Guided Generative Models using Weak Supervision for Detecting Object Spatial Arrangement in Overhead Images

Weiwei Duan
University of Southern California
weiweiduan@usc.edu

Yao-Yi Chiang
University of Minnesota
yaoyi@umn.edu

Stefan Leyk
University of Colorado Boulder
stefan.leyk@colorado.edu

Johannes H. Uhl
University of Colorado Boulder
johannes.uhl@colorado.edu

Craig A. Knoblock
University of Southern California
knoblock@isi.edu

***Abstract*—The increasing availability and accessibility of numerous overhead images allows us to estimate and assess the spatial arrangement of groups of geospatial target objects, which can benefit many applications, such as traffic monitoring and agricultural monitoring. Spatial arrangement estimation is the process of identifying the areas which contain the desired objects in overhead images. Traditional supervised object detection approaches can estimate accurate spatial arrangement but require large amounts of bounding box annotations. Recent semi-supervised clustering approaches can reduce manual labeling but still require annotations for all object categories in the image. This paper presents the target-guided generative model (TGGM), under the Variational Auto-encoder (VAE) framework, which uses Gaussian Mixture Models (GMM) to estimate the distributions of both hidden and decoder variables in VAE. Modeling both hidden and decoder variables by GMM reduces the required manual annotations significantly for spatial arrangement estimation. Unlike existing approaches that the training process can only update the GMM as a whole in the optimization iterations (e.g., a "minibatch"), TGGM allows the update of individual GMM components separately in the same optimization iteration. Optimizing GMM components separately allows TGGM to exploit the semantic relationships in spatial data and requires only a few labels to initiate and guide the generative process. Our experiments shows that TGGM achieves results comparable to the state-of-the-art semi-supervised methods and outperformes unsupervised methods by 10% based on the F_1 scores, while requiring significantly fewer labeled data.**

I. Introduction

Overhead images provide data to unlock information unseen from the ground. Detecting locations for groups of similar-sized geospatial objects of interest (i.e., target objects) is an important computer vision task, which can benefit various applications, such as detecting a group of cars to monitor busy hours in parking lots [1], [2], surveying crowds for disease monitoring [3], and detecting a group of trees in plantations for agricultural monitoring purposes [4], [5]. We define detecting the location of a group of target objects as estimating the spatial arrangement of target objects in this paper. The state-of-the-art supervised object detectors [6], [7], [8] can estimate the spatial arrangement of target object locations by obtaining a precise location of each target object but require thousands of bounding-box level annotations for the specific objects. However, overhead images, such as satellite imagery and scanned topographic maps, do not have sufficient bounding-box level annotations to train supervised object detectors. Although many public scenic image datasets, such as PASCAL Visual Object Classes (VOC) Challenge [9], provide bounding-box level annotations, transfer learning technologies [10], [11], [12] cannot effectively transfer knowledge from scenic images to overhead images because annotations for scenic images are typically for large and notable objects, while overhead images often contain small and densely clustered or dispersed objects over a large area. The spatial arrangement estimation for a group of similar-sized target objects, such as cars, does not require precise location for each target object. Therefore, the bounding-box level annotations are not necessary.

Annotating a region of interest (ROI) and a few boxes covering target objects within the ROI is sufficient to detect a group of target objects in overhead images. ROI is a region covering a group of target objects in overhead images. For example, most of the overlapping green boxes at the bottom right panel in Figure 1 represent areas with a significant overlap with one or more cars. These boxes, typically generated using a sliding-window-based method, can be used to estimate the spatial arrangement of car objects in a parking lot. Therefore, a sliding-window-based method is often sufficient to estimate the spatial arrangement of a group of objects within an ROI.

A sliding-window-based method uses a fix-size window sliding across an ROI and detects the windows that contain the target objects within the ROI. The group of detected windows containing the target objects is assumed to be representive of the spatial arrangement of the target objects. In this case, the sliding-window-based method aims to group window areas in an ROI into two categories: target vs. non-target categories (e.g., cars vs. other objects in a parking lot). The solutions to this type of clustering task can be either unsupervised [13], [14], [15] or semi-supervised methods [16], [17], [18].

Unsupervised clustering methods [14], [13] discover the separable patterns in an ROI to separate windows into clusters. However, the target-vs.-non-target pattern might not be the

only and dominant separable pattern within an ROI. An unsupervised clustering method could use other image patterns (e.g., lighting or texture) to separate the windows. Hence, results from unsupervised methods are not robust for various types of target objects and ROIs (i.e., the target objects might not be grouped into one cluster). In contrast, semi-supervised models [17], [18], [16] can separate the windows into a specific target and non-target category by using annotated areas of example objects (labeled windows containing target and non-target objects) to guide the clustering process. However, these methods typically require substantial manual work of labeling hundreds of target and non-target examples.

This paper proposes a target-guided generative model (TGGM), which exploits a few labeled target windows to detect the spatial arrangement of the target objects within an ROI in overhead images. TGGM has two main advantages: 1. TGGM only needs labeled windows for the target category instead of all categories required by semi-supervised methods; 2. TGGM reduces the number of required labeled target windows from hundreds in the existing approaches to just a few. TGGM exploits a few labeled target windows to initialize the target cluster and then iteratively optimizes the clustering process by accumulating windows assigned to the target cluster. This way, the target cluster can eventually cover all target objects with diverse appearances within an ROI. TGGM also leverages a unique property in spatial data. The ROI boundaries are carefully selected to have a strong semantic relationship with the target objects (e.g., parking lots and cars). Also, within the ROI, the non-target objects are similar (e.g., most non-car areas in a parking lot are parking lot surfaces). If the input image is geo-referenced, TGGM can use external geographic data sources (e.g., OpenStreetMap[1]) to automatically generate the desired ROI boundaries (e.g., parking lot boundaries). In other cases, annotating the ROI boundary is a straightforward task compared to annotating a large number of bounding boxes for objects within the ROI.

Figure 1 illustrates how TGGM detects the target windows (e.g., windows overlapping with one ore more cars) within the ROI iteratively. The first column shows how to generate fixed-size windows over the blue ROI. The window has approximately the same size as the target objects (e.g., cars in the ROI). The second column shows the generated windows across the ROI. The non-blurry windows in the second column indicate the initial manually labeled target windows (in iteration 1) or the detected target windows (in the rest of iterations). The green areas in the third column represent the detected windows containing cars in the parking lot. TGGM leverages an iterative detection process that takes advantage of the detected target windows for the next iteration. The iteration ends when TGGM cannot detect new target windows. The bottom right panel in Figure 1 shows that TGGM estimates the spatial arrangement of cars in the parking lot.

TGGM is a weakly-supervised probabilistic clustering framework based on the Variational Auto-encoder and Gaussian Mixture Models. The main contribution of TGGM is that it can exploit a few labeled data to guide the clustering process so that the data in one of the resulting clusters are similar to the labeled data. Specifically, unlike the existing approaches in which the training process can only update the Mixture of Gaussians (MoG) as a whole or use a subset of the data to update a specific component in the MoG in separate optimization iterations (e.g., a "minibatch"), TGGM's network design allows the update of individual components in the MoG separately in the same optimization iteration. Using labeled and unlabeled data in one optimization iteration is important because when the data in some categories are (partially) labeled, an optimization iteration should update the MoG using both labeled and unlabeled data so that the data distributions of some specific components in MoG is specific for the partially labeled data in some categories.

