first N^* random samples, covered by kernel k. Then N_k^* follows Binomial Distribution $N_k^* \sim B(N^*, \rho_k)$. Using again the Central Limit Theorem, let $M = (N_k^* - N^*\rho_k)/\sqrt{N^*\rho_k q_k}$, where $q_k = 1 - \rho_k$ and M follows Standard

Normal distribution, $M \sim \mathcal{N}(0,1)$. We hope that in the first N^* random samples, the instances covered by kernel k are more than 0.9 times of the expectation, $N_k^* > 0.9N^*\rho_k$; the probability is,

$$\begin{aligned} & P\left\{N_k^* > 0.9N^*\rho_k\right\} \\ & = P\left\{\frac{N_k^* - N^*\rho_k}{\sqrt{N^*\rho_k q_k}} > \frac{0.9N^*\rho_k - N^*\rho_k}{\sqrt{N^*\rho_k q_k}}\right\} \\ & = P\left\{M > -\frac{0.1N^*\rho_k}{\sqrt{N^*\rho_k q_k}}\right\} = \Phi\left(\frac{0.1N^*\rho_k}{\sqrt{N^*\rho_k q_k}}\right), \end{aligned} \tag{4.15}$$

We expect $P\left\{N_k^* > 0.9N^*\rho_k\right\} > 1-\alpha$; if $\alpha = 0.01$, then from eq. (4.15), we have,

$$N^* > \Phi^{-1}(1-\alpha)\frac{1-\rho_k}{\rho_k} > C\frac{1-\rho}{\rho} \tag{4.16}$$

Where $C = 100\Phi^{-1}(1-\alpha)^2$ is a constant number. According to above analysis, given a minimum density ratio of feature learning ρ_1 and part level learning ρ_2, we need at least $N_f^* = C\frac{1-\rho_1}{\rho_1}$ features to build a stable feature code-book and at least $N_p^* = C\frac{1-\rho_2}{\rho_2}$ proposals of object proposals for learning a stable mid-level representation. In object realization, we require N_{obj} normalized hash table to discovery robust part-object relationships. Let R_p be the average number of object proposals per frame (indicating the image diversity and richness) and R_f be the number of features extracted from an object proposal. We need to process at least $\frac{N_f^*}{R_p R_f}$ frames for a confident feature learning, and $\frac{N_p^*}{R_p}$ frames for learning a robust part-level representation. Let R_o be the average number of graphs of each f consecutive frames, then the minimum number of frames for the robust discovery of existing part-object relationship is $\frac{N_{obj}^* f}{R_o}$. Therefore, the minimum number of frames F to fully discover a territory is

$$F = \frac{N_f^*}{R_p R_f} + \frac{N_p^*}{R_p} + \frac{N_{obj}^* f}{R_o} \tag{4.17}$$

4.5 Experiments

4.5.1 Experiment Setting

In this section we evaluate our approach by applying unsupervised object discovery on 3 aerial datasets: UCF Aerial Action Dataset [1], UCLA Aerial Event Dataset [81] and Okutama Action Dataset [115]. All 61 videos from these 3 datasets are taken by flying UAVs in urban areas with sufficiently background complexity and visual diversity. UCLA Event Dataset contains 27 videos taken in two sites. The original research on this dataset is to recognize action and events from these videos. Annotations are made in stabilized videos where more than half of a frame is blank. We run our approach on the original videos with strong turbulence and map the detected bounding box to stabilized frames. UCF Aerial Action Dataset contains 3 videos and is developed to train and validate models that recognize gestures and actions from aerial videos. Okutama Action Dataset is consists of 31 videos taken by two UAVs with high-resolution cameras (3840×2160). Only people are annotated in this dataset.

In our application on on-board videos, it is unnecessary to fully process every frame. Therefore, we perform all 4 levels of learning once every 6 frames and track all recognized bounding boxes across 5 frames between keyframes. In our experiments, the unsupervised learning speed is 5-10 fps, thus the total learning time is the same as the video time.

Since there is no existing work on real-time unsupervised object discovery on aerial videos, we select most related 4 recent state-of-the-art methods on fully unsupervised discovery: [82], [83], [28] and [84]. [82] aims to discover representative mid-level patches and does not rely on a general object proposal. [83] fit object discovery into multiple instances learning thus propose a method for locating foreground objects from massive object proposals. [28] and [84] use Hough Matching to heat up frequently matched regions. All these methods are iterative and designed for non-aerial images/videos.

[1] publlically available on `http://crcv.ucf.edu/data/UCF_Aerial_Action.php`

4.5.2 Parameters and Features

We choose 1 keyframe from every 6 frames for all experiments. We re-size each frame into 1280×720 for all the videos with higher resolution. The neighborhood size θ of clustering for object proposal is 25, and the minimum density ratio ρ is 0.02. Features extracted from each object proposal is in size of 6×6 with stride 4. Features are 128 dimensions of normalized SIFT feature [18] appended with 3 dimensions of normalized RGB colors. In feature learning, we set $\theta = 0.5$ and minimum density ratio $\rho = 0.01$. In part/object level learning, we set the similarity threshold to 0.7, where the similarity between two encoded patches is measured by cosine similarity. We use a naive optical flow tracker and use 20 key points in each tracker. In object realization, we use $f = 20$ and set neighbourhood size $\theta = 1$ and minimum density ratio $\rho = 0.05$.

4.5.3 Evaluation

Ground truth of dataset considered in this chapter provide annotations for objects, e.g. people, cars, shelter, etc. Following [82], we consider two evaluation metrics: purity and coverage. Purity measures if discovered and matched patches are sufficiently distinct from each other and converge measures if interested objects are well discovered. In this chapter, we consider Intersection over Union (IoU) of 50% as a pairing threshold. Specifically, for each bounding box b^* discovered by a given approach, we pair b^* with a ground truth bounding box b if $IoU(b^*, b) \geq 50\%$. Each discovered box has a discovered class label l^* and a paired ground truth label l ($l = -1$ if it is not paired).

For a discovered object class l_0^*, the purity is:

$$Purity = \frac{||\{b^*|l^* = l_0^*, l = k\}||}{||\{b^*|l^* = l_0^*\}||} \tag{4.18}$$

and

$$k = argmax_i||\{b^*|l^* = l_0^*, l = i\}|| \tag{4.19}$$

where $||.||$ is the number of elements of a set. We define discovered object class l_0^* is *matched* to ground truth class k if $k \geq 0$ in eq.4.19. The overall Purity is the average Purity of each discovered class.

Coverage is computed oppositely. For each ground truth bounding box b with label l, we pair a discovered bounding box with class label l^d to it if the previous IOU threshold is satisfied.

Then the coverage of a ground truth class k is computed as:

$$Coverage = \frac{||\{b|l = k, l^d \in M\}||}{||\{b|l = k\}||} \tag{4.20}$$

where M is the set of discovered class labels that are *matched* to ground-truth class k.

We also report the learning speed of all compared methods running on 3.6 GHz i-7 using a single core.

4.5.4 Major Results: Aerial Videos

Table 4.2 and 4.4 list purity and coverage in each video from all 3 datasets and average value of each dataset. Videos in Okutama Action Dataset is taken by 2 drones at different times (morning and noon). We group videos taken by the same drone at the same time. Namely, 1.1, 1.2, 1.3 and 1.4 in table4.2 represents drone 1 morning, drone 1 noon, drone 2 morning and drone 2 noon, respectively. We report both the final results and on-line results. On-line results are image patches matched and clustered on-the-fly and final results are obtained by applying the learned model to the video. Since no object can be learned immediately, the system can not recognize the emerging object during the first few frames. As a result, on-line purity and coverage are lower than the final results.

We found [82] showed slightly lower purity and lower coverage on these video datasets. The most probable reason is patches are extracted randomly from all frames, unlike other methods that use object proposals to extract more robust and meaningful candidate patches. Since the difference between neighboring frames is very small, a proposed bounding box of a certain object or part is very likely to be proposed again in the following frames.

