Journal of Ceramic Processing Research. Vol. 25, No. 4, pp. 572~588 (2024)
(Received 8 March 2024, Received in revised form 26 April 2024, Accepted 29 April 2024)
https://doi.org/10.36410/jcpr.2024.25.4.572

J O U R N A L O F
Ceramic Processing Research

Ceramic tile surface defect detection with integrated feature engineering and defect fuse classifier

Chi Zhang*
Liaocheng Vocational and Technical College, Liaocheng, Shandong, 252000, China

Introduction: Ceramic tile surface defect detection is crucial for ensuring product quality. This study proposes an integrated approach combining feature engineering and a Defect Fuse Classifier for accurate defect detection. Methods: The proposed model utilizes Python and splits the collected data into 70% for training and 30% for testing. Purpose: The purpose section explicitly states the objectives of the study. It highlights the research goals, such as evaluating the effectiveness of the proposed methodology in detecting ceramic tile surface defects and exploring the impact of parameter variations on detection performance. Results: Comparative analysis with state-of-the-art methods is conducted using various metrics such as sensitivity, specificity, accuracy, precision, FPR, FNR, NPV, F-Measure, and MCC. (a) For a Training Rate of 70%: The proposed Defect Fuse Classifier outperforms existing models with an accuracy of 97.4%, precision of 88.5%, sensitivity of 88.5%, specificity of 98.5%, F-Measure of 88.5%, MCC of 87%, NPV of 98.5%, FPR of 1.4%, and FNR of 11.4%. Conclusion: This study introduces a novel deep learning approach for ceramic tile surface defect detection, encompassing data acquisition, pre-processing, feature extraction, feature selection, and deep learning-based defect detection. The proposed Defect Fuse Classifier, integrating CNN, Bi-LSTM, and RNN, demonstrates superior performance, making it a promising solution for defect detection in ceramic tile surfaces.

Keywords: Ceramic tile, Defect detection, Hybrid optimization model, Defectfuse classifier.

Introduction

Ceramic tile is a popular construction decoration, with mechanization and automation in manufacturing. The flaws like dirt, scratches, pinholes, uneven color, and corners can occur during the ceramic tile-making process [1]. High mechanical strength ceramic tile preparation reduces raw material consumption, manufacturing costs, waste emissions, and transportation costs, thereby improving the cost-performance ratio of goods [2]. Ceramic tile has chemical stability, design diversity, and stain resistance, but its high energy consumption and pollution emissions harm the environment despite its economic contributions [3]. Automated equipment is used for tile size and flatness checks, while manual labor is still used for surface quality examination, with defects detected through texture methods, with simple ones visible [4].

The industrial sector faces challenges in maintaining product quality in ceramic tile production, which is evaluated using automated visual assessment. Improving defect identification accuracy is a challenging task that can affect manufacturing quality, customer trust, and business earnings [5]. The ceramic tile industry utilizes quality control and defect detection at various manufacturing stages, often automated using computer vision and image processing techniques, enhancing efficiency and effectiveness [6]. Defects in ceramic tiles can be caused by unreliable manufacturing equipment, production procedures, and raw ingredients, resulting in physical harm, structural weaknesses, and cosmetic issues, affecting the tiles' structural integrity and safety [7]. The technology develops an automatic visual inspection system that can detect surface fractures as thin as a hair's width, evaluating its classification performance using network design optimization strategies and data augmentation methodologies [8]. Utilizing industrial cameras and image processing algorithms, machine vision inspection technology offers a novel method for identifying surface flaws in ceramic tiles [9].

Efficient quality control and performance of ceramic materials rely on non-destructive assessment techniques. The industry seeks automated computer-assisted inspection solutions to eliminate subjectivity and cost inefficiencies [10]. In the defect identification process, neural networks, specifically CNN, are commonly used. CNN is used to detect fractures and surface defects in raw images. Multiple CNN design algorithms exist, and developing the optimal CNN model for a specific task requires expertise and effort [11]. CNN-based target measurement algorithms have demonstrated impressive results in various industries. These algorithms have

*Corresponding author:
Tel : 0635-8334969
Fax: 8322030
E-mail: zhangchi292@163.com

proven invaluable in the manufacturing industry's quality control procedures, especially when it comes to measuring measurements and identifying product flaws. CNN-based target measurement algorithms have also been used in the aerospace sector to check for flaws and irregularities in aircraft components. These algorithms may discover surface flaws, fissures, and abnormalities that could jeopardize the aircraft's structural integrity by analysing photos taken by inspection cameras or drones. Research has indicated that CNN-based algorithms exhibit higher sensitivity and dependability when compared to manual inspection methods or conventional image processing approaches.

The Faster Region-based convolution neural network (R-CNN) model employs the Soft-NMS method to detect flaws and combines multiple layers' features. Faster R-CNN serves as a recognition framework with robust feature expression and easily modifiable structure [12].

According to the HSV color space, the defect discriminator was built using support vector machines (SVM) for two classes after the defects were separated using the conventional saliency detection approach. The color histogram characteristics for segmented defect rectangles were retrieved. At last, the algorithm produced the desired result of increasing fault detection accuracy [13]. The brightness histogram is widely used for identifying defects due to its simplicity and low complexity. The LBP offers improved accuracy but requires longer computation time compared to the luminance histogram [14]. A basic step in image analysis is edge detection, which includes locating the borders between various areas of a picture. A well-liked technique for edge identification in many applications is the Canny edge detection algorithm [15]. This paper aims to find the ceramic tile surface defect detection with integrated feature engineering and defectfuse classifiers.

The main contribution of this research work:

- To introduce a new optimization model: The proposed new hybrid optimization model for feature selection. The model used the Arithmetic optimization Algorithm and Dingo optimization respectively.
- To introduce a new defectfuse classifier model for surface defect detection: The considered defectfuse classifier model is the combination of CNN, Bi-LSTM, and RNN.

Literature Review

In 2020, Wang et al. [16] explored the increasing popularity of vision-based inspection methods in industrial manufacturing, focusing on surface flaw identification. It suggests a technique to minimize false positives that combines feature comparison analysis with adversarial and unsupervised learning.

In 2022, Nogay et al. [17] addressed the application of CNNs in deep machine learning to recognize ceramic flaws, highlighting the significance of both cost containment and product quality. It suggested using thermographic methods and contrasted 1D and 2D CNNs. Defect detection accuracy and resilience were significantly increased with DCNN models with transfer learning, according to a case study utilizing a pre-trained AlexNet-like model.

In 2022, Jin et al. [18] focused on developing a deep learning system for Automated Surface Defect Inspection (ASDI) in decorative ceramics. After preprocessing the images and training YOLOv3 on faulty images, they achieved a 94.90% detection accuracy at 25 frames per second. The decorative ceramics sector benefited economically from this algorithm's increased inspection efficiency and quality.

In 2021, Sušac et al. [19] introduced a multi-line signal change detection method for identifying flaws in ceramic tile images, enhancing defect identification precision and effectiveness. It emphasized technology transfer for quality assurance in the ceramic tile sector, significantly contributing to image processing and flaw identification.

In 2021, Rai et al. [20] discussed the usage of deep learning for finding ceramic surface defects instead of laborious human examination. It examined deep learning techniques, with the greatest outcomes coming from ensemble learning. Customization and flexibility were provided via deep learning. The paper reviewed relevant literature with an emphasis on image-based manufacturing fault identification.

In 2023, Boovaneswari et al. [21] examined the application of AI's deep neural networks are being used to identify defects in ceramic tiles, a shift from human inspection to image processing techniques. This involves addressing issues like form similarity and fabric classification, using various algorithms and techniques.

In 202, Qian et al. [22] explained that the framework automates the color-separation process for ceramic tiles, reducing human judgment. It uses HSV color space histogram statistics for preprocessing and feature extraction, training an SVM model, and balancing training time and accuracy.

In 2020, Le et al. [23] covered a learning-based method that makes use of tiny picture datasets to identify surface flaws. The authors suggested a technique for surface fault identification that made use of machine learning. The study brought attention to the difficulty of dealing with sparse data and offered a way around it.

In 2021, Junior et al. [24] discussed the importance of early detection of ceramic facade fractures in construction to ensure building integrity and passenger safety. Deep learning and image processing have improved crack identification systems, while image segmentation identifies fracture sites. Table 1 tells about the research gaps identified in the existing works.

