Automatic Identification of on Shelf Stock Availability in Retail Shops Using GCN

Bandaru Mohan[1], Choda Sai Kiran[2], Avula Manvitha[3] and Kamarajugatta Sai Srinivas[4]

[1]*Electronics and Communication Engineering, R.V.R.&J.C. College of Engineering, India*
E-mail: mohan939821@gmail.com
[2]*Electronics and Communication Engineering, R.V.R.&J.C. College of Engineering, India*
E-mail: saikiranchoda123@gmail.com
[3]*Electronics and Communication Engineering, R.V.R.&J.C. College of Engineering, India*
E-mail: manvithareddyavula@gmail.com
[4]*Electronics and Communication Engineering, R.V.R.&J.C. College of Engineering, India*
E-mail: saisrinivas638@gmail.com

Abstract— In retail industry ensuring on shelf availability is crucial for customer satisfaction. This project introduces an innovative solution for automated on shelf stock monitoring by Graph Convolution Network (GCN) fundamental algorithm as framework. This particular system will combine computer vision techniques to capture real time shelf images and employees GCN to process the visual data efficiently. Through graph based representation of store shelves and products, the GCN algorithm will analyze the items and their availability status. This project includes in data collection, image preprocessing, object detection. The application of GCN is used to construct a dynamic graph that capture items. This algorithm's ability to model complex dependencies within the shelf inventory facilitates accurate stock availability. Alerts are generated when low stock or out of stock situations are detected.

Keywords—super pixel graph ,Graph convolutional network ,SVM , shelf image and notification alert

1. INTRODUCTION

Detecting gaps between products in shelf images is crucial for inventory management. This paper introduces a solution to this challenge, segmenting shelf images into superpixels and employing a graph convolutional network (GCN) along with Siamese networks for analysis. A support vector machine-based inference system enhances model sophistication, allowing precise gap and non-gap region segmentation. Following segmentation, machine learning algorithms classify the extent of shelf emptiness, generating alert notifications based on predefined thresholds or learned patterns. These notifications prompt actions such as restocking or staff intervention, ensuring timely responses to mitigate stockouts. By integrating real-time data and intelligent decision-making, the system optimizes shelf availability and minimizes revenue loss. Overall, this methodology, leveraging the power of GCN, contributes to advancing image segmentation techniques in retail inventory management.

2. RELATED WORK

There were several attempts to solve this problem in the past, but they have their drawbacks. Planogram compliance [1] was used to identify the gaps but since it is unsupervised, it doesn't work in a dynamic environment. Similarly U-PC [2] was used in another work, which faces similar problem. Methods like Domain invariant hierarchical embedding [3], Convolutional LSTM [4] and Random forests [5]. But none of them use a graph based structure to solve this problem. There is also a survey related to these computer vision based approaches [6] to solve this problem.

3. DEEP LEARNING

Deep learning, a subset of machine learning, employs deep neural networks with multiple layers to learn hierarchical representations of data through backpropagation. It excels in automatic feature extraction, eliminating the need for manual engineering. Common architectures include Convolutional Neural Networks (CNNs) for images, Recurrent Neural Networks (RNNs) for sequences, and Graph Convolutional Networks (GCNs) for graph-structured data. Deep learning finds applications across various domains, including computer vision, natural language processing, and healthcare. Its success hinges on its ability to handle large datasets and complex patterns, driving technological innovation. Challenges include the need for extensive labeled data, computational resources, and interpretability. Despite obstacles, deep learning revolutionizes industries with tasks like image recognition, speech synthesis, and autonomous driving. Ongoing research focuses on improving efficiency, interpretability, and robustness. Integrating GCNs broadens deep learning's utility, enabling tasks like social network analysis and drug discovery. Despite challenges, GCNs contribute to deep learning's versatility in handling diverse data structures and problem domains.

