

A Smart Shopping System for Modern E-Commerce Applications

¹Jyothi P, ²Divya Kumari Tankala, ³Rajesh Kumar A, ⁴Nagarjuna Reddy S, ⁵Nagendar Yamsani and ⁶Jyotsna Devi Kosuru S N V

¹Department of Computer Science and Engineering, VNR Vignana Jyothi Institute of Engineering and Technology, Vignana Jyothi Nagar, Pragathi Nagar, Hyderabad, India.

²Department of Computer Science Engineering, G. Narayanamma Institute of Technology and Science, Hyderabad, India.

³Department of Artificial Intelligence and Data Science, N.S.N. College of Engineering and Technology, Karur, Tamil Nadu, India.

⁴Department of Computer Science and Engineering, Lakireddy Bali Reddy College of Engineering, Mylavaram, Andhra Pradesh, India.

⁵School of Computer Science and Artificial Intelligence, SR University, Warangal - 506371, Telangana, India.

⁶Department of Computer Science and Engineering, Koneru Lakshmaiah Education Foundation, Vaddeswaram, Guntur, Andhra Pradesh, India.

¹jyothi_p@vnrvjiet.in, ²divyatankala@gnits.ac.in, ³arurajesh1980@gmail.com, ⁴nagvik@gmail.com

⁵nagendar.yamsani@gmail.com, ⁶jyotsnakosuru@gmail.com

Correspondence should be addressed to Jyothi P : jyothi_p@vnrvjiet.in

Article Info

Journal of Machine and Computing (<https://anapub.co.ke/journals/jmc/jmc.html>)

Doi : <https://doi.org/10.53759/7669/jmc202505209>

Received 02 June 2025; Revised from 16 August 2025; Accepted 18 September 2025.

Available online 05 October 2025.

©2025 The Authors. Published by AnaPub Publications.

This is an open access article under the CC BY-NC-ND license. (<https://creativecommons.org/licenses/by-nc-nd/4.0/>)

Abstract – In a modern E-commerce system, exact and efficient classification of products is one of the major challenges. The user experience of online purchasing heavily relies on product classification accuracy as well as. Given the large number of products and viable categories, have developing a new framework automatically for assigning the products to suitable categories at scale is desirable. However, inadequate item descriptions, poor data quality and adversarial noise in training data, can lead to low prediction accuracy when applying the Machine Learning (ML) algorithms. In addition, the various influx of new products uploaded daily, and the dynamic nature of categories highlight the need for novel intelligent classification models that can control the cost and time required by human editors. To overcome these kinds of issues, Squirrel search War Strategy Optimization (SSWSO)_LeNet replica is introduced. SSWSO is a revolutionary nature-inspired optimization algorithm developed for unconstrained optimization issues. The foraging behavior of southern flying squirrels is examined and quantitatively simulated, considering every aspect of their food hunt to achieve the desired optimum. Squirrel search Algorithm (SSA) provides global optimum solutions with excellent convergence behavior. Meanwhile, War Strategy Optimization (WSO) algorithm includes an adaptive weight mechanism that varies from one solution (soldier) to another and is updated weights depends on the soldier's rank during the update phase. Thus, the combined optimization strategies ensure an efficient balance among the exploration and exploitation stages. By integrating WSO and SSA provide better product classification accuracy and superior performance. Furthermore, the experimental findings showed that the SSWSO_LeNet performed better in accuracy, sensitivity, and specificity, with values of 0.976, 0.877, and 0.857, which is impressive compared to state-of-the-art results.

Keywords – E-Commerce, Statistical Features, Product Classification, War Strategy Optimization, LeNet.

I. INTRODUCTION

Internet tech keeps innovating and improving year by year. A digital tech representative affects various economic and societal areas and drives up spending growth. The growing use of mobile internet users and rapid advancements in mobile payment technologies have continuously improved and integrated e-commerce transactions into daily life. [24][9]. Anyone can purchase and sell items through e-commerce websites. E-commerce apps employ software designed mainly for offering diverse services to facilitate transactions between consumers and businesses. E-commerce is a developing technology enhancing user communication's speed and effectiveness. The main action in online stores' buying journey is finding items and looking through their list. Products are recognized by how they look, work well, and act. Textual products' names and descriptions, digital media's content, and classifications are used to make decisions throughout the customer's journey.

They are expressed in various forms such as textual product names and descriptions, digital media content, and classifications that are crucial for making choices at every stage of the customer's path. All these traits greatly affect how customers feel about getting things done easily, whether they use technology or not, and what their final impression is. Recently, many businesses and their clients agreed they would share details about products by leaving scores and comments, as well as requesting help from others when deciding which features of the products to use. [24][25].

Product image categorization [2] involves classifying items into suitable categories, making it an essential task in e-commerce processing. Product categorization facilitates efficient organization and quick access to online shopping items. It's advantageous for sophisticated in-store retail systems, like personalized product suggestions [28]. Product categorization takes much effort from both companies and clients. Product categorization is crucial for e-commerce companies. A product's details are frequently described through metadata, which includes things like an image, name, weight, color, among others, usually done by the seller themselves. A product placed on an online store is frequently categorized into numerous groups. E-commerce sites enhance client satisfaction through categorization of products, such as swiftly searching the catalog or creating recommendation systems [26]. The vendor needs to clearly mark and take photos of their goods so they can be easily organized. The customer has to quickly find what they want and get more helpful suggestions. The e-commerce system frequently makes errors due to human mistakes. Misclassification leads to problems like products not showing up in searches, wrong suggestions, low or fewer sales for both sellers and buyers [27].

E-commerce platforms frequently present product profiles with disorganized language, visuals, product names, and pertinent details about products. Profiles help people buy things by giving them ideas, and they also tell computer programs what to do when shopping online. Having too many products and product categories on an e-commerce site can make managing them difficult. Therefore, it's important to use systems that help correctly place these items into their proper categories. [2]. Several machine learning tools, such as the SVM, are commonly employed for categorizing e-commerce image products. Support vector machines (SVM) are highly regarded for their effectiveness in classification tasks, particularly when applied to image data from e-commerce platforms by numerous researchers. [1]. The popularity of AI-driven ML-based categorization technology has grown significantly due to advancements in the industry. Deep learning (DL), an extensive method for classification, excels in successfully categorizing product images. Artificial Neural Networks, K-nearest neighbors, and Naive Bayes are various classifiers used in classification tasks. [28].

The main goal of this research is to create a smart system that can quickly sort through many items sold online. This system uses pictures of the products to help tell if they're good or bad. It's called SSWSO, which stands for Smart Sorting with Pictures Online. The idea is to make sure all the things' people buy are nice and in good condition. Data augmentation is achieved through rotation, cropping, flipping, scaling, and translation. Here, the design of rotation and flipping was intended to partly solve the viewing issue. Cropping and zooming fixed background issues, while scaling changed the image size randomly within limits. For extracting features, CNN, statistics, GLCM, and PHOG methods are utilized, leading to improved accuracy, reduced overfitting, faster training, and better data visualization. LeNet performs the classification task, utilizing the SSWSO method for training, where SSWSO combines SSA and WSO techniques.

The primary contribution of this research is,

- The LeNet is introduced for E-commerce product classification in which the SSWSO is utilized for training LeNet. Here, the SSWSO hybridized SSA and WSO by exploiting the best aspects of both approaches. SSA mimics southern flying squirrels' dynamic feeding behavior and efficient gliding locomotion. Small mammals use gliding to travel great distances efficiently.
- The strategic movement of army soldiers during a war is the foundation of WSO. Each soldier in military strategy advances dynamically toward the optimal value, modeled as an optimization process. The WSO achieves a suitable balance between the exploration and exploitation stages. Thus, SSWSO classifies the E-commerce products into Jeans, T-shirts, Sofas, and Televisions, which facilitates customers' search for what they want and can lessen dissatisfaction and desertion rates.

This paper is organized as follows. An overview of the literature on various E-commerce product classification is given in Section 2; the SSWSO_LeNet scheme is presented in Section 3; the scheme's results and discussion are shown in Section 4; and the paper is concluded in Section 5.

II. LITERATURE SURVEY

S.A. Oyewole and O.O. Olugbara [1] have introduced improved color image-based product categorization architecture, which includes a median filter, Eigen Color features (cHOG-ECF and cULBP-ECF), and an artificial neural network ensemble for the classification of products in E-commerce. The processing steps for product categorization were pre-processing, extraction of features, reducing dimensions, and classification.

Tsegaye Misikir Tashu *et al.* [2] have introduced a multi-modal model, named Hierarchical Fusion for product classification, which integrates extracted features derived by Neural Network (NN) models from visual and textual data with simple fusion methods. In addition, hierarchical fusion was done to merge representations from linguistic and visual models and finally, a fully connected layer for final classification. The technique shortened the training period and produced improved overall performance. However, overfitting issues must be addressed to achieve a robust model.

Wen Zhang *et al.* [3] have devised a deep hierarchical product image classifier using Bidirectional Encoder Representations from Transformers (BERT) pretrained knowledge. Here, the appropriate sampling threshold values were

selected for department and leaf classifiers due to their distinct number of categories. To train the department classifier using Deep NN, hundreds of items were randomly selected per leaf node and merged into a department. For the leaf classifier, thousands of objects were chosen randomly per node without replacement. After preprocessing, the department and leaf classifiers were trained independently using the same deep network topology. Then, the department classifier predicts that the source item belongs to the electronics department. The electronics department's leaf classifier predicts the item to be an iPhone case. After that, the rank model prediction scores were identified in suggested departments, and then a leaf node was inferred within each department. This technique generates departmental and leaf prediction scores. To perform thresholding and model calibration, the confidence score was calculated by multiplying the two values.

Anastraj [4] developed an Anti-Money Laundering (AML) algorithm with Local Binary Pattern-Deep Belief Network (LBP-DBN) to improve E-commerce image classification accuracy and training time. The steps for product classification were pre-processing of images, extraction and selection of features, and classification. In the first step, the images are smoothed, denoised, enhanced, corrected, and background-removed to provide finer edges and stronger contrast. The image attributes, including texture, form, color, and space were retrieved in the second step. Typically, multiple picture features were extracted. Failure to complete feature selection reduces classification efficiency and accuracy. Therefore, it's vital to remove features. In the final step, the classification was done by AML with DBN. Here, the method improved the classification accuracy speed, but still faces computational complexity issues.

Yonatan Hadara and Erez Shmueli [5] have introduced an Ensembled Transferred Embeddings learning (ETE) framework for item categorization. We demonstrated that the suggested framework may be used for item categorization in noisy and short textual attributes without labels. The method's supremacy was due to the item descriptions being brief, loud, and domain-specific in nature of the text used. In contrast, the weakness was that it relied on domain expertise to find relevant large-scale labeled datasets.