Without loss of generality, here, we assume 1) the input data contain two clusters: the target and the non-target clusters, and 2) only limited labeled data in the target category are available. TGGM assumes the hidden space is an MoG where the number of components is the number of the desired clusters

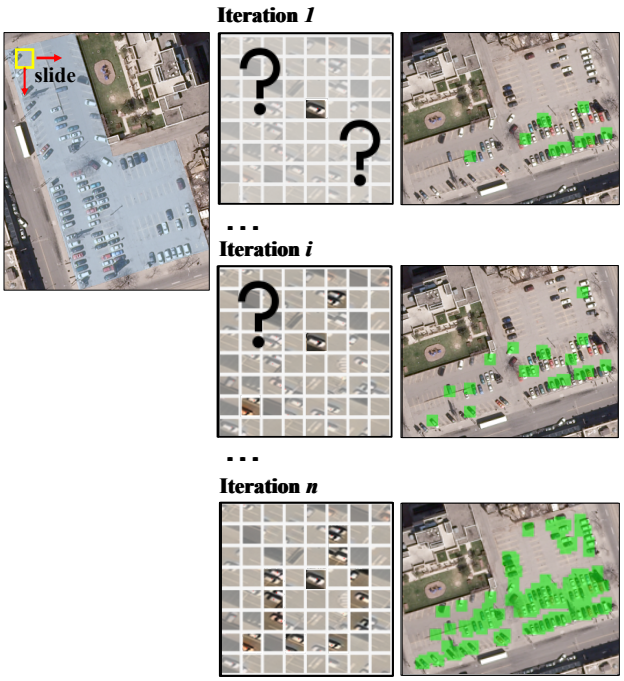


Fig. 1: The first column shows that TGGM uses a window (similar to the size of a car) sliding across the blue ROI. The second column shows the generated windows across the ROI. The non-blurred and blurred images in the second column represent labeled target windows and unlabeled windows, respectively. In the first iteration, the non-blurred image is manually annotated. TGGM takes the manually labeled target window and unlabeled windows as inputs first and detects more windows covering cars in the green areas in the top panel in the third column. The detected target windows are inputs for the next iteration to help detect new target windows. The iteration ends when TGGM cannot detect new target windows. In the end, the green areas in the bottom left panel show the detected target windows representing the spatial arrangement of cars in the parking lot.

[1] https://www.openstreetmap.org/

(e.g., two components in this case) like the other generative clustering approaches [13]. Unlike other approaches, TGGM can use both the labeled and unlabeled data in one optimization iteration. Initially, the labeled data provide examples for the target cluster, and the unlabeled data can contain either the target or non-target data. In one optimization iteration, for the labeled and unlabeled data in the target cluster from the previous iteration, TGGM only updates the parameters for the target component in the MoG. For all other unlabeled data, TGGM updates the entire MoG. The result is that TGGM iteratively learns the MoG parameters that best describe the target data in one component and all other data in the other component. We call this process the target guidance mechanism.

In sum, TGGM exploits a few manually labeled target windows in a spatially-constrained ROI to iteratively detect the spatial arrangement of the target objects within the ROI in overhead images. This paper's contribution is that TGGM provides the optimization procedure that leverages both labeled target and unlabeled windows in one optimization iteration. The optimization procedure enables using the labeled target windows and unlabeled windows to guide the formation of the target and non-target clusters. The resulting windows in the target cluster represent the spatial arrangement of the target objects in the overhead images.

II. Related Work

Weakly supervised object detection [19], [20], [21], [22], [23], [12], [11], [24], [25], [26], [27], aiming to localize objects with image-level annotations, has attracted attention because bounding box annotations require intensive manual work. However, existing weakly supervised object detectors focus on scenic images, such as images in the PASCAL VOC Challenge [9]. The target objects, such as cats, are usually large and notable in scenic images. Because of the notability property and proportional size of target objects in scenic images, existing methods transfer prior knowledge learned from scenic images [11], [12] or utilize the multiple instances learning methods [19] to localize the objects using image annotations. However, the notability property of target objects is usually not valid in overhead images [28], [29]. The target objects in satellite imagery, like cars, are small, densely clustered or spatially dispersed over large areas [30]. Therefore, the weakly supervised object detectors cannot directly apply on overhead images.

Sliding-window-based detectors are commonly used to detect if a window covers small target objects in overhead images [31], [29], [30], [32], because The state-of-the-art object detectors [7], [8] lose the information about small objects when taking entire satellite imagery as the input because of down-sampling operations. The sliding window is a straightforward way to search for small objects across imagery. Detecting if the sliding window contains the target objects is equivalent to classifying the images into the target or non-target categories. State-of-the-art image classifiers without or with a limited number of labeled images are unsupervised and semi-supervised methods.

Unsupervised methods separate data into multiple clusters by identifying separable patterns in the data. Generative models under the Variational Auto-encoder (VAE) framework are common unsupervised clustering methods [15], [13], [18], [33], [14], which learn the distribution to represent and separate inputs in the hidden space. Generative clustering models have shown impressive results by learning flexible distribution representations in the hidden space. However, images in sliding windows have many separable patterns, such as light-vs.-dark or target-vs.-non-target patterns. The methods do not produce target and non-target clusters when unsupervised methods use other separable patterns to cluster the images. In contrast, the target guidance mechanism guides TGGM to separate images in sliding windows into the target and non-target clusters.

Semi-supervised learning methods [34], [18], [16], [35], [36], [37], [38], [39], [40], [41], [42] aim to automatically annotate large amounts of unlabeled data using a small set of labeled data in each category. Recent work [18], [16] shows that unsupervised generative clustering models can convert to semi-supervised models easily by adding a cross-entropy loss for labeled data. The cross-entropy loss optimized by labeled data helps to improve the clustering results. However, labeling both target and non-target windows can require a huge amount of manual work. In contrast, TGGM reduces the manual work for non-target window annotations and separates windows into the target and non-target categories by leveraging the strong semantic relationship between the ROI and the target objects.

III. Target-Guided Generative Model

First, we introduce symbols used in the following subsections. $\boldsymbol{x}_t$ represents labeled target windows. $\boldsymbol{x}_u$ is unlabeled windows. $\hat{\boldsymbol{x}}_u$ stands for the generated unlabeled windows from TGGM. $\boldsymbol{z}$ is a continuous variable to represent the distribution of $\boldsymbol{x}_t$ and $\boldsymbol{x}_u$ in the hidden space. y is a categorical variable representing the labels for $\boldsymbol{x}_u$. $y = \{0, 1\}$, 1 for target windows and 0 for non-target windows.

Figure 2 shows TGGM's architecture. f_{inf} encodes $\boldsymbol{x_u}$ concatenating with y into $\boldsymbol{z}$, then f_{gen} decodes $\boldsymbol{z}$ into $\hat{\boldsymbol{x}}_{\boldsymbol{u}}$. $\hat{\boldsymbol{x}}_{\boldsymbol{u}}$ is the reconstructed $\boldsymbol{x_u}$. f_{cls} assigns $\boldsymbol{x_u}$ into clusters. Section 3.1 describes the distribution of $\boldsymbol{z}$ given $\boldsymbol{x_u}$ and y learned by f_{inf} and f_{cls}. Section 3.2 describes the process for f_{prior}. Section 3.3 formulates the evidence lower bounds of the marginal likelihood of $\boldsymbol{x_u}$ and $\boldsymbol{x_t}$ to optimize TGGM. Section 3.4 explains how TGGM leverages labeled target windows to form target and non-target clusters.

