Is My Object in This Video? Reconstruction-based Object Search in Videos

Tan Yu, Jingjing Meng, Junsong Yuan
ROSE Lab, Interdisciplinary Graduate School, Nanyang Technological Univeristy, Singapore
{tyu008, jingjing.meng, jsyuan}@ntu.edu.sg

Abstract

This paper addresses the problem of video-level object instance search, which aims to retrieve the videos in the database that contain a given query object instance. Without prior knowledge about "when" and "where" an object of interest may appear in a video, determining "whether" a video contains the target object is computationally prohibitive, as it requires exhaustively matching the query against all possible spatial-temporal locations in each video that an object may appear. To alleviate the computational and memory cost, we propose the Reconstruction-based Object SEarch (ROSE) method. It characterizes a huge corpus of features of possible spatial-temporal locations in the video into the parameters of the reconstruction model. Since the memory cost of storing reconstruction model is much less than that of storing features of possible spatial-temporal locations in the video, the efficiency of the search is significantly boosted. Comprehensive experiments on three benchmark datasets demonstrate the promising performance of the proposed ROSE method.

1 Introduction

Given a query object and a database of videos (*e.g.*, video clips collected by mobile robots), video-level object search aims to retrieve the videos containing this object instance without concerning about "when" and "where" it may appear in these videos (*i.e.*, its spatio-temporal locations). It is an important problem which can serve as the stepping stone for many vision and robotics problems that involve fine level video analytics such as object sensing and identification.

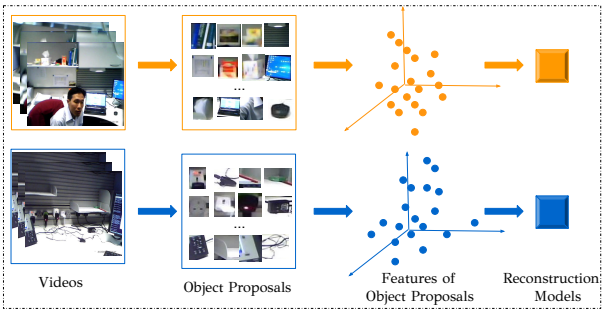
Because the object of interest can be small and appear at any spatio-temporal location in a video, to find out whether a video contains the target object, matching the query directly with the entire video usually does not work. Therefore, previous works try to first produce a set of candidate locations in a video where an object may appear, using either frame-wise sliding windows [Tolias *et al.*, 2016] or object proposals [Bhattacharjee *et al.*, 2016]. Then the relevance of a video to the query can be measured by the matching score of the best-matched candidate location of the query.

However, matching candidate locations is neither necessary nor efficient. First, our goal is to identify "whether" an object of interest is in a video, instead of finding "where" exactly it locates. Second, even with the help of object proposals, it is still computationally prohibitive to perform an exhaustive search for all object proposals as the number of object proposals per video can be extremely large. For example, a short video clip of half a minute can generate tens of thousands of object proposals in order to maintain a high recall of the objects. Hence it is considerably time-consuming to match the query with each individual object proposal, especially if we need to handle a big video database of thousands of video clips.

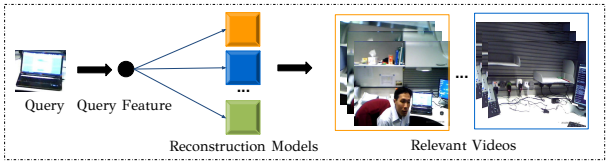
Fortunately, although the number of object proposals per video is huge, the object proposals in a video tend to have high redundancy because of their large spatio-temporal overlaps [Tu *et al.*, 2014]. Thus one plausible solution to speed up the search and reduce the memory cost of storing all the object proposals is to first select the representative ones for each video [Meng *et al.*, 2016a]. Then we only need to match the query with the selected representatives instead of all. According to [Meng *et al.*, 2016a], 10 percent selected representatives can achieve excellent performance in shot-level object instance search. However, to guarantee a reasonable recall, the selection ratio cannot be too small. Therefore, the reduction in memory and computational cost by representative selection is limited.

Different from representative selection, we propose the Reconstruction-based Object SEarch (ROSE) method to further improve the efficiency and accuracy of video-level object instance search, without exhaustively comparing the query with each object proposal. Specifically, for each video, instead of storing all of its object proposals that may capture the query object, we train a compact model for each video to reconstruct all of its object proposals. Then in the search phase, the reconstruction model of each video can answer whether it has ever seen the query object, by trying to reconstruct the query. As the query is not directly compared with individual object proposals, our proposed solution brings a significant reduction in both computational and memory cost, while achieving comparable search accuracy to exhaustive search, due to the following properties:

- Instead of locating all instances of the query accurately in the videos, video-level object search only cares about



(a) Training Phase



(b) Search Phase

Figure 1: In the training phase, for each video, we extract object proposals serving as candidate object locations of the query in the video. We further train a reconstruction model using the set of features of the object proposals generated from each video. In the search phase, given a query object, for each video, we calculate the query's reconstruction error using its reconstruction model. The videos are ranked according to the reconstruction errors.

whether the query object is contained in a video or not. As will be shown in our experiments, such an identification problem can be addressed by a simple reconstruction model.

