Pattern Recognition 121 (2022) 108257

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/patcog

Part-based annotation-free fine-grained classification of images of retail products[☆]

Bikash Santra[a,*], Avishek Kumar Shaw[b], Dipti Prasad Mukherjee[a]

[a] Electronics and Communication Sciences Unit, Indian Statistical Institute, 203, B. T. Road, Kolkata 700108, India
[b] TCS Limited, Ecospace-1A, Ecospace Business Park, Block-1A, Eco Space, Kolkata 700156, India

A R T I C L E I N F O

Article history:
Received 26 January 2021
Revised 30 July 2021
Accepted 13 August 2021
Available online 14 August 2021

Keywords:
Fine-grained classification
Reconstruction-classification network
Supervised convolutional autoencoder
Retail product detection

A B S T R A C T

We propose a novel solution that classifies very similar images (fine-grained classification) of variants of retail products displayed on the racks of supermarkets. The proposed scheme simultaneously captures object-level and part-level cues of the product images. The object-level cues of the product images are captured with our novel reconstruction-classification network (RC-Net). For annotation-free modeling of part-level cues, the discriminatory parts of the product images are identified around the keypoints. The ordered sequences of these discriminatory parts, encoded using convolutional LSTM, describe the products uniquely. Finally, the part-level and object-level models jointly determine the products explicitly explaining coarse to finer descriptions of the products. This bi-level architecture is embedded in R-CNN for recognizing variants of retail products on the rack. We perform extensive experiments on one In-house and three benchmark datasets. The proposed scheme outperforms competing methods in almost all the evaluations.

1. Introduction

Automatic detection of products displayed on the racks in supermarkets is an attractive research problem. This application provides enhanced consumer experience and associated business potential [1]. The problem can be typically defined as detection of objects in a crowded scene where images of only one (or very few) sample(s) per object is (are) available a prior.

Classifying (very) similar (i.e. non-identical) variants of products is one of the most challenging tasks of the above-mentioned problem. We refer this task as fine-grained classification which is the focus of this paper. The fine-grained variations are usually due to slight variations in text, size, or color of the package. A set of examples are shown in Fig. 1. These product images are used as templates and used for training. In contrast the quality of rack images, from where the product images are cropped and used as test images for our problem, is affected due to store level illuminations. An example of such differences between training and test images are shown in Fig. 2. Both marginal variations in image content and illumination, make fine-grained classification of product images much more challenging compared to classical fine-grained object classification [2–7].

This paper presents a novel solution for the fine-grained classification of retail products utilizing object-level and part-level cues. The solution emulates human-like approach. First, features due to overall content of the product are captured followed by features for finer discriminating characteristics. For example, in Fig. 1(a), first product can be differentiated from the second by looking at the weights 805g and 500g mentioned on the package.

In the first step of our proposed approach, we introduce a reconstruction-classification network (RC-Net) which is essentially a deep supervised convolutional autoencoder (SCAE) similar to the supervised autoencoder (SAE) [9]. RC-Net is robust to classification task due to enhanced generalizability of the network (as discussed in Section 3.1). This helps in product recognition under varying store-level illuminations.

The second step of our scheme essentially boosts the classification performance of the first step. The discriminatory parts of the products are searched (in an unsupervised manner) and organized as an ordered sequence to uniquely describe the products. These sequences of parts are modeled using convolutional LSTM (conv-LSTM) [10]. Note that part-level annotations are not used keeping in mind implementation challenges of the system for fast changing product lines [1]. Finally, classification scores from both the steps, jointly decide labels of the products in the test images.

[☆] This work is partially supported by TCS Limited. One Titan XP GPU used for this research was donated by NVIDIA Corporation.
[*] Corresponding author.
E-mail addresses: bikash.santra@isical.ac.in (B. Santra), shaw.avishek@tcs.com (A.K. Shaw), dipti@isical.ac.in (D.P. Mukherjee).

https://doi.org/10.1016/j.patcog.2021.108257
0031-3203/© 2021 Elsevier Ltd. All rights reserved.

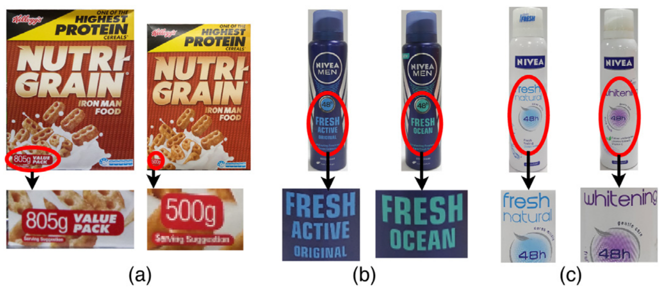

Fig. 1. Top row presents examples of fine-grained products having minute differences (see red ellipses) in (a) size and text and, (b)-(c) color and text which are highlighted in bottom row. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

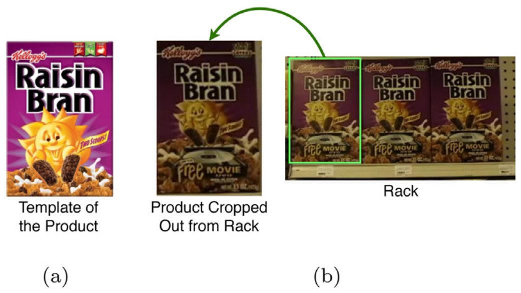

Fig. 2. Differences in (a) training image (template of a product) and (b) test image (cropped product from rack using green bounding box) from GroZi-120 [8] dataset. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

The rest of the paper is structured as follows. Related works and contributions are presented in Section 2. Section 3 explains the proposed fine-grained classification scheme. Experiments and results are provided in Section 4 followed by conclusions in Section 5.

2. Related works and contributions

2.1. Generic fine-grained classification

The state-of-the-art methods for fine-grained classification of objects can be classified into three groups (a) part based models [2,3], (b) two-level attention models [4,5], and (c) CNN with second-order pooling [11,12]. The part based models first determine the key parts of the objects. These key parts are then used to design a CNN (like part-stacked CNN [2]) or LSTM (like complementary parts model [3]) for classification. On the other hand, two-level attention models [4,5] jointly learn object and part information to solve fine-grained classification. The convolution layers at the end of a deep CNN capture the finer descriptions of an object [2]. Moreover, in the CNN, second order pooling (like bilinear pooling [11,12]) models local pairwise feature interactions in a translation invariant manner. Due to this, bilinear pooling is integrated in our part-level modeling of the products. Compared to deep CNNs, our proposed RC-Net is more powerful due to improvement in generalizability. Moreover, our novel part-level module improves the overall fine-grained classification performance.

2.2. Deep reconstruction-classification network

Ghifary et al. [13] present a deep reconstruction-classification network (DRCN) for unsupervised domain adaptation. The architecture of the proposed RC-Net (24 convolutional layers and 2 fully connected layers) is much more deeper than DRCN (6 convolutional layers and 3 fully connected layers) for capturing the fine-grained representation of the products. The layers in encoder-decoder module of DRCN include both convolutional and fully connected layers. On the contrary, all the layers in encoder-decoder module of our proposed network are convolutional like SegNet [14]. This improves the reconstruction power of the proposed network by preserving the local relationship between the neighboring pixels in the image. This is evident in the results shown in Section 4.1.

2.3. Generic object detection

For the given problem, the annotated images of racks are not available [1]. This restricts use of advanced deep learning based object detection methods like YOLO [15] and Mask R-CNN [16]. On the contrary, individual product templates available for our application can be used as training data. Using this data, R-CNN [17], responsible for generation, classification and disambiguation of region proposals, could be an useful approach for detecting products in a rack. The performance of R-CNN is evaluated in Section 4.2.

2.4. Retail product detection

The state-of-the-art methods in this category can be grouped into (a) template matching [8,18–21] and (b) machine learning based methods [21–23].

The template matching based methods first identify a few potential regions for the products displayed in a rack image and crop these regions from the rack image. Subsequently, the cropped regions are matched with the product templates for identification of products. The authors of [19] find the potential regions to be cropped by moving rectangular regions (often referred to as sliding windows) of different scales over the rack image. The closest product class of a cropped region is then determined using a template matching scheme. Finally, the products are detected by determining the maximum weighted path of a directed acyclic graph of cropped regions. Similarly, the authors of [20] slide multiple windows to obtain the (overlapping) cropped regions which are matched with the product templates and disambiguated using non-maximal suppression. In [8], Merler et al. investigate two sliding window based template matching approaches: histogram and SIFT matching methods [24]. In [21], a number of rectangular regions are extracted at each corner point detected using Harris corner detector [25]. Subsequently, each extracted region is matched with the product templates. In [18], Harris-Affine interest region detector [26] is used to obtain the important regions in an image. Subsequently, a SIFT matching technique is implemented to detect the products.

Similar to template matching, the machine learning based methods first extract some probable regions representing the products in a rack as discussed in the previous paragraph. But the matching score and product label for each region are derived using machine learning based models. George et al. [22] use discriminative random forest (DRF) [27] for estimating multi-class ranking of the potential regions. Subsequently, they implement a deformable spatial pyramid based fast dense pixel matching [28] and genetic algorithm based optimization scheme [29] for detecting products. The authors of [21] and [23] detect the products by determining class labels and classification scores of the potential regions using convolutional neural network.