Research Gap

Several authors have addressed the pressing research gaps in ceramic tile defect detection, each proposing distinct methodologies to tackle this challenge. Wang et al. (2020) aimed to develop an unsupervised technique for patterned texture surface images, filling the void in efficient defect detection without manual labeling. Nogay et al. (2022) targeted the detection of subtle deformations, addressing the delay in identifying faults in ceramic products, a gap often overlooked in defect detection systems. Jin et al. (2022) sought to automate decorative ceramic flaw inspection, addressing the need for sophisticated checks in this domain. Sušac et al. (2021) focused on signal change detection methods, a crucial area for enhancing efficiency in ceramic tile flaw detection processes. Rai et al. (2021) investigated deep learning techniques for industrial product surface problem detection, bridging the gap in efficient quality inspection methodologies. Boovaneswari et al. (2023) contributed to improving defect detection through deep neural networks, aiming to address the efficiency gap in quality control processes. Qian et al. (2021) proposed a color-separation framework based on the HSV color space and SVM model, addressing the need for advanced color-based defect detection techniques. Le et al. (2020) focused on learning-based defect identification, especially in scenarios with limited image datasets, filling a significant research gap in data-driven defect detection approaches. Junior et al. (2021) utilized deep learning and automated optical inspection to detect ceramic tile flaws more accurately, addressing the gap in advanced defect detection methodologies. These studies collectively contribute to advancing defect detection in the ceramic tile industry, addressing key research gaps and enhancing quality control processes.

The constraints noted in prior studies on ceramic tile fault identification are intended to be addressed by the suggested model. First off, our model provides a complete approach to defect detection by combining a variety of deep learning approaches, including CNN, Bi-LSTM, and RNN. This overcomes the drawbacks of individual methods as noted by Wang et al. (2020), Nogay et al. (2022), and Rai et al. (2021). Through this integration, the difficulties mentioned by Jin et al. (2022) and Sušac et al. (2021) are overcome and defects, including small deformations and patterned texture surface pictures, may be detected more accurately.

In addition, our model addresses the issues raised by Rai et al. (2021) and Le et al. (2020) about the availability of labeled data by employing sophisticated optimization techniques like ADOA and DOX for feature selection and fault classification. Through the utilization of these methods, our model is able to efficiently extract pertinent information from photos of ceramic tiles without requiring a great deal of human labeling, which improves defect detection procedures' scalability and efficiency.

Additionally, as mentioned by Boovaneswari et al. (2023) and Junior et al. (2021), our model integrates a hybrid defect fusion classifier that combines the characteristics of different deep learning architectures, minimizing the constraints associated with individual approaches. This method boosts overall quality control effectiveness in the ceramic tile manufacturing sector and guarantees reliable fault identification in a variety of manufacturing scenarios. Overall, by fusing cutting-edge deep learning techniques with creative feature selection algorithms, our proposed model provides a comprehensive response to the difficulties in ceramic tile defect detection. This advances the state-of-the-art in quality control procedures for ceramic tile manufacturing and overcomes the limitations found in earlier research.

Proposed Methodology

Overall Architecture

Defect detection in ceramic tile manufacturing is crucial for maintaining integrity, safety, and aesthetics. Machine Learning, trained on specific datasets, may not adapt to evolving defect types. Deep learning helps identify intricate patterns and minimizes the need for human feature engineering by learning and extracting features from images.

In this research work, a new deep learning-based ceramic tile defect detection developed by the mentioned stages of the following: (i) Data Acquisition (ii) Image pre-processing (iii) Feature Extraction (iv) Feature Selection (v) Deep Learning based defect detection

Step 1: Data Acquisition

The dataset utilized in this investigation originates from the Alibaba Tianchi competition. The images are gathered by two separate lighting circumstances. In conclusion, a total of 4956 pictures were employed and saved in JPG format by the dimensions of 8192 × 6000 pixels. The dataset comprises six discrete classifications for defects observed on the external surface of ceramic tiles, namely, abnormalities in the borders, irregularities in the corners, imperfections resembling white dots, flaws in the light-colored blocks, flaws in the dark-colored blocks, and malfunctions in the openings.

Step 2: Image Pre-processing

The collected raw images are pre-processed via Median Filtering and Image Augmentation (rotation, flipping, brightness adjustments)

Step 3: Feature Extraction

From the pre-processed images, features like HOG, LBP, Color Histograms, Gabor Filters, and Edge Detector are extracted.

Step 4: Feature Selection

From the extracted features, the optimal features are selected using new hybrid optimization Algorithm. The Proposed hybrid optimization Algorithm is a combination for Arithmetic Optimization Algorithm and Dingo Optimizer, respectively.

Step 5: Deep Learning based defect detection

The defects in the tiles are identified via the new

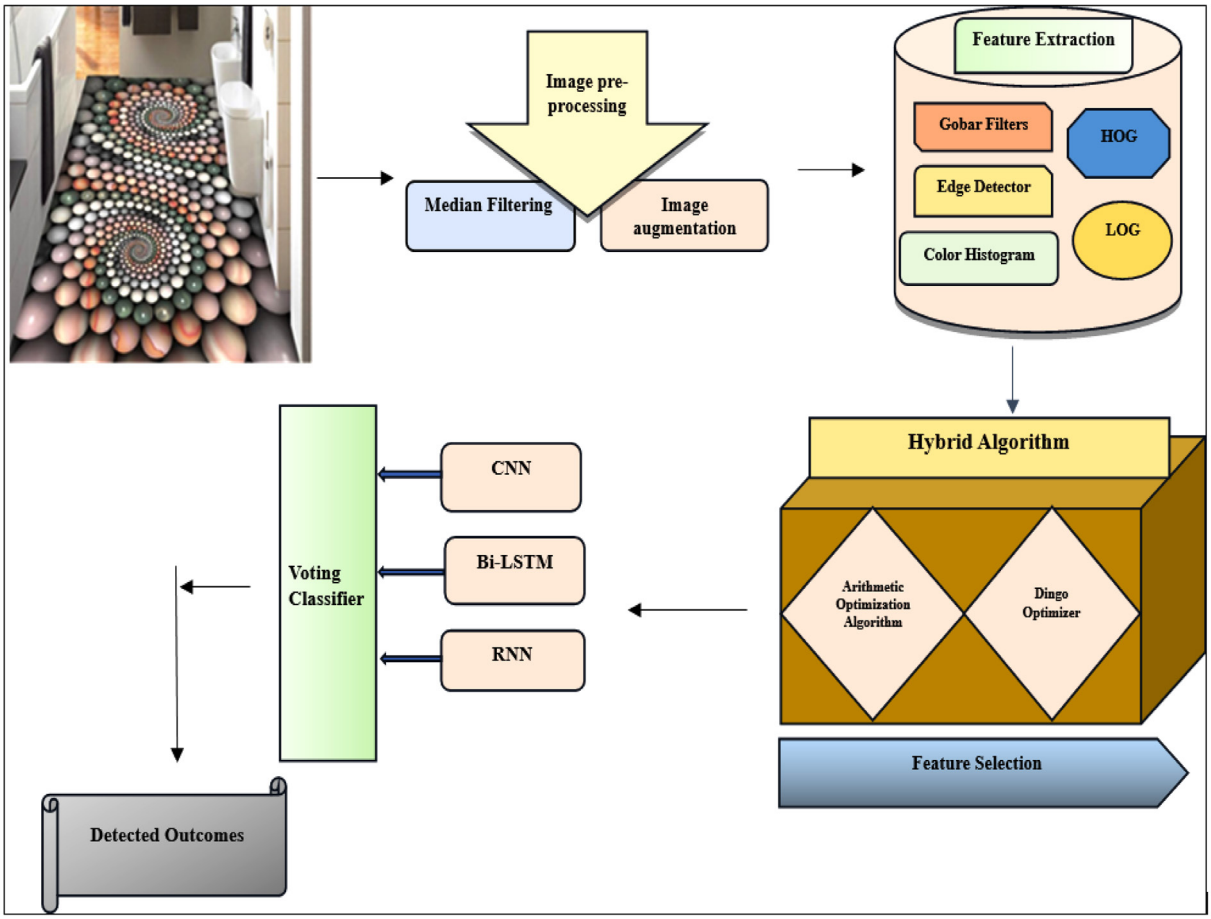

Fig. 1. Overall design for the proposed model.

Defectfuse Classifier. This Defectfuse classifier is developed by combinin Deep Learning classifications like CNN, Bi-LSTM, and RNN, respectively. Finally, result from voting classifiers is validated.

Evaluation Metrics: The performance of the model is evaluated using various metrics such as accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC). These metrics provide insights into the model's ability to correctly classify defective and non-defective tiles, as well as its trade-offs between false positives and false negatives.

Preprocessing

Preprocessing plays a crucial role in ceramic tile defect detection, and its importance can't be overstated. It involves a series of image enhancement and cleaning techniques applied to raw images before defect detection algorithms are employed. Adjusting image contrast and brightness can make defects more visible. By enhancing the contrast between defects and the background, subtle defects that might be initially hard to spot become more apparent. Preprocessing techniques like edge detection help identify boundaries and edges within the image. Detecting edges can assist in segmenting the image and pinpointing defect locations.

In this research work the median filtering and image augmentation in pre-processing is done.