- If a video contains an instance of the query object, more often than not, it will generate many spatio-temporally overlapping object proposals that capture the instance. Therefore, instead of "finding a needle in a haystack", we are actually checking whether there are "a bunch of needles" in a haystack. Although the reconstruction model may not be able to capture a single needle, it is likely to capture "a bunch of needles". Therefore, when the query object comes, the reconstruction model will not miss it.
- Training a model to reconstruct all object proposals can be time consuming, but it is performed offline for only once thus does not affect runtime efficiency. In the search phase, we only need to reconstruct the query to tell whether it is contained in the video, and the same reconstruction model works for different queries.

As illustrated in Figure 1, our Reconstruction-based Object SEarch (ROSE) method converts the set of object proposals into the parameters of the reconstruction model. The learned reconstruction model is a higher-level abstraction of the set of features of object proposals, which is not only compact but also shows better generalization ability. We conduct experiments on three benchmark datasets and compares the state-of-the-art methods to verify the effectiveness and efficiency of our ROSE method.

2 Related Work

Object Instance Search in Videos. In the past decade, the problem of object instance search has been widely exploited on image datasets [Jiang *et al.*, 2015; Tao *et al.*, 2015; Tolias *et al.*, 2016; Bhattacharjee *et al.*, 2016; Yu *et al.*, 2017a] but not as much on video dataset [Meng *et al.*, 2016b; Yu *et al.*, 2017b]. A pioneer work for object instance search in videos is Video Google [Sivic and Zisserman, 2009], which treats each video keyframe independently and ranked the video shots by its best-matched keyframes. Some following works [Zhu and Satoh, 2012; Yang and Satoh, 2013] also process the frames individually and ignore the redundancy across the frames in videos. Until very recently, [Meng *et al.*, 2016a] create a pool of object proposals for each video shot and further utilize the representative selection which exploits the spatio-temporal redundancy in the videos in order to speed up the search.

Reconstruction Model. Reconstruction model has been widely used in outlier/anomaly detection [Cong *et al.*, 2011; Xia *et al.*, 2015]. In this case, a reconstruction model implemented by sparse coding or auto-encoder is trained and the data samples with larger reconstruction errors can be treated as the anomaly/outlier samples. Meanwhile, the reconstruction model has also been used in video highlights detection [Yang *et al.*, 2015] in which the samples reconstructed accurately from the reconstruction model are identified as the samples to summarize the video. Furthermore, the reconstruction model has also been used as classifier [Wright *et al.*, 2009; Hayat *et al.*, 2015]. In this case, the class-specific reconstruction models are trained and the test sample is assigned to the class for which the best reconstruction is obtained.

3 Reconstruction-based Object Search

We denote by $r_{\boldsymbol{\theta}_i}(\cdot) : \mathbb{R}^d \rightarrow \mathbb{R}^d$ the reconstruction model learned by the set of object proposals $\mathcal{S}_i = \{\mathbf{x}_i^1, ..., \mathbf{x}_i^m\}$ from the video V_i, where $\boldsymbol{\theta}_i$ is the parameters of the reconstruction model which are learned by reconstructing all object proposals in the training phase:

$$\boldsymbol{\theta}_i = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{\boldsymbol{x} \in \mathcal{S}_i} \|\boldsymbol{x} - r_{\boldsymbol{\theta}}(\boldsymbol{x})\|_2^2. \tag{1}$$

In the search phase, for each video V_i, we calculate the query's reconstruction error $\|r_{\boldsymbol{\theta}_i}(\boldsymbol{q}) - \boldsymbol{q}\|_2$ using the reconstruction model learned from the video. We use the reconstruction error as a similarity measurement to determine the relevance between the query $\boldsymbol{q}$ and the whole video V_i:

$$dist(\boldsymbol{q}, V_i) = \|\boldsymbol{q} - r_{\boldsymbol{\theta}_i}(\boldsymbol{q})\|_2 \tag{2}$$

The smaller the reconstruction error $\|\boldsymbol{q} - r_{\boldsymbol{\theta}_i}(\boldsymbol{q})\|_2$ is, the more relevant the query is to the video V_i.

If the query object is similar to some object proposals from the video (training data), its reconstruction error should be small. After training the reconstruction model for the video V_i, we no longer rely on the object proposals $\mathcal{S}_i = \{\mathbf{x}_i^1, ..., \mathbf{x}_i^m\}$ and only need to store the parameters $\boldsymbol{\theta}_i$, which is more compact than $\mathcal{S}_i$. Meanwhile, rather than comparing the query $\boldsymbol{q}$ with all the object proposals in the set $\mathcal{S}_i$, the reconstruction model only need to compute $\|\boldsymbol{q} - r_{\boldsymbol{\theta}_i}(\boldsymbol{q})\|_2$ to

obtain the relevance between $\boldsymbol{q}$ and $\mathcal{S}_i$, which can be calculated very efficiently.

The Reconstruction-based Object SEarch (ROSE) method characterizes the set of object proposals $\mathcal{S}_i$ extracted from each video V_i by the reconstruction model $r_{\boldsymbol{\theta}_i}$ trained from V_i. The reconstruction error of the query $\|\boldsymbol{q}-r_{\boldsymbol{\theta}_i}(\boldsymbol{q})\|_2$ serves as a relevance measurement to rank the videos. It should be avoided to build the reconstruction model which can perfectly reconstruct any input vector since it will make the reconstruction error meaningless. This can be achieved by adding constraints or regularization terms in training the reconstruction model. We attempt multiple methods to implement the reconstruction model $r_{\boldsymbol{\theta}_i}(\cdot)$. It includes subspace projection, auto-encoder and sparse dictionary learning.