The performance of all the above methods are compared with that of R-CNN using proposed fine-grained classification scheme in Section 4.2. Further, there exists some state-of-the-art approaches like [30,31] that address the classification (i.e. recognition) of retail products but not the exact localization of products on the rack. The

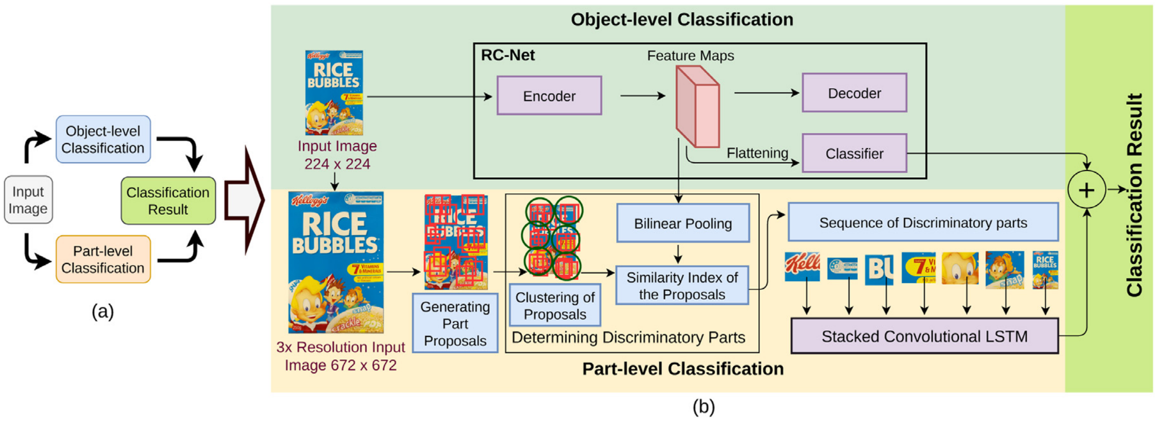
Object-level Classification
Input Image
Classification Result
Part-level Classification
(a)
Object-level Classification
RC-Net
Feature Maps
Encoder
Decoder
Flattening
Classifier
Input Image 224 x 224
3x Resolution Input Image 672 x 672
Generating Part Proposals
Clustering of Proposals
Bilinear Pooling
Similarity Index of the Proposals
Determining Discriminatory Parts
Sequence of Discriminatory parts
Stacked Convolutional LSTM
Part-level Classification
Classification Result
(b)

Fig. 3. (a) Flowchart and (b) corresponding block diagram (highlighting intermediate steps) of the proposed scheme. For part-level classification, example input image is zoomed three times (for details, see Section 3.2).

performance of our fine-grained classifier is compared with [31] in Section 4.1.

The diverse aspects of our proposed architecture are as follows:

(a) A deep convolutional RC-Net for fine-grained classification of products is introduced. The encoder-decoder architecture of RC-Net significantly improves the classification performance.
(b) In order to boost the performance of fine-grained classification, we introduce a conv-LSTM [10] based part-level classification model to encode discriminatory parts (identified by a unique unsupervised technique) of the products without using annotation of sub-parts of the products.
(c) In order to improve the performance of R-CNN for detecting retail products, the proposed fine-grained classifier is embedded in R-CNN without using annotation of racks on which the products are being displayed. This will be discussed in Section 4.2.

Overall the unique contribution of the proposed approach is: Unlike state-of-the-art approaches, we propose a novel fine-grained classification scheme where proposed RC-Net, bilinear pooling [11,12] and Conv-LSTM are utilized in tandem for weighted object-level and part-level classification. Next we present the proposed classification strategy.

3. Classification of fine-grained products

As mentioned in Section 1, the proposed scheme first extracts the coarse representations (or object-level features) of the products followed by detection of fine-grained representations (or local key features) of the products. The schematic of the proposed scheme is shown in Fig. 3. Next we explain the above-mentioned modules of the proposed scheme.

3.1. Object-level classification

The object-level classification of the products are performed using reconstruction-classification network (RC-Net).

RC-Net: It consists of three modules: encoder, decoder and classifier as shown in Fig. 4. The encoder-decoder architecture of RC-Net is built following SegNet [14] for reconstruction of the product images. However, SegNet is a convolutional encoder-decoder architecture designed for segmentation of images. Convolutional autoencoder (CAE) differs from SegNet as CAE reconstructs the input while SegNet produces segmentation mask in its vanilla implementation [14]. As mentioned earlier in Section 1, the proposed RC-Net is a supervised CAE (SCAE). The primary difference between SegNet and RC-Net is that the RC-Net not only reconstructs the input but also determines the classification score for the input image. Therefore, unlike SegNet, RC-Net simultaneously performs the reconstruction and classification of input product images. Due to this, our proposed network is referred to as reconstruction-classification network or RC-Net. The classifier of RC-Net is a fully-connected network, which accepts the output of the encoder for classification.

Let I be an image of a product labelled with the *one-hot* vector $\boldsymbol{\ell} = (\ell_1, \ell_2, \ldots, \ell_c)$ such that, each $\ell_i \in \{0,1\}$ and $\sum_{i=1}^{c} \ell_i = 1$, where c is the number of distinct products (or classes). Assume, $I^{(1)}, I^{(2)}, \ldots, I^{(N)}$ be the N number of training data with the true label vectors $\boldsymbol{\ell}^{(1)}, \boldsymbol{\ell}^{(2)}, \ldots, \boldsymbol{\ell}^{(N)}$ for which RC-Net yields the reconstructed images $I'^{(1)}, I'^{(2)}, \ldots, I'^{(N)}$ and predicted label vectors $\boldsymbol{\ell}'^{(1)}, \boldsymbol{\ell}'^{(2)}, \ldots, \boldsymbol{\ell}'^{(N)}$ respectively.

Let θ and θ' denote all the trainable parameters (i.e., weights and biases) of the encoder and decoder respectively, and θ'' represents the trainable parameters of the classifier of proposed RC-Net. Given this, our objective is to optimize the parameters θ, θ' and θ'' for minimizing the *average reconstruction loss* and the *average classification loss* simultaneously,

$$\theta^*, \theta'^*, \theta''^* = \underset{\theta,\theta',\theta''}{\arg\min} \frac{1}{2N} \sum_{j=1}^{N} \times \Big[\mathscr{L}_{rl}\big(I^{(j)}, I'^{(j)}; \theta, \theta'\big) + \mathscr{L}_{cl}\big(\ell^{(j)}, \ell'^{(j)}; \theta''\big) \Big], \tag{1}$$

where $\mathscr{L}_{rl}(\cdot,\cdot;\theta,\theta')$ and $\mathscr{L}_{cl}(\cdot,\cdot;\theta'')$ are the *reconstruction loss* and *classification loss* respectively. In the proposed RC-Net architecture, *reconstruction loss* $\mathscr{L}_{rl}(\cdot,\cdot;\theta,\theta')$ is the conventional *binary cross-entropy loss* [32,33] and the *classification loss* $\mathscr{L}_{cl}(\cdot,\cdot;\theta'')$ is the *cross-entropy loss* [32,33].

The complete architecture of the proposed RC-Net is shown in Fig. 4. The configurations of various layers of the proposed RC-Net are further tabulated in Table 1, where conv3-64 represents 3×3 padded convolution operation performed 64 times, max-pool2 denotes 2×2 max pooling (with stride 2), max-unpool2 indicates 2×2 max unpooling (with stride 2), and fc-4096 represents fully connected layer with 4096 nodes. Similar interpretation is applicable for rest of the notations. After each conv layer, there exists batch normalization and ReLU layers. The classifier includes batch normalization, ReLU and deterministic dropout [31] layers after fc-4096. c is the number of classes.

However, initialization of the weights is important for any deep neural network with multiple numbers of layers, operations and paths. The weights of our network are strategically initialized with the weights of a pre-trained network (which is trained using the visual information of more than 14 million training images from 1000 classes of ImageNet dataset [34]) as follows.

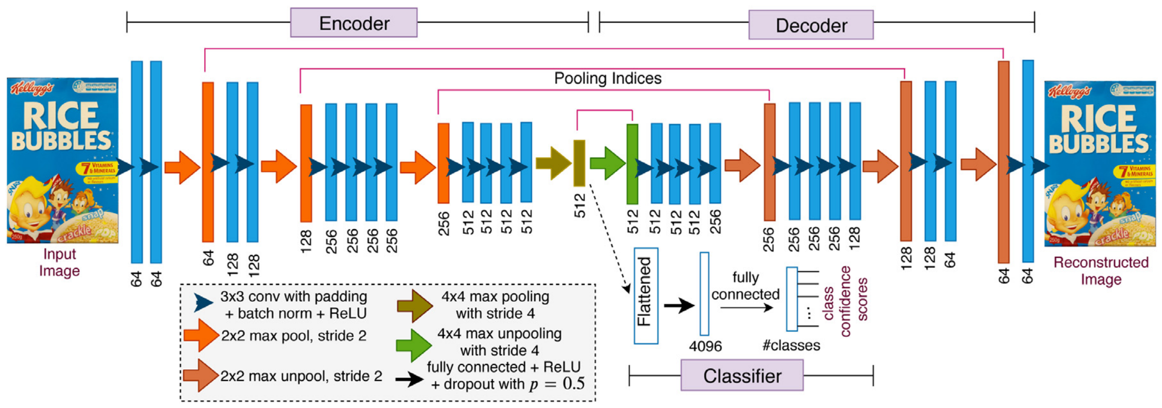

Fig. 4. Architecture of the proposed RC-Net. Colored bars denote the feature maps. The number of feature maps is given just below the bar.

Table 1
Various layers of RC-Net are shown in sequence. For details, refer to the description of Table 1 in Section 3.1.

