Graph-based Modelling of Superpixels for Automatic Identification of Empty Shelves in Supermarkets

Bikash Santra[1,*], Udita Ghosh[1,2], Dipti Prasad Mukherjee

Electronics and Communication Sciences Unit, Indian Statistical Institute, Kolkata

Abstract

Automatic detection of empty spaces (gaps) between the displayed products as seen in the images of shelves of a supermarket is an interesting image segmentation problem. This paper presents the first known attempt to solve this commercially relevant challenge. The shelf image is first over-segmented into a number of superpixels to construct a graph of superpixels (SG). Subsequently, a graph convolutional network and a Siamese network are built to process the SG. Finally, a structural support vector machine based inference model is formulated based on SG for segmenting the gap and non-gap regions. In order to validate our method, we manually annotate the images of shelves of three benchmark datasets of retail products. We have achieved ∼70 to ∼85% segmentation accuracy (in terms of *mean intersection-over-union*) on the annotated datasets. A part of the annotated data is released at `https://github.com/gapDetection/gapDetectionDatasets`.

Keywords: Gap detection, retail store, graph convolutional network, Siamese network, structural support vector machine

*Corresponding author

Email addresses: `bikash.santra@isical.ac.in` (Bikash Santra), `udighosh@gmail.com` (Udita Ghosh), `dipti@isical.ac.in` (Dipti Prasad Mukherjee)

[1]These authors have the equal contributions.

[2]This author is presently at Zendrive Inc.

1. Introduction

In supermarkets, as a part of the marketing strategy, the products are placed on the shelves based on a predefined plan, commonly known as *planogram* [1]. In regular intervals, store associates check the compliance of the planogram. In recent years, various attempts have been made for designing a vision-based system to automatically check the planogram compliance [2, 3]. Classification [4, 5, 6] and accurate localization [7, 8, 9] of retail products on the shelf are the associated sub-problems in order to check the planogram compliance. However, these state-of-the-art solutions ignore a key direction of the problem, which is the automatic identification of the gaps between the products (or empty shelves in the stores). This automatic identification of the gaps on the shelves is the focus of this paper.

As a part of the checking of planogram compliance, the store associate needs to manually identify the gap. This is time-consuming and error-prone. Delays in identifying out-of-stock (OOS) conditions lead to a significant loss in revenue, especially when the product is present in the inventory [10]. There exists several automated OOS detection systems like radio-frequency identification (RFID) tags [11] and weighted sensor shelves [10]. They are expensive and difficult to reconfigure for fast-changing retail product lines. On the other hand, there exists only one known published work [12] that uses computer vision to detect gaps automatically. The use of a camera, either hand-held by a store associate or fixed on the opposite facing rack, seems to be a feasible economic option for continuous monitoring of the gap regions and planogram compliance. In this paper, we introduce a solution for detecting gaps in the images of shelves. Fig. 1 illustrates gap regions in an example shelf image.

We define a gap as an empty space (or a region) created on the shelf after a product is picked up from the shelf. For example, the region covered by the green boundary in Fig. 1 indicates that a (or few) product(s) is (are) missing. Note that different gap regions may exist in the same image having different textures, colors or features. For example, in Fig. 1, the regions highlighted by

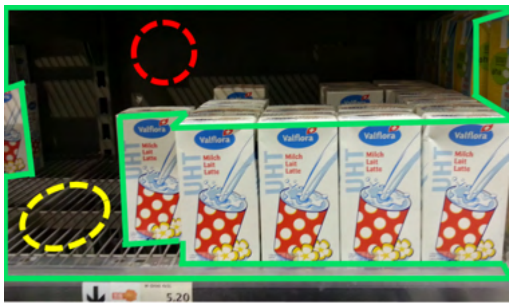


Figure 1: In an example shelf image, the regions covered with the green polygon represent the gap regions. Red and yellow dotted circles highlight different textures for the gaps.

the red and yellow dotted circles on the shelf present different textures for the gap. The absence of unique inherent characteristics of the gap regions amplifies the challenges in solving the gap detection problem.

The sales and promotion programme of products is required to comply with a given planogram (defined in the first line of this section). The gaps created on the racks due to sales or due to interference by the customer disrupts planogram compliance. The proposed algorithm for gap detection helps in assessing planogram compliance.

Further, the detection of gap in a shelf quickly estimates the extent of refilling required in that gap. Store associates can continuously monitor the status of a rack and disruption of the display plan based on the output of the gap detection framework. Temporal study of gap detection may help in studying the consumer behaviour which in turn helps to design a better planogram.

To identify the gaps, one method could be to take the difference of the fully-stacked rack image from the subsequent images of the partially empty rack. This is similar to the background subtraction approach which needs a fully-stacked reference image. On the contrary, our proposed method does not need any reference image.

A straightforward image subtraction does not work in a real store environment. First, the cameras taking pictures of racks are not in fixed locations. Therefore, the sequence of images of any particular rack varies as the pictures are taken by different store associates by different handheld cameras and that too, from different vantage points. Second, the store level illumination and shadow due to occlusion, change the image content and thus making the difference image an unfit case for estimating gap.

Finally, for a wide range of fast-changing product lines, products with minor variations in appearance, dynamic nature of disruption of display due to customer behaviour, all put together, makes processing of background-subtracted results complicated and threshold driven. Such thresholds and rules for image pruning are difficult to set and need to be customized for racks, products and at store level. Compared to this, the proposed approach is far more generalized and better-motivated considering planogram compliance.

We have posed our task of identifying gaps in the shelf images as a segmentation problem and the objective is to estimate the pixel-level binary mask identifying the empty regions as white and the remaining portion as black. The flow chart of our proposal is shown in Fig. 2. Our scheme first over-segments the entire image into superpixel regions and construct a graph of superpixels (say, G), where the edges of G capture the association between two superpixels. Subsequently, the features of each superpixel (or each node of G) are extracted by feeding this G as input to a graph convolutional network (GCN) [13] that imbibes the neighbourhood information of superpixels. Further, the representation of the association/similarity between superpixels i.e. the weights of the edges in G (referred to as edge features) are encoded with a Siamese network

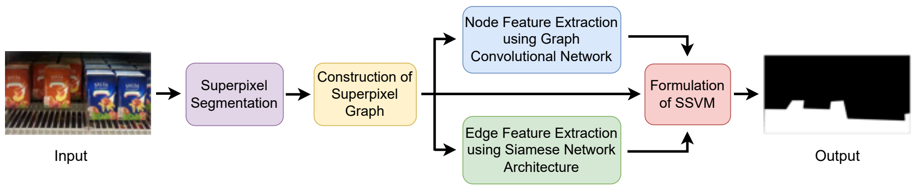

Input
Superpixel Segmentation
Construction of Superpixel Graph
Node Feature Extraction using Graph Convolutional Network
Edge Feature Extraction using Siamese Network Architecture
Formulation of SSVM
Output

Figure 2: Process flow of our proposed scheme for the identification of gaps in the shelves

[14]. This way, our proposal establishes the relationship between the adjacent superpixels being part of a gap or non-gap region in a shelf image. Using these sets of features for the nodes and edges in G, we formulate a structural support vector machine (SSVM) [15] to generate a binary mask that classifies the gap and non-gap regions.

From the above discussion, it is evident that GCN finds representation of superpixels observing their neighbours whereas Siamese network finds the representation of similarity of superpixels. Therefore, GCN and Siamese Network need a classifier for the identification of gaps. SSVM serves the purpose.

The proposed scheme is compared with the recent state-of-the-art deep learning based image segmentation approaches U-Net [16], DeepLabV3 [17], LinkNet [18], FPN [19], PSPNet [20], DeepLabV3+ [21], PAN [22], MA-Net [23] for the identification of gap regions on the shelves. Contributions of our proposed scheme compared to all these state-of-the-art approaches are three-folds:

1) To the best of our knowledge, we are the first to release the benchmark datasets in GitHub [24] for the identification of gaps in the shelves. The identified and annotated gaps are marked in the publicly available datasets Grocery Products [25], WebMarket [26], and GroZi [27].
2) To the best of our knowledge, we are the first to introduce this problem as an image segmentation task to machine vision community and proposing a novel solution that outperforms deep learning based state-of-the-art segmentation approaches in almost all the evaluations.
3) We utilize a graph convolutional neural network and a Siamese network in the formulation of structural support vector machine for the detection of gaps in the shelves.