3.1 Subspace Projection

We define the reconstruction model based on subspace projection as:

$$r_{\boldsymbol{\theta}_i}(\boldsymbol{x}) = \boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i \boldsymbol{x}, \text{ s.t. } \boldsymbol{\theta}_i \boldsymbol{\theta}_i^\top = \boldsymbol{I}, \tag{3}$$

where $\boldsymbol{\theta}_i^\top \boldsymbol{\theta}_i$ is the projection matrix and $\boldsymbol{\theta}_i \in \mathbb{R}^{l\times d}$ consists of the bases of the projected subspace. To avoid $r_{\boldsymbol{\theta}_i}(\boldsymbol{x}) = 0$ for any $\boldsymbol{x} \in \mathbb{R}^d$, we must set $l < d$. Otherwise, if $l = d$, $r_{\boldsymbol{\theta}_i}(\boldsymbol{x}) = \boldsymbol{x}$.

In the training phase, we seek to learn $\boldsymbol{\theta}_i$ by optimizing the following problem:

$$\begin{aligned} \boldsymbol{\theta_i} &= \underset{\boldsymbol{\theta}\in\{\boldsymbol{A}\in\mathbb{R}^{l\times d}|\boldsymbol{A}\boldsymbol{A}^\top=\boldsymbol{I}\}}{\operatorname{argmin}} \sum_{\boldsymbol{x}\in\mathcal{S}_i} \|\boldsymbol{x} - \boldsymbol{\theta}^\top\boldsymbol{\theta}\boldsymbol{x}\|_2^2 \\ &= \underset{\boldsymbol{\theta}\in\{\boldsymbol{A}\in\mathbb{R}^{l\times d}|\boldsymbol{A}\boldsymbol{A}^\top=\boldsymbol{I}\}}{\operatorname{argmin}} \|\boldsymbol{X}_i - \boldsymbol{\theta}^\top\boldsymbol{\theta}\boldsymbol{X}_i\|_F^2, \end{aligned} \tag{4}$$

where $\boldsymbol{X}_i = [\boldsymbol{x}_1, ..., \boldsymbol{x}_m] \in \mathbb{R}^{d\times m}$ is the matrix consisting of features of all the object proposals from the video V_i. The optimal $\boldsymbol{\theta}_i$ is obtained by the singular value decomposition. The singular value decomposition of $\boldsymbol{X}_i$ is defined as $\boldsymbol{X}_i = \boldsymbol{U}_i\boldsymbol{\Sigma}_i\boldsymbol{V}_i^\top$. It can be proven that the optimal $\boldsymbol{\theta}_i \in \mathbb{R}^{l\times d}$ can be simply obtained by choosing the first l columns of the matrix $\boldsymbol{U}_i$, *i.e.*, the left singular vectors of the l largest eigen-values.

In the testing phase, the reconstruction error of $\boldsymbol{q}$ using the reconstruction model learned from each set $\mathcal{S}_i$ can be obtained by $\|\boldsymbol{q} - \boldsymbol{\theta}_i^\top\boldsymbol{\theta}_i\boldsymbol{q}\|_2$. Intuitively, it measures the shortest euclidean distance between the query object $\boldsymbol{q}$ and the data points which lies in the subspace spanned by the bases $\boldsymbol{\theta}_i$.

Compared with exhaustively comparing the query $\boldsymbol{q}$ with all the object proposals $\mathcal{S}_i = \{\boldsymbol{x}_i^1, ..., \boldsymbol{x}_i^m\}$, the reconstruction model implemented by subspace projection is much more efficient, since $l \ll m$. Nevertheless, how to choose l is not a trival problem. On one hand, if l is too small, it will fail to capture the information embeded in the batch of data. On the other hand, if l is too large, considering an extreme case when $l = d$, *i.e.*, $r_{\boldsymbol{\theta}}(\boldsymbol{x}) = \boldsymbol{x}$, it can perfectly reconstruct any d-dimensional vector since ${\boldsymbol{\theta}_i}^\top\boldsymbol{\theta}_i = \boldsymbol{I}$. The above situation limits the usefulness of the reconstruction model implemented by subspace projection.

3.2 Auto-encoder

We define the reconstruction model implemented by auto-encoder as:

$$r_{\boldsymbol{\theta}_i}(\boldsymbol{x}) = f_2(\boldsymbol{W}_i^2 f_1(\boldsymbol{W}_i^1\boldsymbol{x} + \boldsymbol{b}_i^1) + \boldsymbol{b}_i^2), \tag{5}$$

where $\boldsymbol{b}_i^1 \in \mathbb{R}^l$ and $\boldsymbol{b}_i^2 \in \mathbb{R}^d$ are the bias vectors, $\boldsymbol{W}_i^1 \in \mathbb{R}^{l\times d}$ and $\boldsymbol{W}_i^2 \in \mathbb{R}^{l\times d}$ are the weight matrices, l is the number of hidden units, $f_1(\cdot)$ is the encoding activation function and $f_2(\cdot)$ is the decoding activation function.