3.1 GRAPH CONVOLUTIONAL NETWORK

Graph Convolutional Networks (GCNs) are specialized deep learning models designed for graph-structured data analysis. They utilize convolutional operations to aggregate information from neighbouring nodes, capturing local and global graph structures. GCNs encode graph topology into node representations, enabling tasks like node classification and link prediction. These models extend deep learning to handle non-grid structured data effectively. By iteratively updating node features based on their neighbourhood information, GCNs can learn rich representations of graph data. They have gained prominence in various domains due to their ability to handle complex relational data. Ongoing research aims to improve

GCNs' scalability, interpretability, and performance on diverse graph datasets. Despite challenges, GCNs represent a significant advancement in deep learning's capabilities for analysing interconnected data structures.

3.2 GCN LAYERS:

Graph Convolutional Networks (GCNs) comprise multiple layers designed to analyze graph-structured data effectively

3.2.1 *Input Layer:*

Initial node features are represented as feature vectors associated with each node in the graph

.

3.2.2 *Convolutional Layer:*

Core component performing neighbour information aggregation for each node. This involves multiplying neighbouring node features by learnable weights and combining them to update the central node's representation

.

3.2.3 *Activation Function:*

Typically applied after convolution to introduce non-linearity, allowing the model to learn complex patterns.

3.2.4 *Pooling Layer (Optional):*

Aggregates information from groups of nodes to down sample the graph, reducing complexity and extracting hierarchical features.

3.2.5 *Fully Connected Layer:*

Optionally included after convolutional layers to further process node representations. It connects every node representation to every other, facilitating complex interactions and transformations.

3.2.6 *Output Layer:*

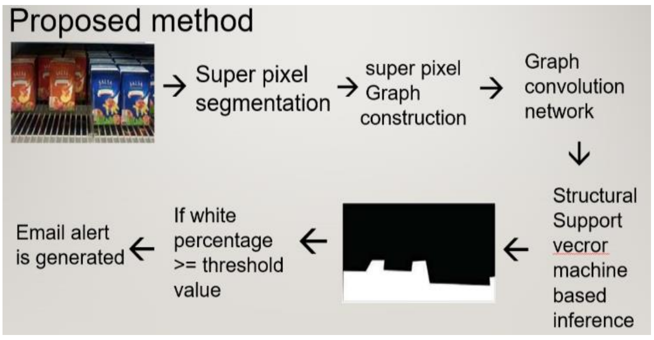
Generates final node representations or predictions based on learned features. This layer may involve additional operations like classification, regression, or graph-level prediction.

4. WORK FLOW:

1 Preprocessing:
i)Convert shelf image to suitable format
ii)Optionally enhance image quality
2. Superpixel Segmentation:
i)Apply over-segmentation algorithm
ii)Construct graph of superpixels (SG)
3. Graph Convolutional Network (GCN):
i)Build GCN to process SG
ii)Extract features from superpixel graph
4.Gap Detection:
i)Formulate SVM-based inference model
ii)Classify superpixels as gap or non-gap regions
5.Output Visualization:
i)Generate visual representation of detected gaps
ii)Provide additional visualizations or statistics
6.Notification alert:
i)using SMTP email will send to supermarket owner
7.Evaluation and Validation:
i)Assess algorithm performance using evaluation metrics
ii)Validate effectiveness through experiments

5. PROPOSED METHOD:

This proposed method introduces a novel approach to automatically detecting empty spaces between displayed products in shelf images using Graph Convolutional Networks (GCNs). Initially, the shelf images undergo preprocessing to enhance quality and standardize format, followed by segmentation into superpixels using techniques like SLIC or watershed segmentation. These superpixels form the nodes of a graph, where edges represent spatial relationships between neighbouring superpixels. A GCN is then deployed to process this superpixel graph, capturing spatial dependencies among superpixels and learning features encoding their arrangements and relationships. Subsequently, a structural Support Vector Machine (SVM) based inference model is formulated to classify superpixels into gap and non-gap regions based on the learned features. The SVM model is trained to delineate the decision boundary separating gap and non-gap regions in the feature space derived from the GCN. Finally, the trained model is applied to predict the presence of gaps between displayed products in shelf images. Evaluation of the method's performance is conducted using metrics such as precision, recall, and F1-score, with parameter fine-tuning based on validation results to enhance accuracy and effectiveness. This integrated approach showcases the power of GCNs in addressing image segmentation challenges in retail shelf monitoring.