A. The Variational Inference

TGGM learns the approximation posterior distribution for $\boldsymbol{z}$, i.e., $q(\boldsymbol{z}|\boldsymbol{x}_u)$ to estimate the true posterior probability $p(\boldsymbol{z}|\boldsymbol{x}_u)$ which is intractable [43]. The approximate posterior probability of $\boldsymbol{z}$ is MoG with two Gaussian components showing in eq. 1. The parameters of Gaussian components are learned by f_{inf} in eq. 2. The weights of two components are the probabilities of the hidden categorical variable y in eq. 3. The hidden categorical variable y is learned by f_{cls} activated by the

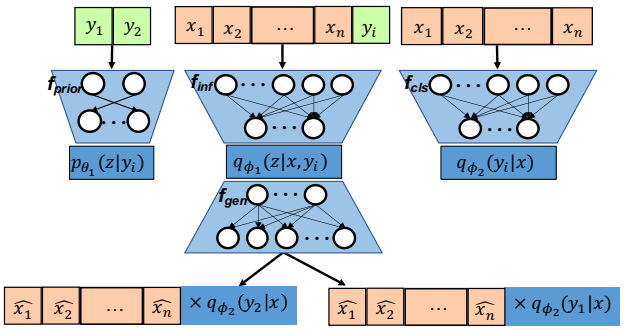


Fig. 2: TGGM's architecture. f_{inf} in the center model encodes $\boldsymbol{x_u}$ concatenating with y into the distribution of the hidden variable z, and f_{gen} decodes $\boldsymbol{z}$ into $\hat{\boldsymbol{x}}_{\boldsymbol{u}}$ weighted by $q(y|\boldsymbol{x_u})$ from f_{cls} represented by the right model. Section 3.1 describes f_{inf} and f_{cls}. Section 3.2 describes f_{prior} represented by the left model. Sections 3.3 and 3.4 describe the optimization procedure.

softmax function. The joint approximate posterior probability of $\boldsymbol{z}$ and y is shown in eq. 4.

$$q(\boldsymbol{z}|\boldsymbol{x}_u) = \sum_y q(y|\boldsymbol{x}_u)q(\boldsymbol{z}|\boldsymbol{x}_u, y) \tag{1}$$

$$q(\boldsymbol{z}|\boldsymbol{x}_u, y) = \mathcal{N}(\tilde{\mu_{\boldsymbol{z}}}, \tilde{\sigma_{\boldsymbol{z}}}), \quad [\tilde{\mu_{\boldsymbol{z}}}, \tilde{\sigma_{\boldsymbol{z}}}] = f_{inf}(\boldsymbol{x}_u, y) \tag{2}$$

$$q(y|\boldsymbol{x}_u) = Cat(y|\pi(\boldsymbol{x}_u)) = softmax(f_{cls}(\boldsymbol{x}_u)) \tag{3}$$

$$q(\boldsymbol{z}, y|\boldsymbol{x}_u) = q(\boldsymbol{z}|\boldsymbol{x}_u, y)q(y|\boldsymbol{x}_u) \tag{4}$$

B. The Generative Process for Windows

Generating a target window follows the steps below:

1) Sample a hidden vector $\boldsymbol{z}$ from $p(\boldsymbol{z}|y=1)$
2) Compute the distribution of $\boldsymbol{x}_t$ following $\mathcal{N}(\mu_{x_t}, \sigma_{x_t})$
3) Sample a $\boldsymbol{x}$ from $\mathcal{N}(\mu_{x_t}, \sigma_{x_t})$

The generative process for non-target windows is similar to the above process, except sampling from $p(\boldsymbol{z}|y=0)$ and sampling $\boldsymbol{x}$ from the Gaussian distribution for non-target windows. $p(\boldsymbol{z}|y)$ in step 1 above follows a Gaussian distribution showing in eq. 5. The parameters of Gaussian distribution are learned by $f_{prior}(y)$ in eq. 6. The parameters of $\mathcal{N}(\mu_{x_t}, \sigma_{x_t})$ in step 2 are learned by f_{gen} in eq. 7.

$$p(\boldsymbol{z}|y) = \mathcal{N}(\mu_{\boldsymbol{z}}, \sigma_{\boldsymbol{z}}) \tag{5}$$

$$[\mu_{\boldsymbol{z}}, \sigma_{\boldsymbol{z}}] = f_{prior}(y) \tag{6}$$

$$[\mu_{x_t}, \sigma_{x_t}] = f_{gen}(\boldsymbol{z}) \tag{7}$$

According to the generative process above, the joint probability $p(\boldsymbol{x}, \boldsymbol{z}, y)$ can be factorized as:

$$p(\boldsymbol{x}, \boldsymbol{z}, y) = p(\boldsymbol{x}|\boldsymbol{z}, y)p(\boldsymbol{z}|y)p(y) \tag{8}$$

Where $p(y)$ is the prior distribution for y defined in eq. 9.

$$p(y) = Cat(1/K) \tag{9}$$

Where K is the number of categories. In our case, the number of categories is two, one for the target category and the other for the non-target one. At the beginning of the iterations, TGGM assumes that the number of target and non-target windows, i.e., $p(y)$ is uniformly distributed since we do not have prior knowledge about the number of target objects in the ROI. This assumption is relaxed in later iterations.

C. The Generative and Inference Processes for Labeled Windows

For the labeled target windows, y becomes an observation instead of a hidden variable, i.e., $y=1$. Because y is an observation, TGGM only uses labeled target windows to learn one Gaussian component of $\boldsymbol{z}$ and generate windows from the component. As a result, the approximate posterior distribution of $\boldsymbol{z}$ and generative $\boldsymbol{x}_t$ distribution can be written as eq. 10 and eq. 11, respectively.

$$q(\boldsymbol{z}|\boldsymbol{x}_t) = q(\boldsymbol{z}|\boldsymbol{x}_t, y=1) \tag{10}$$

$$p(\boldsymbol{x}_t|\boldsymbol{z}, y=1) = p(\boldsymbol{x}_t|\boldsymbol{z}, y=1)p(\boldsymbol{z}|y=1) \tag{11}$$

D. Evidence Lower Bounds

TGGM aims to separate target and non-target windows into two clusters by maximizing the marginal likelihood of unlabeled windows, $\boldsymbol{x}_u$. Because the log-likelihood of $\boldsymbol{x}_u$ (the leftmost term in eq. 12) is intractable, TGGM optimizes the evidence lower bound ($\mathcal{L}_{ELBO}$) (the rightmost term in eq. 12) instead.