In the training phase, $\boldsymbol{\theta}_i = \{\boldsymbol{W}_i^1, \boldsymbol{W}_i^2, \boldsymbol{b}_i^1, \boldsymbol{b}_i^2\}$ can be obtained by optimizing the following problem:

$$\boldsymbol{\theta}_i = \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{\boldsymbol{x}\in\mathcal{S}_i} \|\boldsymbol{x} - f_2(\boldsymbol{W}^2 f_1(\boldsymbol{W}^1\boldsymbol{x} + \boldsymbol{b}^1) + \boldsymbol{b}^2)\|_2^2. \tag{6}$$

In the test phase, the reconstruction error of $\boldsymbol{q}$ for each set $\mathcal{S}_i$ can be calculated by $\|\boldsymbol{q} - f_2(\boldsymbol{W}_i^2 f_1(\boldsymbol{W}_i^1\boldsymbol{q} + \boldsymbol{b}_i^1) + \boldsymbol{b}_i^2)\|_2$.

Based on the reconstruction model implemented by the auto-encoder, for each video V_i, we only need to store $\boldsymbol{\theta}_i = \{\boldsymbol{W}_i^1, \boldsymbol{W}_i^2, \boldsymbol{b}_i^1, \boldsymbol{b}_i^2\}$, which require $\mathcal{O}(dl)$ memory complexity. Meanwhile, the computational cost for calculating the reconstruction error for each video is $\mathcal{O}(dl)$, which is mainly cost on the matrix-vector product $\boldsymbol{W}_i^2 f_1(\cdot)$ and $\boldsymbol{W}_i^1\boldsymbol{q}$.

In fact, auto-encoder is similar to the subspace projection implemented by singular value decomposition except for the nonlinear activation function $f_1(\cdot)$ and $f_2(\cdot)$ and the bias vector $\boldsymbol{b}_i^1$ and $\boldsymbol{b}_i^2$. It can be proven that if the input vector $\boldsymbol{x}$ is zero-mean and both the encoder activation function and decoder activation function are linear functions, optimizing auto-encoder will be equivalent to singular value decomposition. Note that, when the number of hidden units l is large, it tends to be capable of reconstructing any input vector like the situation we encounter in the reconstruction model implemented by the subspace projection. To tackle this problem, a natural solution is adding sparsity constraint. We implement two types of sparsity constraint auto-encoder. They are sparse auto-encoder [Ng, 2011] and k-sparse auto-encoder [Makhzani and Frey, 2013].

Sparse Auto-encoder In order to constrain the sparsity of the activation of the hidden units, sparse auto-encoder adds $KL(\rho\|\hat{\rho}_j)$, the Kullback-Leibler(KL) divergence between average activation of hidden units and sparsity parameter ρ, as a penalty term in the object function in Eq. (6), where $\hat{\rho}_j$ is the average activation of j-th hidden unit.

K-sparse Auto-encoder [Makhzani and Frey, 2013] proposed the k-sparse autoencoder with linear activation function. In the feedforward phase, after computing the hidden code $\boldsymbol{z} = \boldsymbol{W}_1\boldsymbol{x} + \boldsymbol{b}$, rather than constructing the output from all of the hidden units, it identifies the k largest hidden units in $\boldsymbol{z}$ and set the others to zero.

3.3 Sparse Dictionary Learning

We define the reconstruction model implemented by sparse dictionary learning as :

$$r_{\boldsymbol{\theta}_i}(\boldsymbol{x}) = \boldsymbol{\theta}_i\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x}), \tag{7}$$

where $\boldsymbol{\theta}_i \in \mathbb{R}^{d\times l}$ consists of the atoms of the dictionary learned from $\mathcal{S}_i$. $\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ is the sparse code of the vector $\mathbf{x}$ based on $\boldsymbol{\theta}_i$ satisfying $\|\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})\|_0 = z$, where z is the number of non-zero elements of the code.

In the training phase, $\boldsymbol{\theta}_i$ is learned by optimizing the following problem:

$$\begin{aligned} \boldsymbol{\theta}_i = &\underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{\boldsymbol{x}\in\mathcal{S}_i} \|\boldsymbol{x} - \boldsymbol{\theta}_i \boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})\|_2^2 \\ &\text{s.t. } \|\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})\|_0 = z, \forall \boldsymbol{x} \in \mathcal{S}_i. \end{aligned} \tag{8}$$

The above l_0-norm constrained optimization is normally converted into l_1-penalty problem given by

$$\begin{aligned} \boldsymbol{\theta}_i &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \sum_{\boldsymbol{x}\in\mathcal{S}_i} \|\boldsymbol{x} - \boldsymbol{\theta}_i \boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})\|_2^2 + \lambda \|\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})\|_1 \\ &= \underset{\boldsymbol{\theta}}{\operatorname{argmin}} \|\boldsymbol{X}_i - \boldsymbol{\theta}_i \boldsymbol{H}_i\|_F^2 + \lambda \|\boldsymbol{H}_i\|_{1,1}, \end{aligned} \tag{9}$$

where l_1-penalty on $\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ (sum of the magnitude of its elements) encourages a sparse solution of $\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{x})$, $\boldsymbol{X}_i = [\boldsymbol{x}_1, ..., \boldsymbol{x}_m] \in \mathbb{R}^{d\times m}$ is the matrix of all the data points from the set $\mathcal{S}_i$, $\boldsymbol{H}_i = [\boldsymbol{h}_{\boldsymbol{\theta}_i}(\mathbf{x}_i^1), ..., \boldsymbol{h}_{\boldsymbol{\theta}_i}(\mathbf{x}_i^m)]$ is the matrix consisting of the sparse codes, $\|\boldsymbol{H}_i\|_{1,1} = \sum_{j=1}^m \|\boldsymbol{h}_{\boldsymbol{\theta}_i}(\mathbf{x}_i^j)\|_1$ and λ is the parameter to control the sparsity of $\boldsymbol{h}_{\boldsymbol{\theta}_i}(\cdot)$. We use online dictionary learning [Mairal *et al.*, 2009] to solve the above optimization problem since it is efficient to handle large-scale high-dimensional data.