Tianlong Xu *et al.* [6] have introduced a scalable and sustainable picture-type classification system called margin-based submodule sampling. It separates several image types from home improvement retailers' web platform products. An expandable ontology was created to define and arrange image attributes after analyzing the mutually exclusive links between numerous concepts to describe their content and viewpoints. Moreover, to expand the ontology graphs based on feedback from business users and new exceptional circumstances, deep learning, classical computer vision, and Siamese network triplet loss minimization were used to build robust Image Type Classifier (ITC) models based on the ontology. ITC general models successfully recommend the correct image, whereas multiarmed bandit was not used to automatically select the best image for each category for each customer's browsing experience.

Fengzi Li *et al.* [7] have developed Transfer learning (TL) to train neural network models using product images from E-commerce. Here, the TL includes pre-trained models, like VGG19, to classify images. Moreover, Cosine Similarity and Autoencoders were utilized to find similar photos. In addition, shifting, horizontal flipping, rotation, and zoom parameters were done to augment the data. This method only returns randomly altered data throughout the training process. Two-tiered pre-processing approach was used to overcome data imbalance, and then ImageDataGenerator was introduced for data augmentation and rebalancing. Weight balancing can help overcome imbalances in training datasets. It balances data by giving more weight to minority classes. This model performs well overall, with the highest accuracy when trained to predict product sub-categories. However, many products have limited visuals, making it challenging to train models objectively.

To circumvent time constraints, Yarasu Madhavi Latha and Bhukya Srinivasa Rao [8] created an Autoregressive Co-Variance Matrix with Gabor Filter Ensemble Convolutional Neural Network (ARCM-GF-E-CNN) for classification. The internet product database was used as input and then pre-processed with a Gabor filter to remove noise and resize images. Feature extraction was used to estimate components in a desired set by subtracting the backdrop. Image components were derived from background subtraction data, in which product image features are typically extracted online. Images were segmented based on intensity and feature extraction using the ARCM-GF. After feature extraction, features were trained and tested in E-CNN to convert pixel intensity, spatial frequency, and texture for analysis. The ARCMGF-E-CNN achieved greater accuracy in the validation and testing dataset, but still, it took higher processing time. **Table 1** displays the relevant work.

Table 1. Related Work

Author(s)	Proposed Method/Model	Key Features / Steps	Strengths	Limitations
S.A. Oyewole & O.O. Olugbara [1]	ANN Ensemble with Eigen Color Features (cHOG-ECF & curb-ECF)	Pre-processing → Feature extraction → Dimensionality reduction → Classification	Improved categorization using color features	Computational complexity in feature extraction
Tsegaye Misikir Tashu et al. [2]	Hierarchical Fusion (visual + textual features)	NN-based feature extraction → Simple & hierarchical fusion → Fully connected layer → Classification	Reduced training period, improved performance	Overfitting issues remain

Wen Zhang et al. [3]	Deep Hierarchical Product Image Classifier (BERT + NN)	Sampling thresholds for department & leaf classifiers → Independent training → Score multiplication for calibration	Department + leaf predictions, robust hierarchy	Requires large datasets, complex thresholding
Anastraj [4]	AML with LBP-DBN	Pre-processing (denoising, enhancement, background removal) → multi-feature extraction → Feature selection → DBN classification	Improved classification accuracy & training time	Computationally expensive
Yonatan Hadara & Erez Shmueli [5]	Ensembled Transferred Embeddings (ETE)	Noisy/short text handling → Transfer learning embeddings → Item categorization	Works well for noisy, domain-specific text	Requires domain expertise & large labeled datasets
Tianlong Xu et al. [6]	Margin-based Submodule Sampling + Ontology ITC	Ontology design → Deep learning + CV + Siamese network → Triplet loss minimization → ITC general models	Scalable & sustainable classifier, robust ontology	No automated best-image selection (multiarmed bandit not applied)
Fengzi Li et al. [7]	Transfer Learning (VGG19, TL + Autoencoders)	Data augmentation (flipping, zoom, rotation) → Cosine similarity → Weight balancing for imbalance	High accuracy in sub-category prediction	Limited visuals in many products hinder training
Yarasu Madhavi Latha & Bhukya Srinivasa Rao [8]	ARCM-GF-E-CNN	Pre-processing with Gabor filter → Background subtraction → Feature extraction (ARCM-GF) → E-CNN classification	Achieved high accuracy in testing/validation	High processing time

The Challenges of the Related Work are Presented Below

One of the most common difficulties today is the abundance of generally unstructured information on the internet. With the exponential growth of data on the internet, organizing information becomes a vital challenge for supporting users and businesses in storing and retrieving information. One example of such work is the automated classification of E-commerce products, which also includes the issue of retrieving such products from the internet. This issue stems from a high demand for clustered or categorized E-commerce products among market researchers who need to consider various types of statistics in the E-commerce sphere, such as comparison of shopping engines, like Google Shopping or Yandex Market, information retrieving systems, and other types of services. Because of the large range of products sold through E-commerce channels, it is difficult to describe all product attributes accessible (including their unique set of attributes). Catalogs are dynamic in nature, with old products discarded and new products introduced on a daily basis.

III. PROPOSED METHODOLOGY

This section outlines the SWSO_LeNet approach for product classification in e-commerce. The study employs the following steps: pre-processing, feature extraction, data augmentation, and product classification. Initially, input data is sent to the pre-processing stage, where the Region of Interest (ROI) is identified using an Adaptive Weiner filter. The pre-processed data is then fed into the feature extraction module, which utilizes tools like CodeIgniter, XAMPP, GLCM, and PHOG features to recognize and represent various image structures. Feature extraction is effective in reducing computational resources while preserving essential information, thus enhancing the efficiency and accuracy of deep learning models. After feature extraction, the data undergoes augmentation through techniques such as flipping, rotating, cropping, scaling, and translation. Finally, the E-commerce product is classified into categories like sofas, t-shirts,

televisions, and jeans using LeNet. The LeNet model is trained using SSWSO, which integrates WSO and SSA. **Fig 1** illustrates the SSWSO_LeNet block diagram.

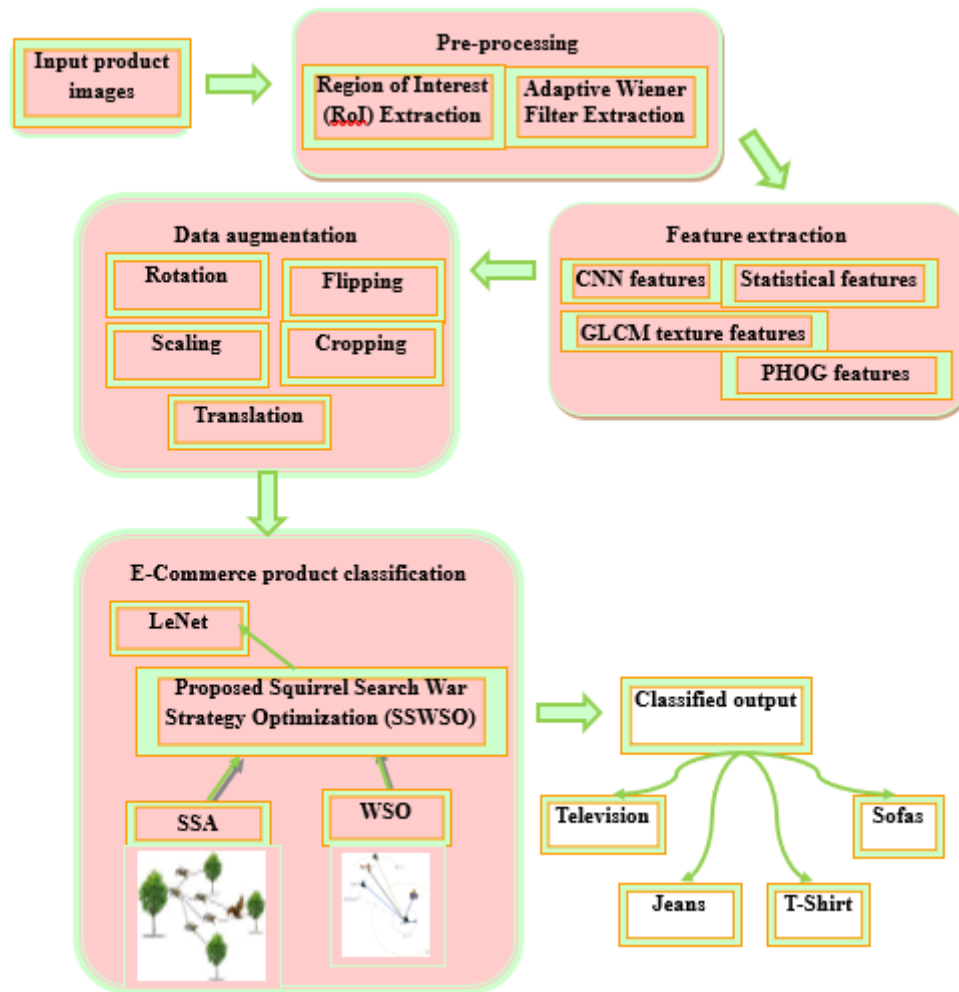


Fig 1. Block Diagram for E-Commerce Product Classification Based on the SSWSO_LeNet.

Prior to E-Commerce product classification, the input image is first aggregated from the dataset. Consider a dataset R containing images S is given by,

$$R = \{J_1, J_2, \dots, J_r, \dots, J_s\} \quad (1)$$

where, R denotes database, S conveys the total number of photos, and J_r represents r^{th} image. As a result, the input image is pre-processed to remove any noise included in it.

Noise Reduction Through Pre-Processing

Pre-processing helps in converting raw data into a more comprehensible format. It also enhances the image's features and minimizes distortions. Therefore, image preprocessing is highly effective in improving image quality and making them more suitable for analysis and further processing. The input image is provided to the preprocessing system, which is developed using a web-based approach. The application is built with Visual Studio Code, MySQL as the database, Codelgniter as the framework, and XAMPP for execution.

Extracting the Region of Interest (ROI)

The ROI is found by measuring pixel brightness and then using a mask. The ROI is used to hide the important areas by covering them with less interesting parts. In this case, the brightness of the pixels is described by a range of brightness values that include the attractive parts of the image. When a user creates a gray-scale brightness value by selecting certain areas and the rest is considered background, the image changes to a black and white version, a process called thresholding.

If the areas we want are in the middle range of brightness, then we can choose a part of the brightness values between 0 (black) and 255 (white) to split the image and separate the parts from the background. Usually, the brightness values of the ROI are called density slices, where the neighboring pixel is given a value of 1 and the value of 0 is ignored.

Adaptive Wiener Filter

To eliminate noise from input images caused by machine vibration, human operation, and image reconstruction flaws, the adaptive Wiener filter [11] applies an image noise filtering method. This enhances the image quality and facilitates accurate object and contour detection. The Wiener filter preserves the image edges while minimizing noise, using Visual Studio Code as the development environment, MySQL as the database, CodeIgniter as the framework, and XAMPP for execution.

$$f = \frac{1}{IM} \sum_{h_1, h_2 \in H} A(h_1, h_2) \quad (2)$$

where, H is the inputted image value selected in the window size, I and M stands for length and breadth of the window size, A denotes original image, and h_1, h_2 are the coordinates of the selected raw image in the window size.