Median Filtering

Median filtering is the method employed in the realm of digital image processing to diminish noise and undesired variations while conserving crucial structural characteristics. It is particularly effective in removing "salt and pepper" noise, which is random, isolated pixels with extreme values [25]. The preprocessing method is highly regarded in surface defect detection, improving image quality and efficiency. It enhances image quality, making them suitable for visual scrutiny and automated identification of defect detection algorithms.

Steps to calculate Median Filtering

- A set of random variables is given. $W = (W_1, W_2, \ldots..W_o)$, The directive statistics $W \leq W_{(2)} \leq \ldots \leq W_{(M)}$ are arbitrary variables. Considered by Ordering the Numbers of W_p in cooperative order. The median value is then expressed as per Eq. (1).

$$median(W) = \begin{cases} W_{(K+1)} = W_{(K+1)}, & for\ M = 2K+1 \\ \frac{1}{2}(W_K + W_{K+1}), & for\ M = 2K \end{cases} \quad (1)$$

Where M=2K+1 is the median rank. The median is utilized in an assortment of denoising and smoothing

methodologies, particularly for signals that have been tainted by abrupt noise. It is acknowledged as a dependable evaluator of the positional element of a dispersion.

- The two-dimensional median filter for a grayscale input image with intensity values $w_{i,l}$ is defined as per Eq. (2).

$$w_{i,l} = \underset{(x,u) \in V}{\boldsymbol{median}} (w_{i+x,l+i}) , \quad (2)$$

where V stands for the window that the filter can be used on. We examine $M \times M$ symmetric square windows with $M=2N+1$, for the rest of the research, where $M=(M^2+1)/2$ is the median rank. Presumably, this is the filter version that is most frequently used.

- The median filter is useful for identifying specific image attributes by examining its output distribution and comparing it to other filters. However, its non-linear nature presents a significant challenge in theoretical investigations into the relationship between input and output distributions.
- Consequently, the widely recognized fact is that the input samples are indistinguishable. The typical Cumulative Distribution Function H_Y for productivity samples $y_{i,l}$ and source samples $w_{i,l}$ with Cumulative Distribution Function H_W is provided as per the Eq. (3).

$$H_Y(y) = \sum_{k=m}^{M^2} \binom{M^2}{k} [H_W(y)]^k [1 - H_W(y)]^{M^2-k} , \quad (3)$$

The sample median of input models with a standard dispersal is a unique yet fascinating example. $w_{i,l} \sim Q(\mu, \sigma)$, which was displayed to asymptotically (as $M \to \infty$) trail the standard distribution as per Eq. (4).

$$y_{i,l} \sim Q(\mu, \sigma_m), \qquad where\ \sigma_m = \sqrt{\frac{\pi}{2}} \cdot \frac{\sigma}{M} . \quad (4)$$

Typically, the combined arrangement of neighboring pixels has significance since close pixels in filtered images are slightly connected due to their overlapping windows. The $M \times M$ median filter with suggested input is $H_W(w)$. It generates an equation for both outcome pixels' bivariate dispersion y_r and y_s (P), $H_y(y_r, y_s)$. Even under the unrealistic premise that pixel intensities are in the case with Appendix (B) calculation, it is validated that defining median filtered images potentially poses challenges.

Image Augmentation

Image augmentation is extensively utilized technique in computer vision, machine learning, and deep learning. It artificially expands the dataset by applying different transformations to existing images. These changes may include cropping, rotating, flipping, adjusting brightness, and resizing, among other things. Tasks involving object identification, image segmentation, and image classification benefit greatly from image augmentation [26].

- Image augmentation allows you to artificially expand your dataset by generating additional images through various transformations. More data helps reduce overfitting in deep learning, where a large dataset is frequently associated with improved model performance.
- Augmentation enhances the utility of labeled data by generating additional examples without manual annotation, and it enables models to recognize defects with different characteristics and patterns, ensuring optimal surface problem detection.
- Augmentation improves surface defect detection models' ability to identify new flaws beyond the initial training set, reducing overfitting, especially in small datasets. It enhances the model's generalizability to new data, not just memorizing the training set.

3.3 Feature Extraction

The images from the pre-processing are extracted by the Feature extraction process. Feature extraction in ceramic tile defect detection involves capturing and representing essential characteristics from the input images to enable effective analysis and classification. Detecting surface defects frequently requires manipulating high-dimensional data, including images. Excessive dimensionality raises the possibility of overfitting and increases computational complexity. Feature extraction is important in ceramic tile surface defect detection process.

In this work, features such as HOG, LBP, Color Histogram, Gabor Filters, and Edge Detectors are

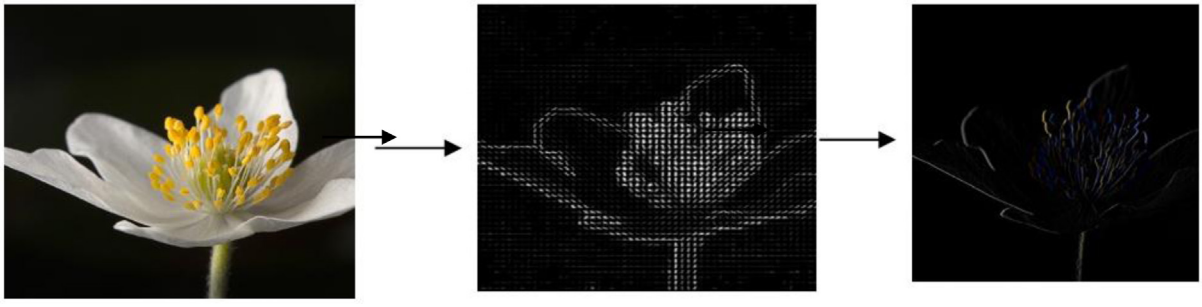

Fig. 2. Image recognition using HOG.

extracted from the pre-processed images.

3.3.1 Histogram Of Oriented Gradient (HOG)

HOG is the feature extraction technique used in computer vision and image processing. It serves a multitude of purposes, encompassing object detection and image classification. The utility of HOG is especially evident in its ability to detect and identify objects or patterns within images [27]. HOG offers a cost-effective method for detecting ceramic tile defects, helping localize their positions within images. It's effective in detecting edges and gradients, identifying fine edges and abrupt changes in the tile's surface, which may indicate defects like cracks, chips, or surface irregularities. This information is crucial for quality control and repair.

Steps to calculate HOG Features

- Select the input image (pre-processed image) whose HOG characteristics need to be computed. 128 by 64 pixels, or 128 pixels in height by 64 pixels in width, is the required scale for the images. Enhancing performance on the site safety detection test was the main goal of employing this type of identification.
- The gradients of the image must be calculated before the HOG feature can be computed. A directed shift in pixel intensity along the *v*-axis and *w*-axis is known as an image gradient. The gradient vector of pixels at point (*v*, *w*) represents as per the Eq. (5)

$$\nabla f(v,w) = \begin{bmatrix} \partial f/\partial v \\ \partial f/\partial w \end{bmatrix} = \begin{bmatrix} f(v+1,w) - f(v-1,w) \\ f(v,w+1) - f(v,w-1) \end{bmatrix} \quad (5)$$

Where $f(v, w)$ is the pixel intensity at coordinates v and w; h_v and h_w are the gradient in v and w direction, respectively. The magnitude, $n(v, w)$ and phase, $\theta(v, w)$ of the gradient, then can be calculated using the following Eq. (6) and Eq. (7), respectively.

$$\text{Magnitude } n(v,w) = \sqrt{h_{v^2} + h_{w^2}} \quad (6)$$

$$\text{Phase } \theta(v,w) = \tan^{-1}\left(\frac{h_w}{h_v}\right) \quad (7)$$

Where h_v and h_w are the gradient in v and w direction, respectively

The gradient matrices are divided into 8×8 cells to form a block once the gradient of each pixel is determined. Each block is then used to compute a nine-point histogram. A nine-point histogram consists of nine bins, each representing a range of angles of twenty degrees. In Fig. 3, a nine-bin histogram is shown with values assigned based on computations. Each of these histograms might be represented as histogram with bins corresponding to angle intensity in the bin.

There are 64 distinct values in a block, so the computation is done for each of these values.

Number of bins = 9(starting from 0° to 180°)

step size (Δθ) = 180°/Number of bins = 20°

Each L^{th} bin, Bin is separated as per the Eq. (8):

$$[\Delta\theta \cdot l, \Delta\theta \cdot (l+1)] \quad (8)$$

The center value of every bin is:

$$B_l = \Delta\theta(l + 0.5) \quad (9)$$

The l^{th} bin and the value supplied to the l^{th} and $(l+1)^{\text{th}}$ bins is computed for every cell in the block. The value is determined using as per Eq. (10), Eq. (11), Eq. (12).

$$L = \left\lfloor \left(\frac{\theta}{\Delta\theta} - \frac{1}{2}\right) \right\rfloor \quad (10)$$

$$O_l = \mu \cdot \left\lfloor \left(\frac{\theta}{\Delta\theta} - \frac{1}{2}\right) \right\rfloor \quad (11)$$

Value									
Bins	0	20	40	60	80	100	120	140	160

Fig. 3. Representation of a 9-bin histogram.