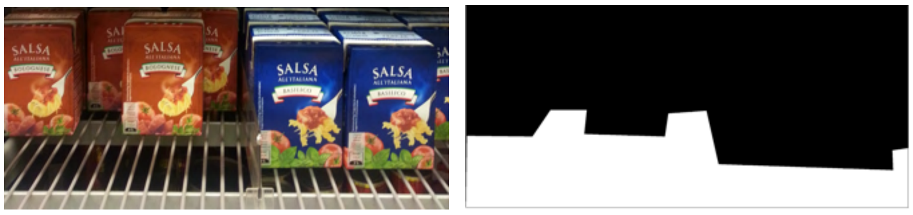
The rest of the paper is organized as follows. The benchmarking procedure of the shelf images from publicly available datasets are described in Section 2. Section 3 explains our proposed method. The experiments are carried out in Section 4. Finally, we conclude the paper in Section 5.

2. Benchmark Datasets for the Identification of Gaps

For designing and validating a model for automatic identification of gaps, we require ground truth for each shelf image specifying the gap and non-gap regions. In order to generate ground truth, we manually annotate the images of shelves by labelling the gap and non-gap regions with the polygons.

We follow a certain convention and annotate the images with two different labels: gap and non-gap polygons drawn on the image. The gap region is defined by (a) the locations of shelves where the background of shelves are visible and (b) the dark regions where the objects are invisible. In the end, we obtain a pixel-level binary mask (which we refer to as ground truth) for each shelf image depicting 1 as the presence of a gap and 0 as the presence of a non-gap (product/non-usable parts of the shelf). A pixel-level binary mask corresponding to an example shelf image of Fig. 3(a) is shown in Fig. 3(b). The regions other than gaps on the shelf image are referred to as non-gap in this paper.

The images of shelves for manual annotation are from the publicly available datasets Grocery Products [25], WebMarket [26], and GroZi [27]. We have used the graphical annotation tool *labelme* [28]. Grocery Products dataset includes 680 images of shelves captured from different supermarkets. We select 305 images where the gap regions are present. Similarly, we annotate 98 and 50 images of shelves from the WebMarket and GroZi datasets respectively. A part of these annotations is made public as benchmark datasets for further studies on the



(a) (b)

Figure 3: (a): An example shelf image, (b): The ground truth i.e. pixel-level binary mask for the shelf image (a) where white and black regions denote gap and non-gap respectively.

identification of gaps. They are available at GitHub [24]. Next, we present the proposed methodology.

3. Methodology

We approach the problem under discussion as a binary segmentation problem in which pixels are classified into either of the two classes, 1 (gap) and 0 (non-gap). The overall block diagram of our proposed scheme is illustrated in Fig. 2. The steps of our scheme are explained in the following subsections.

3.1. Superpixel Segmentation

In order to identify the gap regions on the shelves, the proposed scheme aims to label each pixel of the images of shelves. In our case, the number of pixels in the image of a shelf is in the order of 10^5. The procedure for extracting features from each of these many pixels and labelling them is computationally expensive. In order to reduce the complexity, all the images are initially over-segmented into a few regions consisting of a group of pixels, called superpixels. Assume we obtain N number of superpixels in the images of shelves and each superpixel is denoted as x_i, $\forall\, i = \{1, 2, \ldots, N\}$. In our implementation, we utilize the *simple linear iterative clustering* (SLIC) algorithm [29] for generating superpixels for a shelf image. Next, we construct the graph of superpixels for each image.

3.2. Construction of the Graph of Superpixels

For every shelf image, we construct a graph with the superpixels as the nodes. The edges are connected for the pairs of adjacent superpixels. It is evident that the graph will be a connected graph as each superpixel is adjacent to at least one other superpixel. We refer to this graph of superpixels as *superpixel graph* (SG) in the rest of the paper.

Assume, we have a shelf image I which has four superpixels x_1, x_2, x_3, and x_4 as illustrated in Fig. 4(a). Further assume, G be the SG for the image I. Hence, the set of nodes of G is $V = \{x_1, x_2, x_3, x_4\}$ and the set of edges is $E = \{e_1 = (x_1, x_2), e_2 = (x_1, x_3), e_3 = (x_2, x_3), e_4 = (x_2, x_4),\ e_5 = (x_3, x_4)\}$ as

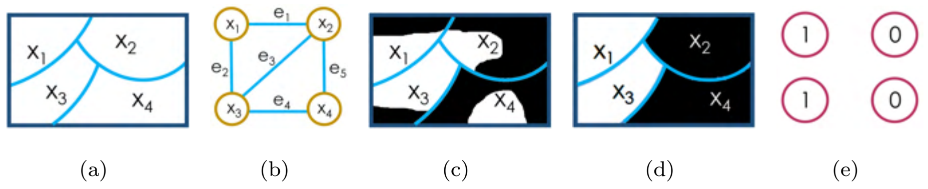


Figure 4: (a): A segmented image I. (b): Superpixel graph of I. (c): Ground truth pixels overlaid on I (referred as pixel-level binary mask). (d): Superpixel-level binary mask for I labelling each superpixel, where the white and black regions indicate gap and non-gap regions, respectively. The majority of pixels in superpixel x_1 is white as shown in (c). Therefore, the label of superpixel x_1 is given as white as shown in (d). The labels of superpixels x_2, x_3 and x_4 are determined in a similar manner. (e): The labels for the nodes of (b).

shown in Fig. 4(b). In this work, this SG is an equivalent representation of the superpixels in I. Thus, in the rest of the paper, the node x_i of G essentially refers to the superpixel x_i of I.

In G, each node x_i is characterized by the unary feature embedding $\mathbf{u}(x_i)$, and each edge (x_i, x_j) is characterized by the pairwise feature embedding $\mathbf{p}(x_i, x_j)$ for the adjacent superpixels x_i and x_j. Thus, the unary feature embedding refers to the feature vector of a superpixel i.e. a node of G, while pairwise feature embedding represents a feature vector for two adjacent superpixels i.e. an edge of G. In the rest of the paper, node and unary features, and edge and pairwise features are interchangeably used. Given these, for the example shown in Fig. 4(a), we define a structured data X which consists of:

(a) Adjacency list A of G,

(b) Pairwise feature embedding for the edges e_1, e_2, e_3, e_4, e_5 and,

(c) Unary feature embedding for the nodes x_1, x_2, x_3, x_4.

This X is essentially the input to Structural Support Vector Machine (SSVM) for the identification of gap/non-gap regions in I. But before discussing SSVM, we define the unary and pairwise feature embedding for nodes and edges using GCN and a Siamese network respectively.

In our proposal, GCN or Siamese network learn the labels of the nodes (gap being labelled as 1, and non-gap as 0) of G (as described at the beginning of Section 3) for all the shelf images in the train set. We extract the feature embedding for nodes and edges of G from learnt GCN and Siamese network respectively. However, in order to learn GCN or Siamese network, an SG and the labels of superpixels are required. We have already explained the procedure of constructing SG. Next, the procedure for assigning labels to the superpixels is described.

Labelling of Superpixels. The training set includes ground truth images i.e. pixel-level binary masks where each pixel is labelled as either gap or non-gap. Given this, we now demonstrate the procedure of labelling superpixels using the example given in Fig. 4. For the shelf image I in Fig. 4(a), Fig. 4(c) is the ground truth image (or the pixel-level binary mask I_{gt}) with each pixel marked either as white (i.e. label 1) or black (i.e. label 0) representing gap or non-gap region respectively. Fig. 4(b) is the superpixel graph of I. Fig. 4(c) shows that superpixel x_1 has more white pixels than black pixels. Therefore, the label of superpixel x_1 is assigned as white as shown in Fig. 4(d). Similarly, based on majority voting of white or black pixels, superpixels x_2, x_3 and x_4 are labelled as black, white and black, respectively as shown in Fig. 4(d). Fig. 4(e) shows the labels of the nodes of the superpixel graph in Fig. 4(b). This way the superpixel-level binary mask B of the shelf image I is determined. Next, we extract features for the nodes of G.

3.3. Feature Representation of the Nodes in SG

The unary feature embedding in the structured data X of G is essentially the feature representation for the nodes (i.e. superpixels) of G. Since we look for a novel feature representation for each superpixel of I considering not only the superpixel itself but also its neighbouring superpixels, we design a GCN [13] that principally accumulates the local neighbourhood information of each superpixel utilizing graph convolution. So naturally, the SG G is input to the GCN.