In the search phase, given a query point $\boldsymbol{q}$, for each set $\mathcal{S}_i$, we can efficiently obtain its sparse code $\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{q})$ using orthogonal matching pursuit (OMP):

$$\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{q}) = \text{OMP}(\boldsymbol{q}, \boldsymbol{\theta}_i, z), \tag{10}$$

The reconstruction error of $\boldsymbol{q}$ from the reconstruction model $r_{\boldsymbol{\theta}_i}(\boldsymbol{x})$ is further obtained by $\|\boldsymbol{q} - \boldsymbol{\theta}\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{q})\|_2$.

The reconstruction model implemented by sparse coding only requires to store the dictionary atoms $\boldsymbol{\theta}_i \in \mathbb{R}^{d\times l}$ which takes $O(ld)$ memory cost. The most time-consuming part in the search phase is to obtain the sparse codes $\boldsymbol{h}_{\boldsymbol{\theta}_i}(\boldsymbol{q})$ through orthogonal matching pursuit. It takes $O(zld)$ computational complexity. In our experiments, we find the number of non-zero element of the code $z = 2$ achieves excellent performance in search precision as well as efficiency.

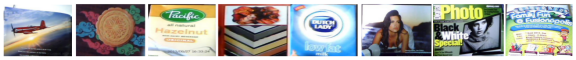
4 Experiments

4.1 Settings

In this paper, we adopt Edge Boxes [Zitnick and Dollár, 2014] to generate 300 object proposals for each frame of the videos. For each object proposal, we further extract its feature by max-pooling the last convolutional layer of VGG-16 CNN model [Simonyan and Zisserman, 2014] pre-trained on Imagenet dataset. The max-pooled 512-dimensional features are further post-processed by principal component analysis (PCA) and whitening in order to suppress the burstiness but the dimension of the feature is kept as 512.

Observing that the object proposals from the same frame tend to overlap with each other and there exist strong redundancy among them. To boost the efficiency of the training process, we use k-means clustering to group 300 object proposals from every frame into 30 clusters and select the centroids of the clusters as compact object proposals. In this scheme, each frame will be represented by 30 compact object proposals and the whole video will be represented by a pool of $30m$ compact object proposals, where m is the number of frames.

(a) CNN2h



(b) Egocentric1

(c) Egocentric2

Figure 2: Query Objects Visualization.

4.2 Evaluation Metric and Dataset

The effectiveness of the proposed method is evaluated by mean average precision (mAP). We conduct the systematic experiments on CNN-2h [Araujo *et al.*, 2014], Egocentric1 [Chandrasekhar *et al.*, 2014] and Egocentric2 [Chandrasekhar *et al.*, 2014] datasets. The CNN-2h dataset contains a long video of $72,000$ frames and we equally divide them into 100 videos. Each video consists of 720 frames and 21600 object proposals. The Egocentric1 dataset consists of 19 long videos. We uniformly sample the keyframes per 20 frames, obtain 51804 keyframes and further equally divide all the keyframes into 101 videos. Each video consists of around 500 keyframes and around 15000 object proposals. The Egocentric2 dataset consists of 21 long videos. We uniformly sample the keyframes per 20 frames, obtain 57802 keyframes and equally divide all the keyframes into 114 videos. Each video consists of around 500 keyframes and around 15000 object proposals.

For the evaluation of object instance retrieval, we used eight query objects for each dataset and skipped those scene query images. Figure 2 visualizes the query objects. In fact, the number of frames where query object appears is much fewer than the total number of frames in the video. We define the frame ratio $\eta(q, V_i)$ as the number of frames in V_i which contain the query object q divided by the number of total frames in the video V_i. Furthermore, we define $\hat{\eta}(q)$ as the average frame ratio of q which is computed by:

$$\hat{\eta}(q) = \sum_{V\in\mathcal{V}_q} \eta(q, V)/|\mathcal{V}_q|, \tag{11}$$

where $\mathcal{V}_q$ denotes the set of videos which are relevant to q. Table 1 shows the average ratio of all the queries on three datasets. It can be seen that the average ratios of queries are quite small. For example, on Egocentric1 dataset, the average frame ratio of 5-th query is only 1.2%. Since each video of Egocentric1 dataset contains around 500 keyframes, it means that, on average, the query object only appears in only 6 keyframes in the relevant videos.