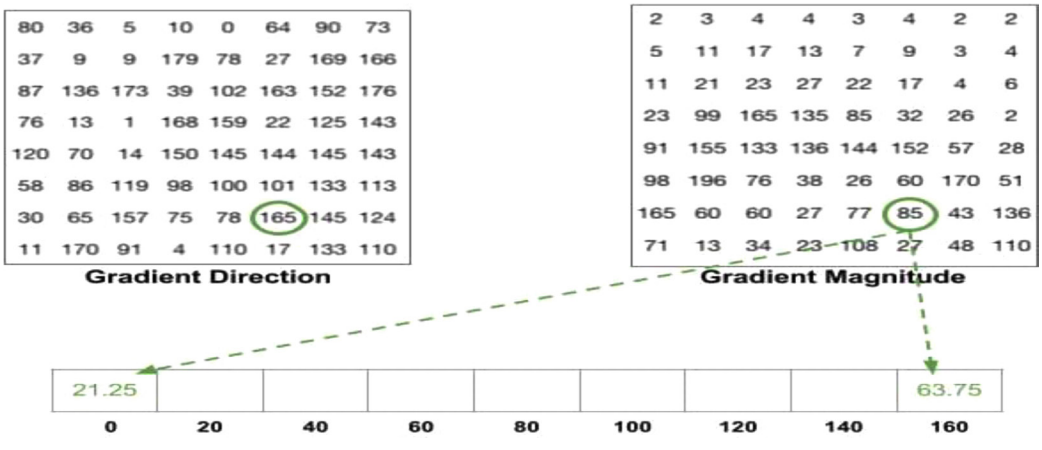

Fig. 4. Example of technique for calculation of 9 bin histograms.

$$L = \left|\left(\frac{\theta}{\Delta\theta} - \frac{1}{2}\right)\right| \quad (12)$$

- ❖ A block's bin is an array, to which the values of O_l and O_{l+1} are attached at index for k^{th} and (l+1) bin, respectively, determined for every pixel.
- ❖ The resultant matrix, as indicated by the computations above, is 16×8×9.
- ❖ The 9-point histogram matrix was combined into a new block (2×2), and the overlapping clubbing with an 8-pixel stride was performed, resulting in a 36-feature vector. Fig. 4. shows the method for calculating 9-bin histograms.

$$h_{ei} = [e_{1,}e_{2,}e.\ldots e_{36,}] \quad (13)$$

- ❖ The $K2$ norm is employed to standardize the standards of fb for each block:

$$h_{ei} \leftarrow \frac{h_{ei}}{\sqrt{\|h_{ei}\|^2 + \in}} \quad (14)$$

- ❖ Before normalizing, the value l is first calculated through the subsequent Eq. (15), Eq. (16).

$$k = \sqrt{e_1^2 + e_2^2 + e_3^2 + \cdots + e_{36}^2} \quad (15)$$

$$h_{ei} = \left[\left(\frac{e_1}{k}\right), \left(\frac{e_2}{k}\right), \left(\frac{e_3}{k}\right), \ldots \left(\frac{e}{k}\right)\right] \quad (16)$$

- ❖ The normalization process reduces contrast discrepancies in images of the same object by normalizing each brick in sequence. A feature vector is created by accumulating 36 data points, 15 blocks in a vertical direction and 7 blocks in a horizontal direction. The cumulative length for HOG features is calculated to be 3780.

Local Binary Patterns (LBP)

LBP is texture descriptor employed in computer vision and image analysis. When analyzing a pixel's surrounding neighborhood, LBP takes into account the local patterns of pixel intensities in that area [28]. LBP is highly effective in describing and capturing local texture patterns in images. It is especially helpful in applications like texture segmentation and classification where texture information is essential. It generates histograms of the frequency of different local patterns, providing a concise representation of the texture distribution in an image or region.

Steps to Calculate LBP Features

- ❖ LBP is a technique that can analyze local textures in ceramic tile images, detecting variations that may indicate defects and can be used to extract texture features.
- ❖ The integration of LBP and deep learning in identifying defects on ceramic tile surfaces can improve detection accuracy by recognizing texture patterns, enhancing generalization, and providing a more reliable method.
- ❖ The combination of handcrafted features like LBP with deep learning techniques leverages the strengths of both approaches for more effective defect detection.

Assume that I_e=I(V, W) is an arbitrary central pixel at the position (V, W) and I_t=I(V_t, W_t) is a neighboring pixel surrounding I_e, with as per the Eq. (17), Eq. (18).

$$v_t = v + s\cos\left(2\pi\,\frac{t}{T}\right) \quad (17)$$

$$w_t = w - s\sin\left(2\pi\,\frac{t}{T}\right) \quad (18)$$

T is the total number of neighboring pixels ($I_t, 0 \le t \le T$) and S is the distance from I_e at which the pixels are sampled. The traditional LBP descriptor as per the following Eq. (18), Eq. (19)

$$L(S,T) = \sum_{t=0}^{T-1} \sigma(I_t - I_e)2^t \quad (19)$$

$$\sigma(\tau) = \begin{cases} 1, if\ \tau \ge 0 \\ 0, otherwise - 1 \end{cases} \quad (20)$$

Where I_e denotes gray value of centre pixel. I_t denotes gray value of its neighbors. S stands for number of neighbor and T be the radius of neighborhood.

The actual 3×3 construction is shown in both Fig. 5(a) and 5(b). The structure in the figure has vectors of "[3,1,5,9,3,2,2,6,9]". The sign vector in figure is "[-1,1,1,1,-1,-1,1,1]". It apparent that the novel regional binary pattern, which codes the binary digit as the 8-bit string "01110011" uses only the sign vector

- ❖ The $LBP_{S,T}$ operator generates ($2T$) output values, forming dissimilar binary patterns by T pixels in neighbor set. Rotation effects are eliminated by assigning unique identifiers. The image rotates, causing gray I_t to move circle perimeter, I_e eliminate the rotation effect, and a unique identifier is assigned

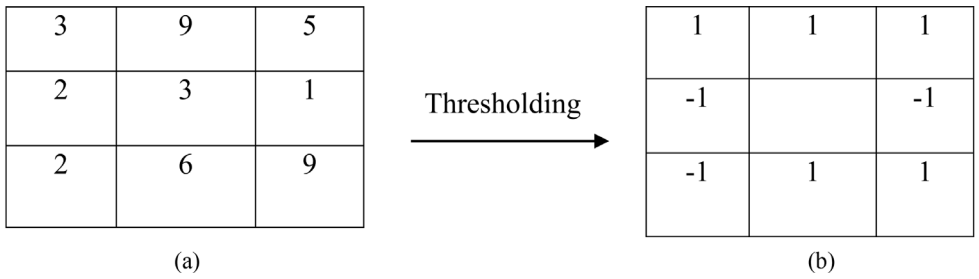

Fig. 5. (a) 3×3 sample block. (b) Sign Component.

to each rotation using invariant local binary patterns. as per Eq. (21).

$$LBP_{S,T}^{ri} = \min\{ROR\left(LBP_{S,T}, \beta\right), \beta = 0,1, \ldots, t-1\} \quad (21)$$

Where *ri* means rotation constant. *ROR* (*v*, β) be a rou nd bitwise turn right on *T*-bit number *v*, β times. Finally, the lowest of computed values of β=0 to *t*–1 are chosen.

Color Histogram

A color histogram is a visual representation of color distribution within an image, providing a numerical assessment of each color's quantity. It is created using the image's color channels, where each channel's histogram represents the intensity distribution for its corresponding color [29]. These are highly efficient feature vectors can be used in various image processing tasks, including object identification, content retrieval, and flaw identification, by condensing the color composition of an image.