$$\begin{aligned}\log p(\boldsymbol{x}_u) &= \log \sum_y \int_{\boldsymbol{z}} p(\boldsymbol{x}_u, \boldsymbol{z}, y) d\boldsymbol{z} \\ &\geq \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log \frac{p(\boldsymbol{x}_u, \boldsymbol{z}, y)}{q(\boldsymbol{z}, y|\boldsymbol{x}_u)}\Big] = \mathcal{L}_{ELBO}(\boldsymbol{x}_u)\end{aligned} \tag{12}$$

Eq. 13 shows the details of $\mathcal{L}_{ELBO}$ for unlabeled windows. $\mathcal{L}_{ELBO}$ after using the reparameterization trick proposed in VAE [43] can be written as:

$$\begin{aligned}\mathcal{L}_{ELBO}(\boldsymbol{x}_u) &= \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log \frac{p(\boldsymbol{x}_u, \boldsymbol{z}, y)}{q(\boldsymbol{z}, y|\boldsymbol{x}_u)}\Big] \\ &= \sum_y q(y|\boldsymbol{x}_u) \log p(\boldsymbol{x_u}|\boldsymbol{z}, y) \\ &\quad - \sum_y q(y|\boldsymbol{x}_u)\mathcal{KL}\Big[q(\boldsymbol{z}|\boldsymbol{x}_u, y)\|p(\boldsymbol{z}|y)\Big] \\ &\quad - \mathcal{KL}\Big[q(y|\boldsymbol{x}_u)\|p(y)\Big]\end{aligned} \tag{13}$$

According to the generative and variational inference processes for labeled target windows, $\boldsymbol{x}_t$, described in the last subsection, TGGM uses $\boldsymbol{x}_t$ to optimize one Gaussian component in $\boldsymbol{z}$ and generate $\boldsymbol{x}_t$ from the component. $\mathcal{L}_{ELBO}$ for $\boldsymbol{x}_t$ is defined in eq. 14.

$$\begin{aligned}\mathcal{L}_{ELBO}(\boldsymbol{x}_t) &= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big[\log \frac{p(\boldsymbol{x}_t, \boldsymbol{z}, y=1)}{q(\boldsymbol{z}|\boldsymbol{x}_t, y=1)}\Big] \\ &= \log p(\boldsymbol{x_t}|\boldsymbol{z}, y=1) \\ &\quad - \mathcal{KL}\Big[q(\boldsymbol{z}|\boldsymbol{x}_t, y=1)\|p(\boldsymbol{z}|y=1)\Big]\end{aligned} \tag{14}$$

The total $\mathcal{L}_{ELBO}$ as the optimization goal of TGGM is the summation of $\mathcal{L}_{ELBO}$ for $\boldsymbol{x}_u$ and $\boldsymbol{x}_t$ showing in eq. 15.

$$\mathcal{L}_{ELBO}(\boldsymbol{x}) = \mathcal{L}_{ELBO}(\boldsymbol{x}_u) + \mathcal{L}_{ELBO}(\boldsymbol{x}_t) \tag{15}$$

E. Network Design

Figure 2 shows TGGM's network architecture. When the input is a labeled target window $\boldsymbol{x}_t$, the encoder (f_{inf}) in TGGM encodes $\boldsymbol{x}_t$ concatenating with $y = 1$ into $q(\boldsymbol{z}|\boldsymbol{x}_u, y = 1)$, then decoder ($f_{gen}$) sampled $\boldsymbol{z}$ from $p(\boldsymbol{z}|y = 1)$ into $\hat{\boldsymbol{x}}_{\boldsymbol{t}}$. In the optimization process, the optimization goal of the second term in eq. 14 is that one component of the hidden variable $\boldsymbol{z}$ represents the labeled target windows. The optimization goal of the first term in eq. 14 is that the decoder generates similar $\hat{\boldsymbol{x}}_{\boldsymbol{t}}$ to $\boldsymbol{x}_t$ from the component of $\boldsymbol{z}$ which represent labeled target windows in the hidden space. TGGM explicitly learns one component in $\boldsymbol{z}$ to represent the labeled target windows. We call this learning process the target guidance mechanism.

When the input is a unlabeled window, $\boldsymbol{x}_u$, f_{inf} encodes $\boldsymbol{x}_u$ concatenating with y into the hidden variable $\boldsymbol{z}$. f_{gen} decodes the sampled $\boldsymbol{z_0}$ from $p(\boldsymbol{z}|y = 0)$ and $\boldsymbol{z_1}$ from $p(\boldsymbol{z}|y = 1)$ into $\hat{\boldsymbol{x}}_{u0}$ and $\hat{\boldsymbol{x}}_{u1}$. In the optimization process, the optimization goal of first term in eq. 13 is assigning a high weight ($q(y|\boldsymbol{x}_u)$) to $\hat{\boldsymbol{x}}_{u0}$ or $\hat{\boldsymbol{x}}_{u1}$, whichever is more similar to $\boldsymbol{x}_u$. For example, when the input is an unlabeled window covering the target objects, the $\hat{\boldsymbol{x}}_{u1}$ from $p(\boldsymbol{z}|y = 1)$ should be more similar to $\hat{\boldsymbol{x}}_u$ than $\hat{\boldsymbol{x}}_{u0}$ from $p(\boldsymbol{z}|y = 0)$ because $p(\boldsymbol{z}|y = 1)$, optimized by labeled target windows in the target guidance mechanism, represents the target windows. The weight for $\hat{\boldsymbol{x}}_{u1}$ is higher than the weight for $\hat{\boldsymbol{x}}_{u0}$ (i.e., $(q(y = 1|\boldsymbol{x}_u) > q(y = 0|\boldsymbol{x}_u))$

Similarly, the optimization goal of the second term in eq. 13 is assigning a high weight ($q(y|\boldsymbol{x}_u)$) to one component of $\boldsymbol{z}$, which has the smaller *KL* value than the other component does. For example, when the input is an unlabeled window covering the target object, the *KL* value between $q(\boldsymbol{z}|\boldsymbol{x}_u, y = 1)$ and $p(\boldsymbol{z}|y = 1)$ should be smaller than the *KL* value between $q(\boldsymbol{z}|\boldsymbol{x}_u, y = 0)$ and $p(\boldsymbol{z}|y = 0)$ (i.e. $\mathcal{KL}[q(\boldsymbol{z}|\boldsymbol{x}_u, y = 1)\|p(\boldsymbol{z}|y = 1)] < \mathcal{KL}[q(\boldsymbol{z}|\boldsymbol{x}_u, y = 0)\|p(\boldsymbol{z}|y = 0)]$) because $q(\boldsymbol{z}|\boldsymbol{x}_u, y = 1)$ and $p(\boldsymbol{z}|y = 1)$, optimized by labeled target windows in the target guidance mechanism, represent the target windows. Hence, TGGM assigns a higher weight to the component in $\boldsymbol{z}$ which represents the target windows (i.e., $(q(y = 1|\boldsymbol{x}_u) > q(y = 0|\boldsymbol{x}_u))$. The weights, $q(y|\boldsymbol{x_u})$, are the probabilities of a window belonging to the target and non-target clusters. With the target guidance mechanism, TGGM assigns unlabeled windows covering the target objects into the target cluster.