The block diagram of the proposed GCN model is presented in Fig. 5. The architecture of the proposed three-layered GCN is detailed in our implementation details in Section 4. However, a GCN usually takes the adjacency matrix of the graph G and an initial feature vector for each node of G. In this work, the initial feature vectors for the nodes or superpixels are determined using a CNN based feature extractor (referred to as initial linear feature extractor or ILFE) designed on top of the pre-trained VGG-19 [30] (see Fig. 5). The architectural details of this ILFE are also provided in the implementation details. Assume, ILFE returns d_1-dimensional feature vector $f_L^{(i)}$ for each i^{th} node in G. In our implementation, $d_1 = 128$.

Then G along with the initial linear feature vectors for all its nodes $f_L^{(i)}, i = 1, 2, \ldots, N$, are sent to GCN and layer-wise propagated in GCN [13] as follows:

$$H^{(l+1)} = \sigma\left(\widetilde{D}^{-\frac{1}{2}} \widetilde{A} \widetilde{D}^{-\frac{1}{2}} H^{(l)} W^{(l)}\right) \tag{1}$$

where $\widetilde{A} = A + I_N$ is the adjacency matrix of G considering self loops, A is the adjacency matrix of G, and I_N is the identity matrix of size N. Here $\widetilde{D}_{ii} = \sum_j \widetilde{A}_{ij}$ and $\widetilde{D}_{ij} = 0,\ \forall i \neq j$. $H^{(l)}$ is the input to the l^{th} layer of GCN and hence $H^{(0)}$ is essentially the input to GCN i.e. $H^{(0)}$ is a vector of length N, where the elements are $f_L^{(i)}, i = 1, 2, \cdots, N$. $W^{(l)}$ is the trainable weight matrix for layer l and $\sigma(\cdot)$ is the activation function. Finally, GCN returns a d_2-dimensional feature vector

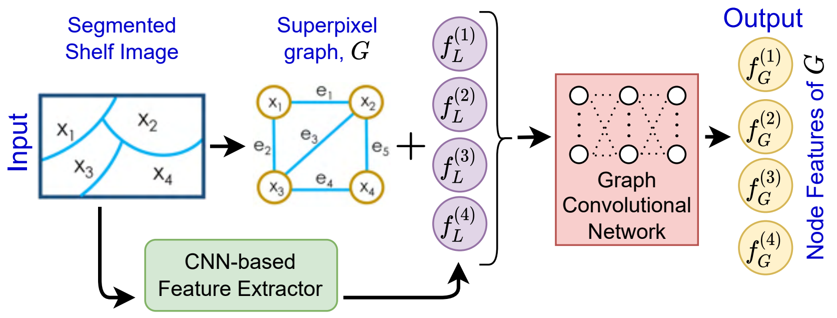


Figure 5: Flow chart of the proposed node feature extraction scheme. For the segmented image in Fig. 4, superpixels x_1, x_2, x_3, and x_4 are sent to CNN-based feature extractor for determining respective initial linear feature vectors $f_L^{(1)}$, $f_L^{(2)}$, $f_L^{(3)}$, and $f_L^{(4)}$. These initial feature vectors and the SG are passed through the graph convolutional network for obtaining the features $f_G^{(1)}$, $f_G^{(2)}$, $f_G^{(3)}$, and $f_G^{(4)}$ of the nodes x_1, x_2, x_3, and x_4 respectively.

$f_G^{(i)}$ for each i^{th} node in G aggregating the local neighborhood information of a superpixel. The aggregation of local information is ensured by the inclusion of $\widetilde{A}$ in (1). $f_G^{(i)}$ is essentially the output of penultimate layer of GCN. In our implementation, $d_2 = 16$. In other words, the unary feature embedding for each i^{th} node x_i of G can be defined as $\mathbf{u}(x_i) = f_G^{(i)}$. Next we present our method for finding out features of edges of G.

3.4. Feature Representation of the Edges in SG

As mentioned earlier, there exists an edge between two nodes x_i and x_j in G, if the superpixels x_i and x_j in I are adjacent. In this work, we aim to define a feature representation for the edge (x_i, x_j) that encodes the similarity (or dissimilarity) between the two neighboring superpixels x_i and x_j in I. Similar superpixels mean either both are gap regions or both are non-gap regions. In other words, we look for a unique representation for the edges between superpixels that are similarly (or dis-similarly) labelled. Therefore, we can pose this as a two-class (similar and dissimilar) classification problem that takes two superpixels (or sub-images) as input. In order to solve this, we build a Siamese network architecture (SNA) [14] from which the features for the edges in G are extracted. The features for the nodes extracted from GCN, although consider the neighbourhood superpixel information, do not take the class similarity of the superpixels into account. On the other hand, SMA learns a similarity measure that tends to bring feature vectors of similar class labels nearer than those of dissimilar class labels in the feature space.

Each pair of adjacent superpixels (x_i, x_j) in I is the input to our SNA for finding out the feature for the edge (x_i, x_j) in G. The schematic of our SNA is provided in Fig. 6 and its architecture is detailed in our implementation details in Section 4. For any pair of adjacent superpixels (x_i, x_j), the superpixel x_i is first fed to the first convolutional block, *conv-block-1* while x_j is sent to the second convolutional block, *conv-block-2*. A convolutional block essentially refers to a stack of a number of convolutional layers. According to the principle of Siamese network, the learnable weights/parameters of the blocks *conv-block-1*

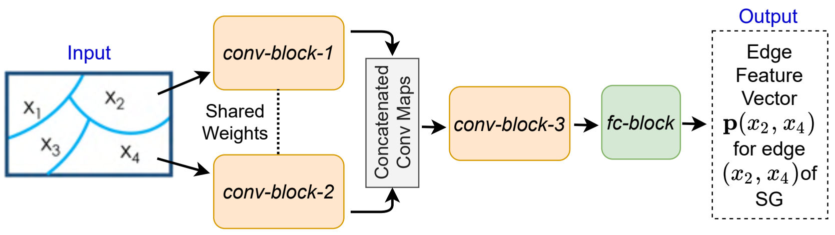


Figure 6: Schematic of the proposed Siamese network architecture for extraction of edge feature. For the example segmented image in Fig. 4, we illustrate the extraction of pairwise feature $\mathbf{p}(x_2, x_4)$ for the edge (x_2, x_4) of SG.

and *conv-block-2* are shared (see Fig. 6). The shared weights result in similar transformation of both the inputs to their respective convolutional maps. The outputs (i.e. convolutional (conv) maps) of both the blocks are then concatenated and fitted to the third block *conv-block-3* of our SNA. The output of *conv-block-3* is finally passed through a *fc-block* comprising of three consecutive fully connected (fc) layers. Last fc layer defines the classification score of similarity and dissimilarity between two input superpixels x_i and x_j. Here, we practically look for a feature representation for the edge between x_i and x_j which is essentially determined by the output of penultimate fc layer of SNA. The SNA provides d_3-dimensional pairwise feature vector/embedding $\mathbf{p}(x_i, x_j)$ for each edge (x_i, x_j) in G. In our implementation, $d_3 = 16$. Next we present the SSVM for the identification of gaps/non-gaps in the shelf images.

3.5. SSVM for the Identification of Gap Regions

SSVM [15] is a classifier that predicts the labels for the nodes of G minimizing the loss between the predicted and true labels. In our problem, we utilize SSVM to learn the labels of the nodes (i.e. 0 or 1) of the G (as described in Section 3.2) for all the shelf images in the train-set. Once the SSVM is learnt, we obtain the labels of the nodes which we assign to the corresponding superpixels of a shelf image. In this way, the entire image is segmented into the gap and non-gap regions (gap being labelled as 1, and non-gap as 0). Next, we formulate the SSVM with the structured data X and the labels of superpixels (see Section 3.2).

3.5.1. Formulation of SSVM

Assume, we have M number of shelf images $I^{(k)}$, $k = 1, 2, \ldots, M$ in the train-set. Corresponding to each training image $I^{(k)}$, we obtain the structured data $X^{(k)}$ and the true labels $Y^{(k)}$ for the superpixels. Given these, the gap identification problem can be posed as follows.