Steps to Calculate Color Histogram Feature

- ❖ Color histograms can be effectively used in ceramic tile defect detection to analyze and characterize the color distribution within tile images. It can inform the setting of adaptive thresholds for defect detection.
- ❖ Adaptive thresholding techniques based on the color distribution can dynamically adjust the sensitivity of the defect detection algorithm.
- ❖ The RGB color space has three color component values. In our method, we utilized the HSV color space which aligns with the human visual system.
  - ○ The HSV color space is used to extract color information from ceramic tile photos for defect detection in the context of the defect discriminator. The defect discriminator makes use of the hue, saturation, and value components of the picture pixels to detect minute color changes that might be signs of defects such surface imperfections, fissures, or discolorations.
  - ○ The hue component can be used by the flaw discriminator to separate particular colors linked to defects, such dark patches or discolored areas. The value component provides information about the defect's brightness or contrast in relation to the surrounding areas, while the saturation component can aid in determining how intense these colors are.
  - ○ The defect discriminator can efficiently discern between normal and faulty regions in ceramic tile photographs by evaluating these color properties in the HSV color space. This allows for precise and efficient defect identification in production processes.
- ❖ RGB image changed into HSV color space as per the Eq. (22), Eq. (23), Eq. (24) respectively. (Here R is represented as *S*, G is represented as *H* and B is represented as *A*)

$$D = \cos^{-1}\frac{{}^{1}/_{2}\,[S-H]+[S-A]}{\sqrt{(S-H)^2-(H-A)(S-A)}} \quad (22)$$

$$S = 1 - \left(\frac{3[\min(S,H,A)]}{S+H+A}\right) \quad (23)$$

$$J = \left(\frac{S+H+A}{3}\right) \quad (24)$$

The color space values in an image are represented by the wavelength (*D*), saturation (*S*), and the intensity value (*J*). range from 0 to 1, representing the intensity of an image, with 0 representing black and 1 representing white.

- ❖ A set of bins representing individual colors in the color space being utilized is color histogram. The quantity of colors in the image determines the number of bins. A vector is what defines a color histogram as per the Eq. (25).

$$D = \{D[0], D[1], D[2], D[3], \ldots \ldots D[p], \ldots . D[n]\} \quad (25)$$

Where *p* denotes the color bin within the color histogram and *D*[*p*] denotes the count for pixels with color *p* in image, while *n* signifies total count of bins utilized in color histogram. Every pixel in image is allocated to bin in the color histogram based on its color. The value for each bin represents count of pixels with similar color. The normalized color histogram, denoted as *D*, is defined as per the Eq. (26).

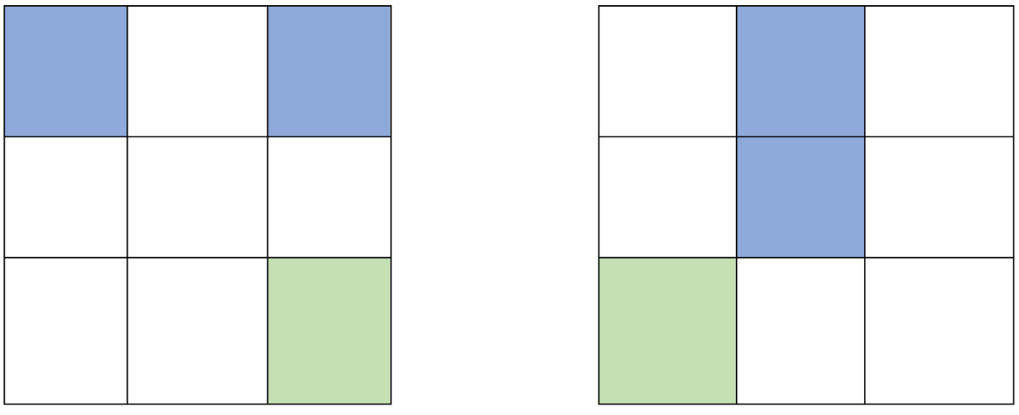

Fig. 6. Two different images having same histogram.

$$D' = \{D'[0], D'[1], D'[2], D'[3], \dots\dots D'[p], \dots . D'[n]\} \quad (26)$$

Where $D'[p] = \frac{D[p]}{t}$, t is total number of pixels of images

However, the color histogram has limitations of its own. The right photos cannot be retrieved if two images have the same color percentage but differ in how the colors are dispersed. Fig. 6. shows two unconnected images that have the same color histogram.

We extract the color histogram of the image and we get a n-dimensional color feature vector:

$$D_R = (d1, d2, d3, \dots\dots dn) \quad (27)$$

$$X_1 = 1 - \sum_{p=0}^{n} \min(D_R[p], D_T[p]) \quad (28)$$

Histogram intersection method is used to measure the distance X_1 between query image R and image T in image database.

Gabor Filter

Gabor filters are used to analyze and characterize textures in images, identifying patterns like edges, lines, and regions with specific textures. They are effective in identifying edges and curves, and their reaction to texture and intensity changes is noticeable [30]. They are particularly useful for capturing and analyzing textures on ceramic tile surfaces, enhancing textural patterns indicative of defects.The images serve as a feature representation that highlights relevant information about the texture and structure of the ceramic tile surface. These features can be fed into machine learning models for defect classification.

Steps to calculate Gabour Filter

(1) Gabour image generation

The following equation, which shows eight amplitudes (λ_α) and eight phases (θ_β), respectively, yields a total of 64 Gabor filter as per the Eq. (29). Each Gabor filter measures 3×3.

$$h(v, w; \lambda_\alpha, \theta_\beta, \psi, \sigma, \gamma) = exp - \left(\frac{v'^2 + \gamma^2 w'^2}{2\sigma^2}\right) \cos\left(2\pi \frac{v'}{\lambda^2} + \psi\right) \quad (29)$$

Where, $v' = v\cos\theta_\beta + w\sin\theta_\beta$, $w' = -v\sin\theta_\beta + w\cos\theta_\beta$,

$$\lambda_\alpha = 2^{\frac{-4\alpha+2}{2}}, \theta_\beta = \frac{\beta}{8}\pi,$$

$$where, \alpha = 0,1, \dots\dots 7 \; and \; \beta = 1,2, \dots\dots 7$$

A picture with a gabor filter P_{Gabour}. The Gabor filter and the I_{Gray} are constructed by convolutionally operating on the former as per the Eq. (30).

$$P_{Gabour}(n, m; \alpha, \beta) = \sum_{v=-1}^{v=1} \sum_{w=-1}^{w=1} h(v, w; \lambda_\alpha, \theta_\beta, \psi, \sigma, \gamma). \; P_{Gray}(n + v, n + w) \quad (30)$$

(2) Steps for Gabour Image Selection

The images that have the lowest Inverse Difference Moment (IDM) value, N are chosen. The image with a significant variation in values between pixels is said to have a small IDM. With the chosen Gabor Image, efficient feature extraction for detection is achievable. The subsequent steps are in the Gabor image selection method.

Step 1: For each phase, construct average image P'_{Gabour} for the Gabor image as per the Eq. (31).

$$P'_{Gabour}(n, m; \beta) = \frac{1}{7}\sum_{\alpha=0}^{7} P_{Gabour}(n, m; \beta) \quad (31)$$

Step 2: The image of the Gabor filter is normalized P'_{Gabour} as per the Eq. (32).

$$P'_{Gabour}(n, m; \beta) = \frac{P'_{Gabour}(n,m;\beta) - min\left(p'_{Gabour}(n,m;\beta)\right)}{max\left(p'_{Gabour}(n,m;\beta)\right) - min\left(p'_{Gabour}(n,m;\beta)\right)} \quad (32)$$

Step 3: According to the above equations the Gabor filter image $P'_{Gabour}(n, m; \beta)$ becomes a GLCM, from which the average matrix T_β is derived.

Step 4: Values for IDM features are taken out of each average matrix T_β as per the Eq. (33).

$$G = \left\{G_\beta = \sum_{p=0}^{O} \sum_{i=0}^{D} \frac{T_\beta(p,i)}{1+(p-i)^2}\right\} \; where, \beta = 0,1,2, \dots\dots,7 \quad (33)$$

Step 5: The highest IDM feature values among the top M Gabor filter images are chosen.

Edge Detectors

An edge detector is an essential part of computer vision as well as the processing of images that locates areas in an image where there are noticeable changes in color or brightness. Edges often correspond to boundaries between objects or changes in texture, and detecting them is fundamental for various tasks, including object recognition, image segmentation, and feature extraction [31]. Understanding the composition and substance of images begins with edge detection. The Canny edge detector can be employed in ceramic tile surface defect detection for various purposes, contributing to the identification and characterization of defects.

Steps to calculate Edge Detectors

- ❖ The detected edges can be utilized as cues for segmenting and isolating defective regions on ceramic tiles. It helps in defining the contours of defects, facilitating the creation of regions of interest for subsequent analysis.
- ❖ Canny edges serve as features that capture the discontinuities and transitions in intensity on ceramic surfaces. Machine learning models that categorize and describe various fault kinds may be trained using extracted edge characteristics as input.
- ❖ Gaussian filtration is used to smooth the gradient's direction and amplitude in the images. The gradient's magnitude and direction are found using the finite difference method of the first-order partial derivative, as per Eq. (34), Eq. (35), Eq. (36), Eq. (37), Eq. (38) respectively.

$$x_v = \begin{bmatrix} -1 & 1 \\ -1 & 1 \end{bmatrix} \quad , \quad x_w = \begin{bmatrix} 1 & 1 \\ -1 & 1 \end{bmatrix} \quad (34)$$

$$T[p,i] = \frac{1}{2}(f[p+1,i] - f[p,i] + f[p+1,i+1] - f[p,i+1]) \quad (35)$$

$$R[p,i] = \frac{1}{2}(f[p,i] - f[p,i+1] + f[p+1,i] - f[p+1,i+1]) \quad (36)$$

$$N[p,i] = \sqrt{T[p,i]^2 + R[p,i]^2} \quad (37)$$

$$\theta[p,i] = arc\tan\left(\frac{R[p,i]}{T[p,i]}\right) \quad (38)$$

Here, the variable f represents gray value of image. The symbol T is used to denote the gradient amplitude in the V direction, while R represents the gradient amplitude in the W direction. The variable N is defined as the amplitude of the point under consideration. Lastly, the symbol θ corresponds to the gradient direction, specifically the angle at which it is measured.