	1	2	3	4	5	6	7	8	Avg
CNN2h	5.0%	10.2%	3.9%	2.0%	3.5%	14.5%	5.6%	9.1%	6.7%
Egocentric1	2.0%	1.4%	1.6%	2.2%	1.2%	3.6%	0.8%	1.1%	1.7%
Egocentric2	1.6%	2.0%	1.8%	0.9%	1.8%	3.6%	4.1%	5.4%	2.7%

Table 1: Average frame ratios of 8 queries on three datasets.

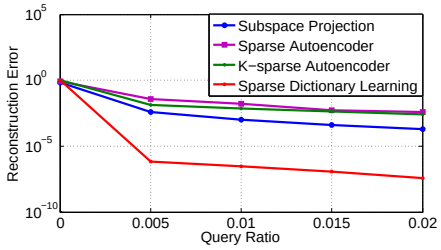


Figure 3: The reconstruction error decreases as the ratio of the query increases.

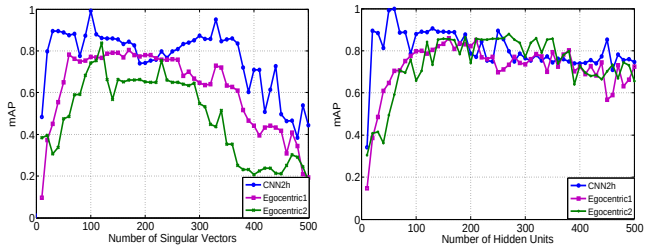
4.3 Effectiveness of ROSE Method

We evaluate the performance of the proposed Reconstruct-based Object SEarch (ROSE) method implemented by subspace projection, sparse autoencoder, k-sparse autoencoder and sparse dictionary learning, in the video-level object instance retrieval task in this section.

Before we evaluate the search precision of ROSE, we design an experiment to demonstrate the effectiveness of reconstruction model in capturing the relevance of the query with the video consisting of a set of object proposals. Particularly, we choose the first query from the CNN2h dataset as our test query. We choose the first video from the CNN2h dataset consisting of 21000 (700 frames * 30 proposals / frame) object proposals, as the training data to learn the reconstruction model. We manually mix the original training data with multiple duplicates of the feature of the query object to form the new training data. We adjust the ratio of the query in the new training data from 0 to 0.02. For example, we add 420 duplicates of the query object to original 21000 object proposals to achieve 0.02 ratio. We can see from Figure 3 that, the reconstruction error of the query object decreases as the ratio of the query object increases, which verifies the effectiveness of the reconstruction error in characterizing the relevance of the query object with the video.

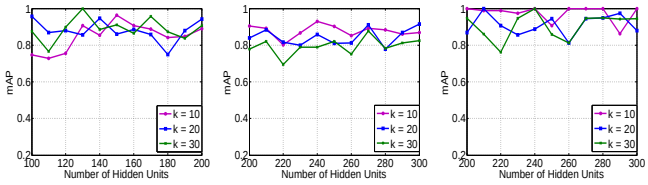
Figure 4(a) shows the mAP from the reconstruction model implemented by subspace projection. We vary the number of singular vectors l from 10 to 500. It is interesting that the mAP increases and then decreases as number of selected singular vectors increase. Particularly, when $l = 10$ and $l = 500$, the mAP is relatively low. This is because of the fact that too few singular vectors can not capture enough information contained in the set of the object proposals. Meanwhile too many singular vectors will make the reconstruction model easily reconstruct any input vectors, which makes it difficult to discriminate different videos.

Figure 4(b) shows the mAP of the proposed reconstruction model implemented by sparse autoenocoder. The encoder activation function $f_1(\cdot)$ is implemented by rectifier defined as $f_1(x) = max(0, x)$. The decoder activation function is implemented by the linear function $f_2(x) = x$. We set the expected average activations $\hat{\rho}$ of all the hidden neural nodes as



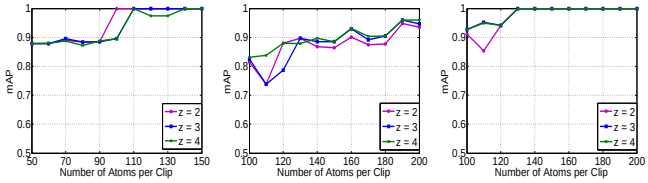
(a) Subspace Projection (b) Sparse Autoencoder

Figure 4: The performance comparison of the reconstruction model implemented by subspace projection and sparse autoencoder.



(a) CNN2h (b) Egocentric1 (c) Egocentric2

Figure 5: The performance of the reconstruction model implemented by k-sparse autoencoder.



(a) CNN2h (b) Egocentric1 (c) Egocentric2

Figure 6: The performance of the reconstruction model implemented by sparse coding.

0.05. It can be observed that from Figure 4(b) that, the reconstruction model implemented by sparse autoencoder can achieve more stable performance compared with the reconstruction model implemented by subspace projection. Particularly, it achieves 0.746 mAP on CNN-2h dataset when the number of hidden units is 500, whereas the subspace projection can only achieves 0.443 mAP when the number of singular vector is 500. The more stable performance can be attributed to the sparse penalty term.

Figure 5 shows the performance of the reconstruction model implemented by k-sparse autoencoder. We vary the number of nonzero activations of the hidden layer k among 10, 20 and 30. Compared with sparse autoencoder, the performance of k-sparse autoencoder is much better and more stable. Especially on Egocentric2 dataset, it can achieve mAP $= 1.00$ when $k = 10$ and the number of hidden units is 200 whereas the sparse autoencoder only achieved around mAP $= 0.75$ when the number of hidden units is 200.