Encoder→	conv3-64 conv3-64	max-pool2	conv3-128 conv3-128	max-pool2	conv3-256 conv3-256 conv3-256 conv3-256	max-pool2	conv3-512 conv3-512 conv3-512 conv3-512	max-pool4
Decoder→	max-unpool4	conv3-512 conv3-512 conv3-512 conv3-512	max-unpool2	conv3-256 conv3-256 conv3-256 conv3-256	max-unpool2	conv3-128 conv3-128	max-unpool2	conv3-64 conv3-64
Classifier→	fc-4096	fc-*c*						

The sequence of the convolution layers of encoder (see Fig. 4) is identical with the sequence of first twelve convolution layers of the VGG-19 network [35]. So, the weights of the convolution layers of the encoder are initialized with the weights of first twelve convolution layers of the pre-trained pytorch [36] implementation of VGG-19. In case of decoder, the initial weights of eight convolution layers, which exactly adhere with that of VGG-19, are initialized with the weights of pre-trained VGG-19. The adherence between the convolution layers of RC-Net and VGG-19 are $1 \leftarrow 12$, $2 \leftarrow 11$, $3 \leftarrow 10$, $5 \leftarrow 8$, $6 \leftarrow 7$, $7 \leftarrow 6$, $9 \leftarrow 4$, and $11 \leftarrow 2$, where $d \leftarrow d'$ represents that dth convolution layer of RC-Net is identical with d'th convolution layer of VGG-19. The initial weights of the remaining layers of decoder are set following [37].

The ability of simultaneous reconstruction and classification of products makes the RC-Net different from benchmark CNNs like VGG [35]. RC-Net is a combination of convolutional autoencoder and CNN classifier. The convolutional autoencoder tries to optimize the *reconstruction loss* while the classifier wants to optimize the *classification loss*.

In RC-Net, the addition of *classification loss* (or supervised loss) to the *reconstruction loss* (or unsupervised loss) makes the convolutional autoencoder capable of learning the underlying pattern of an object with the task specific class information of the object. Conversely, the addition of *reconstruction loss* to the *classification loss* forces the classifier to learn class discriminatory information along with the underlying pattern of an object. This way, the reconstruction and classification joint loss enforces RC-Net to balance both extraction of underlying structure and inference of correct prediction of an object. In other words, *reconstruction loss* regularizes the *classification loss* for the classification task.

As discussed in Section 1, RC-Net is an SCAE. The theoretical justification for improvement of classification performance of SAE over its corresponding fully connected neural network is given in Le et al. [9]. Theorem 1 of [9] provides the generalization bound for SAE and the bound guarantees the stability of the performance of SAE. The addition of *reconstruction loss* with the *classification loss* of a fully connected neural network ensures the generalizability and prevents over-fitting of the SAE network. Since SCAE is the convolutional version of SAE, Theorem 1 of [9] is also valid for the SCAE, which for this paper is RC-Net. Hence, RC-Net is expected to prevent the over-fitting of the network. Moreover, the improved generalizability of the RC-Net is experimentally validated with the ablation study in Section 4.1. Next we present the part-level classification module of our solution.

3.2. Part-level classification

The proposed part-level classification model (a) first generates the part proposals followed by (b) the selection of discriminatory parts and (c) classification of the sequence of those parts using conv-LSTM [10] as illustrated in Fig. 3(b).

(a) Generating part proposals: We introduce an unsupervised keypoint based approach for generating part proposals. We use Binary Robust Invariant Scalable Keypoints (BRISK) [38] to find keypoints on the product image *I*. A brief description of BRISK is given in Appendix A. We observe that these keypoints are the most important locations on the product image which require serious attention to derive the important parts or components of the product. In a local neighborhood of the keypoint, the change in intensity is higher than other regions which do not include keypoints as demonstrated in Fig. 5. In theory, local maxima of gradient of image and its scale-space versions are precursor to localization of keypoints [38]. Naturally, the change in intensity should be higher in and around a keypoint compared to a homogeneous region. This important observation drives us to extract the part proposals around these keypoints.

Assume that BRISK operator finds out η number of keypoints on the image *I*. We extract a patch of height h and width w from *I* centered at each qth keypoint. In our implementation, only square patches (i.e., $h = w$) are considered. In our implementation, the best result is obtained for $h = w = 25$ as detailed in Appendix B. The bounding box defined by the tuple (x_q, y_q, h_q, w_q) extracts the

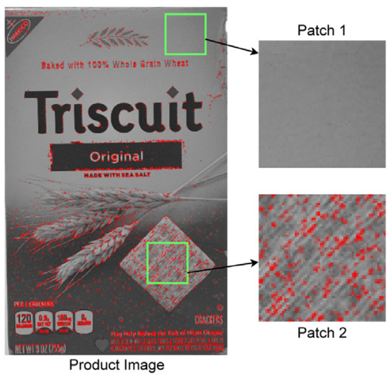

Fig. 5. Patch 1 and Patch 2 are two arbitrary regions of the given product image. Intensity distribution within Patch 1 is almost homogeneous. Naturally, Patch 1 does not include any keypoint. However, there is within-patch intensity variation in case of Patch 2. As a result, Patch 2 includes a number of keypoints (marked in red). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

patch (i.e., sub-image) $\mathscr{P}_q$ from I, where (x_q, y_q), h_q, and w_q be the spatial location of the top left corner, height and width of the qth patch respectively, where $q = 1, 2, \ldots, \eta$. The patches $\{\mathscr{P}_q\}$ are considered as part proposals.

(b) Determining discriminatory parts: Given patches or part proposals $\{\mathscr{P}_q\}$, $q = 1, 2, \ldots, \eta$, selection of discriminatory parts consists of two steps, extraction of features from the part proposals followed by selection of winner proposals from the groups of part proposals. These are detailed next.

Extracting features from the part proposals The layers at the end of a CNN contain more discriminatory information compared to the layers at the beginning [2]. This is an useful clue for fine-grained classification of objects.

We extract features for each qth patch from the last convolution layer of encoder in the RC-Net. First of all, the patch is resized to the size of receptive field of the last convolution layer. But the size of the receptive field of the last layer of encoder in RC-Net is much larger. For example, the size of receptive field is 160×160 for 224×224 input product image. Assume, the size of a patch extracted from the 224×224 input product image is 25×25. Then resizing 25×25 patch into the size of receptive field 160×160 destroys the spatial relationship between the neighboring pixels (or local features). This spatial relationship essentially indicates the product's part-level cue. In order to avoid this problem, we increase the size of input product image during patch extraction (see *Part-level Classification* block of Fig. 3(b)) so that the size of the patch becomes similar to that of the receptive field. Note that we obtain the best performance when the input image is zoomed three times. This way the discriminating power of the layers at the end is retained in our implementation.

Next the resized patch is forward propagated through the convolution layers of encoder in the RC-Net to derive the features. Assume, for any patch $\mathscr{P}$, we obtain a latent representation (or features) $\mathrm{x}_{e \times f \times g}$ from the last convolution layer of encoder in the RC-Net, where g is the number of 2D convolution feature maps, each with e rows and f columns. In other words, x contains ef numbers of g-dimensional vectors $\mathbf{v}_t$, $t = 1, 2, \ldots, ef$.

As discussed in Section 2, bilinear (second order) pooling [11,12] captures the local pairwise feature interactions in a translation invariant manner. The local pairwise feature interactions will certainly encapsulate much more discriminatory information about the objects than capturing same information without the pairwise features. The efficacy of bilinear pooling is evident in our ablation study which is carried out in Section 4.1. The bilinear pooling on x results in A:

$$\mathrm{A} = \frac{1}{ef}\left(\sum_{t=1}^{ef} \mathbf{v}_t \mathbf{v}_t^T\right) + \mathrm{diag}(\epsilon, \epsilon, \ldots, \epsilon), \tag{2}$$

where A is a matrix of size $g \times g$, $\mathrm{diag}(\epsilon, \epsilon, \ldots, \epsilon)$ is a $g \times g$ diagonal matrix and ϵ is a small positive value. The matrix A is now flattened to form a g^2-dimensional vector $\mathbf{A} = (a_1, a_2, \ldots, a_{g^2})$. Inspired by [39], we then calculate the signed square-root of each element a_i, $i = 1, 2, \ldots, g^2$ of $\mathbf{A}$ as $sign(a_i)\sqrt{|a_i|}$, where $sign(a_i)$ and $|a_i|$ denote the sign and absolute value of a_i respectively. Consequently, the vector $\mathbf{A}$ is normalized into a unit vector by dividing $\mathbf{A}$ with its L2 norm $||\mathbf{A}||_2$ i.e. $\frac{\mathbf{A}}{||\mathbf{A}||_2}$. This normalized $\mathbf{A}_q$ is now the feature vector for the qth patch $\mathscr{P}_q$. The procedure for selecting winner proposal is explained next.

Selection of winner proposal Assume β number of discriminatory parts are required to describe the product (ablation study in Section 4.1 suggests $\beta = 8$). First η number of part proposals are spatially clustered into β groups. The spatial clustering of part proposals essentially refers to the clustering of co-ordinates of the keypoints on the product. We implement k-means clustering algorithm [40] to obtain the β group of keypoints (or part proposals). Consequently, one proposal from each group is selected for representing the parts as follows.

In each group, a proposal, which is visually similar with maximum number of proposals in the group, is selected as the potential representative of a part. The visual similarity between the proposals $\mathscr{P}_{q_1}$ and $\mathscr{P}_{q_2}$ is measured by calculating the cosine similarity between their features $\mathbf{A}_{q_1}$ and $\mathbf{A}_{q_2}$.

Assume there are τ proposals in the bth group and $\mathrm{S} = [S_{q_1 q_2}]$ is the $\tau \times \tau$ similarity matrix, where $S_{q_1 q_2}$ denotes the cosine similarity value between the proposals $\mathscr{P}_{q_1}$ and $\mathscr{P}_{q_2}$. S is a symmetric matrix. Then sum of all the elements in the q_1th row (or q_1th column) of S represents the overall similarity between the proposal $\mathscr{P}_{q_1}$ and all other proposals in the group. The sum of elements in the q_1th row (or column) is referred to as the *similarity index* of the proposal $\mathscr{P}_{q_1}$. Let us assume that the highest *similarity index* is obtained for the q_1th row over all the rows in S. Hence, the proposal $\mathscr{P}_{q_1}$ is visually most similar with the other proposals in the bth group. Therefore, $\mathscr{P}_{q_1}$ is selected as a discriminatory part of the product.