The variance is given as follows:

$$\text{Var}(x) = \frac{1}{IM} \sum_{h_1, h_2 \in H} A^2(h_1, h_2) - f^2 \quad (3)$$

The coordinates are as follows:

$$g(h_1, h_2) = f + \frac{\text{Var}(x) - n^2}{\text{Var}(x)} (A(h_1, h_2) - f) \quad (4)$$

where, the noise variance is given as n^2 . Consequently, the extracted features module receives the pre-processed image P^r as an input.

Extracting Features

After preparing the images, we get information from them to check how realistic they are and what important parts they have. The step where we get this information uses the processed images. In this step, we take out features like homogeneity, contrast, correlation, energy, inverse difference moment, and coarseness from GLCM, along with statistical features such as kurtosis, mean, skewness, variance, and entropy. We also get features from CNN and PHOG. These features are explained in detail below:

Statistical Features

Statistical characteristics [12] are features extracted from images at the pixel level through analysis. These include entropy, variance, mean, skewness, and kurtosis. Entropy evaluates the content of segmented images and system dysfunction. A more symmetric temperature distribution results in lower entropy. Entropy is calculated using the expression provided below. The system is built by the author using a web-based application with Visual Studio Code and MySQL as the database. CodeIgniter is used as the framework, and XAMPP is employed to run the application.,

$$L_1 = \sum_{a=1}^v N_a * \log(N_a) \quad (5)$$

Mean

Mean represents the average value of pixels contained in an image, and also focuses on the total distribution of pixels, and is defined as,

$$L_2 = \sum_{a=1}^v P_a * N_a \quad (6)$$

where, N_a denotes pixel's probability a , V defines total amount of pixels in the product image, and P_a is the pre-processed pixel in the image.

Variance

Variance shows how much the grey levels in an image differ from the average grey level. It is calculated using the equation provided below.

$$L_3 = \sum_{a=1}^V (P_a - L_2)^2 * N_a \quad (7)$$

wherein, L_2 represents the mean.

Kurtosis

It is often referred to as the "fourth normalized moment". The below equation calculates the leveling of a distribution in comparison to a normal distribution.

$$L_4 = L_1^{-4} \left[\sum_{\mu=1}^V (P_a - M_2)^4 * N_a \right] \quad (8)$$

Skewness

The degree of asymmetry in the distribution of a certain attribute around the mean. Equation 9 indicates that values might be positive or negative.

$$L_5 = L_2^{-3} \left[\sum_{a=1}^V (P_a - L_2)^3 * N_a \right] \quad (9)$$

Features of GLCM

The GLCM is a statistical method used to study textures by looking at how pixels are arranged in space. GLCM functions examine an image's texture by counting how often pairs of pixels with specific values and positions appear, then creating a matrix called the GLCM. From this matrix, statistical values are calculated. The gray co-occurrence matrix is made by counting how often a pixel with a certain gray level appears next to another pixel with a different gray level in a specific direction. The GLCM matrix has as many rows and columns as there are different gray levels in the image. The value in each cell of the matrix is determined by adding the counts from the rows,

$$S_q(\kappa) = \sum_{\lambda=0}^{num\ grey-1} S(\kappa, \lambda) \text{ and } S_k(\lambda) = \sum_{\kappa=0}^{num\ grey-1} S(\kappa, \lambda) \quad (10)$$

where, (κ, λ) defines total time, pixel with κ can be associated with λ value, $num\ grey$ representing the amount of gray levels, and $S(\kappa, \lambda)$ shows the matrix element. GLCM has six features: correlation, energy, coarseness, homogeneity, contrast, and inverse difference technique.

Energy

Energy is a way to show how evenly the gray levels are spread out, and it is calculated by,

$$L_6 = \sum_{\kappa=1}^{G_h} \sum_{\lambda=1}^{G_h} (S_{\kappa\lambda})^2 \quad (11)$$

where, $S_{\kappa\lambda}$ is the co-occurrence matrix item (κ, λ) , and the image's quantization level is denoted as G_h .

Correlation

This describes the relationship among ordered structures, and if numerous linear characteristics are present in the image, then correlation is high, as stated,

$$L_7 = \sum_{\kappa=1}^{G_h} \sum_{\lambda=1}^{G_h} \frac{(\kappa - L_2)(\lambda - L_2)S_{\kappa\lambda}}{\eta^2} \quad (12)$$

where, L_1 denotes the mean, and η be the standard deviation.

Homogeneity

This measures how uniformly an image's values match up; a homogenous image has a high value; a low value indicates that an image is more random and is represented by,

$$L_8 = \sum_{\kappa=1}^{G_h} \sum_{\lambda=1}^{G_h} \frac{S_{\kappa\lambda}}{1 + |\kappa - \lambda|} \quad (13)$$

Contrast

This demonstrates visual representation with various gray levels. This is indicated by the formula below since it tends to high contrast with greater grey level and vice versa.

$$L_9 = \sum_{\kappa=1}^{G_h} \sum_{\lambda=1}^{G_h} (\kappa - \lambda)^2 \times S_{\kappa\lambda} \quad (14)$$

Coarseness

This refers to image smoothness, whereas coarse structure results in a lower clarity image. This is represented by a term L_{10} .

Inverse Difference Moment (IDM)

The inverse difference moment (IDM) is also referred to as closeness of distribution, and it employs several parameters to assess whether an image possesses texture or not. This metric evaluates uncertainty and value, maintaining a high level when the local gray level is uniform, and is represented as. The system was developed by the author using web-based technology, with the implementation carried out through the Visual Studio Code application, MySQL as the database, CodeIgniter as the framework, and XAMPP as the runtime environment L_{11} .

CNN Features

CNN is made up of four layers: pooling, convolutional, fully connected, and ReLU layers. Blocks are generated by linking a sequence of layers to concentrate on the interference stage [15]. The strata are described below.

Convolutional Layer

This layer performs dot products between filters and associated local regions. D_1 and D_2 do a forward pass of convolution with biases and public weights using an additional protocol. In layer, hidden neuron (i, j) , the conclusion obtained by D_1 is,

$$e'_{i,j} = \sum_{b=0}^{f-1} \sum_{c=0}^{f-1} bia_{b,c} d'_{i+b,j+c} + qt \quad (15)$$

Meanwhile, the output that is produced by D_2 is

$$e''_{i,j} = \sum_{b=0}^{f-1} \sum_{c=0}^{f-1} bia_{b,c} d''_{i+b,j+c} \quad (16)$$

where, $d_{i,j}$ represents input activation in position, filter sizes are $f \times f$ and weights as $bia_{b,c}$ ($b, c = 0, 1, \dots, f-1$)

Fully-Connected

This connects to each layer that computes activation via matrix multiplication, followed by bias off set. Here, identifies a specific component of activation by,

$$e'_i = \sum_j bia_{b,c} d'_j + qt_i \quad (17)$$

Furthermore, D_2 calculate activation by,

$$e'_i = \sum_j bia_{b,c} d'_j \quad (18)$$

where, the term d_j aids in the activation of neuron j in the prior layer. Weights are displayed by $bia_{b,c}$ with qt_i is the bias.

ReLU Layer

When all elements d are shared using D_1 and D_2 , this layer uses function $\max(d, 0)$ thresholding at 0, and it is applied for both D_1 and D_2 based on Media stream Broadcast (MSB) as follows.

$$e'_{i,j} = \begin{cases} d'_{i,j} & \text{if } MSB(d) = 0 \\ 0 & \text{if } MSB(d) = 1 \end{cases} \quad (19)$$

$$e''_{i,j} = \begin{cases} d''_{i,j} & \text{if } MSB(d) = 0 \\ 0 & \text{if } MSB(d) = 1 \end{cases} \quad (20)$$

Pooling Layer

This layer conducts down sampling on spatial parameters, such as width and height. This supports both average and maximum pooling. CNN's output is expressed as L_{12} . **Fig 2** depicts an architectural diagram of CNN.

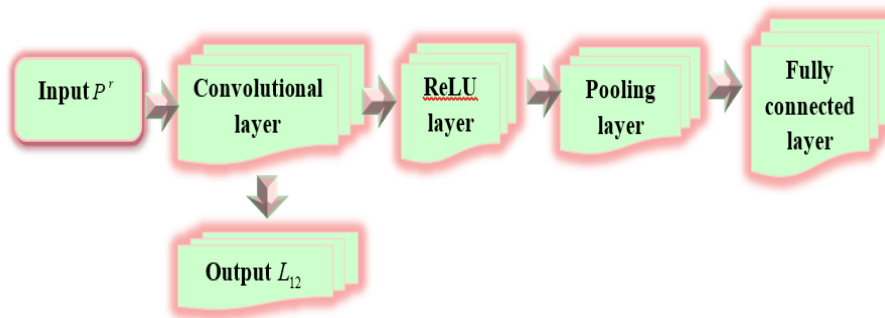


Fig 2. CNN's Feature Architecture.

PHOG Features

PHOG [16], a spatial shape descriptor, has recently been applied to image categorization [17]. It is defined as a vector representation of the spatial distribution of edges. This descriptor is heavily influenced by two sources: (1) the pyramid representation [18], and (2) the Histogram of Orientation Gradients (HOG) [19]. To obtain PHOG, they are concatenated at pyramid resolution and are identified as L_{13} .

All features acquired are represented by a feature vector, which is provided as,

$$B = \{M_1, M_2, M_3, M_4, M_5, M_6, M_7, M_8, M_9, M_{10}, M_{11}, M_{12}, M_{13}\} \quad (21)$$

Augmentation of Data

Data augmentation [20] is carried out by using techniques like rotating, flipping, translating, and cropping the feature vector that is obtained after extracting the features. The main purpose of data augmentation is to enhance the system's performance without needing to collect more data. Let us assume the feature that has been selected with its data dimensions $\beta \times \beta$ that is maximized to $\alpha \times \sigma$, such that $\sigma > \beta$. The content is rephrased on an online platform. In this, the image is enhanced through flipping, rotating, and cropping. For each column, maximum and minimum values are predicted, and the remaining image sample is generated.

Flipping: Flipping an image involves rotating it around its vertical, horizontal, or both axes. It enables users to maximize the number of photos in a collection without requiring any artificial processing. $C_{aug}(1)$ denotes a flipped image.

-Rotation: Image rotation is a type of geometric data augmentation where an image is rotated around an axis with angles ranging from 1 to 359. Images can be rotated at a specific angle using an additive method. This system was developed by the author using a web-based approach with Visual Studio Code as the application, MySQL as the database, CodeIgniter as the framework, and XAMPP to run the application. $C_{aug}(2)$ indicates a rotated image.

-Cropping: Cropping is also called "zooming" or "scaling" in scientific research. It is a method where an image is made bigger. This kind of image data adjustment has two different methods. The first method is cutting an image from one part to another. $C_{aug}(3)$ indicates a cropped image.