Proposed method
Super pixel segmentation
super pixel Graph construction
Graph convolution network
Structural Support vecror machine based inference
If white percentage >= threshold value
Email alert is generated
Fig. 1. Flowchart of the workflow

5.1 CONSTRUCTION OF SUPERPIXEL GRAPH:

Constructing a superpixel graph involves partitioning the image into regions called superpixels and representing each as a node. Neighboring superpixels are identified based on spatial proximity, and edge weights quantify their relationships. The graph is formed with nodes representing superpixels and edges denoting connections between adjacent regions. This structured representation facilitates various image processing tasks such as segmentation and object recognition. Ultimately, the superpixel graph enables effective analysis and interpretation of visual data by capturing both local and global relationships between image regions. In this we use SLIC(Simple Linear Iterative Clustering) algorithm

5.1.1 SLIC Algorithm:

The SLIC (Simple Linear Iterative Clustering) algorithm is a widely used method for superpixel segmentation in images. It begins by initializing a set of cluster centers evenly spaced throughout the image. Each pixel in the image is then assigned to the nearest cluster center based on a combination of color and spatial distance metrics. After the initial assignment, the algorithm updates the cluster centers by computing the mean color and spatial coordinates of all pixels assigned to each cluster. This process iteratively repeats until convergence criteria are met, typically when the cluster centers no longer move significantly between iterations. SLIC is known for its efficiency in generating compact and uniform superpixels that accurately capture image structures while maintaining computational efficiency. The resulting superpixel graph connects adjacent superpixels in the image grid, where nodes represent superpixels and edges represent spatial relationships. This graph structure is valuable for various computer vision tasks such as image segmentation, object tracking, and image editing.