Note that the part proposals may be overlapping. However, the actual parts of a product can never be overlapping as one part always exclusively denotes a specific important region of the product. We select only one part proposal (or patch) from each group (or cluster) of part proposals. In our implementation, there are 8 (i.e. the value of β) selected proposals. They hardly overlap. Even if they overlap, it does not affect the classification accuracy at all.

Finally, we obtain β number of discriminatory parts and form a sequence, $\mathcal{P} = [\mathscr{P}_1^*, \mathscr{P}_2^*, \ldots, \mathscr{P}_\beta^*]$ ordered by the location of the top-left corner of each patch within the product image I. The proposed conv-LSTM network for encoding part-level cues of the products designed with this sequence $\mathcal{P}$ is discussed next.

(c) Classification of sequence of parts using conv-LSTM network: We present a stacked conv-LSTM [10] model for encoding the part-level features (see Fig. 3(b)) and boosting the overall performance of fine-grained classification of the products. The proposed conv-LSTM network essentially works as a classifier. The block diagram of the proposed conv-LSTM network is shown in Fig. 6. A major benefit of using conv-LSTM over LSTM [41] is that the image can be directly fed to conv-LSTM without determining the features.

From the previous step, we obtain the sequence of discriminatory parts $\mathcal{P} = \{\mathscr{P}_1^*, \mathscr{P}_2^*, \ldots, \mathscr{P}_\beta^*\}$. In the sequence $\mathcal{P}$, the product image I is also included as the last member of the sequence to relate the parts with the product image i.e., $\mathscr{P}_{\beta+1}^* = I$. Thus the updated

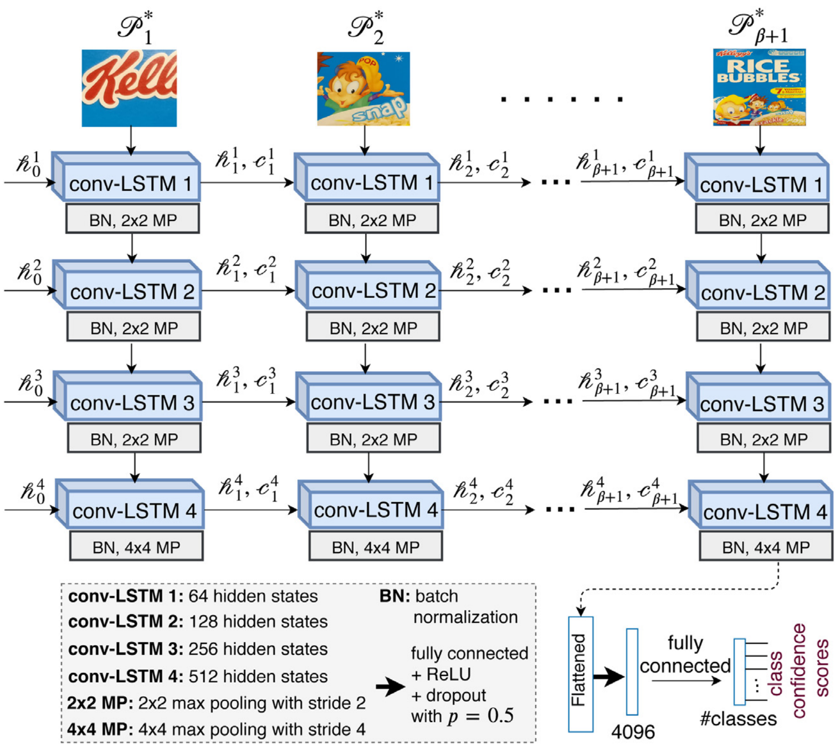
conv-LSTM 1
conv-LSTM 2
conv-LSTM 3
conv-LSTM 4
BN, 2x2 MP
BN, 4x4 MP
conv-LSTM 1: 64 hidden states
conv-LSTM 2: 128 hidden states
conv-LSTM 3: 256 hidden states
conv-LSTM 4: 512 hidden states
2x2 MP: 2x2 max pooling with stride 2
4x4 MP: 4x4 max pooling with stride 4
BN: batch normalization
fully connected + ReLU + dropout with p = 0.5
Flattened
4096
fully connected
#classes
class confidence scores

Fig. 6. Architecture of proposed conv-LSTM network, h_i^j and c_i^j are the hidden states and outputs of jth conv-LSTM unit at ith time step.

sequence is $\mathcal{P} = \{\mathscr{P}_1^*, \mathscr{P}_2^*, \mathscr{P}_3^*, \ldots, \mathscr{P}_\beta^*, \mathscr{P}_{\beta+1}^*\}$ which is the input to the proposed conv-LSTM model.

As illustrated in Fig. 6, we design a four-layered stacked conv-LSTM network for encoding the part-level information. In the hidden states of the proposed conv-LSTM, we apply 64, 128, 256, and 512 convolution filters in the conv-LSTM layers (or units) from start to end respectively. Notably, 3×3 padded convolution operation is applied in the proposed conv-LSTM network. The hidden state of the first conv-LSTM unit is essentially the output from the first conv-LSTM unit. The output from the first conv-LSTM unit is then forwarded through a batch normalization and a max pooling layer. Subsequently, the resultant output is set as input to the next conv-LSTM layer. Similar process is iterated for rest of the layers.

The length of the sequence $\mathcal{P}$ (which is $\beta + 1$) is the number of time steps to unroll the conv-LSTM units. The output of our conv-LSTM network at the last time step defines a feature vector $\mathbf{z}$ which is further connected to a fc-layer with 4096 neurons (referred to as hidden fc-layer). Consequently, conv-LSTM network performs batch normalization followed by ReLU and deterministic dropout [31] operations after the hidden fc-layer successively. However, again these 4096 neurons of the hidden fc-layer are linked to another fc-layer with c number of neurons equivalent to the number of classes (referred to as output layer). The initial weights of the entire network are set following [37].

For the product image I, let $\tilde{\boldsymbol{\ell}}$ be the predicted vector containing class confidence scores using our conv-LSTM network. Assume $\tilde{\theta}$ denotes the weights and biases of convolution and fc-layers of our conv-LSTM network. Now our intention is to train the network with N training samples to optimize the parameter $\tilde{\theta}$ for minimizing the *average classification loss* i.e.,

$$\tilde{\theta}^* = \arg\min_{\tilde{\theta}} \frac{1}{N} \sum_{j=1}^{N} \mathscr{L}_{lcl}\left(\boldsymbol{\ell}^{(j)}, \tilde{\boldsymbol{\ell}}^{(j)}; \tilde{\theta}\right), \tag{3}$$

where $\boldsymbol{\ell}$ is the true label of I and $\mathscr{L}_{lcl}(.,.;\tilde{\theta})$ is the traditional *cross-entropy loss* [32]. The complete pipeline of classification is provided next.

3.3. Complete classification model

Both the modules (object-level and part-level) of the proposed solution perform the classification of fine-grained products (see Fig. 3(a)). The feature reconstruction mechanism of the object-level classifier RC-Net improves the classification performance over any standard CNN models like VGG [35]. Similarly, encoding of part-level features using conv-LSTM helps in accurately classifying fine-grained products. The classification potentiality of RC-Net is much higher than conv-LSTM. But the proposed conv-LSTM network significantly boosts up the classification performance of RC-Net. Therefore we combine them together for accurate classification of fine-grained products as demonstrated in Fig. 3.

The training of object-level and part-level networks are performed separately. Once training is complete, a product image I (having label vector $\boldsymbol{\ell}$) is fed to both the object-level and part-level classification modules to obtain the predicted vectors $\boldsymbol{\ell}'$ (see Section 3.1) and $\tilde{\boldsymbol{\ell}}$ (see Section 3.2) containing class confidence scores. The final classification scores $\boldsymbol{\ell}_F$ for image I is obtained as

$$\boldsymbol{\ell}_F = \boldsymbol{\ell}' + \gamma \tilde{\boldsymbol{\ell}}, \tag{4}$$

where $\gamma \in [0, 1]$ is the weight of the part-level score to improve the object-level classification score for recognition of fine-grained product. Thus, the predicted class confidence scores in $\boldsymbol{\ell}_F$ carry both object-level and part-level cues of the fine-grained products. The class probabilities for the image I is then obtained by applying soft-max as

$$\ell'_{Fi} = \frac{e^{\ell_{Fi}}}{\sum_{a=1}^{c} e^{\ell_{Fa}}}, \quad \forall \ell_{Fi} \in \boldsymbol{\ell}_F, i = 1, 2, \ldots, c. \tag{5}$$

The class with highest probability is the predicted label for the image I. Next we present experiments and analysis.

4. Experiments

Datasets We capture 457 rack images of 352 products in various supermarkets to create **In-house** dataset. This collection of rack images display many fine-grained products. We capture one template image for each product in a controlled scenario. The products belong to six larger classes of products such as *breakfast cereals* (BC) (72 products and 151 racks), *deodorant* (DEO) (55 products and 100 racks), *lip care* (LC) (20 products and 80 racks), *oral care* (OC) (51 products and 30 racks), *personal wash* (PW) (82 products and 36 racks) and *mixed* (MIX) (72 products and 60 racks). For creating ground truth, all the racks are manually annotated by labeling the products using tight rectangular bounding boxes. In addition, we have used following three benchmark datasets.

The **Grocery Products (GP)** [22] dataset consists of 680 rack images that exhibit 3235 products. The products are collected from 27 categories. The rack images display 6 to 30 number of products. The product templates, which are imaged in a controlled environment, are downloaded from the web. While the rack images are clicked in the supermarket environment from various viewing angles with non-identical magnification levels and lighting conditions. The dataset also presents fine-grained products which are displayed on the rack images. The dataset also includes ground truth for all the rack images indicating similar group of products with tight bounding boxes.