IV. Experiment and Analysis

This section presents a comprehensive experiment on TGGM with four types of target objects in four datasets, which are cars, airplanes, and ships in overhead imagery [44], [45], [46], and wetland symbols from scanned topographic maps. The experiment entails two groups of experiments and sensitivity analysis. The first group of experiments applied TGGM on four types of objects in five datasets, and evaluated the spatial arrangement estimation from TGGM by comparing with three baseline models. The second group of experiments applied TGGM on the DIOR dataset and compared results from TGGM with a supervised object detector, YOLOv3 [7].

A. Data Preparation and Experimental Settings

Datasets and Target Objects We tested the TGGM on four objects in two types of images in four datasets. They are cars in the Cars Overhead With Context (COWC) dataset [45], cars and airplanes in the xView dataset [46], and airplanes and ships in the DIOR dataset [44], which are all overhead imagery. The other object is wetland symbols in scanned topographic maps from the United States Geological Survey (USGS) topographic map archive.

Region-Level Annotations We used two approaches to generate the ROI annotations for the two groups of experiments, respectively. In the first group of experiments, we manually drew polygons which have an assumed strong semantic relationship with the target objects in overhead imagery in the xView, COWC, and DIOR datasets. The USGS dataset in the first set of experiments was combined with an external dataset[2] to provide the ROIs for wetland symbols. In the second group of experiments, we tested on the entire imagery covering a group of ships and airplanes in the DIOR dataset.

Target Window Annotations We manually annotate one window covering a target object for each ROI. To augment the number of labeled target windows, we rotated the window and translated the positions of the target object in the window.

Evaluation Methods In the first group of experiments, we used precision, recall, and F_1 at the grid-cells to estimate the performance of spatial arrangement estimation using TGGM. We sliced images into non-overlapped grid cells. Grid-level assessments are commonly used in Geospatial Information Science (GISc) to estimate the detection accuracy [47], [48], [49]. In the second group of experiments, we used mean average precision (mAP). In the first group of experiments, we generated the grid-level ground truth from the bounding-box ground truth. We sliced images using varied grid sizes, i.e., 20×20-, 40×40-, 60×60-, 80×80-pixel. If the overlapping area between the grid cell and the bounding box of the target object is over either 50% bounding-box or 50% grid cell, the grid cell was a true positive. Otherwise, it was a true negative. xView, COWC, and DIOR datasets provide the bounding box annotations for target objects. We manually generated the bounding boxes ground truth for wetland symbols in the USGS dataset. In the second group of experiments, we used the bounding-box ground truth provided by the DIOR dataset. We generated grid-level results from the detected windows. If the overlapping area between the grid and detected target window occupied over either 50% window or 50% grid, the grid was a detected positive grid. Otherwise, it was a detected negative grid. In the second group of experiments, the DIOR dataset provides the bounding boxes ground truth. We generate bounding-box results from detected windows using non-max suppression (NMS) [8].

[2]https://apps.nationalmap.gov/downloader

Baselines In the first group of experiments, we compared TGGM with two unsupervised generative clustering models, VaDE [13] and dualAE [14], and one semi-supervised generative model, AVAE [17]. For the semi-supervised model (AVAE), we used 40% target and non-target windows as labeled data for the training phase. The number of labeled windows for AVAE follows recently reported experiments [17]. In the second group of experiments, we compered TGGM with YOLOv3, a supervised object detector. We adopted the YOLOv3 results from paper [44], which trained YOLOv3 in a supervised manner.

Implementation Details All submodels in TGGM were the multilayer perceptrons with two fully connected layers and optimized by the Adam optimizer with a learning rate of $1e{-}3$. The iterative learning of TGGM for all tasks ended around seven iterations. Each iteration converged around 200 epochs.

B. Experiment Results and Analysis

Figure 3 shows that TGGM outperformed the unsupervised generative clustering models, i.e., dualAE and VaDE, and performed similar to the semi-supervised generative model, i.e., AVAE, in the first group of experiments. An F_1 score that is on average 10% higher than for dualAE and VaDE demonstrates that the spatial arrangement estimation using TGGM was more accurate than the state-of-the-art unsupervised methods. The high recall and low precision from the unsupervised methods show that the clustering results are noisier than those from TGGM. The precise results from TGGM show that the target guidance mechanism improves the clustering results. Compared with the semi-supervised model, AVAE, the F_1 score for TGGM was about 5% lower on average. However, AVAE required 40% labeled target and non-target windows, while TGGM only needed one labeled target window with augmentation, which minimized the manual work needed.

Figure 4 shows the spatial arrangement estimation of cars from TGGM and three baselines in xView. The green and red boxes are true positive and false positive grids, respectively. The bottom two figures show the noisy results from unsupervised models. Although TGGM missed some true positive grids compared to results from AVAE (the semi-supervised method), the spatial arrangement from TGGM covers most cars in the parking lot. In sum, TGGM could obtain more accurate spatial arrangement estimation within the ROI than unsupervised models did and achieved similar results as semi-supervised with much less manual work.

In the second group of experiments, we tested the detection of ships and airplanes in the DIOR dataset and compared the results with the best supervised model, YOLOv3 [44] using mAP. The results from YOLOv3 represent the best case scenario where a large amount of training data for the target objects are available. For airplanes, TGGM obtained 60.15% mAP, while YOLOV3 achieved 72.2%. As for the ships, the mAP for TGGM was 69.92%, while mAP from YOLOv3 was 87.4%. The average mAP from TGGM was 15% lower than YOLOv3, but TGGM only required one labeled target window

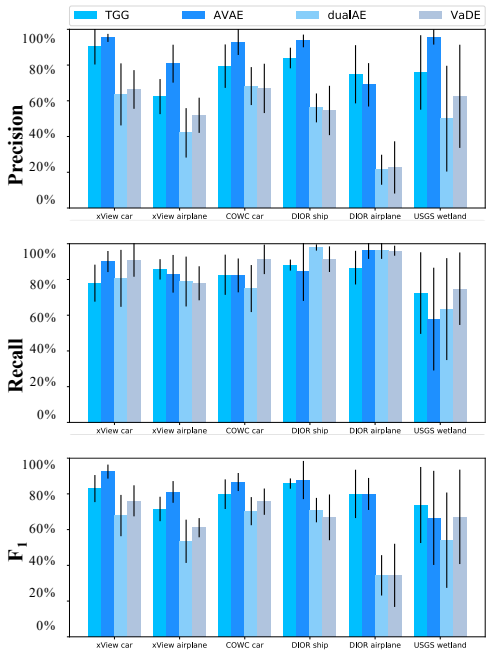


Fig. 3: The performance comparisons among TGGM and three baseline models. VaDE and dualAE are unsupervised baselines, and AVAE is the semi-supervised baseline.

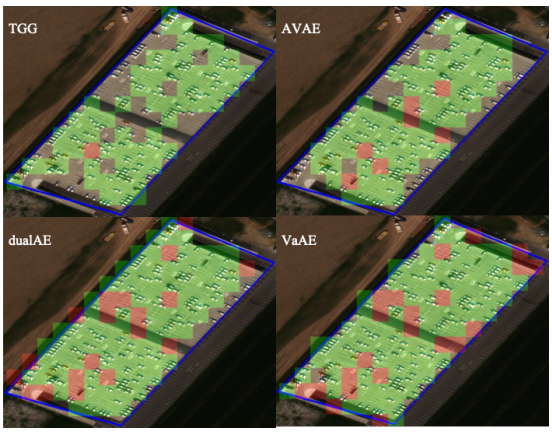


Fig. 4: The evaluation of car spatial arrangement estimation using 40×40-pixel grid size. Green and red boxes represent true positive and false positive grids, respectively.

with data augmentation for each image while YOLOv3 needed 50% target objects for the training process.