Table 4.1.
Purity (%) of classes matched to ground truth

Video	[82]	[83]	[28]	[84]	Ours final	Ours on-line
UCLA 10	82.78	83.51	87.34	81.05	88.76	78.56
UCLA 11	79.01	81.63	81.14	84.58	82.22	77.22
UCLA 12	77.79	80.68	83.59	87.41	86.68	79.30
UCLA 13	74.03	84.03	80.95	86.37	88.97	79.53
UCLA 14	83.02	77.67	87.30	83.60	89.10	79.78
UCLA 15	75.52	81.11	85.24	87.27	89.95	77.56
UCLA 16	76.68	86.38	87.25	85.96	87.35	79.21
UCLA 17	75.93	86.69	82.38	82.87	88.56	76.63
UCLA 18	75.53	84.81	85.71	87.66	83.14	78.50
UCLA 19	74.86	84.29	82.35	81.34	84.83	78.80
UCLA 20	81.91	77.08	79.52	86.12	89.84	76.34
UCLA 56	74.61	83.80	86.34	84.10	89.35	78.12
UCLA 57	80.31	82.68	80.29	86.79	83.55	79.55
UCLA 58	77.16	81.94	83.99	86.63	81.04	77.05
UCLA 59	83.59	84.66	83.05	89.44	88.09	76.94
UCLA 60	78.99	84.57	81.50	88.23	84.65	79.36
UCLA 62	81.39	83.52	88.43	82.46	89.95	77.98
UCLA 63	74.13	81.91	83.81	87.85	84.99	76.61
UCLA 64	80.05	85.43	87.81	82.14	81.54	76.92
UCLA 65	81.63	81.93	81.90	84.96	83.09	78.63
UCLA 68	77.90	80.98	81.81	87.56	87.36	78.25
UCLA 69	74.80	85.36	82.77	88.92	82.65	77.17
UCLA 70	79.76	83.21	80.14	85.27	89.81	78.49
UCLA 71	82.07	77.03	88.65	81.33	87.25	78.86
UCLA 72	75.29	84.70	84.99	87.29	83.35	77.12
UCLA 73	80.55	81.77	79.26	82.54	81.43	77.65
UCLA 74	77.00	86.79	80.55	81.25	83.45	77.45

Table 4.2.
Purity (%) of classes matched to ground truth

Video	[82]	[83]	[28]	[84]	Ours final	Ours on-line
Average	78.38	82.89	83.63	85.22	**85.96**	78.06
UCF 1	88.59	91.64	91.99	91.51	92.10	85.67
UCF 2	88.48	90.80	90.65	92.93	91.36	87.17
UCF 3	89.86	91.50	93.21	91.96	92.66	85.10
Average	88.98	91.32	91.95	**92.13**	92.04	85.98
Okutama1.1	83.97	89.14	90.77	91.67	91.28	80.47
Okutama1.2	82.95	88.33	90.61	92.18	90.47	78.07
Okutama2.1	84.99	87.62	88.50	91.51	91.81	79.54
Okutama2.2	88.55	89.93	89.08	89.19	90.59	78.49
Average	85.12	88.75	89.74	**91.14**	91.04	79.14

Table 4.3.
Coverage (%) of classes matched to ground truth

Video	[82]	[83]	[28]	[84]	Ours final	Ours on-line
UCLA 10	41.36	65.33	63.27	63.67	68.91	39.76
UCLA 11	47.55	59.23	66.39	62.40	61.09	44.11
UCLA 12	42.26	59.17	64.78	62.49	67.98	44.57
UCLA 13	44.80	64.30	62.35	65.92	74.17	41.74
UCLA 14	41.97	59.98	60.98	62.28	72.32	43.74
UCLA 15	45.41	58.23	63.70	66.64	72.21	41.81
UCLA 16	45.19	65.46	61.41	66.55	66.71	41.60
UCLA 17	46.61	63.85	58.33	63.46	69.71	46.91
UCLA 18	42.36	65.93	61.69	64.51	69.38	40.32
UCLA 19	47.39	66.10	61.39	62.50	71.28	46.00
UCLA 20	45.21	59.62	62.03	60.00	68.52	39.99
UCLA 56	38.41	58.07	62.48	66.99	67.69	43.86
UCLA 57	42.28	65.45	59.07	63.87	72.63	44.89
UCLA 58	44.47	61.37	63.22	64.21	66.56	40.83
UCLA 59	40.68	64.95	60.95	66.40	72.72	40.01
UCLA 60	44.80	63.84	62.82	65.12	65.21	41.02
UCLA 62	39.83	62.77	61.57	67.88	71.11	40.25
UCLA 63	44.58	62.87	61.47	62.70	70.98	40.35
UCLA 64	40.79	62.69	60.98	68.44	73.80	40.75
UCLA 65	47.17	65.35	62.21	67.40	72.98	42.48
UCLA 68	40.57	63.87	63.32	61.26	68.80	45.66
UCLA 69	46.86	65.72	64.51	60.06	68.24	45.29
UCLA 70	47.20	60.41	61.56	61.82	69.67	39.12
UCLA 71	41.00	61.96	59.36	60.85	71.69	42.24
UCLA 72	40.56	60.07	62.32	60.65	71.57	41.88
UCLA 73	39.32	58.87	62.22	60.85	72.08	48.51
UCLA 74	38.23	66.29	61.90	63.93	70.68	44.32
Average	43.22	62.66	62.09	63.81	**69.95**	42.67

Table 4.4.
Coverage (%) of classes matched to ground truth

Video	[82]	[83]	[28]	[84]	Ours final	Ours on-line
UCF 1	48.31	75.62	70.25	76.91	81.87	58.61
UCF 2	52.93	69.76	70.37	72.17	79.55	53.61
UCF 3	53.35	72.32	69.13	74.94	79.57	59.83
Average	51.53	72.57	69.92	74.68	**80.33**	57.35
Okutama	68.16	78.69	79.35	79.13	89.25	73.78
Okutama	68.39	77.64	79.27	78.89	88.34	75.17
Okutama	67.48	79.19	80.45	81.36	85.40	74.70
Okutama	69.47	78.99	80.70	82.38	89.58	81.22
Average	68.37	78.63	79.94	80.44	**88.14**	76.22

Therefore, for randomly generated patches, a whole piece of object in one frame may be matched to a part of the same object in other frames, which makes the encoded image patches much noisier.

[83], [28], [84] and our method all showed comparable purity even though the three compared approaches involve more complex encoding method. This phenomenon happens because aerial images process less visual diversity than general indoor or outdoor images. However, our method clearly outperformed other compared approaches in coverage. The major reason is the object proposal method used in these approaches are not suitable for aerial images. Therefore, though proposed bounding boxes are accurately matched, many interesting targets are not well localized (usually contained in a larger bounding box).

We find our approach achieved clearly the highest coverage compared with all other systems. Our approach provides more accurate and cleaner object proposals for aerial videos, where most objects in aerial images are efficiently localized. Our object realization method avoids separating a large object into pieces. Since our objective is to discover the existence and geometric location of interesting targets, it is unnecessary to cover each object in every frame. If an object has been covered by 70%, we can be very confident to confirm its existence and location. Illumination change and turbulence may make a few frames blurry and detection will be very challenging in these frames. This can be partially fixed by using tracking information but with a limited degree.

Running time (second per frame) of each approach is shown in table 4.5. Time reported is used for processing keyframes only. Our approach is able to process at least 4 frames in one second using a single core. The speed can be doubled if object proposal and learning are implemented into two threads using 2 cores. However, we do not assume a drone is equipped with a multi-core processor. Our method is the only one that performs learning on-the-fly that passes the video only once.

Fig.4.8 shows two example of object realization: our method group persistently adjacent object class into a larger object. Since our bottom-up method does not rely on the overlap between saliently matched object proposals, we can discover the part-object relationships efficiently. Approaches based on overlapping and saliency has to encode and