The **WebMarket (WM)** [18] dataset includes 3153 rack images captured from 18 shelves in a supermarket. The dataset contains template images of 100 products which are present only in 402 (out of 3153) rack images. There exists three instances for each of the products in the dataset. This dataset also provides a few fine-grained products. The rack images are clicked when the products are on the shelf. On the contrary, the products are imaged off the shelf. So scale, pose and illumination are not identical for rack and product images. The annotations, for the bounding boxes of the products in the rack images, are not bundled with the dataset. We have manually annotated the racks for evaluation of the methods.

The **GroZi (GZ)** [8] dataset provides 29 videos of racks, which display 120 products captured in a supermarket. The frames after each 5 consecutive frames (e.g., 1st, 6th, 11th, ... frames) are extracted from the videos and annotation for these frames are included in the dataset. The dataset distributes 2 to 14 template images for each product. In this dataset, each product is displayed in 14 to 814 rack images. The dataset also provides (separately) the products cropped from the racks using the annotations for the bounding boxes of the products in the rack images. Most of the products in rack differ from the templates as shown in Fig. 2. This poses additional challenge for the dataset.

4.1. Implementation details, results and analysis

For our product classification task, training data consists of the product templates and the augmented samples from these templates. On the other hand, the cropped (or extracted) products from rack images are the test data. The size of the input image for the object-level classifier is 224×224 whereas the same for part-level classifier is 672×672 after zooming the original input image by a scale of 3. For implementing part-level classifier, we experimentally set the number of discriminatory parts, $\beta = 8$ as detailed in the paragraph for *Ablation Studies* of this section. In our classification scheme, the weight for part-level classification score is experimentally set as $\gamma = 0.6$. Next we describe the normalization of data followed by data augmentation and learning approach of the networks.

Data normalization The training/test images are first transformed into a fixed-size 224×224 images without altering the aspect ratio. We first resize a $\tilde{w} \times \tilde{h}$ image to $224 \times 224\frac{\tilde{h}}{\tilde{w}}$ if $\tilde{w} > \tilde{h}$, else $224\frac{\tilde{w}}{\tilde{h}} \times 224$. We then superimpose the resized image in a white 224×224 frame such a way that the center of the resized image must coincide with the center of the white frame. Next the transformed product image is normalized by dividing all pixel values of the image by the highest pixel value of the image. For part-level classification, we resize the normalized image to the resolution of 672×672.

Data augmentation For the problem under discussion, only one (or very few) template image(s) is (are) available for each product. But training of deep learning model requires large amount of training data per product class. So we synthetically augment $\sim 10^4$ training samples for each product applying various photometric and geometric transformations. We use the python libraries: keras,[1] augmentor,[2] and imgaug,[3] for synthesizing the training samples. Keeping the supermarket like scenario in our mind, we apply the photometric transformations like blurring (Gaussian, mean and median), noise (salt & pepper and Gaussian) addition, random brightness adjustment, and random contrast adjustment. Subsequently, we apply various geometric transformations like distortion, shearing, translation and rotation on the synthesized images using photometric transformations. We utilize these augmented samples for training the object-level and part-level classifiers of the proposed solution.

Learning approach The augmented product images (including the templates) and their class labels are used to train the network with the pytorch implementation of mini-batch stochastic gradient descent (SGD) [43] optimization technique. The hyper-parameters of the SGD for our networks are set experimentally following training algorithms of deep learning based methods [9,31,35,42]. In our experiments, the mini-batch size is set to 2^4. SGD initially starts with a momentum of 0.99, learning rate of 0.001, and weight decay of 10^{-6}. Further the learning rate is updated after each 10 epochs dividing it by 10. Both object-level and part-level classifiers are trained for at most 60 epochs.

Competing methods The pytorch implementation of pre-trained deep learning networks, VGG [35] and ResNet [42] are retrained with our training data for the product classification task following the similar protocol like ours as discussed in the previous paragraph. These networks are also trained for at most 60 epochs. Rest of the competing approaches discussed in Section 2 are reproduced following the respective papers.

Assume we have N_T number of test images out of which N'_T are correctly classified by a specific method. Notably, the test samples are the cropped products from racks. The accuracy of the method is defined as: $\frac{N'_T}{N_T} \times 100$ (%). Following the standard practice in reporting the performance of deep learning models [31,44], we determine five-fold classification accuracy for each dataset and report the mean and standard deviation of these five accuracy values for each dataset. This is done in order to consider (five different) random initialization of the deep learning networks. In other words, all the experiments are repeated five times due to five different random initialization of the training data. The average performance along with the extent of performance variation (*mean* $\pm$ *standard deviation*) of these five trials are shown in Table 2.

The fine-grained product classification performance on various datasets are tabulated in Table 2. It is seen that, our proposed classification scheme, which captures the object and part-level cues, stands out as winner (in most of the cases) among all the methods highlighted in Table 2. We find that the proposed **RC-Net** (object-level classifier) and **Complete** classification scheme (object and part-level classifier) perform equally well when there are a few

[1] https://github.com/keras-team/keras/, accessed as on April, 2021.

[2] https://github.com/mdbloice/Augmentor/, accessed as on April, 2021.

[3] https://github.com/aleju/imgaug/, accessed as on April, 2021.

Table 2
Product classification accuracy (in %, *mean* ± *standard deviation*) on various datasets. VGG, VGG + DD and ResNet refer to VGG-19, VGG-19 with deterministic dropout (DD) [31] and ResNet-101 respectively. The results for DD [31] are provided from respective paper, where only *mean* accuracies are presented, thus we set *standard deviation* to 0.00.

Methods	Categories of in-house dataset						Benchmark datasets		
	BC	DEO	LC	OC	PW	MIX	GP	WM	GZ
VGG [35]	86.03 ± 1.10	88.75 ± 0.56	85.19 ± 0.42	72.01 ± 0.61	78.12 ± 0.78	63.29 ± 0.46	66.05 ± 0.76	64.48 ± 0.70	36.17 ± 0.62
TLA [4]	86.82 ± 0.28	88.12 ± 0.69	85.40 ± 0.89	72.69 ± 0.59	78.46 ± 0.41	64.00 ± 0.37	70.32 ± 0.52	68.21 ± 0.52	39.92 ± 0.69
PSCNN [2]	88.83 ± 0.67	89.75 ± 0.69	88.68 ± 0.54	78.71 ± 0.85	81.89 ± 1.02	70.44 ± 0.45	72.08 ± 0.66	68.12 ± 0.95	40.63 ± 0.57
DRCN [13]	89.15 ± 0.50	89.91 ± 0.43	90.03 ± 0.91	80.43 ± 0.39	83.04 ± 0.77	70.60 ± 0.52	73.36 ± 0.83	70.00 ± 0.48	41.60 ± 0.65
ResNet [42]	90.09 ± 1.16	90.42 ± 0.72	88.59 ± 1.13	73.73 ± 0.66	80.36 ± 0.68	68.55 ± 0.56	69.50 ± 1.10	67.25 ± 0.92	38.27 ± 0.59
OPA [5]	90.13 ± 0.53	89.91 ± 0.51	89.91 ± 0.76	79.58 ± 0.70	83.85 ± 0.73	73.64 ± 0.46	71.77 ± 0.47	69.52 ± 0.88	41.57 ± 0.44
IBP [12]	89.89 ± 0.85	89.88 ± 1.39	89.09 ± 0.66	78.97 ± 0.45	82.63 ± 1.05	71.48 ± 0.83	72.87 ± 0.60	69.06 ± 0.75	41.23 ± 0.30
WSCP [3]	$\mathbf{92.55 \pm 0.44}$	89.84 ± 0.52	89.18 ± 0.27	86.22 ± 0.35	84.05 ± 0.32	73.87 ± 0.47	77.94 ± 0.32	70.60 ± 0.29	44.10 ± 0.27
DD [31]	–	–	–	–	–	–	$\mathbf{81.62 \pm 0.00}$	–	45.15 ± 0.00
VGG + DD	89.21 ± 0.92	90.33 ± 0.39	87.97 ± 0.71	72.13 ± 1.09	79.24 ± 0.84	66.81 ± 0.70	70.81 ± 1.05	67.00 ± 0.59	38.56 ± 0.57
RC-Net	90.19 ± 0.65	90.70 ± 1.10	$\mathbf{91.93 \pm 0.62}$	87.73 ± 0.71	89.51 ± 0.73	79.08 ± 0.99	75.95 ± 0.77	72.40 ± 0.61	46.36 ± 0.53
Complete	92.47 ± 0.32	$\mathbf{90.77 \pm 0.67}$	$\mathbf{91.93 \pm 0.31}$	$\mathbf{90.79 \pm 0.35}$	$\mathbf{92.62 \pm 0.54}$	$\mathbf{84.01 \pm 0.33}$	81.20 ± 0.61	$\mathbf{75.71 \pm 0.18}$	$\mathbf{48.12 \pm 0.58}$

Table 3
Analysis of performance of the proposed approach on different datasets.