Each of the superpixels of any shelf image I can be interpreted as a discrete random variable taking values from the set $\Omega = \big\{0, 1\big\}$, where 1 signifies gap and 0 denotes non-gap. Let us assume that $Y = \{y_1, y_2, \ldots, y_N\} \in \mathcal{Y} = \Omega^N$ be a feasible label vector for the N number of nodes in G. In that case, there exists 2^N possible label vectors (or feasible labeling) for the G with N number of nodes i.e. $|\mathcal{Y}| = 2^N$. Thus, the set of 2^N feasible label vectors must include the true label (Boolean) vector (for the SG). Now the gap identification problem boils down to finding out the true label vector from 2^N possible label vectors for the G. For example, the possible label vectors for the graph shown in Fig. 4(b) is $Y = (y_1, y_2, y_3, y_4)$, where $y_i, i = 1, 2, 3, 4$ can be 0 or 1. Then the number of possible label vectors for this graph is 2^4 out of which we aim to find out the true label vector $(1, 0, 1, 0)$ as shown in Fig. 4(e).

In order to obtain the true label vector $Y^{(k)}$ for any input $X^{(k)}$, we define a potential function $\mathtt{E}(X, Y)$ which will be maximized when $Y = Y^{(k)}$ for the given $X = X^{(k)}$ i.e.

$$Y^{(k)} = \arg\max_{Y} \mathtt{E}(X^{(k)}, Y), \tag{2}$$

and the potential function $\mathtt{E}(X, Y)$ is formulated as:

$$\mathtt{E}(X, Y) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(X, Y), \tag{3}$$

where $\mathbf{w}$ is the weight vector and $\boldsymbol{\phi}$ is the joint feature vector for an input X and its any label vector Y. Our target is to learn this weight vector $\mathbf{w}$ with the train-set such that the potential function $\mathtt{E}$ is maximized for the true label vectors.

The potential function $\mathtt{E}(X, Y)$ can be defined as the sum of the potential functions $\mathtt{E}_{\mathbf{u}}(x_i, y_i)$ and $\mathtt{E}_{\mathbf{p}}(x_i, x_j, y_i, y_j)$ contributed by the unary and pairwise features as:

$$\arg\max_{Y} \mathtt{E}(X,Y) = \arg\max_{Y} \left(\sum_{i=1}^{N} \mathtt{E}_{\mathbf{u}}(x_i, y_i) + \sum_{(x_i,x_j)\in E} \mathtt{E}_{\mathbf{p}}(x_i, x_j, y_i, y_j) \right). \tag{4}$$

Again $\mathtt{E}_{\mathbf{u}}$ can be written as:

$$\mathtt{E}_{\mathbf{u}}(x_i, y_i) = \mathbf{w}_{\mathbf{u}}^{\mathrm{T}} \phi_{\mathbf{u}}(x_i, y_i), \tag{5}$$

where $\phi_{\mathbf{u}}$, which is the joint feature vector (for associating features with labels) of the node x_i and its label y_i for the unary features, is defined as:

$$\phi_{\mathbf{u}}(x_i, y_i) = \Big(\mathbb{I}(y_i = 0)\mathbf{u}(x_i),\ \mathbb{I}(y_i = 1)\mathbf{u}(x_i) \Big), \tag{6}$$

where $\mathbb{I}(\cdot)$ is the indicator function that checks the condition and returns 1 if the condition holds and 0 otherwise. Again the potential function $\mathtt{E}_{\mathbf{p}}$ contributed by pairwise features can be written as:

$$\mathtt{E}_{\mathbf{p}}(x_i, x_j, y_i, y_j) = \mathbf{w}_{\mathbf{p}}^{\mathrm{T}} \phi_{\mathbf{p}}(x_i, x_j, y_i, y_j), \tag{7}$$

where $\phi_{\mathbf{p}}$, which is the joint feature vector of the nodes x_i, x_j and their labels y_i, y_j for pairwise features, is defined as:

$$\begin{aligned} \phi_{\mathbf{p}}(x_i, x_j, y_i, y_j) \quad = \quad & \Big(\mathbb{I}'(y_i = 0, y_j = 0)\mathbf{p}(x_i, x_j), \\ & \mathbb{I}'(y_i = 0, y_j = 1)\mathbf{p}(x_i, x_j), \\ & \mathbb{I}'(y_i = 1, y_j = 0)\mathbf{p}(x_i, x_j), \\ & \mathbb{I}'(y_i = 1, y_j = 1)\mathbf{p}(x_i, x_j) \Big), \end{aligned} \tag{8}$$

where $\mathbb{I}'(\cdot,\cdot)$ is an indicator function that returns 1 if both the conditions (in its arguments) are true and return 0 otherwise. As given in (5), $\phi_{\mathbf{u}}$, which is a function of a node x_i and its corresponding label y_i in SG, returns a vector of length $2d_2$ (as $\mathbf{u}(x_i)$ is a d_2-dimensional vector, see Section 3.3). Similarly from (8), we can see that $\phi_{\mathbf{p}}$ is a function of a pair of adjacent nodes x_i, x_j and their corresponding labels y_i, y_j in G returning a vector of length $4d_3$ (as $\mathbf{p}(x_i)$ is d_3-dimensional vector, see Section 3.4). Thus, the joint feature vector $\boldsymbol{\phi}$ in (3) can be derived as:

$$\boldsymbol{\phi}(\mathrm{X,Y}) = \Big(\sum_{i=1}^{N} \phi_{\mathbf{u}}(x_i, y_i),\ \sum_{(x_i,x_j)\in E} \phi_{\mathbf{p}}(x_i, x_j, y_i, y_j) \Big).$$

Given these, we aim to learn $\mathbf{w}$ in (3) with the SSVM [15]. However, before we train the SSVM for determining $\mathbf{w}$, we have to define the loss between the true and predicted label vectors for any input X by the SSVM model. In order to do that, we calculate the Hamming loss between true and predicted label vectors as:

$$\Delta(Y^{(k)}, Y) = \sum_{i=1}^{N} \mathbb{I}(y_i^{(k)} \neq y_i), \tag{9}$$

where Y and $Y^{(k)}$ are (any) feasible and true label vectors of an input $X^{(k)}$ respectively. In (3), $\mathtt{E}(X, Y)$ eventually is the function which maps each label vector Y of the SG in X to a scalar value (or score). Hence, SSVM is learnt in such a way such that (a) the true label vector has the highest score and (b) the score is lower when the Hamming loss defined in (9) is higher. Keeping all these in mind, we define an SSVM with the formulation of one slack SVM [31] as:

$$\min_{\mathbf{w}} \quad \lambda||\mathbf{w}||^2 + \frac{1}{M}\sum_{k=1}^{M} \varepsilon_k, \tag{10}$$

such that,

$$\sum_{k=1}^{M} \Big(\Delta(Y^{(k)}, \hat{Y}^{(k)}) - \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(X^{(k)}, Y^{(k)}) + \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(X^{(k)}, \hat{Y}^{(k)}) \Big) \leq \sum_{k=1}^{M} \varepsilon_k,$$
$$\forall (\hat{Y}^{(1)}, \hat{Y}^{(2)}, ..., \hat{Y}^{(M)}) \in \mathcal{Y} \times \mathcal{Y} \times ... \times \mathcal{Y}, \tag{11}$$

where ε_k are the slack variables, λ is a positive regularization constant, $\mathcal{Y}$ is the set of all possible label vectors $\hat{Y}^{(k)}$ for $X^{(k)}$. This (10) is a convex quadratic optimization problem with $|\mathcal{Y}|^M$ number of constraints, where (11) represents all the constraints. Finally, we obtain the weight vector $\mathbf{w}$ which is learnt using the structured input of the images of shelves from the train-set following the approach in [31].

For any test shelf image I, we predict the gap/non-gap regions by creating the structured data X as explained in Section 3.2 and using the trained SSVM as:

$$\hat{Y} = F(X) = \underset{Y \in \mathcal{Y}}{\arg\max}\, \mathbf{w}^T \boldsymbol{\phi}(X, Y). \tag{12}$$

We solve (12) using AD^3 algorithm [32]. Hence $\hat{Y} = F(X)$ is the predicted label vector for the nodes of SG in X i.e. the labels of the superpixels in the shelf image I. Thus $\hat{Y}$ produces the predicted binary mask $\hat{B}$ for any shelf image as follows. Subsequently, the predicted label in $\hat{B}$ for each superpixel x_i is assigned to all the pixels contained within the superpixel x_i and thus finally the predicted binary mask is obtained with all pixels labelled either 0 (for non-gap) or 1 (for gap). Next we present the experiments, results and analysis.