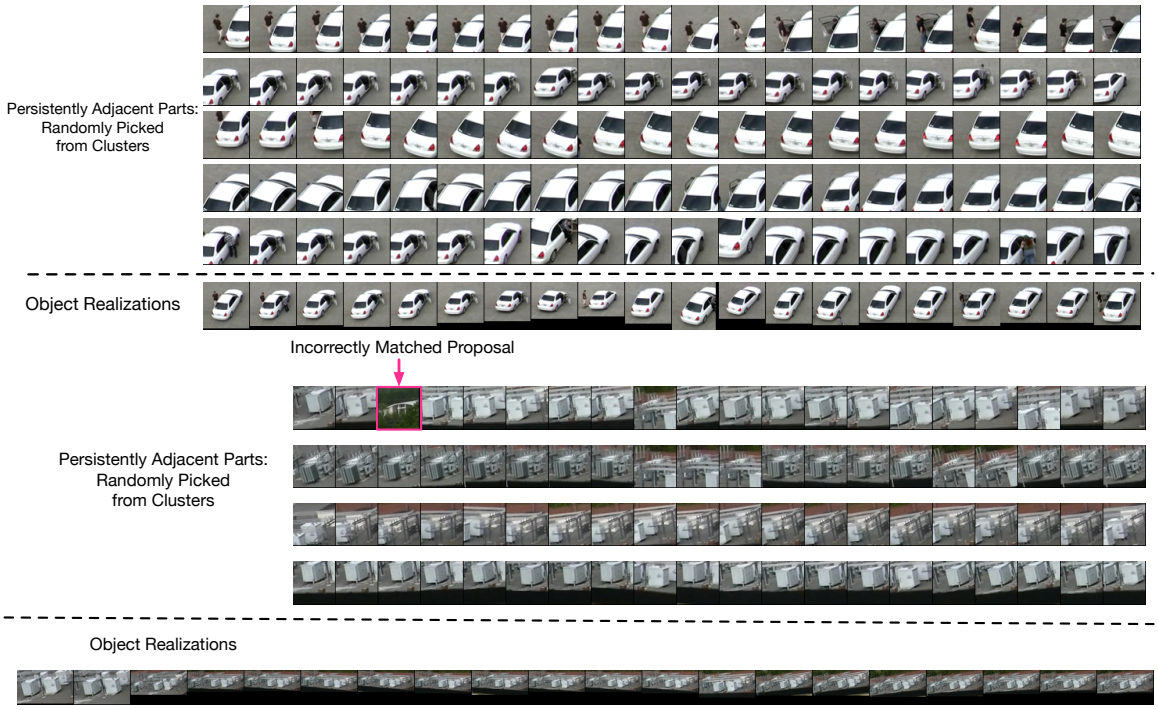

Figure 4.8. Examples of object realizations. Our method groups persistently adjacent parts into a larger object. Patches are possibly incorrectly assigned (red box). However, these noise patches are cleaned up by the object realization process if such patches are not adjacent to the main object.

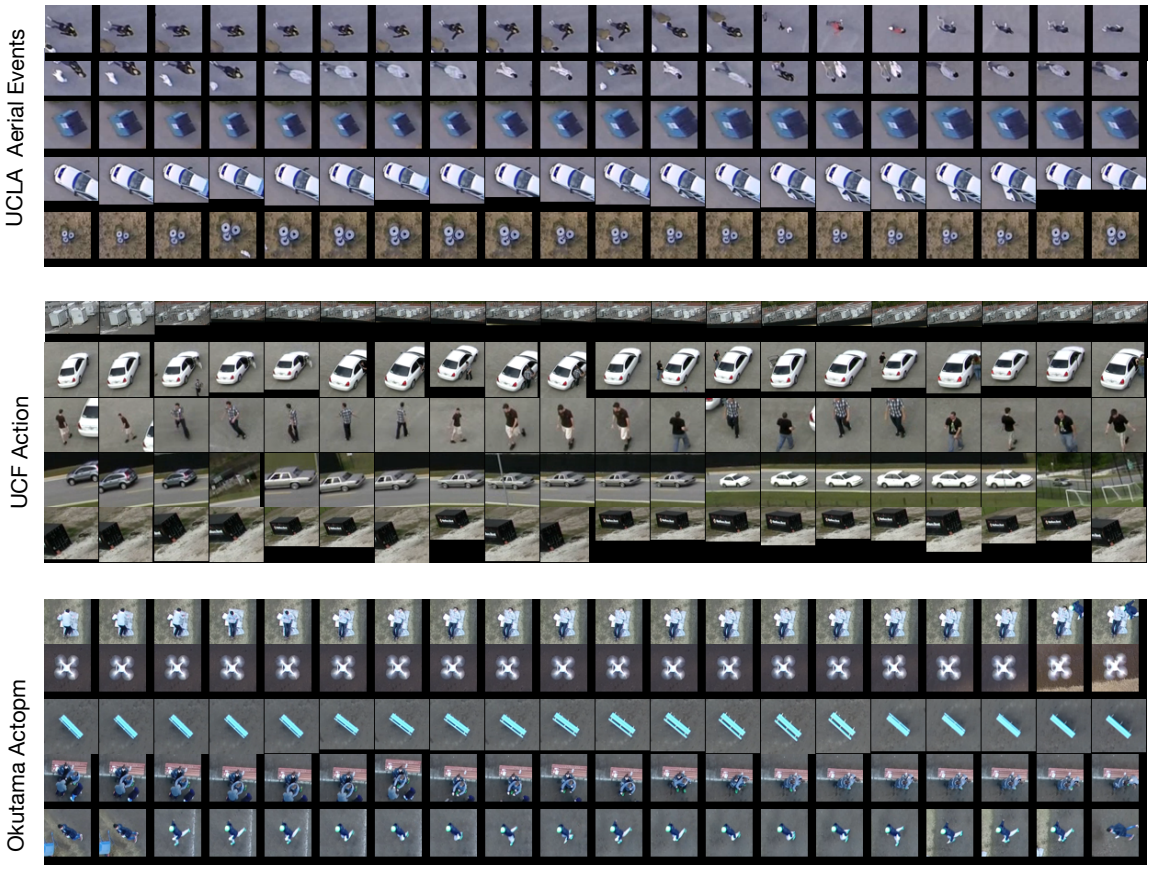

Figure 4.9. 5 most salient objects discovered from a video of each dataset. Each image patch represents a discovered instanced. Patches of each object are randomly selected from all the matched instances across the entire video.

match the same region many times to extract it out from the background, therefore much of the encoding and matching computation is wasted. Another interesting finding is that our object realization method can be used to clean up incorrect matches. As shown in fig.4.8, one noisy patch has been matched to a part of the transformer, however, this patch is not adjacent to any part of the transformer. Therefore, even if incorrectly assigned patches exist in some clusters, as long as such patches are not adjacent to any part of the main object, they will not be merged into the final object realization.

Fig.4.9 shows most 5 salient objects we discovered from each datasets. We randomly pick 20 patches of each object class and plot horizontally in the figure. As we can see from

Table 4.5.
Running Time (second per frame)

Approach	[82]	[83]	[28]	[84]	Ours
UCLA Aerial	4.37	8.42	7.35	14.14	**0.17**
UCF Action	3.92	7.71	7.06	12.97	**0.14**
Okutama Action	5.34	9.80	8.84	17.34	**0.22**

fig.4.9, we generate clean and semantically meaningful object patches from all videos. We found our method not only discovers objects annotated in the ground truth but also ground objects may be interested in other applications.

4.5.5 Non-Aerial Results: Indoor Videos

It is attractive to enable a vision system to understand highly complex, diverse and occluded non-aerial video to lean a well-customized model for specific applications. Recent light and personalized visual applications based on mobile devices give rise to unsupervised object discovery on-the-fly. In this section, we conduct preliminary examinations on indoor videos in a realist office scenario to explore the possibility of further development into a general unsupervised vision system running on mobile devices.

The appearance of objects in ground videos changes more drastically than aerial videos when viewing and distance changes. Unlike aerial videos, objects can be in arbitrary size and not necessarily possess corner-like features. Therefore, we replace the object proposal method with an off-the-shelf algorithm [87] developed for the general object proposal. We still enforce the system to learn features, parts, and correspondence simultaneously. However, the bottom-up object realization is not suitable as we abandon our aerial object proposal method. Therefore, the objective of this research is to test if our method is able to learn well distinct and semantically meaningful parts or objects on-the-fly with very

Figure 4.10. Sample frames of the indoor video taken in an office

limited computational resources (insufficient memory to save the entire video and single-core computation).

Fig.4.10 shows a few frames of our application area. This is a typical office with a realistically messy arrangement to simulate the scenario that related applications may encounter. We take the video with a smartphone and walking around the office for 3 minutes. We visualize in fig.4.11 the top 15 salient object classes discovered by our system from this office. As we can see in fig.4.11, these discovered object classes are all semantically meaningful and represents the key objects in an office, e.g. chairs, desks, monitors, laptops, people. As shown in the last image in fig.4.10, we take some frames outside the window to simulate the environment changing. Our system successfully discovered these emerging object classes such as buildings, windows, and roads. The experiments indicate that our approach built a basic visual understanding of the office within only 3 minutes by simply walking and looking around. The limitation of our system is the object is not well localized which is beyond the focus of this chapter but it is a future work to accomplish.