Two major reasons cause mAP from TGGM lower than YOLOv3. First, the ROI annotations may not have a strong semantic relationship with the target objects. We chose all imagery covering a group of target objects in the DIOR dataset. This criterion cannot guarantee any semantic relationship between the imagery and the target objects. Figure 5a shows

that the imagery covers not only the airplane apron, which has a strong semantic relationship with airplanes but also other areas, such as a lawn. The weak semantic relationship between airplanes and the imagery contents results in rather noisy results (Figure 5a). Figure 5b shows similar failures where the image covers a large area, much of which has no particular semantic relationship with ships. When we manually chose the ROI which has strong semantic relationship with the target objects in the first grounp of experiment, TGGM achieved 77% average F_1 score.

The other reason is the limited performance of TGGM on multi-scaled objects. TGGM uses a fixed-size window slide across all imagery and separates windows into target and non-target categories. The sizes of target objects varied a lot in all imagery. The fixed-size window, which is set based on the size of the small target objects, would often be smaller than those large target objects, and thus the model would fail to cluster the large target objects into the target category. Figure 5c shows that TGGM could detect small ships but missed some of the large ships when using small target objects to set the window size. However, users can adjust the window size to fit both large and small target objects when applying TGGM to one ROI. Consequently, TGGM can estimate the spatial arrangement of the target objects reliably. Figures 5d, 5e, and 5f show that TGGM applying to specific ROIs performs well in estimating the spatial arrangements of airplanes and ships in the overhead imagery, and wetland symbols in the topographic map, respectively.

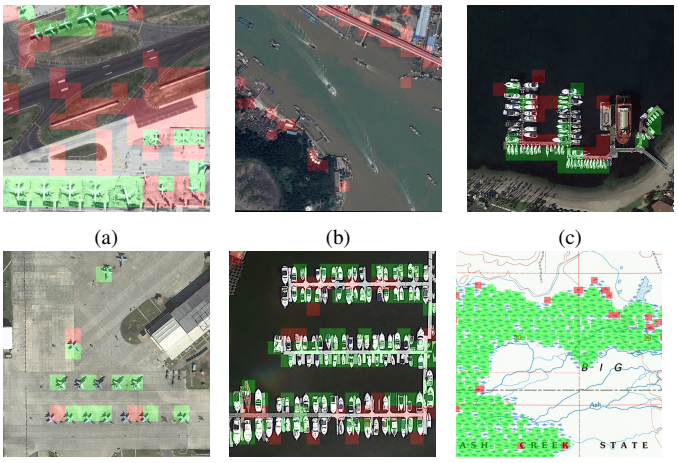


Fig. 5: The figures in the first row show failure cases from TGGM because of ROIs and multi-scale objects. The figures in the second row show the successful cases from TGGM when ROIs are spatially-constrained and the sizes of target objects are similar.

C. Sensitivity Study

1) Accuracy of Spatial Arrangement Estimation: We varied the grid size to assess how different levels of detail affect TGGM's estimation of the spatial arrangement [48]. Figure 6 shows that precision, recall, and F_1 increase with increasing

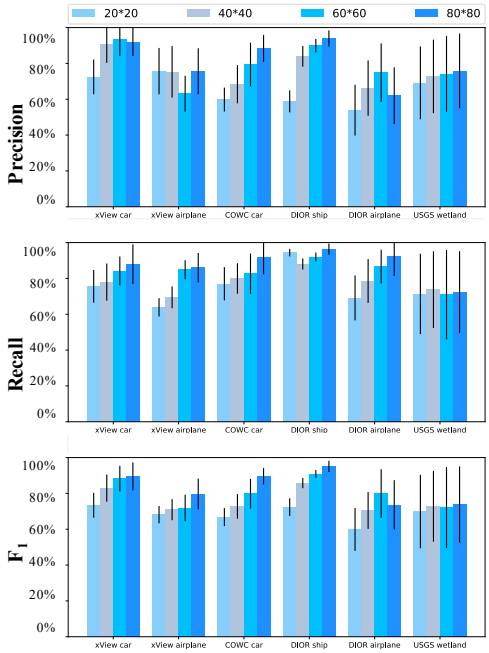


Fig. 6: The evaluations with varied grid sizes from 20×20-pixel to 80×80-pixel. The overall precision, recall and F_1 grow with increasing grid size.

window sizes. In contrast, the spatial arrangement details are lost with increasing window sizes. Figure 7 shows an example of the spatial arrangement estimation for ships in the DIOR dataset using 20×20-, 40×40-, and 60×60-pixel grid dimensions, respectively. Green and red grids are true positive and false positive grids, respectively. The green and red grids in Figure 7 show that the performance of spatial arrangement estimation improves while the details are lost with the increase of the grid sizes. When the grid size is similar to the object size, we find that TGGM can obtain satisfactory performance of spatial arrangement estimation and acceptable spatial arrangement details. TGGM achieved around 80% precision, recall, and F_1 on average when the grid size is similar to the object size. Figure 7b shows the detection results represented by a 40×40-pixel grid, which is similar in size to the ships in the image. The result shows that ships can be detected without losing much detail.

2) Spatial Arrangement Estimation over Iterations: This group of experiments tested the robustness of TGGM to the noise in the detected target windows by showing the performance of spatial arrangement estimation over iterations. Figure 8 shows the grid-level evaluation over iterations using grid extents similar to the object sizes. All detection tasks show high precision and low recall in the first few iterations. The true positive (green) grids in Figures 9a and 9b show that TGGM achieved precise but partial spatial arrangement

(a) 20×20 grid (b) 40×40 grid (c) 60×60 grid

Fig. 7: An example of the accuracy of spatial arrangement estimation using varied grid sizes.

estimation over the first three iterations. After the first few iterations, Figure 8 shows that precision decreases while recall increases. The green grids in Figures 9c and 9d show gradually increasing coverage, while the red grids in Figures 9c and 9d show the estimation contains some noise, such as partial car bodies. The detection evolvement in Figure 9 shows that the TGGM at first can detect cars with similar appearances. For example, most detected cars in the first three iterations have bright colors. With more iterations, the TGGM can detect cars with diverse appearances.

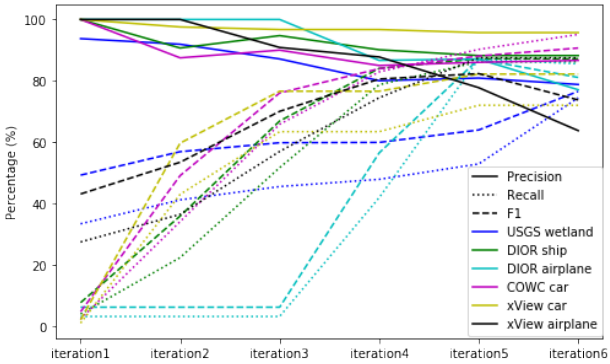
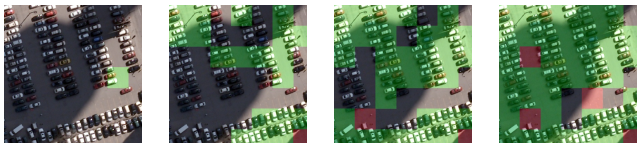


Fig. 8: The performance evaluation over several iterations



(a) Iteration 1 (b) Iteration 3 (c) Iteration 5 (d) Iteration 6

Fig. 9: The spatial arrangement estimation for cars in COWC dataset using 60×60-pixel grids over several iterations.