4. Experiments

We first discuss implementation details of our method followed by the training protocol for various components of the proposed solution, the competing methods, and the performance metrics using which the methods are evaluated. Finally, we present our results and analysis.

Implementation Details. First of all, any shelf image is resized into a fixed-size 700×460 image. Next SLIC segmentation method [29] is run for getting $N = 1000$ number of superpixels. The experimental detail for choosing N is provided in Appendix A. The compactness [29] of SLIC method is experimentally set to 50 in this work.

The superpixels being of irregular shape cannot be sent to a CNN based model (ILFE and Siamese network architecture in our scheme) directly as input. So in our implementation, a patch of shape 32×32 is cropped out for each superpixel centred at the centroid of the superpixel. First, all these patches of superpixels are fed to ILFE for extracting linear feature vectors to fit into GCN.

The architecture of ILFE is composed of a convolutional (conv) block (comprising of a number conv and pooling layers) and two fc layers, fc-1 and fc-2 on top of convolutional block. The architecture of the conv block is identical to the entire conv block of VGG-19 [30]. fc-1 and fc-2 layers have 128 and 2 nodes respectively. fc-2 classifies each superpixel patch to either gap or non-gap. Further, we perform ReLU and dropout (with dropout probability 0.5) operations just after fc-1 and before fc-2. However, the learnable weights of the conv block of ILFE is initialized with the pre-trained weights of pytorch [33] implementation of VGG-19 while the weights of fc-1 and fc-2 layers are randomly initialized with the values in $[-1, 1]$ drawn following normal distribution. In our implementation, the input to fc-2 defines the 128-dimensional feature vectors for each of the superpixels.

The adjacency matrix of the SG and the 128-dimensional feature vectors of each superpixel are the input to our GCN. The proposed GCN is composed of 3 graph convolutional (gc) layers, namely gc-1, gc-2 and gc-3 as shown in Fig. 5. gc-1, gc-2 and gc-3 layers include 64, 16 and 2 nodes respectively. After each of the gc-1 and gc-2 layers, we execute ReLU and dropout (with dropout probability 0.5) operations. The weights of all these gc layers are randomly initialized with the values in $[-1, 1]$ drawn following a normal distribution. gc-3 layer eventually classifies each node of SG i.e. each superpixel to gap or non-gap aggregating the features of adjacent superpixels in gc layers. In this work, the output from gc-2 is the (unary) feature vector for any node of SG.

The feature vector for two adjacent pixels or an edge of the SG is determined by sending their patches to our SNA. As shown in Fig. 6, SNA consists of 3 conv blocks: *conv-block-1*, *conv-block-2*, *conv-block-3* and 1 fc block: *fc-block*. Since, *conv-block-1* and *conv-block-2* share their weights, they are essentially treated as one block whose architecture is identical to the conv block of VGG-16. The outputs from *conv-block-1* (which takes one superpixel as input) and *conv-block-2* (which receives another superpixel as its input) are concatenated and fed to *conv-block-3*. The structure of *conv-block-3* is identical to the chunk of 10-th to 16-th conv layers (including intermediate maxpool layer) of VGG-19. The weights of these blocks are initialized with weights of respective blocks of the VGG pre-trained models. The *fc-block* (consisting of 3 fc layers having 64, 16 and 2 nodes respectively) takes the output of *conv-block-3* and classifies the adjacent superpixels to be identically labeled or not. *fc-block* also performs ReLU and dropout (with dropout probability 0.5) operations after each of first two fc layers. Weights of all these fc layers are randomly initialized with the values in $[-1, 1]$ drawn following normal distribution. In our model, the output from penultimate layer of *fc-block* is eventually the (pairwise) feature vector for the edge between the adjacent superpixels in SG. The SG along with these unary and pairwise feature vectors are sent to SSVM for the identification of gaps and non-gaps in shelf images.

All above discussed deep learning based models ILFE, GCN and SNA are implemented with pytorch library [33] while SSVM is designed with pystruct library [34] of python. Next, we explain the training strategies of the above-discussed models.

Training of Various Networks. During training of the deep learning based models ILFE, GCN, and SNA of our proposed approach, approximately 80% images of the train-set are used for training while the rest 20% images of the train-set are utilized for validation of the network. All the networks are trained by applying the softmax function on output (referred to as softmax output) and then calculating the cross-entropy loss [35] between the softmax output and one hot label vector. Adam optimizer at learning rate of 0.0001, weight decay of 5e-4, and mini-batch (of shelf images) of 1 used to learn the networks. ILFE and SNA are optimized up to 150 epochs while GCN is trained for at most 400 epochs. On the other hand, for the training of SSVM, the chosen tunable parameters are a maximum iteration of 100, a regularization parameter of 0.1 and a convergence tolerance of 0.1. Next, we present the competing methods.

Competing Methods. We compare our proposal with the competing methods: U-Net [16], DeepLabV3 [17], LinkNet [18], FPN [19], PSPNet [20], DeepLabV3+ [21], PAN [22], MA-Net [23]. We have run pytorch implementation of all these methods with default setup available in GitHub [36]. All these networks are trained for 100 epochs. The performance metric is defined next.

Performance Measure. The methods are evaluated using the metric, *intersection-over-union* (IoU) [37, 38] used in evaluating the performances of the methods for semantic segmentation. The IoU essentially determines the similarity between the predicted binary mask $\hat{B}$ and pixel-level binary mask I_{gt} (i.e. the ground truth) of an image of a shelf. Subsequently, the gap identification performance for the M_{tst} number of test images in a dataset is defined by the *mean IoU* ($mIoU$) as :

$$mIoU = \frac{1}{M_{tst}} \left(\frac{|I_{gt} \cap \hat{B}|}{|I_{gt} \cup \hat{B}|} \right). \tag{13}$$

Next, we present the results and analysis.

4.1. Results and Analysis

The experiments are carried out on three publicly available datasets Grocery Products (GrocProd) [25], WebMarket (WebMkt) [26], and GroZi [27] from which we select 305, 98, and 50 number of shelf images and create ground truth specifying gap/non-gap regions as explained in Section 2. For each dataset, we randomly choose approximately 60% of these shelf images as the train-set and the rest 40% as the test-set. The train-set includes 184, 59, and 30 shelf images in GrocProd, WebMkt, and GroZi respectively while the test-set contains 121, 39, and 20 images in the respective datasets. These train-set and test-set containing images and their ground truths (or pixel-level binary masks) are made public in GitHub [24].

Quantitative Results: Table 1 presents the gap identification accuracy in terms of $mIoU$ (%) of the proposed approach including the competing ones. The proposed scheme outperforms deep learning based state-of-the-art methods in all the evaluations by at least ∼1%. The maximum performance improvement from the nearest competitor is ∼3% (see rightmost column for GroZi in Table 1) which is indeed remarkable. In fact, our method achieves higher accuracy for the GroZi dataset. Our method performs equally well for the datasets having a larger or a smaller number of training images (for Grocery Products there are 184 training images while for GroZi there are only 30 training images). On the contrary, the performance of solely deep learning based methods deteriorates with the decreasing size of the train-set, which is why the margin of $mIoU$ for our method w.r.t. others becomes higher when the train-set is smaller as witnessed for GroZi. However, the results for all the methods are inferior on the

Table 1: Gap identification results of various methods on benchmark datasets. DLV3 and DLV3+ represents DeepLabV3 and DeepLabV3+ respectively.

Methods	*mIoU* (%) on Benchmark Datasets		
	GrocProd [25]	WebMkt [26]	GroZi [27]
U-Net [16]	69.36	66.83	81.76
DLV3 [17]	67.82	64.89	78.47
LinkNet [18]	68.73	66.28	79.25
FPN [19]	67.36	66.19	77.60
PSPNet [20]	66.19	64.55	72.30
DLV3+ [21]	68.98	64.75	75.70
PAN [22]	68.95	64.93	77.31
MA-Net [23]	69.64	63.60	81.66
Proposed	**70.62**	**69.20**	**84.58**

GrocProd and WebMkt datasets compared to GroZi due to the large variation in the texture of the gaps and in the packaging of products.