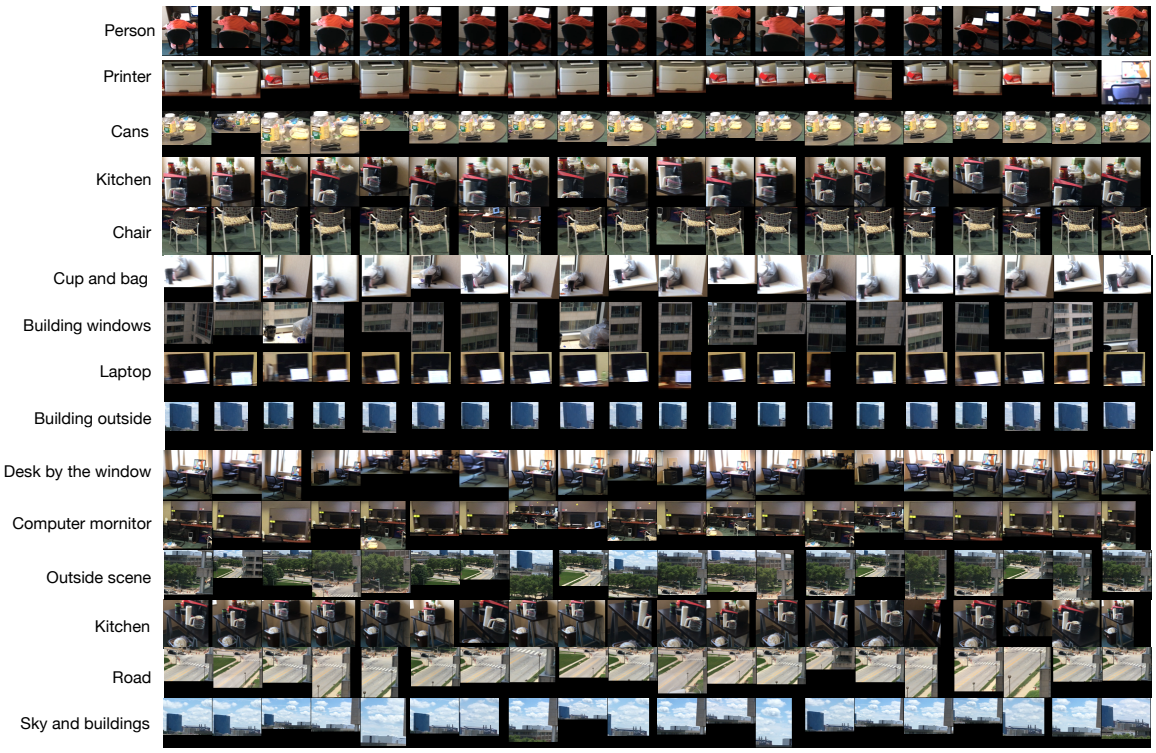

Figure 4.11. The top 15 salient object classes discovered on-the-fly in the indoor area.

Figure 4.12. Sample frames of the outdoor video taken in an grass field

4.5.6 Non-Aerial Videos: Outdoor Videos

We also apply our method to outdoor videos. Unlike indoor videos, outdoor videos are more difficult because of severe illumination change. For example, an object under the sunlight may look different from the ones under the shadow. In this experiment, we fly a quadrotor at a low altitude to capture non-aerial videos. The quadrotor was flying in a random direction around the environment to reproduce the randomness assumptions in this application. Fig.4.12 shows 6 representative frames of the area which includes all the salient objects, e.g. buildings, boxes.

We present the discovered objects in Fig.4.13. As the experiments in indoor videos, we use EdgeBox as our object proposal method. The results indicate that our approach built a basic visual understanding of the field within only 6 minutes by simply walking and looking around, processing more than 10000 frames. Though in such a short running period, we still detect and accurately distinguish most salient objects in the environment (boxes, desks, different buildings, and trees.). Similar to indoor experiments, the limitation of our system is the objects are not well localized, which may be fixed by involving much more computation and supervision by building complex models.

4.6 Summary

In this chapter, we proposed and evaluated a fully unsupervised vision system that builds visual knowledge on-the-fly from aerial videos. The system is computationally efficient and requires no pre-training, which ideally fit hash applications such as military scouting and post-disaster rescue, as well as light applications on mobile devices. It worth noting that our approach is not modularized as a pipeline. Instead, each part of our method is carefully designed to work seamlessly with each other. Our experiments show that the approach proposed in this chapter is fully functioning on the application we focus on. With reasonable coverage, an agent is able to discover the existence of a certain ground object and locate geometrically. The learned object classes can be easily visualized by a few sample patches so that people at the back end can efficiently recognize and take action.

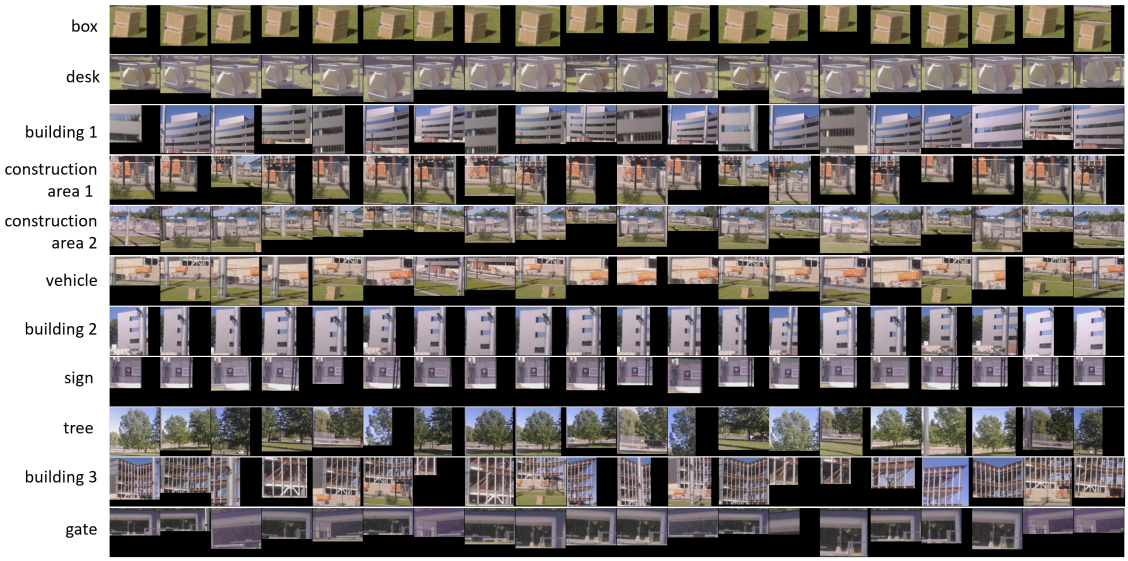

Figure 4.13. The top 10 salient object classes discovered on-the-fly in the outdoor area.

The comparison with most related unsupervised object discovery approaches shows that our method is the most suitable for aerial imagery and the only solution to unsupervised ground object discovery in real-time.

5. APPLICATION ON STATIC DATA: PRE-TRAINING THE CNN BY UNSUPERVISED DEEP ENCODING

In this chapter, we apply Unsupervised Deep Encoding to pre-training Deep CNNs such that we can achieve better learning dynamics, performance and memory efficiency when training a Deep CNN. We test our model on Bench Mark Dataset and compare it with state-of-the-art models in similar parameter space. Note that our model is free from Batch Normalizations, which means at least 50% memory will be saved under the same architectures. Indeed, with a similar amount of parameters, we achieved a lower error rate in general.

5.1 Network Architectures and Implementation Details

In this section, we propose a hybrid architecture with our similarity-based Convolutional Layers and Resnet blocks, as shown in fig.5.1. We test three architectures for ILSVRC2018 [111], from shallow to deep. The first architecture stacks 4 similarity layers, each of which followed by a maxpooling layer. We then add a Resnet Block onto the last maxpooling layer. Standard average pooling and fully connected layer follow the Resnet Block. We find the final Resnet Block increases training speed. The second architecture is similar to the first with the difference that a Resnet block is inserted after each maxpooling layer, which results in 13 layers in total. At the same time, to guarantee information flows majorly through pre-trained similarity layers in early stages of training, we initialize the weights of Resnet blocks from a small interval $(-10^{-5}, 10^{-5})$). One advantage of this design is that we do not need an auxiliary projection shortcut used in [10].