V. Discussion

This paper presents TGGM to estimate the spatial arrangement of the target objects within a spatially-constrained ROI in overhead images. TGGM's advantage is reducing the manual work to a few windows and an ROI annotations. The experiments show that TGGM outperforms baseline models in terms of spatial arrangement estimation accuracy and the amount of manual work. We are going to integrate TGGM into a pipeline for automatic information extraction from historical topographic maps which have little available labeled data. Historical topographic map archives store valuable information about the evolution of natural features and human activities. Additionally, we will work to address TGGM's multi-scale objects limitation by using windows with multiple sizes.

VI. Acknowledge

This material is based upon work supported in part by the National Science Foundation under Grant Nos. IIS 1564164 (to the University of Southern California) and IIS 1563933 (to the University of Colorado at Boulder), NVIDIA Corporation, the National Endowment for the Humanities under Award No. HC-278125-21, and the University of Minnesota, Computer Science & Engineering Faculty startup funds.

Appendix A

Here is the detailed evidence lower bounds (ELBO) deduction for the unlabeled data $\boldsymbol{x}_u$ and labeled target data $\boldsymbol{x}_t$.

$$
\begin{aligned}
&\mathcal{L}_{ELBO}(\boldsymbol{x}_u) = \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log \frac{p(\boldsymbol{x}_u,\boldsymbol{z},y)}{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log p(\boldsymbol{x}_u|\boldsymbol{z},y) + \log p(\boldsymbol{z}|y) + \log p(y) \\
&\quad - \log q(\boldsymbol{z}|\boldsymbol{x}_u,y) - \log q(y|\boldsymbol{x}_u)\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log p(\boldsymbol{x}_u|\boldsymbol{z},y)\Big] \\
&\quad + \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log p(\boldsymbol{z}|y) - \log q(\boldsymbol{z}|\boldsymbol{x}_u,y)\Big] \\
&\quad + \mathbb{E}_{q(\boldsymbol{z},y|\boldsymbol{x}_u)}\Big[\log p(y) - \log q(y|\boldsymbol{x}_u)\Big] \\
&= \sum_y q(y|\boldsymbol{x}_u)\int_{\boldsymbol{z}} q(\boldsymbol{z}|\boldsymbol{x_u},y)\Big[\log p(\boldsymbol{x}_u|\boldsymbol{z},y)\Big]d\boldsymbol{z} \\
&- \sum_y q(y|\boldsymbol{x}_u)\int_{\boldsymbol{z}} q(\boldsymbol{z}|\boldsymbol{x}_u,y)\Big[\log q(\boldsymbol{z}|\boldsymbol{x}_u,y) - \log p(\boldsymbol{z}|y)\Big]d\boldsymbol{z} \\
&- \sum_y q(y|\boldsymbol{x}_u)\Big[\log q(y|\boldsymbol{x_u}) - \log p(y)\Big] \\
&= \sum_y q(y|\boldsymbol{x}_u)\int_{\boldsymbol{z}} q(\boldsymbol{z}|\boldsymbol{x_u},y)\Big[\log p(\boldsymbol{x}_u|\boldsymbol{z},y)\Big]d\boldsymbol{z} \\
&- \sum_y q(y|\boldsymbol{x}_u)\mathcal{KL}\Big[q(\boldsymbol{z}|\boldsymbol{x}_u,y)\|p(\boldsymbol{z}|y)\Big] - \mathcal{KL}\Big[q(y|\boldsymbol{x}_u)\|p(y)\Big]
\end{aligned}
\tag{16}
$$

$$
\begin{aligned}
&\mathcal{L}_{ELBO}(\boldsymbol{x}_t) = \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big[\log \frac{p(\boldsymbol{x}_t,\boldsymbol{z},y=1)}{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big[\log p(\boldsymbol{x}_t|\boldsymbol{z},y=1) + \log p(\boldsymbol{z}|y=1) \\
&\quad - q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big[\log p(\boldsymbol{x}_t|\boldsymbol{z},y=1)\Big] \\
&\quad + \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big[\log p(\boldsymbol{z}|y=1) - \log q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)\Big] \\
&= \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)}\Big[\log p(\boldsymbol{x_t}|\boldsymbol{z},y=1)\Big] \\
&\quad - \mathcal{KL}\Big[q(\boldsymbol{z}|\boldsymbol{x}_t,y=1)\|p(\boldsymbol{z}|y=1)\Big]
\end{aligned}
\tag{17}
$$

REFERENCES

[1] G. Palubinskas, F. Kurz, and P. Reinartz, "Detection of traffic congestion in optical remote sensing imagery," in *IGARSS 2008-2008 IEEE International Geoscience and Remote Sensing Symposium*, vol. 2. IEEE, 2008, pp. II–426. 1

[2] J. Kurniawan, S. G. Syahra, C. K. Dewa *et al.*, "Traffic congestion detection: Learning from cctv monitoring images using convolutional neural network," *Procedia computer science*, vol. 144, pp. 291–297, 2018. 1

[3] E. O. Nsoesie, B. Rader, Y. L. Barnoon, L. Goodwin, and J. Brownstein, "Analysis of hospital traffic and search engine data in wuhan china indicates early disease activity in the fall of 2019," 2020. [Online]. Available: http://nrs.harvard.edu/urn-3:HUL.InstRepos:42669767 1

[4] N. A. Mubin, E. Nadarajoo, H. Z. M. Shafri, and A. Hamedianfar, "Young and mature oil palm tree detection and counting using convolutional neural network deep learning method," *International Journal of Remote Sensing*, vol. 40, no. 19, pp. 7500–7515, 2019. 1

[5] B. A. G. de Oliveira, F. M. F. Ferreira, and C. A. P. da Silva Martins, "Fast and lightweight object detection network: Detection and recognition on resource constrained devices," *IEEE Access*, vol. 6, pp. 8714–8724, 2018. 1

[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. 1

[7] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv:1804.02767*, 2018. 1, 3, 5

[8] 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. 1, 3, 5

[9] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html. 1, 3

[10] Q. Tao, H. Yang, and J. Cai, "Zero-annotation object detection with web knowledge transfer," in *Proceedings of European Conference on Computer Vision*, 2018, pp. 369–384. 1

[11] M. Shi, H. Caesar, and V. Ferrari, "Weakly supervised object localization using things and stuff transfer," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3381–3390. 1, 3

[12] T. Hu, P. Mettes, J.-H. Huang, and C. G. Snoek, "Silco: Show a few images, localize the common object," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 5067–5076. 1, 3

[13] Z. Jiang, Y. Zheng, H. Tan, B. Tang, and H. Zhou, "Variational deep embedding: An unsupervised and generative approach to clustering," *arXiv:1611.05148*, 2016. 1, 3, 6