-Scaling: This is how the original image is made bigger using two different ways. First, the image is split from one place to another. Second, the image is resized back to its original size. The way the resized image looks is shown like this:

$$\begin{bmatrix} U_t \\ U_y \end{bmatrix} = \begin{bmatrix} Xscale & 0 \\ 0 & Yscale \end{bmatrix} \begin{bmatrix} t \\ y \end{bmatrix} \quad (22)$$

where, the image's position is denoted as X, Y , U_t, U_y be the scaled pixel's new coordinates, and t, y are the coordinates of original point on the image. $C_{aug}(4)$ represents a rotated image.

-Translation: Moving an object from one place to another is called translation. To keep the image the same after moving it, it stays either black or white and moves at the same time in both directions. This method can cause important parts of the image to be lost during the process, so it's better to do it by hand. This affects how images are placed in data and is shown in the formula as,

$$\begin{bmatrix} U_t \\ U_y \end{bmatrix} = \begin{bmatrix} t \\ y \end{bmatrix} + \begin{bmatrix} T_{rans(t)} \\ T_{rans(y)} \end{bmatrix} \quad (23)$$

where, new coordinates, denoted by U_t, U_y , the coordinates of raw position on image is denoted by t, y , $T_{rans(t)}, T_{rans(y)}$ be the image translated pixels. Here, the translated image is labelled as $C_{aug}(5)$.

The set of augmented images is shown below,

$$C_{aug} = \{C_{aug}(1), C_{aug}(2), C_{aug}(3), C_{aug}(4), C_{aug}(5)\} \quad (24)$$

where, $C_{aug}(1)$, $C_{aug}(2)$, $C_{aug}(3)$, $C_{aug}(4)$ and $C_{aug}(5)$ are the flipped, rotated, cropped, scaled and translated image. In order to classify the E-commerce products, LeNet receives the augmented data output C_{aug} .

LeNet Structure

The LeNet [21] is a great example of the first CNN architecture. It is simple and easy to understand, yet large enough to lead to interesting results. LeNet includes neurons that have weights and biases that can be learned. Each neuron takes in many inputs, adds them up with weights, applies an activation function, and gives an output. The structure of CNNs uses spatial relationships to reduce the number of parameters needed. LeNet has five layers, including two convolutional layers and three fully connected layers.

Convolutional layer: The convolutional layer is the key part of a CNN

It contains several filters that work independently. Each filter creates a feature map by applying it to the image. If the resulting image after convolution has a certain size, it helps in extracting important features. $O_{sj} \times Q_{sp}$ and a filter size of $u \times v$, the output feature map has a size of $Z_u \times Z_v$, and is defined as,

$$Z_u = \frac{O_{sj} - u + 2W_u}{K_u} + 1 \quad (25)$$

$$Z_v = \frac{Q_{sp} - v + 2W_v}{K_v} + 1 \quad (26)$$

where, K_u and K_v represent the horizontal and vertical strides, and W_u and W_v represent the padding of zeros in the width and height.

With the inclusion of bias, linear filter, and nonlinear function, each output feature map is obtained by convolution of the input maps.

$$EF_{js}^{zv} = M_s \left(\sum_{uo \in \text{TN}_{sr}} EF_{uo}^{zv-1} * Of_{uo,js}^{zv} + bias_{js}^{zv} \right) \quad (27)$$

If $bias_{js}$ represents bias, zv is the layer number, TN denotes the input map group, and $M_s(\cdot)$ is the activation function.

This function helps in learning and doing difficult tasks. These functions aren't straight-line changes that can be used as input here. It shows if a neuron is getting the right information. ReLU is a special kind of function that moves errors in the opposite direction. Many neurons get activated with ReLU. The best part of ReLU is that it uses fewer neurons, making the network less complicated, easier to work with, and faster. The ReLU equation is written as,

$$M_s(EF) = \begin{cases} EF, & EF \geq 0 \\ 0, & EF < 0 \end{cases} \quad (28)$$

The first convolutional layer considers several low-level features, including lines, edges, and corners. The network learns global characteristics by stacking multiple convolutional layers.

Pooling layer: This limits the spatial scale of the visualization and the number of parameters used in the network calculations. The system was created by the author using a web-based approach with Visual Studio Code as the development tool, MySQL as the database, CodeIgniter as the framework, and XAMPP for execution. Spatial pooling includes methods such as max, average, and min. When maxpooling is used for spatial pooling, the highest value in the feature map is selected. Average pooling considers the average of the values. Maxpooling provides good results because it reduces computation in the top layer by eliminating non-maximal values and allows for translational variance. It is the most effective method for reducing the dimensionality of intermediate representations, as it provides greater location robustness.

Fully connected layers: This refers to a neuron in the fourth layer that is connected to all neurons in the current layer. Activations are performed using matrix multiplication followed by a bias offset. The neurons in the final fully connected layer match the number of classes that needed to be identified. Features from subsampling and convolutional layers are useful for classification, but feature integration is more effective. Thus, the fully connected layer includes all the features from the previous convolutional and subsampling layers. Finally, the fully connected layer uses softmax activation, a global logistic function designed for multiclass classification. As a result, the outcome of LeNet is achieved, and the LeNet architecture is illustrated in **Fig 3**.

LeNet Training using SSWSO

The SSWSO approach for finding products on e-commerce websites is discussed in this section. WSO [23] is a type of algorithm that works like a war strategy. In WSO, soldiers are placed randomly on a battlefield, and the process keeps going on continuously. Two main tactics are used for optimization: one is an attack tactic that depends on the king and commander's positions, and the other is a defense tactic where other soldiers help protect. WSO is straightforward, but it has some issues with diversity. So, the SSA technique is combined with WSO, which makes the process better and uses less computer power. SSA [22] was made based on how squirrels find food. It also has a good way of moving called gliding.

Squirrels look for food by gliding between trees during warm weather like fall. They explore different parts of the forest. Because of the hot weather, they need acorns to stay energized and eat them as soon as they find them. After meeting their daily energy needs, they look for the best food for winter: hickory nuts. Storing hickory nuts helps keep their energy up during tough weather, reduces the number of times they need to search for food, and improves their chances of surviving. In deciduous forests, the lack of leaves in winter makes them more vulnerable to predators, so they are less active but don't hibernate. Flying squirrels become active again at the end of winter. This whole process repeats throughout their lives and forms the basis of the SSA. SSA finds the best possible solutions quickly and accurately. It is more effective and reliable and gives better results for real-life problems.

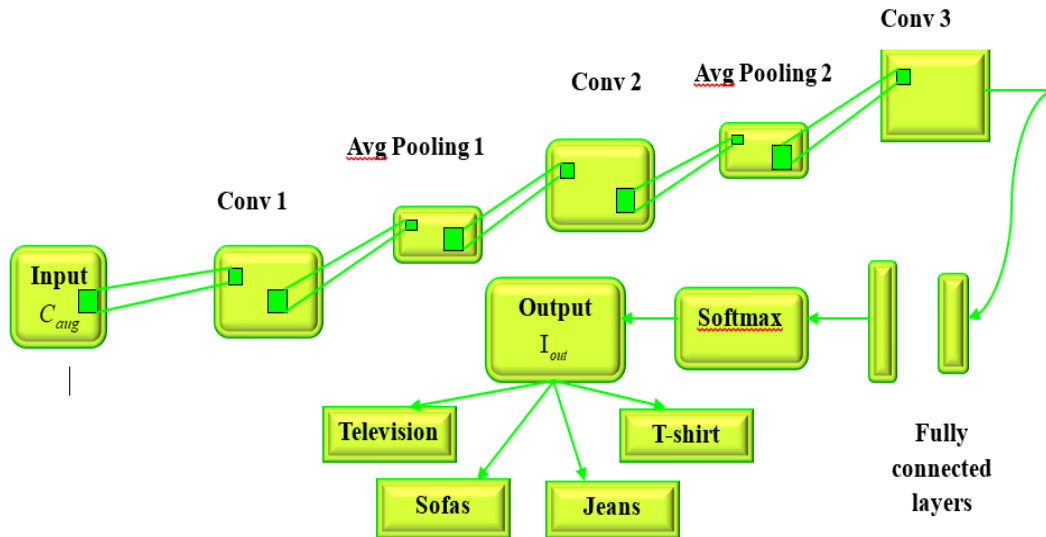


Fig 3. Architecture of LeNet.

The algorithmic phases of SSWSO are described in depth below.

Initialization

Consider the following: Flying squirrels W in V woodland, where the flying squirrel's principal position is shown by,

$$HA_j = HA_Y + Z(0,1) \times (HA_Z - HA_Y) \quad (29)$$

where, HA_Y denotes the flying squirrel's lowest bound in a χ^{th} dimension. $Z(0,1)$ represents the evenly distributed random integer among 0 and 1, HA_j represents the flying squirrel's position, and HA_Z represents the upper bound of the flying squirrel's j in dimension χ .

Calculating the Fitness Function

The fitness of each flying squirrel's position is calculated using a user-defined fitness function based on MSE. The best ideal value is found by identifying the fitness measure with the lowest value, which is provided by,

$$Err = \frac{1}{O} \sum_{z=1}^O [I_{out} - I_{out}^*] \quad (30)$$

where, the symbol O stands for the total number of image samples, I_{out} denotes the categorized output of LeNet, I_{out}^* denotes target output, and Err represents fitness measure.

Declaration, Sorting, and Random Selection

The fitness values of each flying squirrel's position are stored and sorted in ascending order. The flying squirrel with the lowest fitness value is found on hickory nut trees. The top three flying squirrels are located on acorn nut trees and are

believed to migrate towards hickory nut trees. The remaining flying squirrels are expected to be developed by the author using a web-based approach with Visual Studio Code and MySQL as the database, CodeIgniter as the framework, and XAMPP for execution. These remaining squirrels forage on acorn nut trees to meet their daily energy needs. The foraging behavior of flying squirrels is permanently affected by predators. The natural behavior is modeled using a location update mechanism that incorporates the probability of predator presence γ .

Identify New Solution

The following three conditions are based on the foraging behavior of squirrels. When there are no predators around, flying squirrels actively search for food across the forest. However, when predators are present, they become cautious and take short, random walks to a nearby hiding spot. The dynamic foraging behavior is modeled as follows.

This system was developed by the author using a web-based approach.

It was created with Visual Studio Code as the development environment, MySQL as the database, CodeIgniter as the framework, and XAMPP to run the application. No other tools or systems were used. HA_{mz} to hickory nut trees. This is demonstrated by the following scenario:

$$HA_{mz}^{z+1} = \begin{cases} HA_{mz}^z + \nu_u \times R_l (HA_{gz}^z - HA_{mz}^z); & X_1 \geq \gamma \\ \text{Random position} & ; \text{Otherwise} \end{cases} \quad (31)$$

where, ν_u be the distance of random gliding, X_1 is a random number ranging from 0 to 1, the flying squirrel's location when it arrived at the hickory nut tree is shown by HA_{gz}^z , and the current iteration is indicated by Z . The mathematical model uses the gliding constant R_l to establish a balance of exploration and exploitation, where the value considered is 1.9.