Dataset		Characteristics of training and test images
In-house		To generate training images, the products arranged on the racks of a supermarket, are taken out from the racks and imaged in isolation in the store environment. Naturally, the product templates used in training and product test images cropped from the rack are identical, captured under same lighting condition using the same camera. As a result, there is minimal intensity variation between training and test images and the classification performance for different categories of In-house dataset are roughly between 84% to 92%.
Benchmark	GP [22]	The product templates for training are high resolution images downloaded from the web. Therefore, the intensity variations in the product templates and the cropped product images from the supermarket rack are significant. The training and test images have different resolutions and are possibly captured using different cameras. Further, the dataset includes around 3000+ product classes. As a result, the performance of our method drops to around 81%.
	WM [18]	The products are taken out from the racks and made to lie on the floor face up to capture training images. The imaging system from the top introduces tilt and skew. Naturally, the product templates are not ideally captured. The illumination difference between the rack images and product template images is visible. We have achieved around 75% classification accuracy on this dataset.
	GZ [8]	The resolution of both product templates and rack images are poor. The product template images are downloaded from the web and thus, most of the cases, they are not entirely identical to the products displayed on the rack images. Therefore, the test data largely differ from the training data. As a result, the accuracy on this dataset is nearly 48%.

number of fine-grained products in a dataset like LC and DEO of In-house dataset.

Our proposed approach also outperforms other methods for benchmark datasets except for GP dataset that includes huge number of product classes (around 3 K). Still, our **Complete** scheme (integrating part-level classifier with RC-Net) performs similar to state-of-the-art methods on GP dataset. This supports the potential of our **Complete** scheme that achieves similar performance like state-of-the-art approaches when the number of classes is high. Overall, our **Complete** classification model significantly enhances the fine-grained classification performance. However, Table 2 shows that the classification accuracy on In-house dataset is higher compared to the accuracy on benchmark datasets for all the methods. In Table 3, we explain why the performance on In-house datasets are impressive compared to the performance on benchmark datasets. In Fig. 7, example classification results due to various methods for some fine-grained products are demonstrated. The superiority of our proposed scheme is also evident in these example results.

Ablation study The ablation study is carried out on both object-level and part-level classification modules of the proposed scheme. We have performed ablation studies on (a) the influence of decoder in RC-Net for improving performance, (b) the importance of part-level classifier along with object-level classifier, (c) optimal number of discriminatory parts in building part-level classifier, and (d) effectiveness of bilinear pooling.

(a) Influence of decoder in RC-Net for improving performance: In our object-level classifier RC-Net, the contribution of reconstruction segment of the network is examined for classification of the products. Note that the reconstruction segment is the decoder of RC-Net (see Fig. 4). The entire RC-Net is referred to as **Encoder + Decoder + Classifier** while the network without reconstruction segment is symbolized as **Encoder + Classifier**. Fig. 8 presents the performance of these two networks on both In-house and benchmark datasets. It can be seen that the improvement in classification performance is 8% to 18% when the reconstruction segment is integrated with the **Encoder + Classifier** network.

(b) Importance of part-level classifier along with object-level classifier: From Table 2, it can be clearly seen that the classification performance drops 1% to 5% for all the datasets except DEO and LC, if we do not integrate part-level module with our object-level module. In Table 2, **RC-Net** refers to object-level classifier while **Complete** denotes object-level with part-level classifier. Therefore, part-level classifier indeed boosts up the performance of fine-grained classification.

(c) Optimal number of discriminatory parts in building part-level classifier: Our study on optimal number of discriminatory parts in building part-level classifier is performed on PW category of In-house dataset. The experiments are carried out for $\beta = 2, 4, 6, 8, 10, 12, 16,$ and 18 and the corresponding classification accuracy are provided in Fig. 9 keeping other parameters unchanged. We can clearly see that the optimal performance is obtained for $\beta = 8$. Moreover, if we do not consider ordered sequence of the discriminatory parts, the performance drops from 92.62% (which is the highest accuracy obtained with $\beta = 8$) to 88.93%. Thus ordered sequence of the parts is important in our model.

(d) Effectiveness of bilinear pooling: Bilinear pooling shows good performance in representing fine-grained features of the objects

	True Class Label (or Product Label) ⟶	Lakme Apricot	Lakme Cherry	NutriGrain 290g	NutriGrain 500g	NutriGrain 805g	RiceBubbles 250g	RiceBubbles 705g
	Cropped Images from Rack (or Test Images) ⟶							
Methods	VGG [35]	×	✔	×	×	✔	×	✔
	TLA [4]	✔	✔	×	✔	×	×	✔
	PSCNN [2]	✔	✔	×	×	✔	✔	✔
	DRCN [13]	✔	×	✔	×	✔	×	✔
	ResNet [42]	✔	✔	✔	✔	×	×	✔
	OPA [5]	✔	✔	✔	×	✔	×	✔
	IBP [12]	✔	✔	×	✔	✔	✔	×
	WSCP [3]	✔	✔	×	✔	✔	×	✔
	DD [31]	✔	✔	✔	×	✔	✔	×
	VGG+DD [31]	×	✔	×	✔	✔	✔	×
	RC-Net	✔	✔	✔	×	✔	✔	✔
	Complete	✔	✔	✔	✔	✔	✔	✔

Fig. 7. Example classification results of a few fine-grained products for which the cropped images from rack (considered as test image) are shown along with the true class labels (see top of the products). Each of these products is classified using various methods. ✔ indicates that the test image is correctly classified as the true product label while × denotes the incorrect classification of test image.

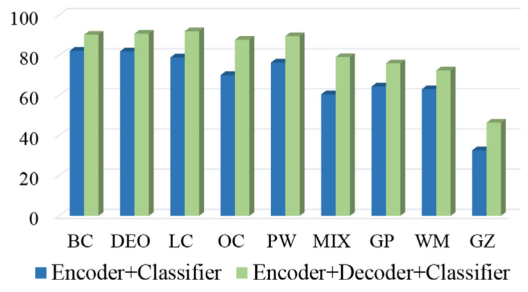

Fig. 8. Accuracy (in %) of RC-Net with (Encoder + Decoder + Classifier) and without (Encoder + Classifier) Decoder.

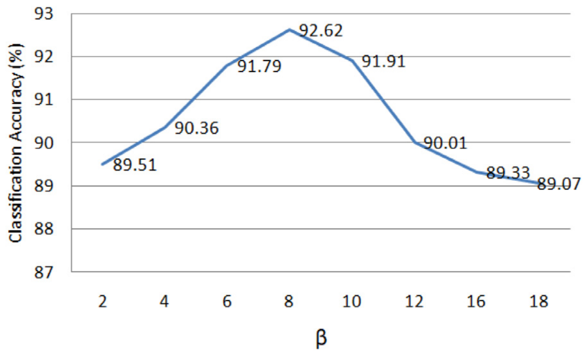

Fig. 9. Classification accuracy (%) for different values of β.

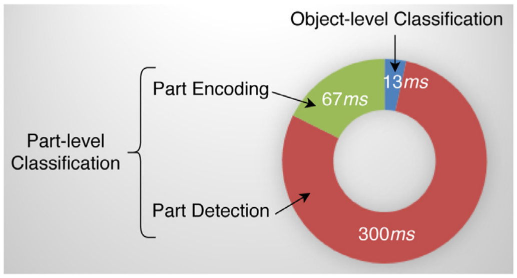

Fig. 10. Execution time per test image (in ms) of various modules.

[11,12]. Here we carry out an important ablation study for evaluating the effectiveness of bilinear pooling. Our experiments find that the performance improvement (in %) is 2.05, 0.02, 0.59, 1.79, 2.41, 1.88, 3.67, 1.92 and 0.81 on BC, DEO, LC, OC, PW, MIX, GP, WM, and GZ datasets respectively, if we use bilinear pooling in our part-level classifier. These results exhibit the significant contribution of bilinear pooling in our part-level classifier. Since bilinear pooling consumes insignificant processing time in GPU, this does not add any computational overhead to our part-level classifier.

Notes on test time We implement the proposed algorithm in python in a computing system with the specs as follows: 64 GB RAM, Intel Core i7-7700 K CPU @ 4.2 GHz $\times$ 8 and TITAN XP GPU. Fig. 10 illustrates the time (in milliseconds (ms)) consumed by different modules of the proposed classifier for processing a single test image. Our classification algorithm recognizes an image of a retail product in 380 ms. The object-level classifier consumes only 13 ms to process its input image of resolution 224×224. While the part-level classifier completes its execution in 367 ms, the part detection and part encoding sub-modules take 300 ms for processing 672×672 image and 67 ms for processing 224×224 images respectively. Note that part detection refers to the sub-routines *generating part proposals* and *determining discriminatory parts*, and part encoding refers to *conv-LSTM classification network* explained in Section 3.2. Therefore, the part-level classifier improves the accuracy at least 2% (see Table 2) at the cost of 367 ms. Next we present the efficacy of our fine-grained classifier when we embed this in R-CNN [17].

4.2. Proposed fine-grained classifier in R-CNN

The block diagram of R-CNN is demonstrated in Fig. 11. It has three primary modules: (i) generation, (ii) classification and (iii) non-maximal suppression of the region proposals. The proposed fine-grained classification module is built inside the classification step (indicated by blue dotted rectangle in Fig. 11) of R-CNN. Thus in this paper, we assess the performance of modified R-CNN with

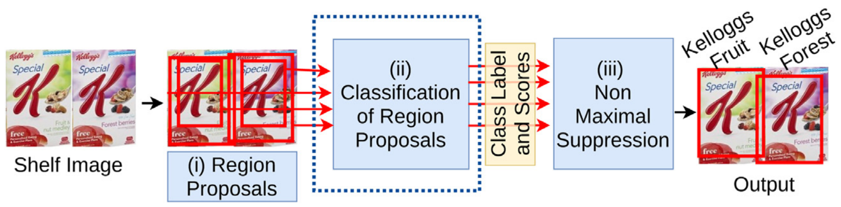

Fig. 11. Pipeline of the R-CNN algorithm for detecting fine-grained products on the rack. Dotted rectangle highlights our contribution i.e. the module of R-CNN where the classification of products/region proposals are performed using our proposed fine-grained classifier.

our proposed fine-grained classifier for detecting products sitting in the rack.