[14] X. Yang, C. Deng, F. Zheng, J. Yan, and W. Liu, "Deep spectral clustering using dual autoencoder network," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 4066–4075. 1, 3, 6

[15] N. Dilokthanakul, P. A. Mediano, M. Garnelo, M. C. Lee, H. Salimbeni, K. Arulkumaran, and M. Shanahan, "Deep unsupervised clustering with gaussian mixture variational autoencoders," *arXiv:1611.02648*, 2016. 1, 3

[16] D. P. Kingma, S. Mohamed, D. J. Rezende, and M. Welling, "Semi-supervised learning with deep generative models," in *Proceedings of Advances in Neural Information Processing Systems*, 2014, pp. 3581–3589. 1, 2, 3

[17] X. Zhang, L. Yao, and F. Yuan, "Adversarial variational embedding for robust semi-supervised learning," in *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 139–147. 1, 2, 6

[18] L. Maaløe, M. Fraccaro, V. Liévin, and O. Winther, "Biva: A very deep hierarchy of latent variables for generative modeling," in *Proceedings of Advances in Neural Information Processing Systems*, 2019, pp. 6548–6558. 1, 2, 3

[19] Y. Gao, B. Liu, N. Guo, X. Ye, F. Wan, H. You, and D. Fan, "C-midn: Coupled multiple instance detection network with segmentation guidance for weakly supervised object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9834–9843. 3

[20] X. Li, M. Kan, S. Shan, and X. Chen, "Weakly supervised object detection with segmentation collaboration," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9735–9744. 3

[21] K. Yang, D. Li, and Y. Dou, "Towards precise end-to-end weakly supervised object detection network," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8372–8381. 3

[22] Z. Zeng, B. Liu, J. Fu, H. Chao, and L. Zhang, "Wsod2: Learning bottom-up and top-down objectness distillation for weakly-supervised object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8292–8300. 3

[23] S. Rahman, S. Khan, and N. Barnes, "Transductive learning for zero-shot object detection," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 6082–6091. 3

[24] Y. Zhu, Y. Zhou, Q. Ye, Q. Qiu, and J. Jiao, "Soft proposal networks for weakly supervised object localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 1841–1850. 3

[25] D. Kim, D. Cho, D. Yoo, and I. So Kweon, "Two-phase learning for weakly supervised object localization," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3534–3543. 3

[26] K. K. Singh and Y. J. Lee, "Hide-and-seek: Forcing a network to be meticulous for weakly-supervised object and action localization," in *2017 IEEE international conference on computer vision (ICCV)*. IEEE, 2017, pp. 3544–3553. 3

[27] X. Liang, S. Liu, Y. Wei, L. Liu, L. Lin, and S. Yan, "Towards computational baby learning: A weakly-supervised approach for object detection," in *Proceedings of the IEEE International Conference on Computer Vision*, 2015, pp. 999–1007. 3

[28] F. Yang, H. Fan, P. Chu, E. Blasch, and H. Ling, "Clustered object detection in aerial images," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 8311–8320. 3

[29] M. Pritt, "Deep learning for recognizing mobile targets in satellite imagery," in *2018 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE, 2018, pp. 1–7. 3

[30] J. Shermeyer and A. Van Etten, "The effects of super-resolution on object detection performance in satellite imagery," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2019, pp. 0–0. 3

[31] A. Van Etten, "Satellite imagery multiscale rapid detection with windowed networks," in *2019 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE, 2019, pp. 735–743. 3

[32] A. Groener, G. Chern, and M. Pritt, "A comparison of deep learning object detection models for satellite imagery," in *2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE, 2019, pp. 1–10. 3

[33] L. Maaløe, M. Fraccaro, and O. Winther, "Semi-supervised generation with cluster-aware generative models," *arXiv:1704.00637*, 2017. 3

[34] J. Liu, J. Yao, M. Bagheri, V. Sandfort, and R. M. Summers, "A semi-supervised cnn learning method with pseudo-class labels for atherosclerotic vascular calcification detection," in *Proceedings of IEEE International Symposium on Biomedical Imaging*, 2019, pp. 780–783. 3

[35] C. Xu, Y. Dai, R. Lin, and S. Wang, "Social image refinement and annotation via weakly-supervised variational auto-encoder," *Knowledge-Based Systems*, vol. 192, p. 105259, 2020. 3

[36] J. Jeong, S. Lee, J. Kim, and N. Kwak, "Consistency-based semi-supervised learning for object detection," in *Proceedings of Advances in Neural Information Processing Systems*, 2019, pp. 10 759–10 768. 3

[37] F. Xing, T. C. Cornish, T. Bennett, D. Ghosh, and L. Yang, "Pixel-to-pixel learning with weak supervision for single-stage nucleus recognition in ki67 images," *IEEE Transactions on Biomedical Engineering*, vol. 66, no. 11, pp. 3088–3097, 2019. 3

[38] Y. Zeng, Y. Zhuge, H. Lu, L. Zhang, M. Qian, and Y. Yu, "Multi-source weak supervision for saliency detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 6074–6083. 3

[39] Y. Shen, R. Ji, S. Zhang, W. Zuo, and Y. Wang, "Generative adversarial learning towards fast weakly supervised detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 5764–5773. 3

[40] Y. K. Jang and N. I. Cho, "Generalized product quantization network for semi-supervised image retrieval," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020, pp. 3420–3429. 3

[41] L. Yang, N.-M. Cheung, J. Li, and J. Fang, "Deep clustering by gaussian mixture variational autoencoders with graph embedding," in *Proceedings of the IEEE International Conference on Computer Vision*, 2019, pp. 6440–6449. 3

[42] M. Ehsan Abbasnejad, A. Dick, and A. van den Hengel, "Infinite variational autoencoder for semi-supervised learning," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 5888–5897. 3

[43] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv:1312.6114*, 2013. 3, 4

[44] K. Li, G. Wan, G. Cheng, L. Meng, and J. Han, "Object detection in optical remote sensing images: A survey and a new benchmark," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 159, pp. 296–307, 2020. 5, 6

[45] T. N. Mundhenk, G. Konjevod, W. A. Sakla, and K. Boakye, "A large contextual dataset for classification, detection and counting of cars with deep learning," in *Proceedings of European Conference on Computer Vision*, 2016, pp. 785–800. 5

[46] D. Lam, R. Kuzma, K. McGee, S. Dooley, M. Laielli, M. Klaric, Y. Bulatov, and B. McCord, "xview: Objects in context in overhead imagery," *arXiv:1802.07856*, 2018. 5

[47] R. G. Congalton and K. Green, *Assessing the accuracy of remotely sensed data: principles and practices*. CRC press, 2019. 5

[48] K. Kuzera and R. G. Pontius, "Importance of matrix construction for multiple-resolution categorical map comparison," *GIScience & Remote Sensing*, vol. 45, no. 3, pp. 249–274, 2008. 5, 7

[49] J. H. Uhl, S. Leyk, C. M. McShane, A. E. Braswell, D. S. Connor, and D. Balk, "Fine-grained, spatiotemporal datasets measuring 200 years of land development in the united states," *Earth System Science Data*, vol. 13, no. 1, pp. 119–153, 2021. 5