We also explore a deep architecture consisting of 33 layers, with multiple Restnet Blocks inserted after each maxpooling. In this architecture, we add additional shortcuts between each similarity Convolutional Layers, bypassing all the Resnet layers in between,

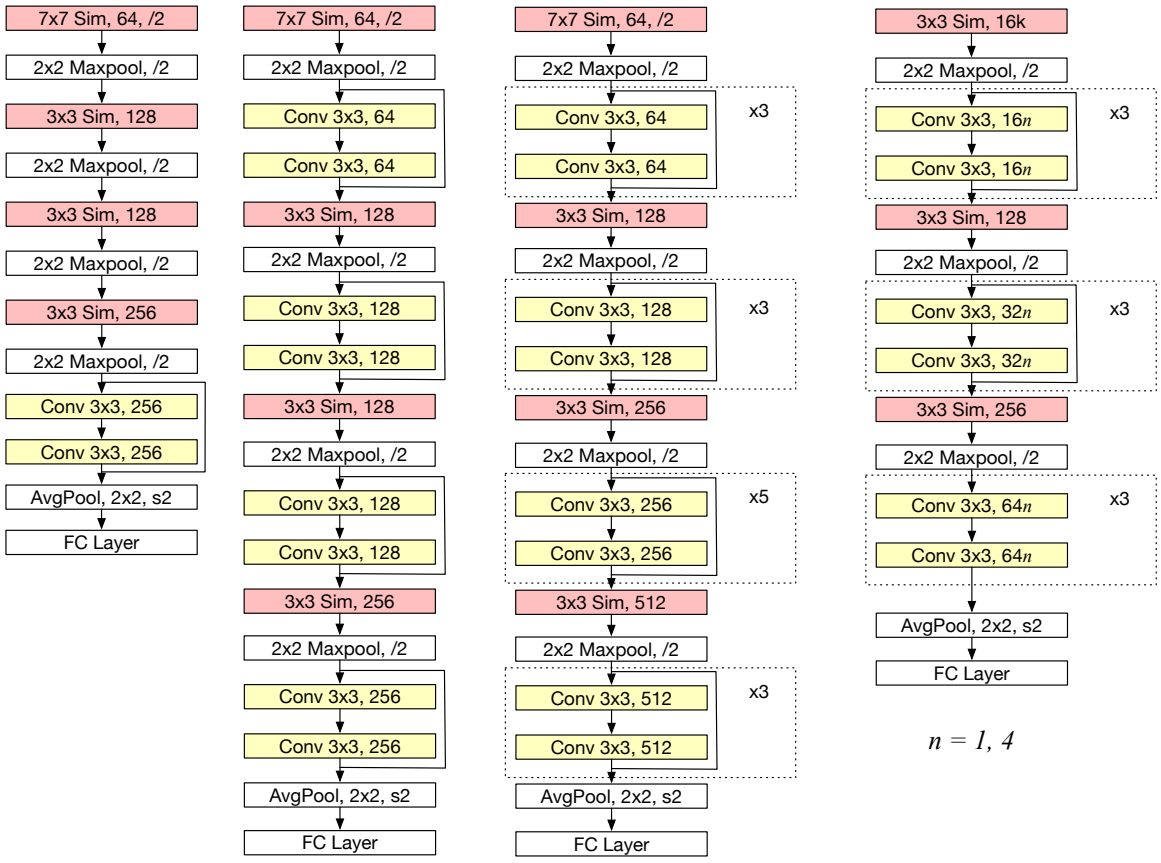

Figure 5.1. Architectures explored in this chapter. The left two architectures are tested for subsets of ILSVRC2018 to study the parameter efficiency of Similarity Layers. The third architecture is designed for ILSVRC 2018. The rightmost architecture is designed for CIFAR 10 and CIFAR 100 datasets. Its width is controlled by n. "Sim" stands for Similarity Layer proposed in this chapter and "Conv" stands for general Convolutional Layer. Note that when the number of stacked Resnet blocks is greater than one, we also add a shortcut for each Similarity Layer from the previous maxpooling layer bypassing all the Resnet Blocks.

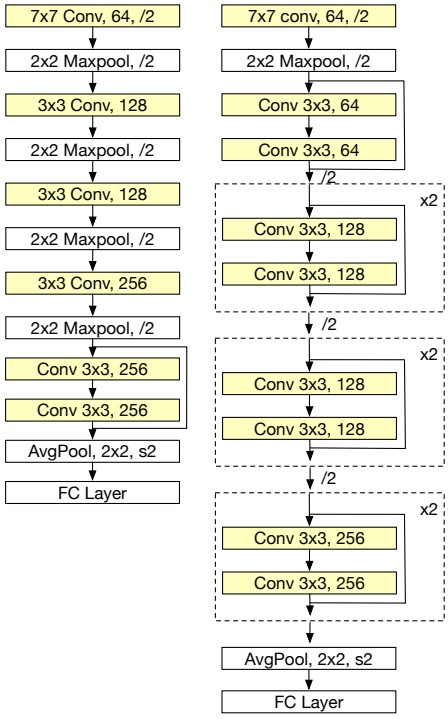

Figure 5.2. Compared CNN (left) and Resnet (right) Models.

to let information dominated by similarity Convolutional Layers in early stages of training. We found such shortcut increases training speed. To show the benefits of proposed similarity Convolutional Layers, we compare it with a number of CNN and Resnet models, which have similar or larger parameter space (details in the experiment section).

We consider similar architectures designed for CIFAR 10 [116] and CIFAR 100 [117]. As shown in fig.5.1, we construct 3 similarity Convolutional Layers and stacked Resnet blocks in between. We follow the idea of [10] and [109], consider 3 Resnet blocks after each maxpooling. We test 2 architectures: first a narrow network with 16, 32, and 64 convolutional filters in each level; and second a wide network with 64, 128 and 256 convolutional filters at each level.

5.2 Layer Efficiency

In this section, we evaluate the layer efficiency of the Similarity Layer and general Convolutional Layer, investigating whether a Similarity Layer can encode more information than a general Convolutional Layer, namely achieving better accuracy with the same or fewer layers. The motivation of considering layer efficiency is to use less GPU Memory as possible, such that an application can be fitted into smaller platforms, e.g. mobile devices. We consider 3 subsets of ILSVRC2018: randomly select 10, 50 and 100 classes from the entire dataset. For systematic comparison, layers used in Similarity Net must be a subset of its compared counterpart. For example, if a General CNN or a ResNet uses n convolutional layers with 64 3×3 filters, the compared Similarity Net can use at most n similarity layers with the same number of 3×3 filters.

For insufficiently large datasets, complex models may be outperformed by simpler ones due to overfitting, which may bias our experiments. Therefore in this section, we consider the first two shallower nets introduced in the previous section to remove bias caused by possible overfitting of deep nets trained on small datasets. We compare our constructed Similarity CNNs with two architectures (Fig.5.2). The first architecture is identical to our 7-layer Similarity Net except replacing all Similarity Layers with general Convolutional Layers, keeping all hyper-parameters the same (receptive field, number of filters, etc.). To compare models with more layers, we consider a 16-layer Resnet that replaces each Similarity Layer as a Resnet Block. Thus each Similarity Layer becomes two Convolutional Layers with the shortcut, as shown in fig.5.2. Batch Normalization is applied for every Convolutional Layer for compared models. We use linear projection when Feature Maps shrinks (option B in [10]).

All the models are trained by Adam Optimizer [112] with learning rate 0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 0.001$. We do not apply Batch Normalization neither in similarity Convolutional Layers nor in Resnet blocks of the Similarity Nets. We augment each image and its horizontal reflections by randomly resize the longer edge into $[256, 480]$, and take random 224×224 crop, following [10].

Table 5.1.
Error rate (single crop) in the subsets of the ILSVRC 2018. For Similarity Net, we apply the model with pre-training and random initialization. For each model, we also consider situations that Batch Normalization is applied and absent. Suffix "BN" stands for Batch Normalization.