Feature Selection

The models from the extraction are selected in the feature selection process. Feature selection is a relevant attribute improves machine learning models by enhancing efficiency, preventing overfitting, and aiding interpretability. In deep learning, neural networks automatically learn features, and techniques such as regularization, attention mechanisms, and ensemble methods contribute to implicit feature selection. The features are selected via ADOA (Arithmetic Dingo Optimization Algorithm).

The significance of the newly developed hybrid optimization model for feature selection is noteworthy when it comes to the identification of surface defects in ceramic tile, and it may also find use in other fields. Here are some main justifications for its significance:

- Enhanced Performance of the Model: This integration of AOA and DOA, makes it possible to search for ideal characteristics more thoroughly and effectively, which improves the model's performance in defect identification.
- Effective Feature Selection: Finding the most pertinent characteristics while lowering dimensionality is made possible by feature selection, which is a crucial stage in the creation of machine learning models. By effectively navigating the feature space and choosing the most discriminative features for defect detection, the hybrid optimization model maximizes the feature selection procedure.
- Adaptability to Data Characteristics: The hybrid optimization model's adaptive nature enables it to dynamically modify its methods in response to the properties of the incoming data. This flexibility is especially important for defect identification, as different types and patterns of flaws may occur, and it guarantees that the model can handle a wide range of circumstances.

Overall, the proposed hybrid optimization model for feature selection plays a crucial role in improving the efficiency, effectiveness, and interpretability of defect detection models, thereby contributing to higher product quality and operational efficiency in industries such as ceramic tile manufacturing.

Arithmetic Optimization Algorithm

Arithmetic Optimization Algorithms improve numerical computation efficiency and accuracy by optimizing precision, minimizing errors, and improving mathematical performance. They address challenges in floating-point arithmetic, reduce complexity, and explore parallel computing for faster execution [32]. Adaptive approaches dynamically adjust algorithms based on input characteristics, crucial in scientific computing, machine learning, and numerical simulations.

Population-based algorithms use random solutions, while detection-based algorithms use randomization for optimal solutions. Optimization involves exploration and exploitation, with exploration preventing local solutions and exploitation improving the precision of answers during the investigation phase.

Initialization phase

The collection of candidate solutions for AOA is chosen randomly and starts the optimization process (W), as indicated in matrix Eq. (39).

$$W = \begin{bmatrix} w_{1,1} & \cdots & \cdots & w_{1,p} & w_{1,m-1} & w_{1,l} \\ w_{2,1} & \cdots & \cdots & w_{1,p} & \cdots & w_{2,l} \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{N-1,1} & \cdots & \cdots & w_{N-1,p} & \cdots & w_{N-1,l} \\ w_{N,1} & \cdots & \cdots & w_{N,p} & w_{N,l-1} & w_{N,l} \end{bmatrix} \quad (39)$$

Before commencing work, the AOA ought to determine which search phase exploration or exploitation. The following search phrases employ the Math Optimizer Accelerated (MOA) function, whose factor was obtained using as per the Eq. (40).

$$MOA(B\_Iter) = Min + B\_Iter \times \left(\frac{Max-Min}{m\_Iter}\right) \quad (40)$$

$MOA(B\_Iter)$ is the function value at the C^{th}iteration, as identified as per Eq. (40). B_Iter defines current iteration between 1 and greatest number of iterations (m_Iter). The characters Min and Max, respectively, refer to minimum and maximum values of accelerated function.

Exploration phase

The AOA explores using the division (d) and multiplication (m) operators, which have scattered values and cannot easily approach the objective. By using four mathematical operations, the exploration search finds the nearly ideal solution. The operators (d and m) assist with the exploitation step of the search process.

The exploration stage of the AOA involves randomly investigating the search area using the division (d) and multiplication (m) operators. The Math Optimizer Accelerated (MOA) function guides this process based on a random number. The first operator (d) is activated

when $t2 < 0.5$, and it performs its task until completed. During this time, the second operator (m) remains idle. If $t2 >= 0.5$, the second operator (m) takes over the current task rather than d. The operators are activated based on specific conditions and promote diversification and exploration through a stochastic scaling coefficient. The exploration phase in AOA efficiently searches for better solutions while promoting exploration and diversification. The position-updating equations are proposed to facilitate the process as per the Eq. (41).

$$w_{d,p}(B\_Iter + 1) = \begin{cases} best(w_p) \div (mov+\in) \times \left((FE_p - LE_p) \times \mu + LE_p\right), & t < 0.5 \\ best(w_p) \div mov \times \left((FE_p - LE_p) \times \mu + LE_p\right), & otherwise \end{cases} \quad (41)$$

where $w_{d,p}(B\_Iter)$ indicates the p^{th} position in the d^{th} solution in the current iteration, $w_{d,p}(B\_Iter + 1)$ indicates the d^{th} solution in the following iteration, and $best(w_p)$ represents the p^{th} place in the best solution thus far, KC_i and AC_i represent upper and lower limit values of p^{th} slot, respectively, while ε is a small integer. The search process is controlled by μ, which has a fixed value of 0.5 based on the tests reported in this work as per the Eq. (42).

$$mov(B\_Iter) = 1 - \frac{B\_Iter^{1/\propto}}{m\_Iter^{1/\propto}} \quad (42)$$

Where (m_Iter) is extreme number of iterations, (mop) is coefficient, and function value at u^{th} iteration indicates mop (B_Iter) signifies the current iteration. The work's experiments indicate that the exploitation accuracy across repetitions, denoted by the sensitivity limit α, is set at 5.

Exploitation phase

The AOA employs an exploitation approach based on mathematical computations using Subtraction (s) or Addition (a) operators. These operators have low dispersion characteristics and are proficient at detecting near-optimal solutions. The exploitation phase utilizes these operators to facilitate enhanced communication and support within the optimization process.

The initialization of this stage is determined by the MOA function value, with specific requirements outlined. The exploitation operators thoroughly examine dense regions in the search area to approach and discover enhanced solutions based on two primary search strategies. These strategies are effectively represented as per the Eq. (43).

$$w_{d,p}(B\_Iter + 1) = \begin{cases} best(w_p) - mov \times \left((FE_p - LE_p) \times \mu + LE_p\right), & t3 < 0.5 \\ best(w_p) + mov \times \left((FE_p - LE_p) \times \mu + LE_p\right), & otherwise \end{cases} \quad (43)$$

Deep search is used to explore the search space, employing two main operators, s and a, to exploit the space. The first operator s is activated based on a condition, while the second operator a remains inactive. If $t3$ is less than 0.5 a takes over. These procedures resemble the partitioning approach but are designed to avoid getting stuck in local areas, ensuring optimal solutions and maintaining solution diversity. Stochastic values are generated to ensure exploration throughout iterations.

The final position in a search can be stochastic, estimating the near-optimal solution, while other solutions update their positions around this solution.

Algorithm 1 pseudocode of AOA

Start AOA

1. Setup the AOA Limits (Arithmetic Optimization Algorithm) α, μ.
2. Set the location of resolutions arbitrarily. (Solution: i=1,…,M.)
3. **While** ($B\_Iter < m\_Iter$) **do**
4. For specified resolution calculate the Fitness Function (FF)
5. Find finest answer as yet.
6. Utilizing Equation (40) modernize the mov
7. Using Equation (40) update the mov
8. **For** (d=1 to Answers) **do**
9. **For** (p=1 to Answers) **do**
10. Create arbitrary value among [0,1] ($t1$, $t2$, and $t3$)
11. **If** $t1$ > mov **then**
12. **Exploration phase**
13. **If** $t2 > 0.5$ **then**
14. (1) Implement division math operator (d "÷")
15. Using 1st rule in the Equation (41) apprise the d^{th} solution
16. **Else**
17. (2) Implement multiplication math operator (m "×")
18. employing 2nd rule in Equation (41) upgrade the d^{th} solution
19. **End if**
20. **Else**
21. **Exploitation phase**
22. **If** $t3 > 0.5$ **then**
23. (1) Implement subtraction math operator (s "-").
24. Utilizing 1st rule in the Equation (43) update the d^{th} solution.
25. **Else**
26. (2) Employ addition math operator (m "+").
27. Using 2nd rule in the Equation (43) update the d^{th} solution.
28. **End if**
29. **End if**
30. **End for**
31. **End for**
32. $B_{Iter} = B_{Iter} + 1$
33. **end while**
34. Return best result (w).