Case 2: Flying squirrels on conventional trees HA_{cz} may migrate to acorn nut trees to meet their energy demands. The new squirrel location can be acquired as follows:

$$HA_{cz}^{z+1} = \begin{cases} HA_{cz}^z + \nu_u \times R_l (HA_{mz}^z - HA_{cz}^z); & X_2 \geq \gamma \\ \text{Random position} & ; \text{Otherwise} \end{cases} \quad (32)$$

where, the symbol X_1 is the random number in the range $[0,1]$.

Case 3: Squirrels that have already consumed acorn nuts may travel to hickory nut trees to store them for future consumption during food scarcity. The new squirrel location can be acquired as follows:

$$HA_{cz}^{z+1} = \begin{cases} HA_{cz}^z + \nu_u \times R_l (HA_{gq}^z - HA_{cz}^z); & X_3 \geq \gamma \\ \text{Random position} & ; \text{Otherwise} \end{cases} \quad (33)$$

where, X_1 be the random number in range $[0,1]$.

To improve product classification in e-commerce, the updated attack strategy equation from WSO is combined with equation (33) in SSA. In the WSO attack strategy, the monarch stays in a strong position to launch big attacks. Here, a soldier with a strong attack is considered the king. A soldier who wins a battle gets promoted. Each soldier's position is adjusted based on their attack performance, and the positions of the army leader, king, and soldiers stay close to the target. Therefore, the standard update equation for WSO is as follows,

$$HA_{cz}^{z+1} = HA_{cz}^z + 2 \times m \times (J - Y) + rand \times (N_{cz} \times Y - HA_{cz}^z) \quad (34)$$

$$HA_{cz}^{z+1} = HA_{cz}^z + 2 \times m \times (J - Y) + rand \times N_{cz} \times Y - rand HA_{cz}^z \quad (35)$$

$$HA_{cz}^{z+1} = HA_{cz}^z (1 - rand) + 2 \times m \times (J - Y) + rand \times N_{cz} \times Y \quad (36)$$

$$HA_{cz}^z (1 - rand) = 2 \times m \times (J - Y) + rand \times N_{cz} \times Y - HA_{cz}^{z+1} \quad (37)$$

$$HA_{cz}^z = \frac{2 \times m \times (J - Y) + rand \times N_{cz} \times Y - HA_{cz}^{z+1}}{1 - rand} \quad (38)$$

$$\text{Here, } N_{cz} = N_{cz} \times \left(1 - \frac{B_{cz}}{Max_iter}\right)^\sigma \quad (39)$$

where, HA_{cz}^{z+1} and HA_{cz}^z denotes the new and previous location, N_{cz} represents the weight, m symbolizes the strategy, $rand$ is the random value, king's position is represented as Y , commander's position is denoted as J , B_{cz} is the soldier's rank, and σ is factor.

Rearrange equation (33),

$$HA_{cz}^{z+1} = HA_{cz}^z + \nu_u R_l HA_{gq}^z - \nu_u R_l HA_{cz}^z \quad (40)$$

$$HA_{cz}^{z+1} = HA_{cz}^z (1 - \nu_u R_l) + \nu_u R_l HA_{gq}^z \quad (41)$$

Substitute equation (38) in (41),

$$HA_{cz}^{z+1} = \left[\frac{2 \times m \times (J - Y) + rand \times N_{cz} \times Y - HA_{cz}^{z+1}}{1 - rand} \right] (1 - \nu_u R_l) + \nu_u R_l HA_{gq}^z \quad (42)$$

$$HA_{cz}^{z+1} + \frac{HA_{cz}^{z+1}}{1 - rand} (1 - \nu_u R_l) = \left[\frac{2 \times m \times (J - Y) + rand \times N_{cz} \times Y}{1 - rand} \right] (1 - \nu_u R_l) + \nu_u R_l HA_{gq}^z \quad (43)$$

$$HA_{cz}^{z+1} \left(\frac{(1 - rand) + (1 - \nu_u R_l)}{1 - rand} \right) = \left[\frac{2 \times m \times (J - Y) + rand \times N_{cz} \times Y}{1 - rand} \right] (1 - \nu_u R_l) + \nu_u R_l HA_{gq}^z \quad (44)$$

$$HA_{cz}^{z+1} = \left[\frac{2 \times m \times (J - Y) + rand \times N_{cz} \times Y}{1 - rand} (1 - \nu_u R_l) + \nu_u R_l HA_{gq}^z \right] \left(\frac{1 - rand}{(1 - rand) + (1 - \nu_u R_l)} \right) \quad (45)$$

Substitute equation (43) in (33),

$$HA_{cz}^{z+1} = \begin{cases} \left[\frac{2 \times m \times (J - Y) + rand \times N_{cz} \times Y}{1 - rand} (1 - \nu_u R_l) + \nu_u R_l HA_{gq}^z \right] \left(\frac{1 - rand}{(1 - rand) + (1 - \nu_u R_l)} \right); X_3 \\ \text{Random position} \end{cases} ; \text{Otherwise} \quad (46)$$

The above equation represents the updated equation of SSWSO, which is applied for E-Commerce product classification.

Aerodynamics of Gliding

Flying squirrels employ an equilibrium glide, where the combined lift and drag forces equal the force opposing their weight. This glide is achieved through a controlled flight path, maintaining a consistent speed. The glide ratio, which is the lift-to-drag ratio, is determined by the angle of the flight path relative to the horizontal. The system was developed using a web-based approach, with Visual Studio Code as the coding environment, MySQL as the database, CodeIgniter as the framework, and XAMPP for running the application.

Seasonal Constant Estimation

Seasonal variations significantly influence the foraging behavior of flying squirrels. Due to their high body temperature and small size, they suffer from substantial heat loss during colder periods. Foraging is energy-intensive and risky due to predation. As a result, flying squirrels are less active in winter compared to autumn, influenced by climatic conditions. Changes in weather affect their movement patterns, making optimization more challenging. To address this, a seasonal monitoring condition has been integrated into SSA, preventing the method from becoming trapped in local optima.

The flying squirrel's seasonal constant is calculated by,

$$HA_l^z = \sqrt{\sum_{k=1}^{\Re} (HA_{mz,k}^z - HA_{gz,k})^2} \quad (47)$$

Here, $z = 1, 2, 3$.

Furthermore, the minimum seasonal constant value is estimated as follows:

$$HA_{\min} = \frac{10E^{-6}}{(365)^{z/(z_k/2.5)}} \quad (48)$$

where, Z_k and Z represents maximum and current iteration. If the seasonal monitoring scenario is actual, then randomly rearrange those flying squirrels that could not find the forest for the best winter food source.

Random Migration Around the End of the Winter Season

Flying squirrels become more active as winter ends due to the reduced cost of foraging. Those that couldn't find sufficient food in the forest during winter may now search for new areas. Incorporating this behavior into modeling can enhance the exploration capabilities of the proposed approach. This system was developed by the author using web-based technology, with Visual Studio Code as the application, MySQL as the database, CodeIgniter as the framework, and XAMPP to run the application. The following equation models the relocation behavior of flying squirrels.

$$HA_{cz}^{new} = HA_{\ell} + \chi(\omega) + (HA_O - HA_P) \quad (49)$$

where, $\chi(\omega)$ denotes the Levy distribution, which promotes effective and efficient search space exploration.

$$\chi(\omega) = 0.01 \times \frac{j_s \times \mu}{|j_r|^{1/\varphi}} \quad (50)$$

$$\mu = \left[\frac{\xi(1+\varphi) \times \sin\left(\frac{\pi\varphi}{2}\right)}{\xi\left(\frac{1+\varphi}{2}\right) \times \varphi \times 2^{\left(\frac{\varphi-1}{2}\right)}} \right]^{1/\mu} \quad (51)$$

$$\zeta(u) = (u-1)! \quad (52)$$

where, j_s and j_r denotes the random numbers, which are distributed normally, ranging from 0 to 1. The term φ is a constant value, where the value considered here is 1.5.

Check the Feasibility of the Solution

The optimal solution is estimated using the fitness measure described in equation (30). If a new solution outperforms the prior solution, the previous value is updated to reflect the new solution.

Termination

All preceding processes are repeated until the best possible solution for lung cancer detection is achieved. Algorithm 1 describes the pseudo-code of the SSWSO algorithm.

Algorithm 1. SSWSO's Pseudocode

Input: HA_j
Output: The best solution
Start
Generate HA 's random location
Use equation (30) to estimate the fitness measure
Arrange the HA 's location
Declare flying squirrels on hickory nuts, normal trees, and acorn nuts
Choose squirrels on regular trees randomly
While (stopping condition is not satisfied)
for ($z = 1 to g_1$); g_1 - flying squirrels on acorn trees
if $X_1 \geq \gamma$ X_1 is a random number
$HA_{mz}^{z+1} = HA_{mz}^z + v_u \times R_l (HA_{gz}^z - HA_{mz}^z)$
else
HA_{mz}^{z+1} is equal to search space location randomly
end
end
for ($z = 1 to g_2$); g_2 - squirrels flying on normal trees
if $X_2 \geq \gamma$ X_2 is random integer
$HA_{cz}^{z+1} = HA_{cz}^z + v_u \times R_l (HA_{mz}^z - HA_{cz}^z)$
else
HA_{cz}^{z+1} is equal to search space random location
end
end
for ($z = 1 to g_3$); g_3 is normal trees flying squirrels
if $X_3 \geq \gamma$ X_3 is random integer
Identify the location using equation (46)
else
HA_{cz}^{z+1} is equal to random location of search space
end
end
Estimate seasonal constant
If $HA_j^z < HA_{min}$
Flying squirrel are relocate randomly using equation (48)
end
Update HA_{min}
end
Return best solution
End

By using both the WSO and SSA methods together, the performance of E-Commerce product classification improves a lot. Also, using the LeNet classifier helps lower the cost of computation, though it takes more time to train. Because of this, LeNet is able to correctly classify E-Commerce products into categories like T-Shirts, Sofas, Jeans, and Television.

IV. RESULTS AND DISCUSSION

By changing the K-fold and training data, the effectiveness of SSWSO_LeNet is evaluated in terms of specificity, accuracy, and sensitivity.

Experimental Setup

The SSWSO_LeNet runs in Matlab on a Windows 10 operating system with 2GB RAM and an Intel core processor.

Dataset Description

The E-Commerce Product Classification Dataset [29] is used to classify products online with high accuracy. It includes many images of different products along with labels that describe each product. The dataset has 796 files grouped by product types such as televisions, sofas, jeans, and t-shirts. All the images come from various websites and are raw, unstructured data collected from the web. The performance metrics used for SSWSO_LeNet are accuracy, sensitivity, and specificity.

Accuracy

It shows how close the predicted results are to the actual results in E-Commerce product classification and is expressed as a percentage,

$$Acc = \frac{T^{pos} + F^{neg}}{T^{pos} + T^{neg} + F^{pos} + F^{neg}} \quad (53)$$

where, T^{pos} , and T^{neg} signifies true positive, and negative, F^{pos} , and F^{neg} denotes the false positive, and negative.