Statistical Significance Test: We perform an in-depth comparison of the proposed approach with two of its closest competitors (identified from Table 1) U-Net [16] and MA-Net [23]. We have performed 10-fold cross-validation on all three datasets. The mean (μ) and standard deviation (σ) of the $mIoU$ accuracy obtained for 10 folds of each dataset are tabulated in Table 2. In order to highlight the efficacy of the proposed approach against its closest competitors, we have carried out a Wilcoxon Rank-Sum (WRS) test [39]. Table 2 provides the outcome of the WRS test which we express in the form of p-values of rejecting the null hypothesis.

The null hypothesis for each test is $\mathcal{H}_0$: the performances of the control method (i.e. the proposed one) and the competing method are identical over the 10-fold cross-validation, versus $\mathcal{H}_1$: the performances of the control method and the competitor are different over the 10-fold cross-validation. In our experiment,

Table 2: Performance comparison: the results of the 10-fold cross-validation and corresponding Wilcoxon Rank-Sum (WRS) test. For each dataset, $\mu \pm \sigma$ of the 10-fold cross-validation results ($mIoU$ in %) are shown, where μ and σ denote the mean and standard deviation respectively. Subsequently, the p-values obtained from the WRS and the corresponding selected hypothesis are presented for each of the competitors on each of the datasets.

		U-Net [16]	MA-Net [23]	Proposed (Control Method)
GrocProd	$\mu \pm \sigma$	71.64 ± 3.77	71.21 ± 4.73	72.36 ± 4.38
	p-values	0.064	0.045	-
	Hypothesis	$\mathcal{H}_0$	$\mathcal{H}_1$	-
WebMkt	$\mu \pm \sigma$	64.34 ± 8.66	63.96 ± 8.73	65.76 ± 7.97
	p-values	0.045	0.021	-
	Hypothesis	$\mathcal{H}_1$	$\mathcal{H}_1$	-
GroZi	$\mu \pm \sigma$	83.92 ± 4.64	83.62 ± 4.47	85.13 ± 5.18
	p-values	0.038	0.025	-
	Hypothesis	$\mathcal{H}_1$	$\mathcal{H}_1$	-

NOTES:

$\mathcal{H}_0$: The performances of the control method (i.e. the proposed one) and the competing method are statistically comparable over the 10-fold cross-validation.

$\mathcal{H}_1$: The performances of the control method and the competitor are significantly different over the 10-fold cross-validation.

Threshold for p: 0.05

we consider the 5% significance level, i.e. $\mathcal{H}_0$ is rejected if $p < 0.05$. On any dataset, for any competitor, assume the hypothesis $\mathcal{H}_1$ is selected. Then our algorithm performs significantly better than its competitor, if the mean (μ) performance of the proposed method is greater than that of the competitor. Our solution performs significantly worse, if the mean performance of our method is lower than the mean performance of the competitor. In case the hypothesis $\mathcal{H}_0$ is selected, both the methods are statistically comparable. From Table 2, we can clearly conclude that the proposed method is significantly better than its competitors except in one case (see U-Net on GrocProd dataset), where our method and the competitor are significantly comparable.

Qualitative Results: A few example qualitative results of our method are provided in Fig 7. The efficiency of our method is established in the six example results shown in the top six rows of Fig. 7, where the predicted binary masks are almost similar to the true pixel-level binary masks. The bottom two rows of Fig. 7 illustrate two notable failure cases. Our analysis finds that the non-gap is misidentified as gap in both the images due to darkness in the packaging of the product. Next, we perform the ablation study.

4.1.1. Ablation Study

The ablation study is carried out on all benchmark datasets for investigating the contributions of different components of the proposed scheme. Our proposal has three primary components such as: node feature extractor (NFE) i.e. ILFE + GCN, edge feature extractor (EFE) i.e. GCN, and SSVM i.e. our classifier. In the proposed scheme, NFE is the basic component without which SSVM can not be executed. Therefore, next, we provide the efficacy of the remaining two components EFE and SSVM of our proposal to identify the gaps or non-gaps in the shelf images.

Contribution of EFE: SSVM can be modelled using the SG and its node features NFE, without explicitly extracting edge features of SG. In that case,

Table 3: Performances of our method removing or adding different components of it on the test-set of our benchmark datasets

Components in our scheme	*mIoU* (%)		
	GrocProd [25]	WebMkt [26]	GroZi [27]
(i) NFE	65.98	60.06	79.11
(ii) NFE + SSVM	66.75	63.76	79.89
(iii) **Proposal:** NFE + EFE + SSVM	70.62	69.20	84.58

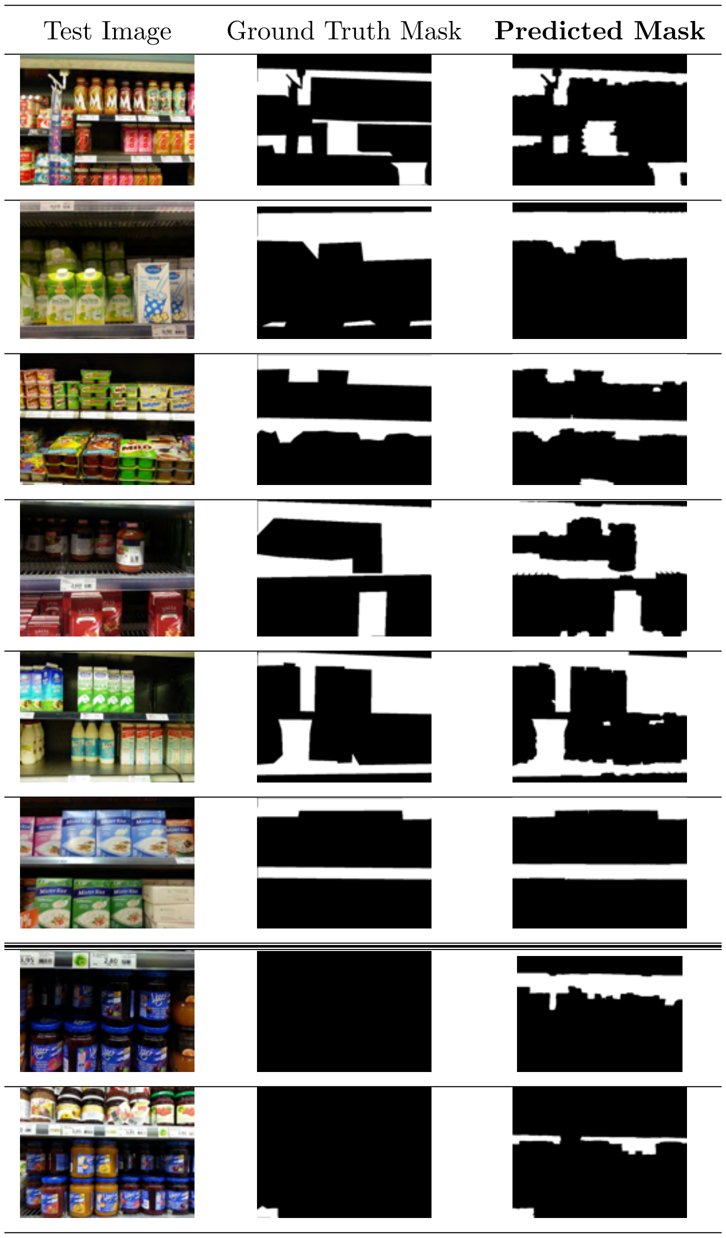


Figure 7: A few qualitative results from the test-set of various datasets. The top six rows show the efficacy of the proposed scheme while the last two rows present the failure cases of our solution when the products with darker packaging appear like a gap.

SSVM considers the adjacency value (1 or 0) in the adjacency matrix of SG as an edge feature. This setup, which can be denoted as NFE + SSVM, examines the necessity of EFE i.e. SNA in our proposal. If we remove the EFE module from our proposal, the performance drops at least ∼5% (compare rows (ii) and (iii) of Table 3) that clearly shows the importance of EFE. This happens because the Siamese network considers each pair of adjacent superpixels and efficiently captures the discriminatory characteristics between them as the edge feature.

Contribution of SSVM: We can use only NFE for obtaining the gap identification results. The results of this model essentially illustrate the contribution of SSVM. If we look at the performances of our proposal (row (iii)) and NFE (row (i)) in Table 3, the difference is at least ∼5% and at most ∼10%. Thus the necessity of SSVM can be clearly noticed.