Dingo Optimization Algorithm

The Dingo Optimization Algorithm mimics the behavior of dingoes to solve optimization problems. It employs a population-based approach. Dingoes collaborate to find the optimal solution through exploration and exploitation of solution space. Each dingo represents a potential solution, also the algorithm evolves the population by adapting to the fittest solutions. The Dingo Optimization Algorithm is effective in diverse fields such as engineering, machine learning, and logistics, as it harnesses principles from the natural behavior of dingoes.

Mathematical Models

Dingo optimization is carried out in this part by the mathematical design of the model of the hunting, surrounding, and attacking prey.

Encircling

Dingoes are intelligent and locate their prey, surrounded by pack and alpha. The social hierarchy of dingos suggests the objective prey represents the best agent strategy, as the search region is unknown. Other search companies are updating their plans for potential strategies. The dingoes' behavior is patterned as per the Eq. (44), Eq. (45), Eq. (46), Eq. (47), Eq. (48).

$$\overrightarrow{B_b} = \left|\overrightarrow{B}.\overrightarrow{T_t}(v) - \vec{T}\,(p)\right| \tag{44}$$

$$\vec{T}(p+1) = \overrightarrow{T_t}\,(p) - \vec{A}.\vec{B}\,(b) \tag{45}$$

$$\vec{F} = 2.\overrightarrow{f_1} \tag{46}$$

$$\vec{A} = 2\vec{a}.\overrightarrow{f_2} - \vec{a} \tag{47}$$

$$\vec{a} = 3 - \left(P * \left(\frac{3}{P_{max}}\right)\right) \tag{48}$$

Where $\overrightarrow{B_b}$ is distance among dingo and prey. $\overrightarrow{T_t}$ be position vector (prey). $\vec{T}$ be the position vector (dingo). $\vec{F}$ and $\vec{A}$ be a Coefficient vector. $\overrightarrow{f_2}$ and $\overrightarrow{f_1}$ be the Random vector in [0,1]. $\vec{a}$ be a Linear reduction from 3 to 0 at all iterations. P be 1, 2, 3, ……, P_{max}. P_{max} be Maximum no. of iteration.

Hunting

Dingoes are unaware of the optimal prey location in their search area. However, alpha, beta, and other dingos know the potential location and can mimic hunting behavior. Alpha usually leads hunting, but beta or other dingos may also hunt. Two best values are considered to determine the optimal position for the prey. To do this, all dingoes must update their locations, which may be expressed mathematically as per the Eq. (49), Eq. (50), Eq. (51), Eq. (52), Eq. (53), Eq. (54).

$$\overrightarrow{B_\alpha} = \left|\overrightarrow{F_1}.\overrightarrow{T_\alpha} - \vec{T}\right| \tag{49}$$

$$\overrightarrow{B_\beta} = \left|\overrightarrow{F_2}.\overrightarrow{T_\beta} - \vec{T}\right| \tag{50}$$

$$\overrightarrow{B_q} = \left|\overrightarrow{F_3}.\overrightarrow{T_q} - \vec{T}\right| \tag{51}$$

$$\overrightarrow{T_1} = \left|\overrightarrow{T_\alpha} - \vec{A}.\ \overrightarrow{B_\alpha}\right| \tag{52}$$

$$\overrightarrow{T_2} = \left|\overrightarrow{T_\beta} - \vec{A}.\overrightarrow{B_\beta}\right| \tag{53}$$

$$\overrightarrow{T_3} = \left|\overrightarrow{T_q} - \vec{A}.\overrightarrow{B_q}\right| \tag{54}$$

To calculate intensity of every dingo, following as per the Eq. (55), Eq. (56), and Eq. (57) are being used,

$$P_\alpha = log\left(\frac{1}{G_\alpha - (1Z-100)} + 1\right) \tag{55}$$

$$P_\beta = log\left(\frac{1}{G_\beta - (1Z-100)} + 1\right) \tag{56}$$

$$P_q = log\left(\frac{1}{G_q - (1Z-100)} + 1\right) \tag{57}$$

where, G_α and G_β=α and β-dingo fitness values, correspondingly, G_q=another dingo fitness worth.

Attacking Prey

Dingo completes hunt by attacking if no position update. Approach is mathematically constructed by lowering value of $\vec{A}$. The range $\overrightarrow{B_\alpha}$ is decreased by $\vec{A}$, variable $\overrightarrow{B_\alpha}$ within an [-3a, 3a] interval is randomized, a value is decreased, when a variable has random values between [1, 1], the next position could be anywhere between current and preys.

Searching

Dingoes hunt according to the group's position. The DOX uses random values, $\vec{A}$, to scan targets worldwide, indicating approaching or running prey. A value greater than 1 indicates approaching prey, while a value less than -1 indicates running prey. $\vec{F}$ also increases exploration

Algorithm 2 pseudocode of Dingo Optimization Algorithm

Start DOA Input: The populace of dingoes D_n (i=1,2,..., n)
Output: The best dingo. (This being the case for detraction issue)

1. Create primary search agents Din
2. Set value for $\vec{a}$, $\vec{F}$ then $\vec{A}$
3. While completion situation not achieved do
4. Assess every dingo's capability then concentration charge.
5. B_α = dingo over top search
6. B_β = dingo over second-top search
7. B_q = dingoes finding outcomes subsequently
8. Iteration1
9. Repeat
10. For i=1: D_n do
11. Restart modern quest agent status
12. End for
13. Evaluate suitability then strength cost for dingoes
14. Record value for G_α, G_β and G_q
15. Record value of $\vec{a}$, $\vec{F}$, and $\vec{A}$
16. *Iteration* = *Iteration*+1
17. Check if *Iteration* >= *stopping*
18. Output End while

possibilities. The vector $\vec{F}$ in Eq. (46) can generate any random number between [0, 3] for varied prey weights. The variable DOX is a probabilistic variable, where vector ≤1 comes before vector ≥1 for analyze impact of the gap as stated in Eq. (44).

ADOA

In ADOA, the mean of the acquired best solution from AOA and DOX is computed. This means the computed best solution is the global best solution.

Advantages:

- The best solution will be efficient and a positive impact on defect detection.
- The best solution will find the defect fastly and accurately.
- It enhances workflow, enabling both teams and individuals to work more and provide better job outcomes.

The mean is for the utilization of the best solution in both algorithm

$$\boldsymbol{X_{best}} = \frac{\boldsymbol{best(w_p)} + \boldsymbol{B_{best}}}{2} \tag{58}$$

$\boldsymbol{X_{best}}$, be the overall global best solution. $\boldsymbol{best(w_p)}$ be the best solution (in terms of position) acquired from AOA. $\boldsymbol{B_{best}}$ be the best solution (in terms of position) acquired from DOX. Mathematically, ADOA algorithm can be given as per Eq. (58).

DefectFuse Classifier

The selection feature next comes to the defectfuse classifiers. The defectfuse classifiers is the combination of CNN, Bi-LSTM, RNN, and Voting classifier.

CNN

CNN are Strong deep learning models were developed particularly for applications related to computer vision and image processing. CNNs consist of pooling, fully connected, and complexity layers. These layers automatically extract layers of information from input images [33]. CNNs' capacity to capture spatial hierarchies and patterns makes them useful for applications like item identification, facial recognition, and image classification. CNNs are ANNs (artificial neural networks) extended to feature extraction from matrix datasets with grid-like structures. Shared-parameter neural networks are called CNNs. Consider a representation of image as a cuboid whose length, width, and height are the RGB channels and image's dimensions.

In a CNN model, the input x of every layer is arranged in three dimensions: height, width, and depth, or $n \times n \times s$, where n is similar as the width. Another name for the depth is the channel number. For instance, the depth (r) of an RGB picture is three. There are several kernels, or filters, denoted by l in each convolutional layer. Similar to the input image, they contain three dimensions ($m \times m \times r$); the only requirements are that m must be smaller than n and that q must either be equal to or smaller than s.

$$d^l = f(O^l * v + a^l) \tag{59}$$

Furthermore, the kernels serve as the foundation for the local connections, which, as was previously indicated, convolve with input and share identical characteristics (weight O^l and bias a^l) for producing l feature maps d^l with a size of $(n - m - 1)$ each f is the activation function applied element-wise v represents input data * represents the convolution operation. The inputs are small portions for original picture size, and convolution layer computes dot product among its input and weights as in Eq. (59).