Sensitivity: It is expressed as the ratio of positives accurately detected by the E-Commerce product classification method.

$$Sen = \frac{T^{pos}}{T^{pos} + F^{neg}} \quad (54)$$

Specificity: It expresses the precise ratio of negatives identified using SSWSO_LeNet for E-Commerce product classification.

$$Spe = \frac{T^{neg}}{T^{neg} + F^{pos}} \quad (55)$$

Experimental Outputs

Fig 4 illustrates the experimental outcomes of the SSWSO_LeNet model using a collection of original photographs. **Fig 4a)** displays the original input images, which consist of various E-Commerce product images. **Fig 4b)** presents the output E-Commerce product image, including items such as Television, Sofas, Jeans, and T-Shirts. The system was developed by the author using a web-based approach with Visual Studio Code as the application, MySQL as the database, CodeIgniter as the framework, and XAMPP for execution.

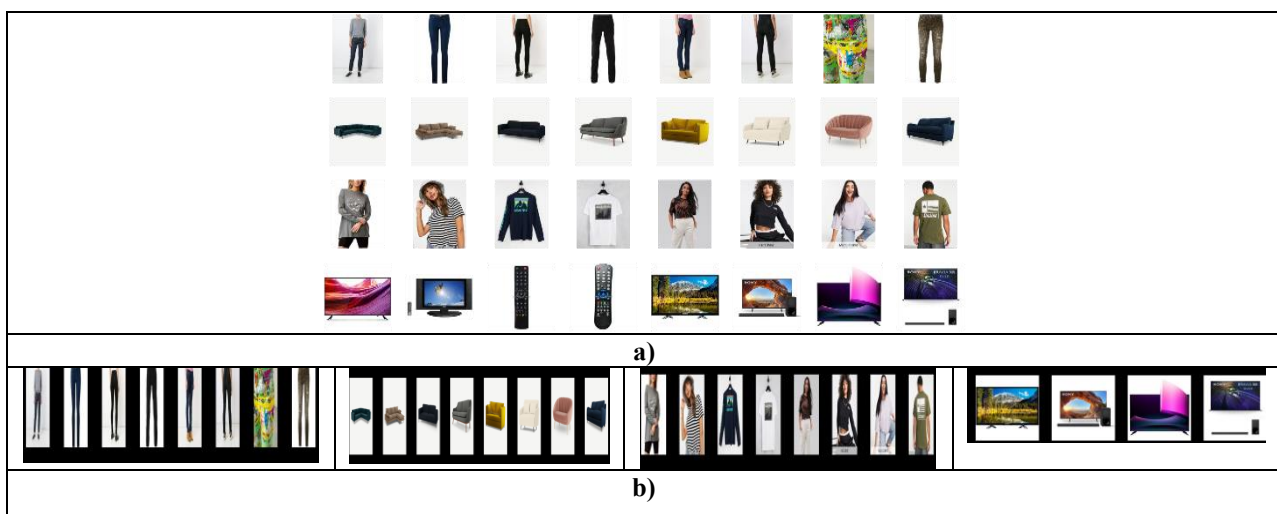


Fig 4. Experimental Results Of SSWSO_Lenet A) Original Input Image, B) Classified Outputs.

Evaluation of Achievement

The following **Fig 5** and **Fig 6** explain how training data and the K-fold are changed to evaluate the generated SSWSO_LeNet utilizing specificity, sensitivity, and testing accuracy.

Performance Evaluation of Training data

Fig 5a shows how the performance of SSWSO_LeNet was checked using accuracy after changing the training data many times. The SSWSO_LeNet had an accuracy of 0.791 after 10 iterations, 0.814 after 20 iterations, 0.836 after 30 iterations, and 0.869 after 40 iterations when using 60% of the training data. Compared to other iterations and training data, the highest accuracy for SSWSO_LeNet was 0.875 at the 70th iteration, meaning more iterations led to better performance. When using 80% of the training data, SSWSO_LeNet had accuracy values of 0.801, 0.834, 0.856, and 0.877 after 10, 20, 30, and 40 iterations respectively. When using 90% of the training data, the accuracy values were 0.833, 0.862, 0.874, and 0.917 for 10, 20, 30, and 40 iterations. Accuracy went up as the number of iterations increased. **Fig 5b** shows the sensitivity evaluation. The training data percentage goes from 60 to 90, and sensitivity values go from 0 to 0.6. Sensitivity went down as the number of iterations and training data increased, showing that the approach reduces error. When using 60% of the training data, SSWSO_LeNet with 10, 20, 30, and 40 iterations had sensitivity values of 0.771, 0.784, 0.816, and 0.859 respectively. For 70% of the training data, sensitivity values for SSWSO_LeNet with different iterations were 0.790, 0.829, 0.854, and 0.881. For 80% of the training data, sensitivity values for SSWSO_LeNet with 10, 20, 30, and 40 iterations were 0.792, 0.809, 0.832, and 0.865. With the 90th iteration, sensitivity values for 10, 20, 30, and 40 iterations were 0.839, 0.856, 0.866, and 0.927. The performance analysis shows that as the number of iterations increases, sensitivity values go up for sorting E-Commerce products. **Fig 5c** shows a specificity evaluation. Specificity values go from 0 to 0.9, and training data ranges from 60 to 90. SSWSO_LeNet's specificity for E-Commerce product sorting with various iterations is 0.739, 0.747, 0.766, and 0.809 when using 60% of the training data. At iteration 10 with 70% of the training data, the lowest specificity of 0.701 was achieved, while iteration 40 had the best specificity of 0.799. More iterations and training data give better specificity for sorting E-Commerce products. For 80% of the training data, SSWSO_LeNet with 10, 20, 30, and 40 iterations had specificity values of 0.753, 0.772, 0.787, and 0.819. Changing the number of iterations improved the specificity value. Without proper product classification, many customers are not happy and their expectations are not met, which leads to fewer sales.

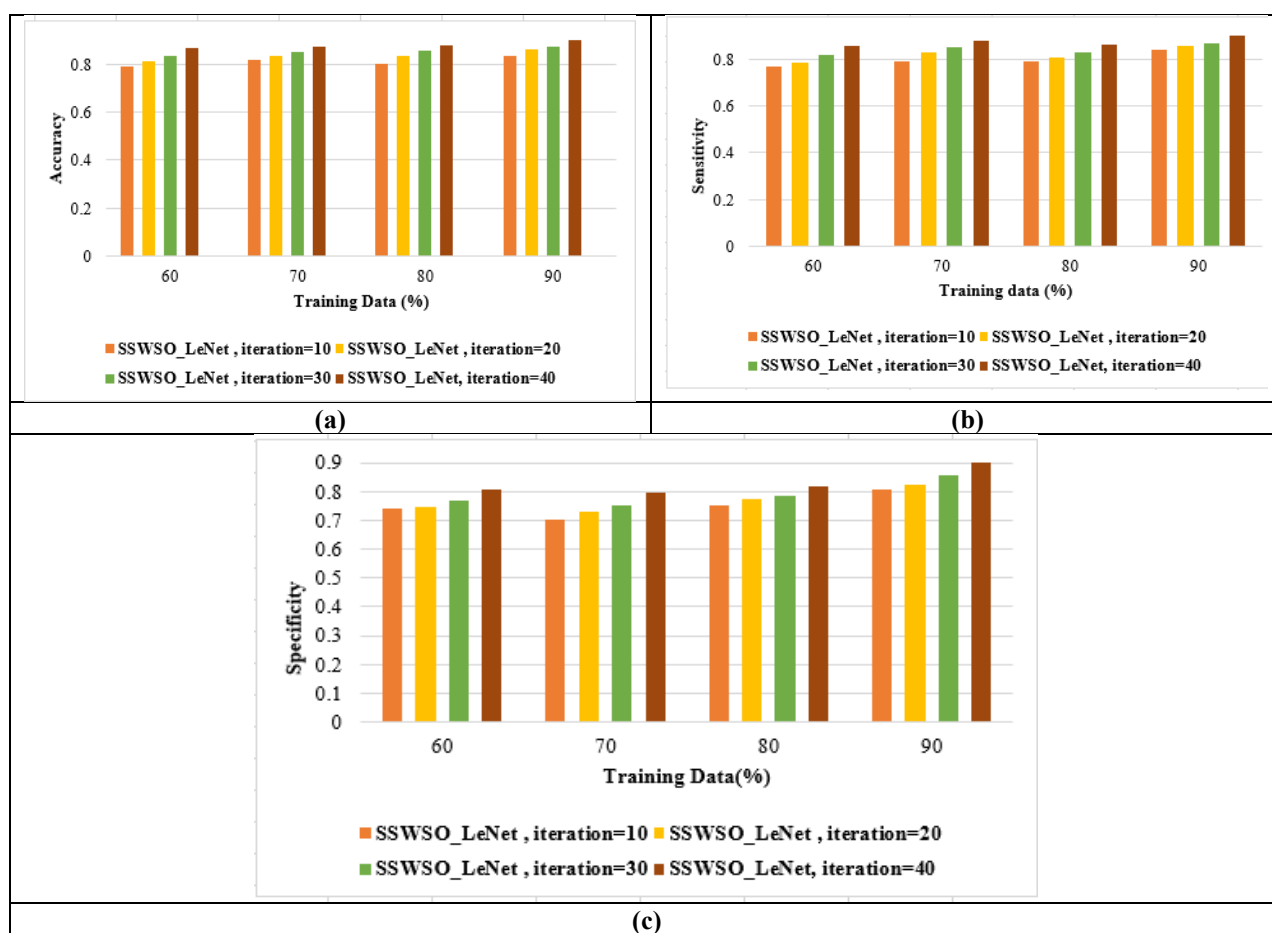


Fig 5. Evaluation of the SSWSO_LeNet Through Varying Training Data Utilizing a) Accuracy b) Sensitivity c) Specificity.

Performance Evaluation of K-Fold

Fig 6a) shows how well the SSWSO_LeNet model performs based on accuracy. When using 10, 20, 30, and 40 iterations, the accuracy for K-fold=5 is 0.767, 0.789, 0.796, and 0.829 respectively. For K-fold=6, the accuracy values with the same iterations are 0.776, 0.794, 0.811, and 0.835. **Fig 6b)** shows the sensitivity of the SSWSO_LeNet model. The sensitivity values for K-fold=5 with iterations 10, 20, 30, and 40 are 0.756, 0.778, 0.785, and 0.810. For K-fold=8, the sensitivity values are 0.821, 0.833, 0.863, and 0.906 with the same iterations. In addition to improving the LeNet-based model, the SSWSO_LeNet model also improves the ability to classify E-Commerce products. This means the product categorization system can handle noisy data and make accurate predictions, which is very useful. **Fig 6c)** shows how well the SSWSO_LeNet model performs based on specificity. Specificity ranges from 0 to 0.9, depending on the K-fold value, which ranges from 5 to 8. For K-fold=5, the specificity values for iterations 10, 20, 30, and 40 are 0.628, 0.656, 0.675, and 0.718. The lowest specificity is 0.683 when K-fold=6 and iteration=10, while the highest specificity is 0.768 with K-fold=6 and iteration=40. As iterations and K-fold values increase together, specificity improves, allowing for faster and more accurate classification of E-Commerce products. For K-fold=8, the specificity values after iterations 10, 20, 30, and 40 are 0.739, 0.755, 0.775, and 0.832. As the number of iterations increases, specificity also improves, which leads to better product classification and helps users work more efficiently and communicate better.