	#para	10 classes	50 classes	100 classes
Sim 13 pre-train	2.37M	**7.29**	**16.78**	**22.68**
Sim 7 pre-train	1.71M	11.72	21.09	25.02
Sim 13 random	2.37M	12.16	19.85	24.08
Sim 7 random	1.71M	14.09	24.02	27.58
Sim 13 BN	2.37M	9.38	18.13	22.96
Sim 7 BN	1.71M	13.80	21.49	27.25
Res 16 BN	3.25M	10.49	20.31	24.38
CNN 7 BN	1.71 M	14.28	23.75	28.21
Res 16	3.25M	14.58	25.29	31.25
CNN 7	1.71 M	18.24	26.89	30.71

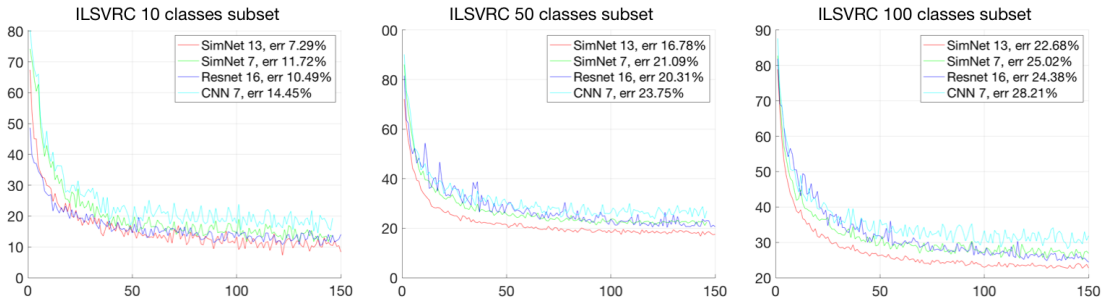

Figure 5.3. Learning Curve (error rate in the validation set) of the 4 models on the ILSVRC subset with 10, 50 and 100 randomly selected classes. Batch Normalization is applied to each layer of Resnet 16 and CNN 7, but not in Similarity Net. ("BN" suffix is omitted to void occlusion)

The final validation accuracy (top 1) is summarized in Table.5.1. We found Similarity Net outperformed compared architectures in general. First, the 7-layer Similarity Net achieved better validation accuracy than 7-layer CNN. Indeed, 7-layer Similarity Net reached the same error rate level of 16-layer Resnet. Following this, 13-layer Similarity Net outperformed 16-layer Resnet. This observation stands for all three subsets, which indicates that the advanced layer efficiency of Similarity Net is consistent in different data sizes. experiments.

The learning curve is plotted in Fig.5.3. Even though Batch Normalization is not applied to any Similarity Layer nor Convolutional Layer in our model, Similarity Net still converges as fast as models with Batch Normalization, where this fast convergence is benefited from pre-training (next subsection).

5.3 Classifications on Benchmark Dataset

ILSVRC2018. In this section, we compare the overall performance of the 33-layer Similarity Net with state-of-the-art models that have the same level of parameter space. We train the network on ILSVRC 2018 and evaluate by the validation set, computing top 1 and top 5 error rate of a single crop. The data augmentation method is the same as in previous experiments. We train the network by Gradient Descend with Momentum of 0.9. We start learning rate with 0.01 and divide by 10 when error rate plateaus. The entire training takes 100 epochs. The final evaluation and compared models are shown in Table.5.2.

As we can see, though the architecture is not carefully tuned, Similarity Net still presents sufficient parameter efficiency. As the closest sibling, Resnet-50, we achieve higher accuracy with less number of parameters. SE-Resnet 50 considers feature dependency across channels in a feature volume. Without such feature recalibration mechanism nor Batch Normalization, Similarity Net still achieved the competitive results using 19% fewer parameters. DenseNet architecture showed advanced parameter efficiency. However, each convolutional layer of DenseNet concatenates Feature Maps from all previous layers, thus

Table 5.2.
Single model top 1 and top 5 validation error rate of the ILSVRC dataset. All the results are from a single crop.

Model	#para	top 1 err	top 5 err
VGG 16 [11]	138.3M	27.02	8.81
Resnet 50 [10]	25.6M	24.70	7.48
WRN 18 [109]	11.7M	30.40	10.93
WRN 34 [109]	21.8M	26.77	8.67
DenseNet 121 [110]	6.9 M	25.02	7.71
DenseNet 200 [110]	17.9M	22.58	6.34
SE-Resnet 50 [12]	26.9M	23.29	6.62
SimNet 33	22.7M	23.52	6.91

Table 5.3.
The single model validation error rate of CIFAR 10 and CIFAR 100 dataset. All the results are from a single crop.

Model	#parameter	C10	C100
Resnet 20	0.27M	8.75	-
Resnet 101 [10]	1.7M	6.41	27.22
WRN 40-1 [109]	0.6M	6.85	30.89
WRN 40-4 [109]	8.9M	4.97	22.89
DenseNet 40 [110]	1.0M	5.24	24.42
DenseNet 100 [110]	7.0M	4.10	20.20
SimNet (1)	0.25M	6.92	31.72
SimNet (4)	3.9M	5.08	24.94

it still needs intensive computational resources. Improving the architectures of Similarity Net can be the focus of our future work.

CIFAR10 and **CIFAR100**. We further test Similarity Net in CIFAR 10 [116] and CIFAR 100 [117] dataset for the architectures on the right of Fig.5.1. We consider two networks with different width, where the larger net is 4 times wider than the smaller one. We adopt a widely used data augmentation approach for CIFAR 10 and CIFAR 100. Each 32×32 image is padded with reflection by 8 pixels into 40×40 images. And then a 32×32 patch is randomly cropped from the padded image. We do not apply data augmentation on the test set. We compare our model with the state-of-the-art architectures of similar parameter space. The results are shown in Table.5.3.

Comparing to its direct predecessor, Resnet 20 and WRN040-1, Similarity Net shows a better performance score under approximately the same number of parameters. Our narrow Similarity Net Architecture ($n = 1$) outperformed Resnet 20. It also performs as well as WRN-40-1, which is 2 more times larger and performs slightly lower accuracy than Resnet 101, which is 6 times larger. The wide architecture of Similarity Net ($n = 4$) performs as

well as WRN-4 (2 times larger). DenseNet [110] showed advanced parameter efficiency in both ILSVRC and CIFAR datasets. However, such efficiency is sacrificed from Memory efficiency. Since each Convolutional Layer of the Dense Net is concatenate to the next layer, the Feature Maps can be very large for the last few layers in each Dense Block. Such memory demand does not exist in the Similarity Net.

5.4 Summary

We apply the Unsupervised Deep Encoding to pre-training Deep Convolutional Neural Networks, to achieve better performance, learning dynamics, and memory efficiency. We conduct systematic experiments on benchmark datasets to validate the advances we make.

The equivalence between a single layer of Neural Network and Clustering provides a way of pre-train a Convolutional Neural Network by unsupervised feature learning. Our experiments and analysis demonstrate that the pre-training schema is crucial to Similarity Net, which also makes the model not rely on Batch Normalization. As a result, all the models showed advanced parameter efficiency compared to the models in similar parameter space. We argue that unsupervised feature learning initiates the Network that closes to a better local minimum, such that we are more likely to find better optimum by the same training method. Finally, unlike general CNNs that perform linear classification at each filter, Similarity Net learns a number of kernels that represent the corresponding feature at each level. For each input region, the convolution output is the similarity measure between input and the filter. This may bring us richer potential to reuse well-learned feature maps.

6. CONCLUSION

In this thesis, we tackled a series of Computer Vision problems in Dynamic Environments, which is featured by unsupervised learning, fast processing, and light training-deploying cycle. There are multiple ways to establish machine intelligence in Vision. The approaches we proposed and studied in this thesis contribute to building a customized vision for harsh applications.

We first developed a stream clustering algorithm that is repeatedly used by the solutions to practical vision tasks applied in Dynamic Environments. The algorithm possesses a number of characteristics that are irreplaceable by existing works. First, it is a fully incremental stream algorithm, thus the cluster centers are promptly updated and well organized such that the efficiency of higher-level learning is optimized. Second, the algorithm is universal, i.e. suitable for arbitrary types of data with arbitrary similarity measure. In the task of online unsupervised object discovery, complex data fed into clustering are histogram with randomly increasing dimension and harshed graphs, each with its own similarity measure. Different similarity measures are also used by general data, i.e. Euclidean Distance and Cosine Similarity. Third, we provided complete theoretical guarantees in statistics. We first show that a given kernel is always moved towards its nearest density peak. We then give proof of a reliable way to initial kernels to the most promising location such that all valid density peaks will be discovered. Our experiments strongly supported the above conclusion. Compared with known state-of-the-art, it is superior in general to both stream and non-stream algorithms. It is not only a solution to vision problems, but also a promising candidate for any data clustering problems.