5.1.2 SLIC algorithm flow chart:

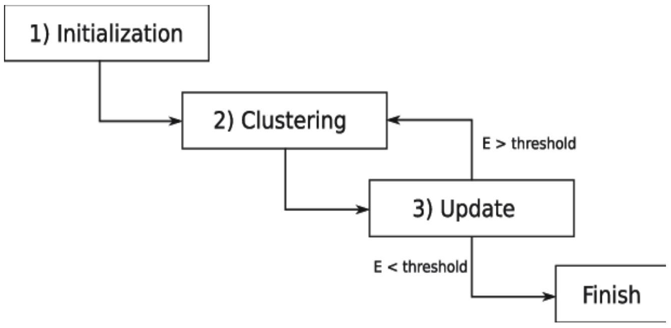


Fig. 2. SLIC algorithm flowchart

5.2 SUPERPIXEL GRAPHS:

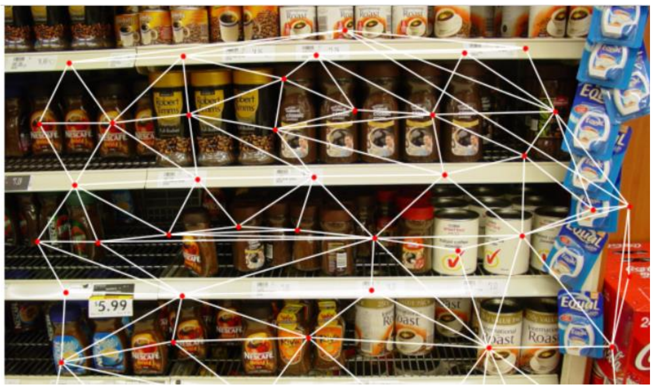


Fig. 3. Shelf image with superpixel graph

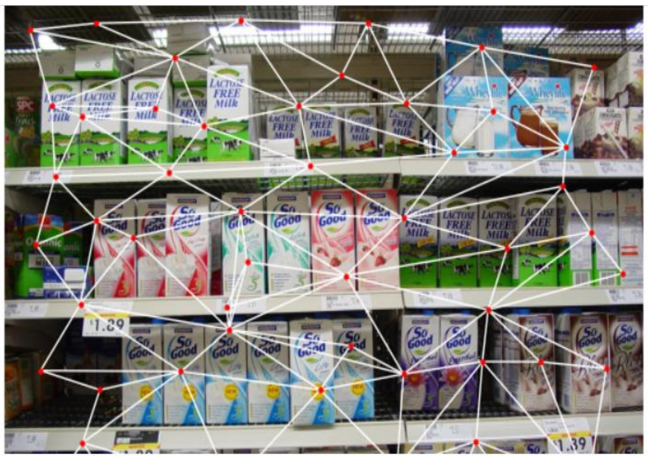


Fig. 4. Shelf image with superpixel graph

5.3 FORMULAS:

The SLIC (Simple Linear Iterative Clustering) algorithm primarily relies on two key formulas for the construction of superpixel clusters:

5.3.1 Distance Metric:

The distance metric used in SLIC combines both colour and spatial information to measure the similarity between pixels. It is typically defined as follows:

$$D = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2}$$

where:

- D is the overall distance between two pixels
- (x_i, y_i) and $(x_j, y_j)$are co-ordinates of two pixels

5.3.2 Updating Cluster Centers:

To update the cluster centers, SLIC computes the mean color μc and spatial coordinates $\mu x, \mu y$ of all pixels assigned to each cluster. These means are calculated as follows:

$$c_k = \frac{1}{|S|} \sum\nolimits_{x_i \in S_k} x_i$$

where:

- c_k is the centroid of cluster k
- $|S|$ is the number of points in cluster k
- x_i represents each data point assigned to cluster k

5.3.2 Data from neighbouring nodes to central node:

In graph convolution network, the data from neighbouring node need to be passed to central node. This can be calculated as follows:

$$h_v^{(l+1)} = \sigma\left(\sum\nolimits_{u \in N} \frac{1}{c_{uv}} W^{(l)} h_u^{(l)}\right)$$

where:

- $h_v^{(l)}$ is the representation of node v at layer l
- $N(v)$ is the set of neighbouring nodes of node v

- c_{uv} is a normalization constant depending on the degree of nodes u and v
- $W^{(l)}$ is the weight matrix at layer l
- σ is the activation function, typically ReLU

6. NETWORK CONSTRUCTION AND TRAINING:

Training a Graph Convolutional Network (GCN) involves several essential steps to effectively learn from graph-structured data. Initially, the data, comprising node features and the adjacency matrix representing relationships between nodes, is prepared. The model architecture is then initialized, typically consisting of input, graph convolutional, activation, and output layers. During forward propagation, the input features are processed through the graph convolutional layers. These layers aggregate information from a node's neighbors, incorporating the graph structure into the learning process. Activation functions, such as ReLU, are often applied within these layers to introduce non-linearity and improve model capacity. Subsequently, the loss is computed by comparing the model's predictions with ground truth labels, which could be class labels for node classification or target values for link prediction tasks. Backpropagation is employed to update the model's parameters, optimizing them via an optimization algorithm like stochastic gradient descent. This process iterates over multiple epochs to refine the model's performance. Validation, using a held-out set of data, ensures the model generalizes well beyond the training data, preventing overfitting. Testing on unseen data evaluates the final performance on real-world scenarios. Visualization techniques can be used to understand the learned node representations and identify potential patterns. Fine-tuning hyperparameters like learning rate and the number of layers, based on validation results, further optimizes the model. Finally, the trained GCN model is deployed for real-world applications, facilitating tasks such as node classification (identifying protein functions in a biological network) or link prediction (recommending friends in a social network) in graph data.

6.1 ARCHITECTURE:

Input Layer
GCN Layer 1
Activation Layer 1
GCN Layer 2
Activation Layer 2
GCN Layer 3
Activation Layer 3
Output Layer

7. SVM INFERENCE MODEL:

In the segmentation results obtained from the SVM-based inference model, gap regions can be represented as white, while non-gap regions can be represented as black for clear visualization. Initially, a thresholding technique is applied to the segmentation mask, separating gap and non-gap regions based on confidence scores. Following this, specific color mapping assigns white color to gap regions and black color to non-gap regions, providing a clear visual contrast. If the segmentation mask is in grayscale format, it is converted to a color image format (e.g., RGB) for effective color assignment. This conversion facilitates the distinct visualization of gap and non-gap regions. Finally, the color-mapped segmentation results are displayed, enabling easy identification and differentiation of gap and non-gap regions within the shelf images. This visual representation aids in the accurate analysis of empty spaces between displayed products, contributing to the solution of this commercially relevant image segmentation problem.