Figure 6 shows the mAP from the reconstruction model implemented by sparse dictionary learning as the number of atoms and the number of non-zero element in the codes z changes. It can be observed that the value of z does not influence the performance significantly when it varies between 2 and 4. Therefore, we set the default number of z as 2 on

Method	CNN-2h		Egocentric1		Egocentric2	
	Memory	mAP	Memory	mAP	Memory	mAP
ROSE-SDL (ours)	400KB	1.00	400KB	0.94	400KB	1.00
SMRS	400KB	0.73	400KB	0.83	400KB	0.62
K-mediods	400KB	0.63	400KB	0.69	400KB	0.89
Exhaustive Search	42MB	1.00	30MB	1.00	30MB	1.00

Table 2: Comparison of our ROSE method with the representative selection method and exhaustive search method.

all three datasets. Compared with the reconstruction model implemented by sparse autoencoder or k-sparse autoencoder, the reconstruction model implemented by sparse dictionary learning is more accurate and stable. It can achieve mAP $= 1.00$ on both CNN-2h and Egocentric2 datasets when the number of atoms per video is only 150. We choose the sparse dictionary learning as the default implementation of the proposed reconstruction model.

4.4 Comparison with Baselines

In order to further demonstrate the effectiveness of the proposed ROSE method, we compare it with exhaustive search scheme and representative selection scheme [Elhamifar *et al.*, 2012]. Particularly, the exhaustive search scheme directly compares the query with all the object proposals from each video to determine the relevance of the video with the query. The representative selection scheme selects a set of representative objects $\mathcal{O}_i$ from the set of object proposals $\mathcal{S}_i$ for each video V_i. The relevance of the video V_i with the query $\boldsymbol{q}$ is determined by the similarity score of the best-matched representatives as

$$R(\boldsymbol{q}, V_i) = \max_{\boldsymbol{o} \in \mathcal{O}_i} \text{sim}(\boldsymbol{q}, \boldsymbol{o}) \tag{12}$$

We compare with two types of representative selection methods. They are k-mediods [Park and Jun, 2009] and SMRS [Elhamifar *et al.*, 2012]. The reconstruction model we use for comparison is implemented by sparse dictionary learning and the number of non-zero elements in the codes z is set as 2. We conduct the comparison on the condition when the number of atoms/representatives l is set to be 200, which costs 400KB memory for each video.

From Table 2, we can see that our ROSE implemented by sparse dictionary learning (ROSE-SDL) achieves comparable search precision as exhaustive search but takes much less memory. For example, on the CNN2h dataset, the exhaustive search takes 42MB to achieve mAP $= 1.00$. In contrast, to achieve mAP $= 1.00$, our ROSE-SDL only takes 400KB memory, which is only around 1% as that for the exhaustive search method. Meanwhile, using comparable memory cost, our ROSE-SDL achieves significantly higher precision than the representative selection implemented by k-mediods and SMRS on all three datasets. For example, on CNN-2h dataset, the proposed reconstruction model can achieve mAP $= 1.00$, whereas the mAP of the representative selection scheme implemented by k-mediods and SMRS are both less than 0.75.

We visualize the search results from our ROSE-DSL method on CNN2h, Egocentric1 and Egocentric2 datasets in Figure 7. For each dataset, we show one query and its top-3 retrieved videos. We can see that, in many cases the query object only appears in few frames of the video and occupies a very small area surrounded by dense clutter in the frame but our method can still successfully retrieve the videos where the query object appears.

(a) CNN2h

(b) Egocentric1

(c) Egocentric2

Figure 7: Visualization of top-3 search results of the proposed ROSE method on three datasets.

5 Conclusion

Our work provides a novel perspective to video-level object instance search and proposes the Reconstruction-based Object SEarch (ROSE) method to achieve search efficiency. In this method, a huge number of object proposals generated from the frames of a video are compactly characterized by the parameters of a learned reconstruction model. Then the video-level object instance search is converted into computing the reconstruction error of the query object using the reconstruction model learned for each video. It is significantly faster than exhaustively comparing the query object with all the object proposals from the video. Meanwhile, storing the reconstruction model of the video greatly reduces the memory cost. Comprehensive experiments on three benchmark datasets verify the high efficiency and effectiveness of our ROSE method.

Acknowledgements

This work is supported in part by Singapore Ministry of Education Academic Research Fund Tier 2 MOE2015-T2-2-114. This research was carried out at the ROSE Lab at the Nanyang Technological University, Singapore. The ROSE Lab is supported by the National Research Foundation, Prime Ministers Office, Singapore, under its IDM Futures Funding Initiative and administered by the Interactive and Digital Media Programme Office.

References

[Araujo *et al.*, 2014] A Araujo, Mina Makar, Vijay Chandrasekhar, D Chen, S Tsai, Huizhong Chen, Roland Angst, and Bernd Girod. Efficient video search using image queries. In *ICIP*, pages 3082–3086. IEEE, 2014.

[Bhattacharjee *et al.*, 2016] Sreyasee Das Bhattacharjee, Junsong Yuan, Yap-Peng Tan, and Ling-Yu Duan. Query-adaptive small object search using object proposals and shape-aware descriptors. *IEEE Transactions on Multimedia*, 18(4):726–737, 2016.