We implement the R-CNN [17] for detecting products in a rack as follows. We first run selective-window search [45] algorithm on a rack image to generate a number of proposals which are essentially the initial set of detected products. Each proposal is then fed to the fine-tuned pre-trained CNN classifier (here ResNet-101) to determine the class label and class confidence score. Finally, the products are detected by removing the overlapping proposals using greedy non-maximal suppression (greedy-NMS) [46] technique. **R-CNN** refers to the above implementation while **R-CNN-M** refers to the above implementation substituting ResNet-101 with the proposed fine-grained classifier.

We reproduce the competing methods in Merler et al. [8], Girshick et al. [17], Zhang et al. [18], Ray et al. [19], Marder et al. [20], Franco et al. [21], George and Floerkemeier [22]. The competing methods in Ray et al. [19], George and Floerkemeier [22] are referred to as U-PC and MLIC respectively. We implement the histogram of oriented gradients (HOG) and bag of words (BoW) based schemes from [20] and color histogram matching (CHM) based scheme from [8]. From [18], the best method S1 is reproduced. Both bag of words (BoW) and deep neural network (DNN) based methods from [21] are implemented and referred to as GBW and GDN respectively in our comparative study. The publicly available code of the method of [23], which we refer as SET, is downloaded from github (https://github.com/leokarlin/msmo_star_model).

A number of performance metrics for identification of retail products are listed in Santra and Mukherjee [1]. We calculate F_1 score for analyzing the detection performance. The standard performance metrics for generic object detection such as *precision*, *recall*, and *intersection-over-union* (IoU) measures are combined in calculating F_1 score in our evaluation. For product detection problem, the product on the rack is labelled using a (rectangular) bounding box associated with class label of the product. Therefore, we define TP (true positives), FN (false negatives), and FP (false positives) for each (labelled) product sitting in the rack. Assume a rack image R displays the product Q. If the center of any detection is recognized as Q and falls within Q (as defined in ground truth) in the rack, the TP count for the rack R is increased by 1. If the center of any detection is not recognized as Q but falls within Q in the rack, the FP count for the rack R is increased by 1. Moreover, the FP count for the rack R is increased by 1, if the center of a detection does not fall within any true product in R. The FN count of the rack R is increased by 1, if the center of any detection does not fall within Q in the rack R. Then the F_1 score for the rack R is defined as $\frac{2(recall \times precision)}{recall + precision}$, where $recall = \frac{\text{TP}}{\text{TP+FN}}$, $precision = \frac{\text{TP}}{\text{TP+FP}}$.

Table 4 tabulates the performance of the methods on both In-house and benchmark datasets. Due to correct classification of products, the improvement in performance of **R-CNN-M** over **R-CNN** is remarkable as shown in Table 4. Furthermore, this is evident from the table that **R-CNN-M** outperforms other methods by at least 3% except GDN [21] on GroZi dataset.

The superiority of the **R-CNN-M** with our proposed fine-grained classifier over vanilla **R-CNN** can be noticed in the example output in Fig. 12. In the first row of Fig. 12, it can be seen that **R-CNN** draws two bounding boxes where the products are not present i.e. **R-CNN** results in false positives. This problem is resolved with the **R-CNN-M** due to the accurate classification of region proposals using the proposed fine-grained classifier. Both the outputs for **R-CNN** and **R-CNN-M** in the 2nd row of Fig. 12 do not identify two products for which the ground truth is available (refer to as miss-detection). Our analysis finds that this is happened due to either of the following two reasons (a) the region proposal algorithm in **R-CNN** (or **R-CNN-M**) does not provide tight fitted bounding boxes (or region proposal) for these products (b) due to the inaccurate classification scores, the proposals for these products get suppressed by the proposals for the neighboring products. Moreover, **R-CNN** results five more detection, which are incorrect or false positives (see 1st column of 2nd row). These problems are resolved with the **R-CNN-M** (see 2nd column of 2nd row). Similarly, the remaining (third) row of Fig. 12 also establishes the eminence of **R-CNN-M** over **R-CNN** by minimizing the number of false positives and incorrect detection.

When we embed the proposed fine-grained classifier in R-CNN, our un-optimized GPU code classifies 20 region proposals in parallel in 367 ms. If we generate 500 proposals per rack using region proposal algorithm (in R-CNN), the classification time for a rack is approximately $\frac{500}{20} \times 367$ ms $= 9.17$ s. We can see that significant improvement in product detection performance (refer Table 4) can be obtained at the expense of ~ 9 s. Next section concludes the paper.

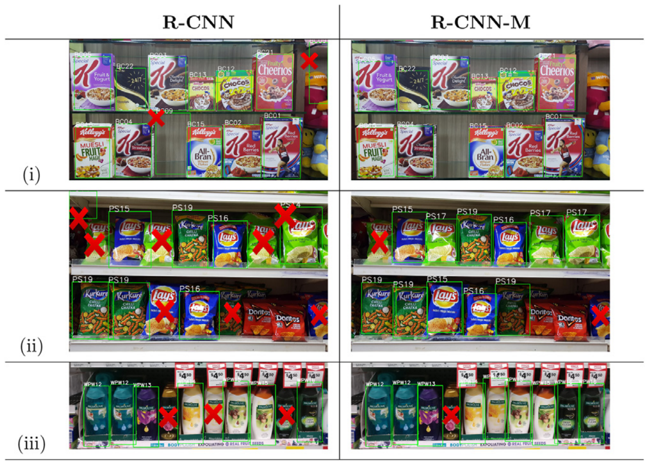

Fig. 12. Example output of **R-CNN** (left column) and **R-CNN-M** (right column) with our proposed fine-grained classifier. The incorrect or miss-detection are highlighted with the red cross mark. The products which are not detected by the algorithm but for which ground truths are available, are also highlighted using the red cross mark. Note that the ground truths are not available for second and third last products present in the lower shelf of the example image (ii). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Table 4
Product detection results (F_1 score in %) on In-house and benchmark datasets.

Methods	Categories of in-house dataset						Benchmark datasets		
	BC	DEO	LC	OC	PW	MIX	GP	WM	GZ
S1 [18]	41.01	45.21	47.87	48.08	54.91	49.01	58.39	49.19	31.71
CHM [8]	48.04	33.56	60.12	30.77	36.40	44.74	51.20	52.81	24.70
MLIC [22]	64.45	50.08	54.91	40.23	59.97	48.76	59.07	53.33	33.10
R-CNN [17]	82.04	**83.76**	87.99	79.72	88.05	73.16	78.99	72.01	40.91
HOG [20]	62.00	28.52	49.37	28.73	44.06	50.62	58.11	43.03	28.33
BoW [20]	65.05	45.10	70.72	53.31	71.23	59.91	59.91	55.15	26.83
GBW [21]	72.07	49.98	68.29	46.79	77.22	53.41	74.34	65.59	39.66
GDN [21]	82.12	55.49	82.55	51.32	87.64	61.98	73.09	71.13	**43.99**
SET [23]	83.61	83.95	88.36	81.77	88.16	74.22	79.05	72.13	43.78
U-PC [19]	84.77	52.59	86.29	55.65	81.15	65.49	76.20	67.79	40.10
R-CNN-M	**87.94**	**83.76**	**89.69**	**85.17**	**90.79**	**77.73**	**79.66**	**74.00**	**43.99**

5. Conclusions

We have proposed a two-level fine-grained classification scheme considering both part and object-level cues. Our study finds that the proposed object-level classifier RC-Net generalizes better while the part-level classifier significantly boosts the overall fine-grained classification performance. The overall product classification accuracy is in the range of 90% for In-house dataset while the accuracy is less for benchmark datasets due to large variations between training and test data. This indicates a significant scope of research in this particular application domain. We plan to boost fine-grained classification performance using local patch-based texture features as a guide to already developed conv-LSTM architecture.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. BRISK [38]

Scale-space based keypoint detection Points of interest are identified in the image and its various scaled versions. The scaled images are obtained by progressive half-sampling of the original image. A continuous saliency score is calculated at each scaled version of the image. A 2D quadratic function is approximated in the least-squares sense considering three consecutive layers of the scale-space stack of images. The objective is to detect local maxima of a combined score involving local intensity maxima and saliency score considering three consecutive layers of the scale-space stack of images. The locations of these local maxima are potential keypoints.

Keypoint description A sampling pattern consisting of points lying on appropriately scaled concentric circles, similar to a DAISY descriptor, is applied at the neighborhood of each potential keypoint. The characteristic direction of each keypoint is evaluated comparing the local image gradient direction between the keypoint and the sampled points defined in the preset concentric circles around the keypoint. Brightness comparisons of points as defined above generate bit vectors for a 512-bit long BRISK descriptor. Once generated, the BRISK keypoints can be matched efficiently thanks to the binary nature of the descriptor and dense description of the bit vectors due to the use of preset DAISY like pattern of the descriptor.

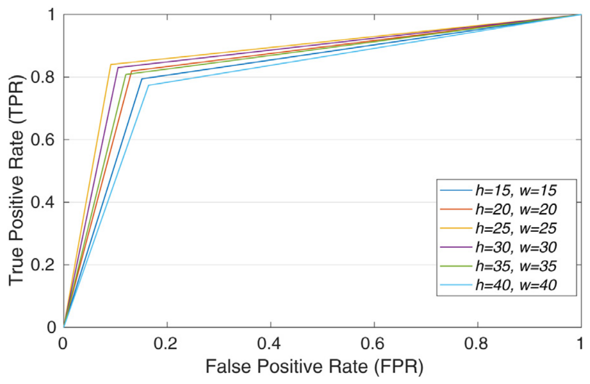

Fig. B.13. ROC curves plotting TPR against FPR for different h and w.