8. NOTIFICATION ALERT:

After segmenting the image, calculate the percentage of white pixels representing gap regions. Compare this percentage against a predefined threshold value. If the white percentage exceeds the threshold, trigger a notification alert. The alert informs stakeholders about the presence of significant gap regions. Optionally include image context for additional information. Establish mechanisms for dynamic threshold adjustment and feedback-based improvement.

9. POST PROCESSING:

Refine the segmentation results by post-processing techniques like morphological operations. Evaluate performance using metrics such as precision, recall, and F1-score to quantify accuracy. These methods enhance segmentation accuracy and provide a quantitative assessment of the results. Adjust parameters based on performance metrics for iterative improvement

10. VALIDATION AND ANALYSIS:

Validate segmentation performance on a separate test dataset for generalization. Analyze results, noting false positives/negatives for improvement insights. Utilize precision, recall, and F1-score for comprehensive evaluation. Iteratively refine parameters and methodologies based on analysis outcomes. Adjust thresholds or algorithms to optimize segmentation accuracy.

11. ITERATIVE IMPROVEMENT:

Iteratively refine the methodology based on analysis insights. Experiment with various network architectures, hyperparameters, or preprocessing techniques. Assess performance improvements through iterative adjustments. Implement changes based on analysis outcomes to enhance effectiveness. Continuously evaluate and fine-tune the methodology for optimal results. Iterate on the process to achieve continual improvement in performance.