Therefore, in Table 3, the three possible cases are studied to perform superpixel classification for identifying the gaps. NFE can do the task alone; NFE and SSVM together can also perform the task; finally, the proposed scheme i.e. NFE, EFE, and SSVM put together, is capable of completing the task. We can see that the improvement of NFE + SSVM over NFE is marginal. When we add EFE with the NFE + SSVM, we can see a significant performance jump. But, in the proposed solution, EFE without SSVM cannot exist. EFE is utilized to extract the edge weights to be used in the SSVM to classify the superpixels. And we have witnessed the effectiveness of EFE (our novel contribution in this work) in Table 3. Therefore, we can see that EFE embedded in SSVM provides improved performance.

However, our ablation study suggests that all three components of our proposal are significant in accurately identifying gaps and non-gaps in shelf images. To be specific, this study infers that EFE in SSVM contributes most in achieving higher gap identification performance with respect to competing approaches. Next we analyze the inference time (i.e. test time) of the proposed method.

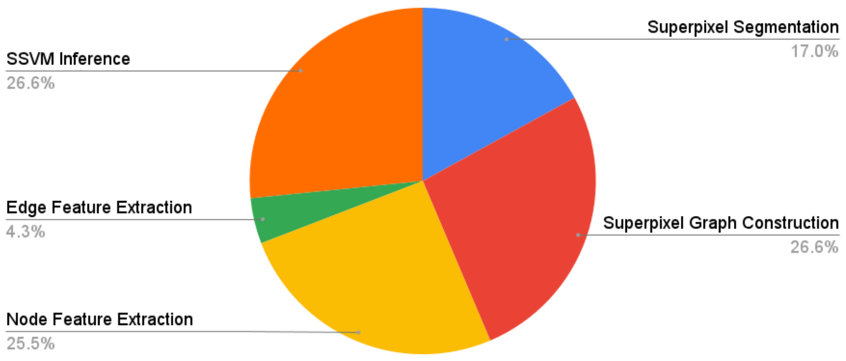


Figure 8: The pie-chart representing the distribution of the execution time consumed by the different building blocks of the proposed approach for identifying the gaps in a shelf image

4.1.2. Notes on Inference Time

The proposed algorithm is implemented in python and tested in a computing system with the following specifications: 96GB RAM, Intel Core i9-9820X CPU 3.30GHz×20 and 24GB TITAN RTX GPU. The modules of the proposed approach involved during inference are: (a) Superpixel Segmentation (Section 3.1), (b) Superpixel Graph Construction (Section 3.2), (c) Node Feature Extraction (Section 3.3), (d) Edge Feature Extraction (Section 3.4) and (e) SSVM Inference (Section 3.5). For identifying the gaps in a (test) shelf image, the time consumed by each of these modules of our scheme is presented using a pie-chart in Fig. 8.

The total time taken by the un-optimized code per rack of the proposed approach is ∼0.93 seconds. Among all the modules, as expected, Node Feature Extraction consumes higher time due to the Graph Convolution process explained in Section 3.3. Further, the CPU implementations of graph manipulation (Superpixel Segmentation and Superpixel Graph Construction) and the SSVM inference process have increased the overall execution time. However, the (deep learning based) competing methods take ∼0.45 seconds for identifying the gaps in a (test) shelf image. All the competing methods are end-to-end deep learning based methods, which are entirely implemented in GPU. On the contrary, our current implementation of the proposed approach involves CPU along with GPU implementation. This essentially increases the test time. Our analysis finds that the test time of our scheme should be close to that of the competitors if we implement graph manipulation and SSVM inference in the GPU. However, with this fraction of second increase of test time w.r.t. the competing approaches, the proposed scheme yields significantly better performances in almost all the cases. Next, we discuss the importance of our method in context of retail stores.

4.1.3. Suitability of the proposal for retail store environment

The deep learning based approaches usually require enormous training data. Limited training images result in over-fitting of the model during training and hence poor generalized performance. Due to the availability of limited training data for the application under consideration, we have used the structured support vector machine that learns a much lesser number of parameters ($2 \times$ number of node features $+ 4 \times$ number of edge features) compared to any deep learning based methods.

The deep learning models, that we have utilized in this work, are GCN to extract the features of the superpixels and a Siamese network to extract the features of a pair of superpixels. In order to train these networks, a minimal set of labelled data (i.e. annotated shelf images) is good enough.

Assume there are 30 shelf images in the training set, each of which has 1000 superpixels as decided through experiments. In that case, the training data, which is used for training the node feature extractor (GCN), contain 30×1000 samples. For training the edge feature extractor (Siamese network), we have $30 \times$ number of edges in each SG (obviously more than 1000) training samples.

Such a training scheme is large enough to train the proposed GCN or Siamese network. On the contrary, all other deep learning based segmentation methods considered in our comparative study (see Table 1), require to train their millions of parameters. As a result, the proposed scheme outperforms all these methods as evident in Table 1. Hence, in the context of a retail store with a limited number of training images, given that the product display plan in supermarkets changes quickly, the proposed scheme is expected to be a better choice for an application like identification of gaps. Next, we conclude the paper.

5. Conclusions

The method presented in this paper uses graph convolutional network (GCN) for feature extraction of the superpixels independently while Siamese network architecture (SNA) captures the similarity of the neighbouring superpixels in a feature embedding framework. Finally, the features extracted from GCN and SNA are fed to SSVM for the classification of the superpixels. Utilizing GCN and SNA to obtain the node and edge features of a superpixel graph for training SSVM has never been attempted. We have shown their importance in the classification of gaps on the rack with SSVM. We consider this to be the key contribution of our proposal. We believe that the release of datasets for the gap detection problem is an important opportunity for the application-driven computer vision research community. We further plan to add more annotated data. In future, we aim to formulate an end-to-end strategy for training the learners (ILFE, SNA and SSVM) jointly, to extract node & edge features of superpixel graphs and to classify the superpixels.

Acknowledgements

The authors wish to thank Mr. Avishek Kumar Shaw for helping us in annotating the shelf images for creating ground truths of the benchmark datasets.

Appendix A. Choice of Number of Superpixels (N)

An example shelf image I, segmented into four superpixels is shown in Fig. 4(a). The pixel-level binary mask of I is I_{gt} as shown in Fig. 4(c). In order to train the SSVM, we label each superpixel by majority voting of white or black pixels to create the superpixel-level binary mask B as explained in Section 3.2 and as shown in Fig. 4(d).

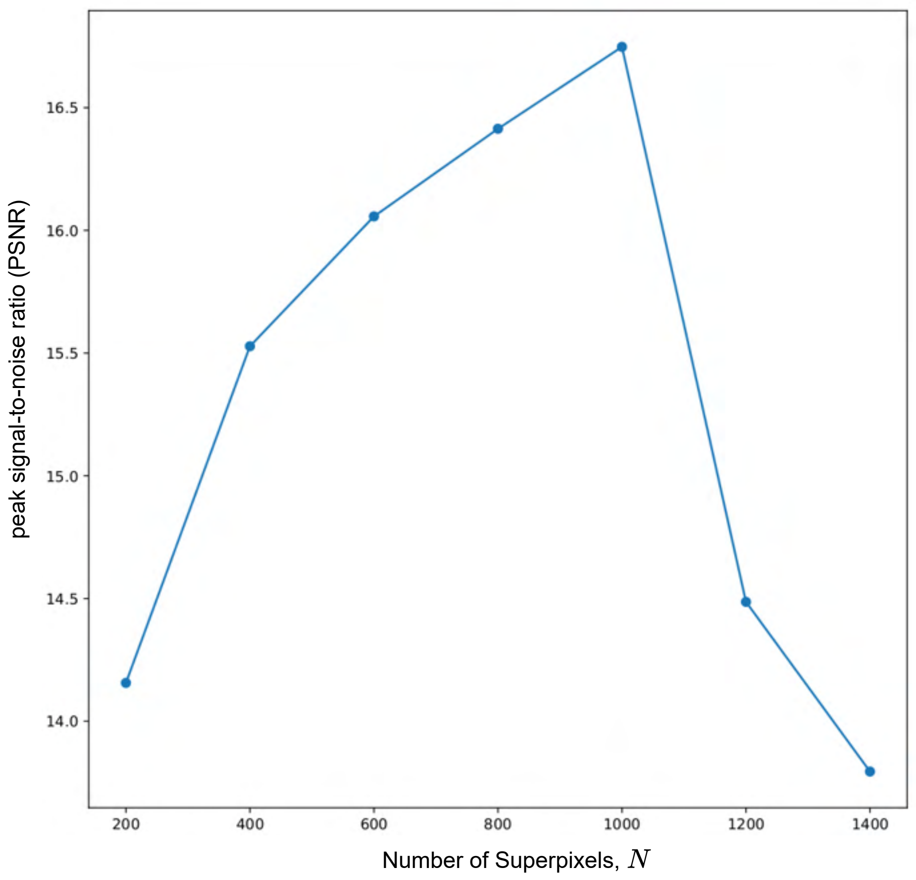


Figure 9: Peak signal-to-noise ratio (PSNR) (in dB) values between I_{gt} and B for different numbers of superpixels N generated by SLIC superpixel segmentation algorithm