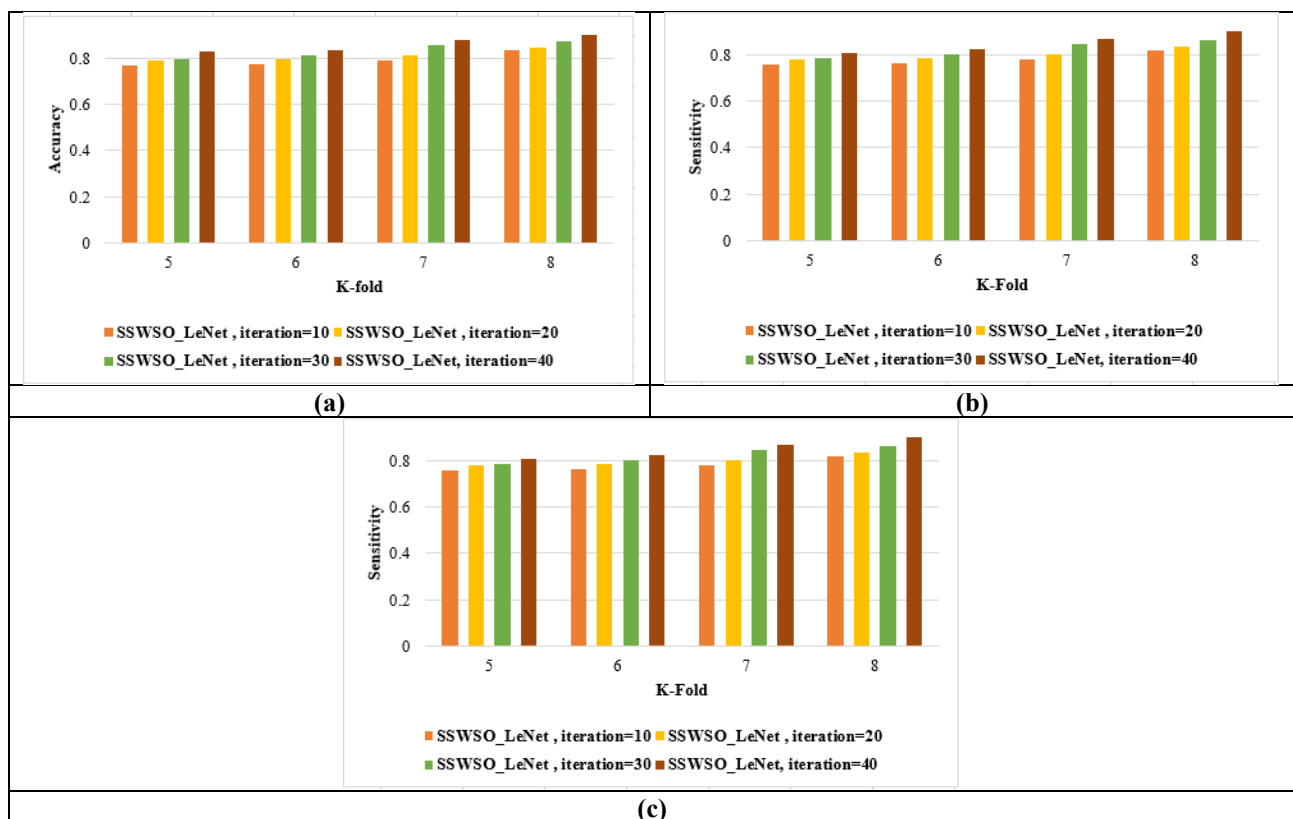


Fig 6. Evaluation of the SSWSO_LeNet Through K-Fold a) Accuracy b) Sensitivity c) Specificity.

Comparative Methods

The performance gain of SSWSO_LeNet is assessed by contrasting it with existing techniques, such as AML-LBP-DBN [4], TL [7], ARCM-GF-E-CNN [8].

Comparative Assessment with Training Data

Fig 7 compares the performance of the SSWSO_LeNet model, where the percentage of training data was changed to find accuracy, sensitivity, and specificity. **Fig 7a** shows the accuracy results based on different training data percentages. When the training data was 60%, the accuracy of SSWSO_LeNet was 0.847, while the accuracy of standard methods like LBP-DBN, TL, and ARCM-GF-E-CNN were 0.782, 0.815, and 0.834, respectively. The SSWSO_LeNet showed a 6.5%, 3.2%, and 1.3% improvement in accuracy compared to these methods. When the training data was 70%, SSWSO_LeNet had the highest accuracy of 0.862, while LBP-DBN had the lowest at 0.795. As the training data percentage increased, the accuracy of SSWSO_LeNet also increased, but the performance remained fairly stable. The E-Commerce product classification model performs best when the training data is 90%, which ensures the results of further experiments are reliable. TL and ARCM-GF-E-CNN can also improve classification of E-Commerce products. SSWSO_LeNet is useful for E-Commerce product classification because it uses a technique based on training data percentage, leading to high accuracy.

Fig 7b shows the sensitivity graph after adjusting the training data percentage. At 60% training data, LBP-DBN, TL, and ARCM-GF-E-CNN had sensitivity values of 0.671, 0.704, and 0.723, while SSWSO_LeNet had a value of 0.756. The

highest sensitivity value shows the best performance. At 70% training data, the sensitivity values of LBP-DBN, TL, ARCM-GF-E-CNN, and SSWSO_LeNet were 0.684, 0.724, 0.736, and 0.771, respectively. ARCM-GF-E-CNN performed second best for improving E-Commerce product classification compared to SSWSO_LeNet because it had the highest sensitivity. At 90% training data, the sensitivity values for LBP-DBN, TL, ARCM-GF-E-CNN, and SSWSO_LeNet were 0.783, 0.802, 0.844, and 0.877, respectively.

Fig 7c presents the results of the specificity-based assessment. This study found that SSWSO_LeNet performed better than the other three classic product classification algorithms. At 70% training data, the specificity values for the different algorithms were 0.684, 0.724, 0.736, and 0.771, respectively. Because of its low performance on datasets and limited ability to integrate parameters into the classification model for E-Commerce products, the ARCM-GF-E-CNN method could have performed better overall. As a result, SSWSO_LeNet improves the performance of the product classification model by including a set of parameters and a random location similar to squirrels. This improves the system performance for product classification in E-Commerce for modern consumers shopping from home.

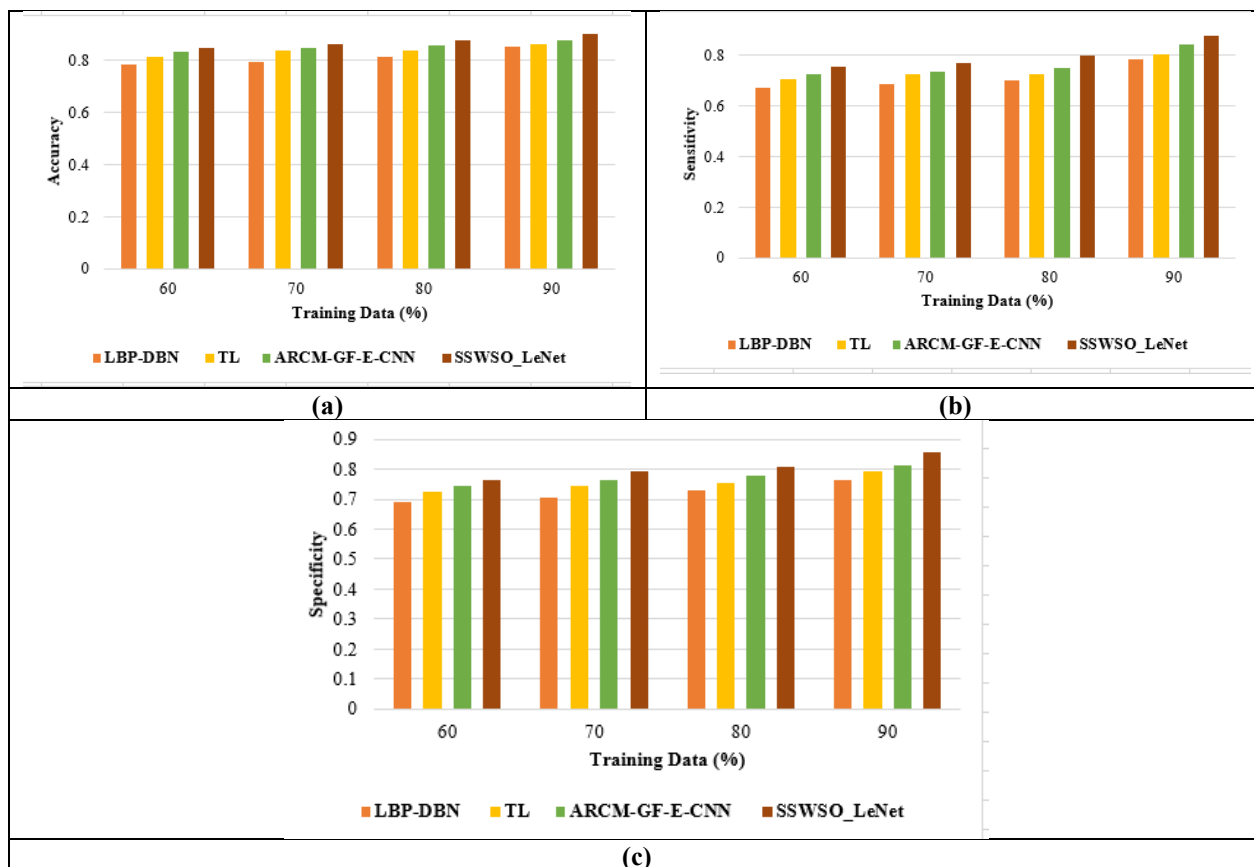


Fig 7. Evaluation of SSWSO_LeNet's a) Accuracy, b) Sensitivity, and c) Specificity.

Comparative Discussion

Table 2 shows how well the SSWSO_LeNet model performed based on different measures. The model worked best when the percentage of training data was set to 90. Its accuracy, sensitivity, and specificity were 0.976, 0.877, and 0.857, respectively. Other methods, like LBP-DBN, TL, and ARCM-GF-E-CNN, had lower results: their accuracy was 0.854, 0.863, and 0.875; their sensitivity was 0.783, 0.802, and 0.844; and their specificity was 0.763, 0.792, and 0.814. The hybrid model, which uses both SSA and WSO, improved the performance of SSWSO_LeNet. Product categorization is a basic part of online shopping, making it easier for users to navigate e-commerce websites. SSWSO_LeNet did better than ARCM-GF-E-CNN, especially when training data was higher. This shows that using random positions in the search space during optimization can improve classification results. Accurate categorization also makes shopping better for customers and helps merchants find the right buyers. Even small changes can greatly affect how well products are classified, even if SSWSO_LeNet's accuracy with 80% training data is close to that with 90%. The hybrid model generally performs better than others in most cases. Both ARCM-GF-E-CNN and TL don't reach the highest accuracy, meaning they often misidentify bad examples. With SSWSO_LeNet, customers can save time when looking for products. They can quickly go through many items and buy what they need.