[Chandrasekhar *et al.*, 2014] Vijay Chandrasekhar, Wu Min, Xiao Li, Cheston Tan, Bappaditya Mandal, Liyuan Li, and Joo Hwee Lim. Efficient retrieval from large-scale egocentric visual data using a sparse graph representation. In *CVPR Workshops*, pages 527–534, 2014.

[Cong *et al.*, 2011] Yang Cong, Junsong Yuan, and Ji Liu. Sparse reconstruction cost for abnormal event detection. In *CVPR*, pages 3449–3456. IEEE, 2011.

[Elhamifar *et al.*, 2012] Ehsan Elhamifar, Guillermo Sapiro, and Rene Vidal. See all by looking at a few: Sparse modeling for finding representative objects. In *CVPR*, pages 1600–1607. IEEE, 2012.

[Hayat *et al.*, 2015] Munawar Hayat, Mohammed Bennamoun, and Senjian An. Deep reconstruction models for image set classification. *IEEE transactions on pattern analysis and machine intelligence*, 37(4):713–727, 2015.

[Jiang *et al.*, 2015] Yuning Jiang, Jingjing Meng, Junsong Yuan, and Jiebo Luo. Randomized spatial context for object search. *IEEE Trans. Image Processing*, 24(6):1748–1762, 2015.

[Mairal *et al.*, 2009] Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th annual international conference on machine learning*, pages 689–696. ACM, 2009.

[Makhzani and Frey, 2013] Alireza Makhzani and Brendan J. Frey. k-sparse autoencoders. *CoRR*, abs/1312.5663, 2013.

[Meng *et al.*, 2016a] Jingjing Meng, Hongxing Wang, Junsong Yuan, and Yap-Peng Tan. From keyframes to key objects: Video summarization by representative object proposal selection. In *CVPR*, pages 1039–1048. IEEE, 2016.

[Meng *et al.*, 2016b] Jingjing Meng, Junsong Yuan, Jiong Yang, Gang Wang, and Yap-Peng Tan. Object instance search in videos via spatio-temporal trajectory discovery. *IEEE Transactions on Multimedia*, 18(1):116–127, 2016.

[Ng, 2011] Andrew Ng. Sparse autoencoder. *CS294A Lecture notes*, 72:1–19, 2011.

[Park and Jun, 2009] Hae-Sang Park and Chi-Hyuck Jun. A simple and fast algorithm for k-medoids clustering. *Expert Systems with Applications*, 36(2):3336–3341, 2009.

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

[Sivic and Zisserman, 2009] Josef Sivic and Andrew Zisserman. Efficient visual search of videos cast as text retrieval. *IEEE transactions on pattern analysis and machine intelligence*, 31(4):591–606, 2009.

[Tao *et al.*, 2015] Ran Tao, Arnold WM Smeulders, and Shih-Fu Chang. Attributes and categories for generic instance search from one example. In *CVPR*, pages 177–186. IEEE, 2015.

[Tolias *et al.*, 2016] Giorgos Tolias, Ronan Sicre, and Hervé Jégou. Particular object retrieval with integral max-pooling of cnn activations. In *Proc. International Conference on Learning Representations (ICLR)*, 2016.

[Tu *et al.*, 2014] Zhigang Tu, Nico Van Der Aa, Coert Van Gemeren, and Remco C Veltkamp. A combined post-filtering method to improve accuracy of variational optical flow estimation. *Pattern Recognition*, 47(5):1926–1940, 2014.

[Wright *et al.*, 2009] John Wright, Allen Y Yang, Arvind Ganesh, S Shankar Sastry, and Yi Ma. Robust face recognition via sparse representation. *IEEE transactions on pattern analysis and machine intelligence*, 31(2):210–227, 2009.

[Xia *et al.*, 2015] Yan Xia, Xudong Cao, Fang Wen, Gang Hua, and Jian Sun. Learning discriminative reconstructions for unsupervised outlier removal. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1511–1519, 2015.

[Yang and Satoh, 2013] Yan Yang and Shin'ichi Satoh. Efficient instance search from large video database via sparse filters in subspaces. In *2013 IEEE International Conference on Image Processing*, pages 3972–3976. IEEE, 2013.

[Yang *et al.*, 2015] Huan Yang, Baoyuan Wang, Stephen Lin, David Wipf, Minyi Guo, and Baining Guo. Unsupervised extraction of video highlights via robust recurrent auto-encoders. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4633–4641, 2015.

[Yu *et al.*, 2017a] Tan Yu, Yuwei Wu, Sreyasee Das Bhattacharjee, and Junsong Yuan. Efficient object instance search using fuzzy objects matching. In *AAAI*, pages 4320–4326, 2017.

[Yu *et al.*, 2017b] Tan Yu, Yuwei Wu, and Junsong Yuan. Hope: Hierarchical object prototype encoding for efficient object instance search in videos. In *CVPR*. IEEE, 2017.

[Zhu and Satoh, 2012] Cai-Zhi Zhu and Shin'ichi Satoh. Large vocabulary quantization for searching instances from videos. In *ICMR*, page 52, 2012.

[Zitnick and Dollár, 2014] C Lawrence Zitnick and Piotr Dollár. Edge boxes: Locating object proposals from edges. In *ECCV*, pages 391–405. Springer, 2014.