We then developed a UAV-based vision system that discovers unknown objects on-the-fly. This real-life task can be thought of as the most challenging problem in Dynamic Environments. First, there is no data available before deploying the agents. Thus any sort of pre-training is impossible. The system has to learn all the knowledge from the feature

level to object level on-the-fly with very limited computation. The system strongly benefits from the clustering algorithm we proposed, not only because of its advanced speed, but also the data structure and fully incremental property that optimized the learning at each level. We further developed our stream clustering algorithm into a hierarchical algorithm while still possess all the characteristics of the original one. The clustering result is stored in a data structure, called Memory Tree, that can be used to match features in sub-linear time. We then developed a fast and accurate object proposal method for aerial images, which drastically improved the efficiency of the entire system. This method takes the heuristics that group objects in aerial view possess more corners than the background. Along with the object proposal method, we developed a bottom-up object realization method that groups the pieces of a large object into one according to sustained adjacency. At the proposal(part) level, we also built a Latent Dirichlet Model to discover the different appearance caused by viewpoint. We applied our method to a large number of aerial videos, and compare them with most related methods. Though compared methods are in a non-stream setting, we still reach competitive accuracy. The major advances of our paper are that we achieved much higher coverage, namely better detection and discovery rate.

To further explore the possibility of models in Dynamic Environments, we developed Unsupervised Deep Encoding that bridges traditional Visual Encoding and Convolutional Neural Networks. The motivation of this work is to explore complex models in Dynamic Environments. We first discovered the relation between clustering and single-layer neural networks. We found that by letting the pre-activation function to be a similarity measure between the input and the weight vector, a single-layer neural network can be equivalent to clustering. When cosine similarity is considered as the pre-activation function, clustering can be performed by Back Propagation. Once we replace the dot product by cosine similarity in Convolutional Neural Networks, each convolution becomes an unnormalized soft assignment of the feature in that layer. Naturally, we can learn these features unsupervisedly by clustering. Though we can use Gradient Descend to perform clustering, this process significantly benefits from the advanced efficiency and accuracy of the stream clustering proposed in this thesis. By regarding each feature map as an image, we can

conduct unsupervised feature learning by clustering in every layer of the Similarity-based Convolutional Neural Networks. One application of the Unsupervised Deep Encoding is to pre-train the CNN for realist tasks such as image recognition. We demonstrate by systematic experiments that the features unsupervisedly learned by our approaches are strongly related to semantic labels. When the pre-train is done, freezing early layers in Similarity Layers reaches significantly better classification accuracy than the randomly initialized counterparts. Also, because pre-training finds a better initial state for gradient descend, we achieve the same or higher outcome by using fewer parameters, which makes the model thinner for applications with limited computation.

The methods we developed are designed specifically for the above two applications, however, the theories, ideas, and algorithms are also suitable for many other similar applications.

REFERENCES

REFERENCES

[1] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2016.

[2] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[3] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[4] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*. IEEE, 2017, pp. 2980–2988.

[5] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[6] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.

[7] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587.

[8] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, 2012, pp. 1097–1105.

[10] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[11] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[12] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 7132–7141.

[13] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, p. 436, 2015.

[14] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[15] J. Sivic, A. Zisserman *et al.*, "Video google: A text retrieval approach to object matching in videos." in *Proceedings of the IEEE international conference on computer vision*, vol. 2, no. 1470, 2003, pp. 1470–1477.

[16] H. Jégou, M. Douze, C. Schmid, and P. Pérez, "Aggregating local descriptors into a compact image representation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2010, pp. 3304–3311.

[17] F. Perronnin and C. Dance, "Fisher kernels on visual vocabularies for image categorization," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[18] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[19] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2005.

[20] M. R. Ackermann, M. Märtens, C. Raupach, K. Swierkot, C. Lammersen, and C. Sohler, "Streamkm++: A clustering algorithm for data streams," *Journal of experimental algorithmics (JEA)*, vol. 17, pp. 2–4, 2012.

[21] H. Fichtenberger, M. Gillé, M. Schmidt, C. Schwiegelshohn, and C. Sohler, "Bico: Birch meets coresets for k-means clustering," in *European Symposium on Algorithms*. Springer, 2013, pp. 481–492.

[22] T. Zhang, R. Ramakrishnan, and M. Livny, "Birch: an efficient data clustering method for very large databases," in *ACM Sigmod Record*, vol. 25, no. 2. ACM, 1996, pp. 103–114.

[23] A. Kobren, N. Monath, A. Krishnamurthy, and A. McCallum, "A hierarchical algorithm for extreme clustering," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2017, pp. 255–264.

[24] D. Sculley, "Web-scale k-means clustering," in *Proceedings of the International Conference on World Wide Web*. ACM, 2010, pp. 1177–1178.

[25] E. Liberty, R. Sriharsha, and M. Sviridenko, "An algorithm for online k-means clustering," in *Proceedings of the Workshop on Algorithm Engineering and Experiments*. SIAM, 2016, pp. 81–89.

[26] O. Bachem, M. Lucic, S. H. Hassani, and A. Krause, "Approximate k-means++ in sublinear time." in *AAAI*, 2016, pp. 1459–1467.

[27] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[28] M. Cho, S. Kwak, C. Schmid, and J. Ponce, "Unsupervised object discovery and localization in the wild: Part-based matching with bottom-up region proposals," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1201–1210.

[29] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural networks*, vol. 61, pp. 85–117, 2015.

[30] B. J. Frey and D. Dueck, "Clustering by passing messages between data points," *Science*, vol. 315, no. 5814, pp. 972–976, 2007.

[31] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492–1496, 2014.

[32] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[33] D. Arthur and S. Vassilvitskii, "k-means++: The advantages of careful seeding," in *Proceedings of the annual ACM-SIAM symposium on discrete algorithms*. Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035.

[34] T. K. Moon, "The expectation-maximization algorithm," *IEEE Signal Processing Magazine*, vol. 13, no. 6, pp. 47–60, 1996.

[35] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[36] L. O'callaghan, N. Mishra, A. Meyerson, S. Guha, and R. Motwani, "Streaming-data algorithms for high-quality clustering," in *Proceedings of the International Conference on Data Engineering*. IEEE, 2002, pp. 685–694.

[37] C. C. Aggarwal, J. Han, J. Wang, and P. S. Yu, "A framework for clustering evolving data streams," in *Proceedings of the international conference on very large data Bases-Volume 29*. VLDB Endowment, 2003, pp. 81–92.

[38] F. Cao, M. Estert, W. Qian, and A. Zhou, "Density-based clustering over an evolving data stream with noise," in *Proceedings of the SIAM international conference on data mining*. SIAM, 2006, pp. 328–339.

[39] J. A. Silva, E. R. Faria, R. C. Barros, E. R. Hruschka, A. C. de Carvalho, and J. Gama, "Data stream clustering: A survey," *ACM Computing Surveys (CSUR)*, vol. 46, no. 1, p. 13, 2013.

[40] L. M. L. Alvarez, "Data stream management systems," Dec. 25 2014, uS Patent App. 14/375,845.

[41] A. Ene, S. Im, and B. Moseley, "Fast clustering using mapreduce," in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2011, pp. 681–689.

[42] X. Cui, P. Zhu, X. Yang, K. Li, and C. Ji, "Optimized big data k-means clustering using mapreduce," *The Journal of Supercomputing*, vol. 70, no. 3, pp. 1249–1259, 2014.

[43] M.-F. F. Balcan, S. Ehrlich, and Y. Liang, "Distributed k-means and k-median clustering on general topologies," in *Advances in neural information processing systems*, 2013, pp. 1995–2003.

[44] B. Bahmani, B. Moseley, A. Vattani, R. Kumar, and S. Vassilvitskii, "Scalable k-means++," *Proceedings of the VLDB endowment*, vol. 5, no. 7, pp. 622–633, 2012.

[45] Y. W. Teh, M. I. Jordan, M. J. Beal, and D. M. Blei, "Sharing clusters among related groups: Hierarchical dirichlet processes," in *Advances in neural information processing systems*, 2005, pp. 1385–1392.

[46] B. Kulis and M. I. Jordan, "Revisiting k-means: New algorithms via bayesian nonparametrics," *arXiv preprint arXiv:1111.0352*, 2011.

[47] A. Coates, A. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 215–223.

[48] J. Philbin, O. Chum, M. Isard, J. Sivic, and A. Zisserman, "Object retrieval with large vocabularies and fast spatial matching," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–8.

[49] D. Nister and H. Stewenius, "Scalable recognition with a vocabulary tree," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, vol. 2. Ieee, 2006, pp. 2161–2168.

[50] C. Luo, J. Zhan, X. Xue, L. Wang, R. Ren, and Q. Yang, "Cosine normalization: Using cosine similarity instead of dot product in neural networks," in *International conference on artificial neural networks*. Springer, 2018, pp. 382–391.

[51] G. E. Hinton and R. R. Salakhutdinov, "Reducing the dimensionality of data with neural networks," *science*, vol. 313, no. 5786, pp. 504–507, 2006.