Appendix B. Choice of h and w in part-level classifier

As discussed in Section 3.2, in our implementation of part-level classification, we consider only square image patches (i.e. $h = w$) for the deep learning framework of our proposal following [35,42]. The experiments are carried out for various h (or w) such as $h = 15, 20, 25, 30, 35, 40$ on MIX category of In-house dataset (as this category includes mostly fine-grained products). The respective receiver operating characteristic (ROC) curves are drawn by plotting the true positive rate (TPR) against the false positive rate (FPR) in Fig. B.13 during training. The figure clearly depicts that the area under the curve (AUC) for the ROC curve with respect to h (or w) $= 25$ is higher than the AUCs for other ROC curves. Hence we have chosen $h = w = 25$ 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 Vis. Comput. 86 (2019) 45–63.
[2] S. Huang, Z. Xu, D. Tao, Y. Zhang, Part-stacked CNN for fine-grained visual categorization, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 1173–1182.
[3] W. Ge, X. Lin, Y. Yu, Weakly supervised complementary parts models for fine-grained image classification from the bottom up, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2019, pp. 3034–3043.
[4] T. Xiao, Y. Xu, K. Yang, J. Zhang, Y. Peng, Z. Zhang, The application of two-level attention models in deep convolutional neural network for fine-grained image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 842–850.
[5] Y. Peng, X. He, J. Zhao, Object-part attention model for fine-grained image classification, IEEE Trans. Image Process. 27 (3) (2017) 1487–1500.
[6] C.F. Flores, A. Gonzalez-Garcia, J. van de Weijer, B. Raducanu, Saliency for fine-grained object recognition in domains with scarce training data, Pattern Recognit. 94 (2019) 62–73.

[7] G.-S. Xie, X.-Y. Zhang, W. Yang, M. Xu, S. Yan, C.-L. Liu, LG-CNN: from local parts to global discrimination for fine-grained recognition, Pattern Recognit. 71 (2017) 118–131.
[8] 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.
[9] L. Le, A. Patterson, M. White, Supervised autoencoders: improving generalization performance with unsupervised regularizers, in: Advances in Neural Information Processing Systems, 2018, pp. 107–117.
[10] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, W.-c. Woo, Convolutional LSTM network: a machine learning approach for precipitation nowcasting, in: Advances in Neural Information Processing Systems, 2015, pp. 802–810.
[11] T.-Y. Lin, A. RoyChowdhury, S. Maji, Bilinear CNN models for fine-grained visual recognition, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 1449–1457.
[12] T.-Y. Lin, S. Maji, Improved bilinear pooling with CNNs, in: British Machine Vision Conference (BMVC), 2017.
[13] M. Ghifary, W.B. Kleijn, M. Zhang, D. Balduzzi, W. Li, Deep reconstruction-classification networks for unsupervised domain adaptation, in: European Conference on Computer Vision, Springer, 2016, pp. 597–613.
[14] V. Badrinarayanan, A. Kendall, R. Cipolla, Segnet: a deep convolutional encoder-decoder architecture for image segmentation, IEEE Trans. Pattern Anal. Mach. Intell. 39 (12) (2017) 2481–2495.
[15] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 779–788.
[16] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask R-CNN, in: Computer Vision (ICCV), 2017 IEEE International Conference on, IEEE, 2017, pp. 2980–2988.
[17] R. Girshick, J. Donahue, T. Darrell, J. Malik, Rich feature hierarchies for accurate object detection and semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 580–587.
[18] Y. Zhang, L. Wang, R. Hartley, H. Li, Where's the Weet-Bix? in: Asian Conference on Computer Vision, Springer, 2007, pp. 800–810.
[19] 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.
[20] M. Marder, S. Harary, A. Ribak, Y. Tzur, S. Alpert, A. Tzadok, Using image analytics to monitor retail store shelves, IBM J. Res. Dev. 59 (2/3) (2015) 3:1–3:11.
[21] A. Franco, D. Maltoni, S. Papi, Grocery product detection and recognition, Expert Syst. Appl. 81 (2017) 163–176.
[22] 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.
[23] 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.
[24] D.G. Lowe, Distinctive image features from scale-invariant keypoints, Int. J. Comput. Vis. 60 (2) (2004) 91–110.
[25] C. Harris, M. Stephens, A combined corner and edge detector., in: Alvey Vision Conference, Vol. 15, Manchester, UK, 1988, pp. 10–5244.
[26] K. Mikolajczyk, C. Schmid, Scale & affine invariant interest point detectors, Int. J. Comput. Vis. 60 (1) (2004) 63–86.
[27] B. Yao, A. Khosla, L. Fei-Fei, Combining randomization and discrimination for fine-grained image categorization, in: Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on, IEEE, 2011, pp. 1577–1584.
[28] J. Kim, C. Liu, F. Sha, K. Grauman, Deformable spatial pyramid matching for fast dense correspondences, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2307–2314.
[29] D. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, 1989.
[30] A. Tonioni, L. Di Stefano, Domain invariant hierarchical embedding for grocery products recognition, Comput. Vis. Image Underst. 182 (2019) 81–92.
[31] B. Santra, A. Paul, D.P. Mukherjee, Deterministic dropout for deep neural networks using composite random forest, Pattern Recognit. Lett. 131 (2020) 205–212.
[32] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, 2016.
[33] P. Pawara, E. Okafor, M. Groefsema, S. He, L.R. Schomaker, M.A. Wiering, One-vs-one classification for deep neural networks, Pattern Recognit. 108 (2020) 107528.
[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Imagenet: a large-scale hierarchical image database, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, IEEE, 2009, pp. 248–255.
[35] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, in: International Conference on Learning Representations, 2015.
[36] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, A. Lerer, Automatic differentiation in PyTorch, NIPS-W, 2017.
[37] 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.
[38] S. Leutenegger, M. Chli, R.Y. Siegwart, Brisk: binary robust invariant scalable keypoints, in: Computer Vision (ICCV), 2011 IEEE International Conference on, IEEE, 2011, pp. 2548–2555.
[39] F. Perronnin, J. Sánchez, T. Mensink, Improving the fisher kernel for large-scale image classification, in: European Conference on Computer Vision, Springer, 2010, pp. 143–156.
[40] S. Lloyd, Least squares quantization in PCM, IEEE Trans. Inf. Theory 28 (2) (1982) 129–137.
[41] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[42] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[43] H. Robbins, S. Monro, A stochastic approximation method, Ann. Math. Stat. 22 (3) (1951) 400–407.
[44] L. Wan, M. Zeiler, S. Zhang, Y. Le Cun, R. Fergus, Regularization of neural networks using dropconnect, in: International Conference on Machine Learning, 2013, pp. 1058–1066.
[45] J.R. Uijlings, K.E. Van De Sande, T. Gevers, A.W. Smeulders, Selective search for object recognition, Int. J. Comput. Vis. 104 (2) (2013) 154–171.
[46] P.F. Felzenszwalb, R.B. Girshick, D. McAllester, D. Ramanan, Object detection with discriminatively trained part-based models, IEEE Trans. Pattern Anal. Mach. Intell. 32 (9) (2010) 1627–1645.

Bikash Santra is currently a Senior Research Fellow at the Electronics and Communication Sciences Unit, Indian Statistical Institute pursuing Ph.D. in Computer Science. He received the M.Tech., M.Sc. and B.Sc. degrees from Indian Institute of Engineering Science and Technology (2016), West Bengal University of Technology (2010) and Vidyasagar University (2007), respectively. His primary research interest is in computer vision, deep learning and image processing. Mr. Santra authored many peer-reviewed research papers and has two patent applications pending. He had held project-linked scholar positions in the research projects at the Indian Institute of Engineering Science and Technology (2010–2011) and Indian Statistical Institute (2013–2016). He has been associated with the conferences like ICVGIP, ICAPR and NCVPRIPG in reviewing the papers and various organizational activities. He serves as a Reviewer of the IEEE Transactions on Image Processing, IEEE Access, and IET Image Processing. Mr. Santra is one of the recipients of 2013 Young IT Professional Award from the Computer Society of India. He is also one of the finalists of Qualcomm Innovation Fellowship (QIF) 2020, India.

Avishek Kumar Shaw is currently a Data Science Engineer at Tata Consultancy Services Limited. He received the B.Sc. (Honours) degree in Computer Science from the West Bengal State University (2013). His primary research interest is in computer vision, deep learning and image processing. Mr. Shaw has two peer-reviewed research papers. He holds a patent (USA and Australia) and also has two patent applications pending. Mr. Shaw serves as a Reviewer of IET Image Processing.

Dipti Prasad Mukherjee is currently a Professor (HAG) at the Electronics and Communication Sciences Unit, Indian Statistical Institute. His primary research interest is in computer vision and applications of machine learning. He has guided several Master's and Ph.D. level dissertations. He has written two books on Computer Graphics and more than hundred thirty peer-reviewed research articles. He had held visiting faculty positions at the Oklahoma State University (1998-99), the University of Virginia (2002, 2013), Alcorn State University (2011), USA, the University of Alberta, Canada (2006, 2007, 2008, 2009, 2020), the University of Guanajuato, Mexico (2015, 2016), and the University of Aegean, Greece (2018). Prior to this, in 1992, Dr. Mukherjee is the recipient of the pre-doctoral UNDP fellowship at the Robotics Research Group, University of Oxford, and the UNESCO-CIMPA fellowship to INRIA, Sophia-Antipolis, France in 1991, 1993, 1995 and to ICTP, Trieste, Italy in 2000. In 2010, he had received Japan Society for the Promotion of Science (JSPS) Invitation fellowship to the Department of Radiology, Graduate School of Medicine, Osaka University and continued to visit Osaka University in the summer of 2012. He is a Fellow of the Computer Society of India, West Bengal Academy of Science and Technology and the Institution of Engineers (India). He had served as an Associate Editor of the IEEE Signal Processing Letters, SADHANA, Springer journal of the Indian Academy of Sciences, the IEEE Transactions on Image Processing and IET Image Processing journal.