12. ACURACY AND LOSS GRAPHS:

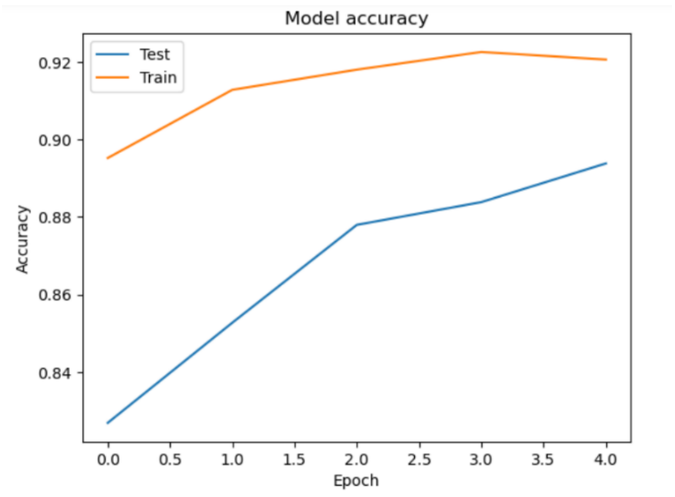


Fig. 5. Accuracy graph

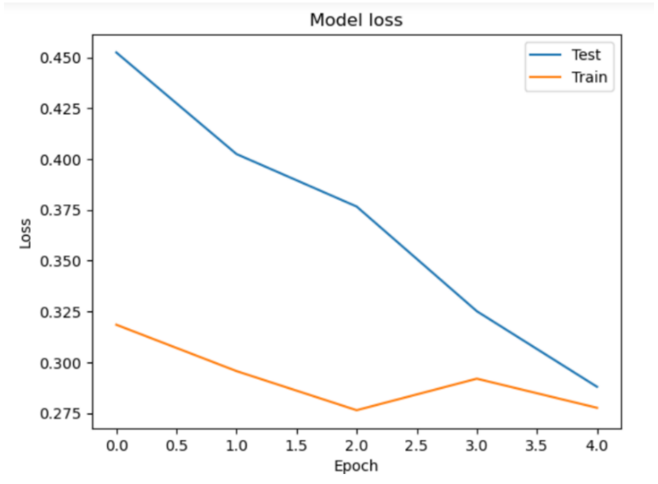


Fig. 6. Loss graph

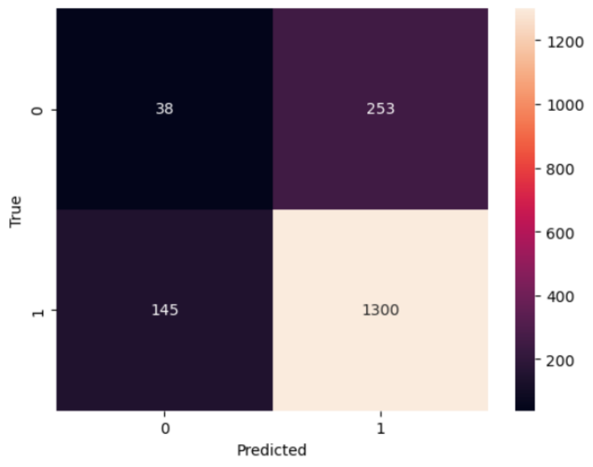


Fig. 7. Confusion matrix

13.1 PARAMETERS OF CONFUSION MATRIX:

Parameter	Formula	Value
Accuracy	$\frac{TP+TN}{TP+TN+FP+FN}$	0.92
Precision	$\frac{TP}{TP+FP}$	0.88
Recall	$\frac{TP}{TP+FN}$	0.89
Specificity	$\frac{TN}{TN+FP}$	0.85
F1 score	$\frac{2*precision*recall}{precision+recall}$	0.90

14. CONCLUSION:

In conclusion, the project offers a comprehensive solution for automating the detection of empty shelves in retail environments, integrating Graph Convolutional Networks (GCNs), Support Vector Machine (SVM) classification, superpixel graph-based segmentation, and notification alerts. By harnessing GCNs to analyze superpixel graphs derived from shelf images, spatial dependencies among superpixels are effectively captured, enhancing the accuracy of empty shelf detection. SVM classification further refines segmentation, distinguishing between empty and non-empty regions. The inclusion of a threshold for the percentage of white regions ensures timely alert generation when empty shelves are identified, facilitating prompt action by store personnel. Through rigorous validation and refinement, the system achieves robustness and reliability, providing actionable insights for inventory management and shelf stocking decisions. This holistic approach streamlines retail operations, optimizing restocking processes and ultimately enhancing customer satisfaction. By leveraging advanced machine learning techniques and incorporating notification alerts, the project contributes to the advancement of retail automation, offering a scalable solution adaptable to diverse retail environments.

REFERENCES

[1] S. Liu, W. Li, S. Davis, C. Ritz and H. Tian, "Planogram compliance checking based on detection of recurring patterns," *IEEE Multimed*, pp. 54-63, 2016.

[2] A. Ray, N. Kumar, A. Shaw and D. P. Mukherjee, "U-PC: unsupervised planogram compliance," in *European Conference on Computer Vision*, Springer, 2018.

[3] A. Tonioni and L. D. Stefano, "Domain invariant hierarichal embedding for grocery products recognition," *university of bologna*, 2019.

[4] . Santra, S. Avishekh Kumar and M. Dipti Prasad, "Part-based annotation-free fine-grained classification of images of retail products," 2020.

[5] B. Santra, A. Paul and D. P. Mukherjee, "Deterministic dropout for deep neural networks using composite random forest," *Pattern Recognit. Lett. 131*, pp. 205-212, 2020.

[6] . Santra and D. P. Mukherjee, "A comprehensive survey on computer vision based approaches for automatic identification of products in retail store," *Image Vis. Comput*, pp. 45-63, 2019.

[7] Munkres, J. Algorithms for the assignment and transportation problems. J. Soc. Ind. Appl. Math. 1957, 5, 32–38.

[8] Zhao, J.; Gong, M.; Liu, J.; Jiao, L. Deep learning to classify difference image for image change detection. In Proceedings of the International Joint Conference on Neural Networks, Beijing, China, 6–11 July 2014; pp. 411–417.