[52] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P.-A. Manzagol, "Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion," *Journal of machine learning research*, vol. 11, no. Dec, pp. 3371–3408, 2010.

[53] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *European conference on computer vision*. Springer, 2014, pp. 818–833.

[54] M. D. Zeiler, G. W. Taylor, and R. Fergus, "Adaptive deconvolutional networks for mid and high level feature learning," in *Proceedings of the IEEE international conference on computer vision*. IEEE, 2011, pp. 2018–2025.

[55] X. Glorot and Y. Bengio, "Understanding the difficulty of training deep feedforward neural networks," in *Proceedings of the international conference on artificial intelligence and statistics*, 2010, pp. 249–256.

[56] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Advances in neural information processing systems*, 2007, pp. 153–160.

[57] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[58] Q. V. Le, "Building high-level features using large scale unsupervised learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 8595–8598.

[59] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[60] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," *arXiv preprint arXiv:1511.05644*, 2015.

[61] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, 1990.

[62] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[63] J. Donahue, P. Krähenbühl, and T. Darrell, "Adversarial feature learning," *arXiv preprint arXiv:1605.09782*, 2016.

[64] X. Huang, Y. Li, O. Poursaeed, J. Hopcroft, and S. Belongie, "Stacked generative adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2017, pp. 1866–1875.

[65] C. Doersch, A. Gupta, and A. A. Efros, "Unsupervised visual representation learning by context prediction," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1422–1430.

[66] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, and A. A. Efros, "Context encoders: Feature learning by inpainting," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2536–2544.

[67] M. Noroozi and P. Favaro, "Unsupervised learning of visual representations by solving jigsaw puzzles," in *European conference on computer vision*. Springer, 2016, pp. 69–84.

[68] X. Wang and A. Gupta, "Unsupervised learning of visual representations using videos," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 2794–2802.

[69] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv:1502.03167*, 2015.

[70] J. Lei Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[71] T. Salimans and D. P. Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," in *Advances in neural information processing systems*, 2016, pp. 901–909.

[72] L. Huang, D. Yang, B. Lang, and J. Deng, "Decorrelated batch normalization," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[73] I. Sutskever, J. Martens, G. Dahl, and G. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, 2013, pp. 1139–1147.

[74] D. Erhan, P.-A. Manzagol, Y. Bengio, S. Bengio, and P. Vincent, "The difficulty of training deep architectures and the effect of unsupervised pre-training," in *Artificial Intelligence and Statistics*, 2009, pp. 153–160.

[75] J. Xiao, C. Yang, F. Han, and H. Cheng, "Vehicle and person tracking in aerial videos," in *Multimodal technologies for perception of humans*. Springer, 2007, pp. 203–214.

[76] J. Xiao, H. Cheng, H. Sawhney, and F. Han, "Vehicle detection and tracking in wide field-of-view aerial video," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*. IEEE, 2010, pp. 679–684.

[77] S. Tuermer, F. Kurz, P. Reinartz, and U. Stilla, "Airborne vehicle detection in dense urban areas using hog features and disparity maps," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 6, no. 6, pp. 2327–2337, 2013.

[78] M. Teutsch and W. Krüger, "Detection, segmentation, and tracking of moving objects in uav videos," in *IEEE International Conference on Advanced Video and Signal-Based Surveillance*. IEEE, 2012, pp. 313–318.

[79] M. Teutsch and W. Kruger, "Robust and fast detection of moving vehicles in aerial videos using sliding windows," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2015, pp. 26–34.

[80] S. Ali, V. Reilly, and M. Shah, "Motion and appearance contexts for tracking and reacquiring targets in aerial videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*. IEEE, 2007, pp. 1–6.

[81] T. Shu, D. Xie, B. Rothrock, S. Todorovic, and S. Chun Zhu, "Joint inference of groups, events and human roles in aerial videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 4576–4584.

[82] S. Singh, A. Gupta, and A. A. Efros, "Unsupervised discovery of mid-level discriminative patches," in *European conference on computer vision*. Springer, 2012, pp. 73–86.

[83] J.-Y. Zhu, J. Wu, Y. Xu, E. Chang, and Z. Tu, "Unsupervised object class discovery via saliency-guided multiple class learning," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 4, pp. 862–875, 2014.

[84] S. Kwak, M. Cho, I. Laptev, J. Ponce, and C. Schmid, "Unsupervised object discovery and tracking in video collections," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 3173–3181.

[85] A. Ošep, P. Voigtlaender, J. Luiten, S. Breuers, and B. Leibe, "Large-scale object discovery and detector adaptation from unlabeled video," *arXiv preprint arXiv:1712.08832*, 2017.

[86] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[87] C. L. Zitnick and P. Dollár, "Edge boxes: Locating object proposals from edges," in *European conference on computer vision*. Springer, 2014, pp. 391–405.

[88] B. Alexe, T. Deselaers, and V. Ferrari, "Measuring the objectness of image windows," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2189–2202, 2012.

[89] J. R. Uijlings, K. E. Van De Sande, T. Gevers, and A. W. Smeulders, "Selective search for object recognition," *International journal of computer vision*, vol. 104, no. 2, pp. 154–171, 2013.

[90] J. Carreira and C. Sminchisescu, "Cpmc: Automatic object segmentation using constrained parametric min-cuts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1312–1328, 2011.

[91] E. Rahtu, J. Kannala, and M. Blaschko, "Learning a category independent object detection cascade," in *2011 international conference on Computer Vision*. IEEE, 2011, pp. 1052–1059.

[92] S. Manen, M. Guillaumin, and L. Van Gool, "Prime object proposals with randomized prim's algorithm," in *Proceedings of the IEEE international conference on computer vision*, 2013, pp. 2536–2543.

[93] I. Endres and D. Hoiem, "Category-independent object proposals with diverse ranking," *IEEE transactions on pattern analysis and machine intelligence*, vol. 36, no. 2, pp. 222–234, 2013.

[94] P. Rantalankila, J. Kannala, and E. Rahtu, "Generating object segmentation proposals using global and local search," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 2417–2424.

[95] M.-M. Cheng, Z. Zhang, W.-Y. Lin, and P. Torr, "Bing: Binarized normed gradients for objectness estimation at 300fps," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 3286–3293.

[96] M. Everingham, S. A. Eslami, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International journal of computer vision*, vol. 111, no. 1, pp. 98–136, 2015.

[97] F. S. Samaria and A. C. Harter, "Parameterisation of a stochastic model for human face identification," in *Proceedings of the IEEE Workshop on Applications of Computer Vision*. IEEE, 1994, pp. 138–142.

[98] Y. LeCun, C. Cortes, and C. J. Burges, "The mnist database of handwritten digits," 1998.

[99] J.-M. Geusebroek, G. J. Burghouts, and A. W. Smeulders, "The amsterdam library of object images," *International Journal of Computer Vision*, vol. 61, no. 1, pp. 103–112, 2005.

[100] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[101] A. Asuncion and D. Newman, "Uci machine learning repository," 2007.

[102] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[103] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

[104] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the international conference on machine learning*, 2010, pp. 807–814.

[105] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1026–1034.

[106] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the international conference on artificial intelligence and statistics*, 2011, pp. 315–323.

[107] B. Xu, N. Wang, T. Chen, and M. Li, "Empirical evaluation of rectified activations in convolutional network," *arXiv preprint arXiv:1505.00853*, 2015.

[108] Z. Wang and G. Tsechpenakis, "Stream clustering with dynamic estimation of emerging local densities," in *International Conference on Pattern Recognition*. IEEE, 2018, pp. 2100–2105.

[109] S. Zagoruyko and N. Komodakis, "Wide residual networks," *arXiv preprint arXiv:1605.07146*, 2016.

[110] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 4700–4708.

[111] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "ImageNet: A Large-Scale Hierarchical Image Database," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2009.

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

[113] R. Pascanu, T. Mikolov, and Y. Bengio, "On the difficulty of training recurrent neural networks," in *International conference on machine learning*, 2013, pp. 1310–1318.

[114] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244.

[115] M. Barekatain, M. Martí, H.-F. Shih, S. Murray, K. Nakayama, Y. Matsuo, and H. Prendinger, "Okutama-action: An aerial view video dataset for concurrent human action detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition Workshops*, 2017, pp. 28–35.

[116] A. Krizhevsky, V. Nair, and G. Hinton, "Cifar-10 (canadian institute for advanced research)." [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html

[117] ——, “Cifar-100 (canadian institute for advanced research).” [Online]. Available: http://www.cs.toronto.edu/~kriz/cifar.html