Table 2. Comparative Discussion

Metrics	LBP-DBN	TL	ARCM-GF-E-CNN	SSWSO_LeNet
Accuracy	0.854	0.863	0.875	0.976
Sensitivity	0.783	0.802	0.844	0.877
Specificity	0.763	0.792	0.814	0.857

V. CONCLUSION

As the internet and online shopping grow, new shopping websites keep appearing. People using these sites get information through pictures and text. Pictures are better at showing product features and help users find the right items more easily than just reading text. This paper talks about a new method called SSWSO_LeNet that helps classify images on e-commerce sites. This method can accurately identify products based on pictures, helping users find what they need quickly. To prepare the images, they use two filters: RoI and Adaptive Wiener. This help clean up the images by reducing unwanted effects or making important parts clearer. Then, they use data augmentation to make more copies of each image. This makes the dataset bigger and helps the model work better in different situations. For the classification part, they use LeNet, which is good because it has a convolution layer with a stride of two and an average pooling layer with a stride of one. The results showed that SSWSO_LeNet was better than other models in accuracy, sensitivity, and specificity, with scores of 0.976, 0.877, and 0.857. In the future, this method could be used in other areas like health, law, and social media. With a few changes, it might also help with tasks like computer vision, audio processing, and analyzing time series data.

CRedit Author Statement

The authors confirm contribution to the paper as follows:

Conceptualization: Jyothi P, Divya Kumari Tankala, Rajesh Kumar A, Nagarjuna Reddy S, Nagendar Yamsani and Jyotsna Devi Kosuru S N V; **Methodology:** Jyothi P, Divya Kumari Tankala and Rajesh Kumar A; **Software:** Nagarjuna Reddy S, Nagendar Yamsani and Jyotsna Devi Kosuru S N V; **Data Curation:** Jyothi P, Divya Kumari Tankala and Rajesh Kumar A; **Writing- Original Draft Preparation:** Jyothi P, Divya Kumari Tankala, Rajesh Kumar A, Nagarjuna Reddy S, Nagendar Yamsani and Jyotsna Devi Kosuru S N V; **Visualization:** Jyothi P, Divya Kumari Tankala and Rajesh Kumar A; **Investigation:** Nagarjuna Reddy S, Nagendar Yamsani and Jyotsna Devi Kosuru S N V; **Supervision:** Jyothi P, Divya Kumari Tankala and Rajesh Kumar A; **Validation:** Nagarjuna Reddy S, Nagendar Yamsani and Jyotsna Devi Kosuru S N V; **Writing- Reviewing and Editing:** Jyothi P, Divya Kumari Tankala, Rajesh Kumar A, Nagarjuna Reddy S, Nagendar Yamsani and Jyotsna Devi Kosuru S N V; All authors reviewed the results and approved the final version of the manuscript.

Data Availability Statement

The Datasets used and /or analysed during the current study are available from the corresponding author on reasonable request.

Conflict of interest

The authors declare that they have no conflict of interest.

Funding

No funding agency is associated with this research.

Consent for publication

All authors gave permission to consent to publish.

References

- [1]. S. A. Oyewole and O. O. Olugbara, "Product image classification using Eigen Colour feature with ensemble machine learning," *Egyptian Informatics Journal*, vol. 19, no. 2, pp. 83–100, Jul. 2018, doi: 10.1016/j.eij.2017.10.002.
- [2]. T. M. Tashu, S. Fattouh, P. Kiss, and T. Horvath, "Multimodal E-Commerce Product Classification Using Hierarchical Fusion," 2022 IEEE 2nd Conference on Information Technology and Data Science (CITDS), pp. 279–284, May 2022, doi: 10.1109/citds54976.2022.9914136.
- [3]. Wen Zhang, Yanbin Lu, Bella Dubrov, and Zhi Xu, "Deep Hierarchical Product Classification Based on Pre-Trained Multilingual Knowledge", 2021.
- [4]. Anastraj K, "Optimizing Edge Intelligence in Satellite IoT Networks via Computational Offloading and AI Inference", *Journal of Computer and Communication Networks*, pp. 001-012, 05 January 2025, doi: 10.64026/JCCN/2025001.
- [5]. Yonatan Hadara, and Erez Shmueli, "Categorizing Items with Short and Noisy Descriptions using Ensembled Transferred Embeddings", *arXiv preprint arXiv:2110.11431*, 2021.
- [6]. Tianlong Xu, Le Yu, Yuanbo Wang, Morgan White, and Estelle Afshar, "A Scalable Framework for Product Image Classification applied to Home Improvement E-commerce", *Association for Computing Machinery, DLP-KDD*, Singapore, 2021.
- [7]. L. Fengzi, S. Kant, S. Araki, S. Bangera, and S. S. Shukla, "Neural Networks for Fashion Image Classification and Visual Search," *SSRN Electronic Journal*, 2020, doi: 10.2139/ssrn.3602664.
- [8]. Y. M. Latha and B. S. Rao, "A Novel Autoregressive Co-Variance Matrix and Gabor Filter Ensemble Convolutional Neural Network (ARCM-GF-E-CNN) Model for E-Commerce Product Classification," *Revue d'Intelligence Artificielle*, vol. 36, no. 1, pp. 163–168, Feb. 2022, doi: 10.18280/ria.360119.
- [9]. L. Xu and X. Sang, "E-Commerce Online Shopping Platform Recommendation Model Based on Integrated Personalized Recommendation," *Scientific Programming*, vol. 2022, pp. 1–9, Apr. 2022, doi: 10.1155/2022/4823828.

- [10]. Y. Liu and Z. Ding, “Personalized recommendation model of electronic commerce in new media era based on semantic emotion analysis,” *Frontiers in Psychology*, vol. 13, Jul. 2022, doi: 10.3389/fpsyg.2022.952622.
- [11]. C.-F. J. Kuo et al., “Automatic lung nodule detection system using image processing techniques in computed tomography,” *Biomedical Signal Processing and Control*, vol. 56, p. 101659, Feb. 2020, doi: 10.1016/j.bspc.2019.101659.
- [12]. V. Lessa and M. Marengoni, “Applying Artificial Neural Network for the Classification of Breast Cancer Using Infrared Thermographic Images,” *Computer Vision and Graphics*, pp. 429–438, 2016, doi: 10.1007/978-3-319-46418-3_38.
- [13]. Zulpe, N. and Pawar, V., “GLCM textural features for brain tumor classification,” *International Journal of Computer Science Issues (IJCSI)*, vol.9, no.3, p.354, 2012.
- [14]. A. F. Weller, J. Corcoran, A. J. Harris, and J. Andrew Ware, “The semi-automated classification of sedimentary organic matter in palynological preparations,” *Computers & Geosciences*, vol. 31, no. 10, pp. 1213–1223, Dec. 2005, doi: 10.1016/j.cageo.2005.03.011.
- [15]. K. Huang, X. Liu, S. Fu, D. Guo, and M. Xu, “A Lightweight Privacy-Preserving CNN Feature Extraction Framework for Mobile Sensing,” *IEEE Transactions on Dependable and Secure Computing*, pp. 1–1, 2020, doi: 10.1109/tdsc.2019.2913362.
- [16]. Yang Bai, Lihua Guo, Lianwen Jin, and Qinghua Huang, “A novel feature extraction method using Pyramid Histogram of Orientation Gradients for smile recognition,” *2009 16th IEEE International Conference on Image Processing (ICIP)*, pp. 3305–3308, Nov. 2009, doi: 10.1109/icip.2009.5413938.
- [17]. A. Bosch, A. Zisserman, and X. Munoz, “Representing shape with a spatial pyramid kernel,” *Proceedings of the 6th ACM international conference on Image and video retrieval*, pp. 401–408, Jul. 2007, doi: 10.1145/1282280.1282340.
- [18]. S. Lazebnik, C. Schmid, and J. Ponce, “Beyond Bags of Features: Spatial Pyramid Matching for Recognizing Natural Scene Categories,” *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition - Volume 2 (CVPR’06)*, vol. 2, pp. 2169–2178, doi: 10.1109/cvpr.2006.68.
- [19]. N. Dalal and B. Triggs, “Histograms of Oriented Gradients for Human Detection,” *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR’05)*, vol. 1, pp. 886–893, doi: 10.1109/cvpr.2005.177.
- [20]. N. E. Khalifa, M. Loey, and S. Mirjalili, “A comprehensive survey of recent trends in deep learning for digital images augmentation,” *Artificial Intelligence Review*, vol. 55, no. 3, pp. 2351–2377, Sep. 2021, doi: 10.1007/s10462-021-10066-4.
- [21]. R. Ahila Priyadharshini, S. Arivazhagan, M. Arun, and A. Mirmalini, “Maize leaf disease classification using deep convolutional neural networks,” *Neural Computing and Applications*, vol. 31, no. 12, pp. 8887–8895, May 2019, doi: 10.1007/s00521-019-04228-3.
- [22]. M. Jain, V. Singh, and A. Rani, “A novel nature-inspired algorithm for optimization: Squirrel search algorithm,” *Swarm and Evolutionary Computation*, vol. 44, pp. 148–175, Feb. 2019, doi: 10.1016/j.swevo.2018.02.013.
- [23]. Tummala, S. L. V. Ayyarao et al., “War Strategy Optimization Algorithm: A New Effective Metaheuristic Algorithm for Global Optimization,” *IEEE Access*, vol. 10, pp. 25073–25105, 2022, doi: 10.1109/access.2022.3153493.
- [24]. D. Gefen, “E-commerce: the role of familiarity and trust,” *Omega*, vol. 28, no. 6, pp. 725–737, Dec. 2000, doi: 10.1016/s0305-0483(00)00021-9.
- [25]. M. Pawłowski, “Machine Learning Based Product Classification for eCommerce,” *Journal of Computer Information Systems*, vol. 62, no. 4, pp. 730–739, May 2021, doi: 10.1080/08874417.2021.1910880.
- [26]. T. Zahavy, A. Krishnan, A. Magnani, and S. Mannor, “Is a Picture Worth a Thousand Words? A Deep Multi-Modal Architecture for Product Classification in E-Commerce,” *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, Apr. 2018, doi: 10.1609/aaai.v32i1.11419.
- [27]. B. K. Jha, S. G. G, and V. K. R, “E-Commerce Product Image Classification using Transfer Learning,” *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 904–912, Apr. 2021, doi: 10.1109/icmc51019.2021.9418371.
- [28]. J. Dai, T. Wang, and S. Wang, “A Deep Forest Method for Classifying E-Commerce Products by Using Title Information,” *2020 International Conference on Computing, Networking and Communications (ICNC)*, pp. 1–5, Feb. 2020, doi: 10.1109/icnc47757.2020.9049751.
- [29]. E-Commerce Product Image Dataset taken from, “<https://www.kaggle.com/datasets/sunnykusawa/ecommerce-products-image-dataset>”, accessed on February 2024.