The choice of number of superpixels (N) should be such that the (pixel-wise) difference between the pixel-level binary mask i.e. ground truth $I_{gt}^{(\mathrm{k})}$ and superpixel-level binary mask $B^{(\mathrm{k})}$ is minimum for the training image $I^{(\mathrm{k})}$. That is, Fig. 4(c) and Fig. 4(d) become almost identical. In order to ensure that, we choose N in a way such that the peak signal-to-noise ratio (PSNR) [40] (in db) between $I_{gt}^{(\mathrm{k})}$ and $B^{(\mathrm{k})}$ is maximum. Thus, we compute the mean PSNR for the images in the training set of the WM dataset varying N from 200 to

1400 in intervals of 200. The mean PSNR for various N is plotted in Fig. 9. It can be seen that till about $N = 1000$, the PSNR increases. This means that more the granularity in segmentation, more (pixel-wise) similar are $I_{gt}^{(\mathrm{k})}$ and $B^{(\mathrm{k})}$. However, for $N > 1000$, PSNR starts to fall due to the inconsistent superpixel boundaries determined by the SLIC algorithm. Therefore, we set $N = 1000$ in our implementation.

References

[1] B. Santra, D. P. Mukherjee, A comprehensive survey on computer vision based approaches for automatic identification of products in retail store, Image and Vision Computing 86 (2019) 45–63.

[2] S. Liu, W. Li, S. Davis, C. Ritz, H. Tian, Planogram compliance checking based on detection of recurring patterns, IEEE MultiMedia 23 (2) (2016) 54–63.

[3] A. Ray, N. Kumar, A. Shaw, D. P. Mukherjee, U-pc: Unsupervised planogram compliance, in: European Conference on Computer Vision, Springer, 2018, pp. 598–613.

[4] A. Tonioni, L. Di Stefano, Domain invariant hierarchical embedding for grocery products recognition, Computer Vision and Image Understanding 182 (2019) 81–92.

[5] B. Santra, A. Paul, D. P. Mukherjee, Deterministic dropout for deep neural networks using composite random forest, Pattern Recognition Letters 131 (2020) 205 – 212.

[6] B. Santra, A. K. Shaw, D. P. Mukherjee, Part-based annotation-free fine-grained classification of images of retail products, Pattern Recognition 121 (2022) 108257.

[7] L. Karlinsky, J. Shtok, Y. Tzur, A. Tzadok, Fine-grained recognition of thousands of object categories with single-example training, in: Proceedings

of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4113–4122.

[8] B. Santra, A. K. Shaw, D. P. Mukherjee, Graph-based non-maximal suppression for detecting products on the rack, Pattern Recognition Letters 140 (2020) 73 – 80.

[9] B. Santra, A. K. Shaw, D. P. Mukherjee, An end-to-end annotation-free machine vision system for detection of products on the rack, Machine Vision and Applications 32 (3) (2021) 1–13.

[10] R. Moorthy, S. Behera, S. Verma, On-shelf availability in retailing, International Journal of Computer Applications 115 (2015) 47–51.

[11] K. Michael, L. McCathie, The pros and cons of rfid in supply chain management, in: International Conference on Mobile Business (ICMB'05), 2005, pp. 623–629.

[12] R. Yılmazer, D. Birant, Shelf auditing based on image classification using semi-supervised deep learning to increase on-shelf availability in grocery stores, Sensors 21 (2021) 327.

[13] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks (2017). `arXiv:1609.02907`.

[14] G. Koch, R. Zemel, R. Salakhutdinov, Siamese neural networks for one-shot image recognition, 2015.

[15] H. Xue, S. Chen, Q. Yang, Structural support vector machine, in: F. Sun, J. Zhang, Y. Tan, J. Cao, W. Yu (Eds.), Advances in Neural Networks - ISNN 2008, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 501–511.

[16] O. Ronneberger, P. Fischer, T. Brox, U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical

image computing and computer-assisted intervention, Springer, 2015, pp. 234–241.

[17] L.-C. Chen, G. Papandreou, F. Schroff, H. Adam, Rethinking atrous convolution for semantic image segmentation, arXiv preprint arXiv:1706.05587.

[18] A. Chaurasia, E. Culurciello, Linknet: Exploiting encoder representations for efficient semantic segmentation, in: 2017 IEEE Visual Communications and Image Processing (VCIP), IEEE, 2017, pp. 1–4.

[19] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2117–2125.

[20] H. Zhao, J. Shi, X. Qi, X. Wang, J. Jia, Pyramid scene parsing network, in: Proceedings of the IEEE conference on computer vision and pattern recognition, 2017, pp. 2881–2890.

[21] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, H. Adam, Encoder-decoder with atrous separable convolution for semantic image segmentation, in: Proceedings of the European conference on computer vision (ECCV), 2018, pp. 801–818.

[22] H. Li, P. Xiong, J. An, L. Wang, Pyramid attention network for semantic segmentation, in: BMVC, 2018.

[23] T. Fan, G. Wang, Y. Li, H. Wang, Ma-net: A multi-scale attention network for liver and tumor segmentation, IEEE Access 8 (2020) 179656–179665.

[24] B. Santra, U. Ghosh, D. P. Mukherjee, Datasets for identification of gaps in the images of shleves in supermarkets, `https://github.com/gapDetection/gapDetectionDatasets` (2021).

[25] M. George, C. Floerkemeier, Recognizing products: A per-exemplar multi-label image classification approach, in: European Conference on Computer Vision, Springer, 2014, pp. 440–455.

[26] Y. Zhang, L. Wang, R. Hartley, H. Li, Where's the weet-bix?, in: Asian Conference on Computer Vision, Springer, 2007, pp. 800–810.

[27] M. Merler, C. Galleguillos, S. Belongie, Recognizing groceries in situ using in vitro training data, in: Computer Vision and Pattern Recognition, 2007. CVPR'07. IEEE Conference on, IEEE, 2007, pp. 1–8.

[28] K. Wada, labelme: Image Polygonal Annotation with Python, `https://github.com/wkentaro/labelme` (2016).

[29] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, Slic superpixels, Tech. rep. (2010).

[30] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.

[31] T. Joachims, T. Finley, C.-N. Yu, Cutting-plane training of structural svms, Machine Learning 77 (2009) 27–59.

[32] A. Martins, M. Figueiredo, P. Aguiar, N. Smith, E. Xing, Ad3: Alternating directions dual decomposition for map inference in graphical models, Journal of Machine Learning Research 16 (2015) 495–545.

[33] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in pytorch, in: NIPS-W, 2017.

[34] A. C. Müller, S. Behnke, Pystruct: learning structured prediction in python., J. Mach. Learn. Res. 15 (1) (2014) 2055–2060.

[35] M. Tzelepi, A. Tefas, Improving the performance of lightweight cnns for binary classification using quadratic mutual information regularization, Pattern Recognition 106 (2020) 107407.

[36] P. Yakubovskiy, Segmentation models pytorch, `https://github.com/qubvel/segmentation_models.pytorch` (2020).

[37] L. R. Dice, Measures of the amount of ecologic association between species, Ecology 26 (3) (1945) 297–302.

[38] L. Wang, J. Gu, Y. Chen, Y. Liang, W. Zhang, J. Pu, H. Chen, Automated segmentation of the optic disc from fundus images using an asymmetric deep learning network, Pattern Recognition 112 (2021) 107810.

[39] M. Hollander, D. A. Wolfe, E. Chicken, Nonparametric statistical methods, Vol. 751, John Wiley & Sons, 2013.

[40] S. T. Welstead, Fractal and wavelet image compression techniques, Vol. 40, Spie Press, 1999.