Bi-LSTM

Bi-LSTM, is pattern model has two LSTM layers: one to process inputs in the direction of progress and another for handling in the reverse direction. It is typically applied to jobs using NLP. The idea behind this strategy is that the model can better comprehend the link between sequences by processing input in both directions.

The gradient of a recurrent neural network can quickly inflate also vanish in the gradient method when the time steps are excessively little or big. Consequently, LSTM utilizes a gating mechanism to regulate information to tackle this difficulty. The hidden state among earlier time step D_{u-1} and the present time step input V_u is the input of an LSTM gate. The entire connecting layer calculates the output as per the Eq. (60), Eq. (61), Eq. (62), Eq. (63), Eq. (64).

$$P_u = \sigma\left(V_u O_{vp} + D_{u-1} O_{dp} + a_p\right) \tag{60}$$

$$G_s = \sigma\left(V_u O_{vg} + D_{u-1} O_{dg} + a_g\right) \tag{61}$$

$$\widetilde{E_u} = \tan d\left(V_u O_{ve} + D_{u-1} O_{de} + a_e\right) \tag{62}$$

$$E_u = G_u \odot E_{u-1} + P_u \odot \widetilde{E_u} \tag{63}$$

$$Q_u = \sigma\left(V_u O_{vq} + D_{u-1} O_{dq} + a_q\right) \tag{64}$$

d signifies the number of secret units, V_u means the tiny batch input over the time step p, D_{u-1} denotes the secret state for preceding time step, σ represents the sigmoid function, O_{vp} and O_{dp} denote the weight matrix for gate used for input, also a_p is the input gate's offset term. O_{vg} and O_{dg} represent weight matrices for forgetting gate, and a_g is forgetting gate's offset term. $\widetilde{E_u}$ As potential memory cells, E_u represents the current cell state, E_{u-1} represents the prior cell state, O_{ve} and O_{de} are weight matrices of gated unit, and a_e is offset term for gated unit. O_{vq} and O_{dq} are weight matrix for output gate, then a_q is the offset term for output gate. The concealed state's information flow is regulated by multiplication by elements ⊙.

Data flow from memory cell to hidden state is regulated by output gate Q_u, and the resultant output D_u as per the Eq. (65).

$$D_u = Q_u \odot tan\, d(C_u) \tag{65}$$

The Bi-LSTM technique uses both forward and reverse

LSTM to extract features from data, considering all hidden details. This method can improve the results by merging two-way extraction results from two dimensions in a specific method, reducing the data provided by single LSTM and resulting in more comprehensive results.

RNN

An artificial neural network class called RNN is made for processing data sequentially. Because of their special design, they may be used for tasks like time series prediction, voice recognition, and language modeling, and they can retain hidden states that reflect temporal relationships. RNNs preserve the recollection of previously processed data as they step-by-step process incoming sequences. However, the problem of vanishing gradients is the reason behind its incapacity to express long-term dependency.

The Gated Recurrent Unit (GRU) is popular RNN unit that can capture relationships over various time scales adaptively. It features gating units that influence stream of detail, similar to LSTM units. However, GRU exposes the entire state every time and lacks regulation mechanism. Due to the disparity in length between questions and answers in AS, it is more suitable. To learn sentence representations, the hidden state d_u is calculated as per the Eq. (66), Eq. (67), Eq. (68), Eq. (69).

$$d_u = (1 - y_u) \circ d_{u-1} + y_u \circ \widetilde{d_u} \tag{66}$$

$$\widetilde{d_u} = \sigma\left(O\, o_u + C\left[s_u \circ d_{u\_1}\right] + a\right) \tag{67}$$

$$y_u = \sigma\left(O_y o_u + C_y d_{u-1} + a_y\right) \tag{68}$$

$$s_u = \sigma(O_s o_u + C_s d_{u-1} + a_s) \tag{69}$$

where O, O_y, O_s; C, C_y, C_s and a, a_y, a_s are network parameters.

Voting Classifiers

Voting classifiers in deep learning are used to detect defects by integrating the outputs of several different models to determine whether or not a flaw exists. This ensemble approach improves accuracy and resilience by using many models [34]. Voting classifiers are essential to obtaining high accuracy and resilient performance in defect detection applications because they use the advantages of many deep learning architectures to make collective choices regarding the existence of faults in a variety of materials or surfaces.

Result and Discussion

In this subdivision result and discussion of the suggested model is presented.

Experimental setup

The proposed model has applied Python. 70% of collected data has been used for training and 30% for testing. A comparative analysis has been made with state-of-the-art methods. The assessment considered several metrics like sensitivity, specificity, accuracy, precision, FPR, FNR, NPV, F-Measure and MCC.

Overall performance analysis for deep learning models

For Training Rate=70%

Table 1 shows the results acquired with the proposed as well as existing classifiers for 70%. The accuracy recorded by the proposed model (defectfuse classifier) is 97.4%, which is better than CNN=96.8%, Bi-LSTM =93.3%, RNN=95.6%, DCNN=93.4% due to the proposed model is the combination of CNN, Bi-LSTM, RNN. The precision is recorded by the proposed model is 88.5% which is better than CNN, Bi-LSTM, RNN, DCNN due to the involvement for ADOA. In addition, the planned model has recorded the highest sensitivity (88.5%) and specificity (98.5%) which is better than CNN, Bi-LSTM, RNN, DCNN because of the proposed model of feature selection process. The f_measure recorded by the proposed model is 88.5% which is better than remaining model due to the performance for combination of proposed model. The model recorded the high MCC of 87% than other models due to the performance of ADOA. The proposed model achives high NPV of 98.5% than other models for finding the Negative value. The model achives less FPR 1.4% and FNR 11.4% of proposed false positive and negative rate due to the involvement of deep learning technique.

For Training Rate=80%

Table 2 shows the results acquired with proposed as well as existing classifiers for 80%. The defectfuse classifier, a suggested model, outperforms other models with a 99% accuracy, outperforming CNN (97.1%), Bi-lstm (95.1%), RNN (96.4%), and DCNN (94.3%). The reason for this advantage is that the suggested model integrates RNN, CNN, and Bi_LSTM. In addition, the suggested model's precision of 95.7% beats CNN,

Table 1. Overall performance analysis for proposed Vs existing classifiers: for training rate=70%.

	Accuracy	Precision	Sensitivity	Specificity	F_Measure	MCC	NPV	FPR	FNR
CNN	0.968322	0.857447	0.857447	0.982181	0.857447	0.839628	0.982181	0.017819	0.142553
Bi-LSTM	0.933333	0.7	0.7	0.9625	0.7	0.6625	0.9625	0.0375	0.3
RNN	0.956974	0.806383	0.806383	0.975798	0.806383	0.782181	0.975798	0.024202	0.193617
DCNN	0.934279	0.704255	0.704255	0.963032	0.704255	0.667287	0.963032	0.036968	0.295745
PROPOSED	0.974468	0.885106	0.885106	0.985638	0.885106	0.870745	0.985638	0.014362	0.114894

Table 2. Overall performance analysis for proposed Vs existing classifiers: for training rate=80%.

	Accuracy	Precision	Sensitivity	Specificity	F_measure	MCC	NPV	FPR	FNR
CNN	0.971158	0.870213	0.870213	0.983777	0.870213	0.853989	0.983777	0.016223	0.129787
Bi-LSTM	0.9513	0.780851	0.780851	0.972606	0.780851	0.753457	0.972606	0.027394	0.219149
RNN	0.964539	0.840426	0.840426	0.980053	0.840426	0.820479	0.980053	0.019947	0.159574
DCNN	0.943262	0.744681	0.744681	0.968085	0.744681	0.712766	0.968085	0.031915	0.255319
PROPOSED	0.990544	0.957447	0.957447	0.994681	0.957447	0.952128	0.994681	0.005319	0.042553

Bi-LSTM, RNN, and DCNN since it makes use of a hybrid optimization technique. Furthermore, when contrasted to CNN, Bi-LSTM, RNN, and DCNN, the model that was suggested has the greatest sensitivity (95.7%) then specificity (99.4%). This benefit is ascribed to the property selection procedure that the proposed framework uses. The hybrid optimization approach allows the suggested model to outperform alternatives, resulting in a high MCC score of 95.2%. When it comes to recognizing negative values, the model recommended outperforms other models with a high NPV of 99.4%. The recommended model uses deep learning techniques, which lowers the FPR to 0.5% and the FNR to 4.2%. So, it is best approach for ceramic tile surface defect detection.

The superiority of the proposed defectfuse classifier over existing methods lies in its innovative design and its ability to address key challenges inherent in ceramic tile surface defect detection. Here's a breakdown of the justification based on the results:

- Integration of Deep Learning Architectures: By combining CNN, Bi-LSTM, and RNN architectures, the proposed model harnesses the complementary strengths of these networks. CNNs excel at spatial feature extraction, Bi-LSTMs handle sequential data effectively, and RNNs capture temporal

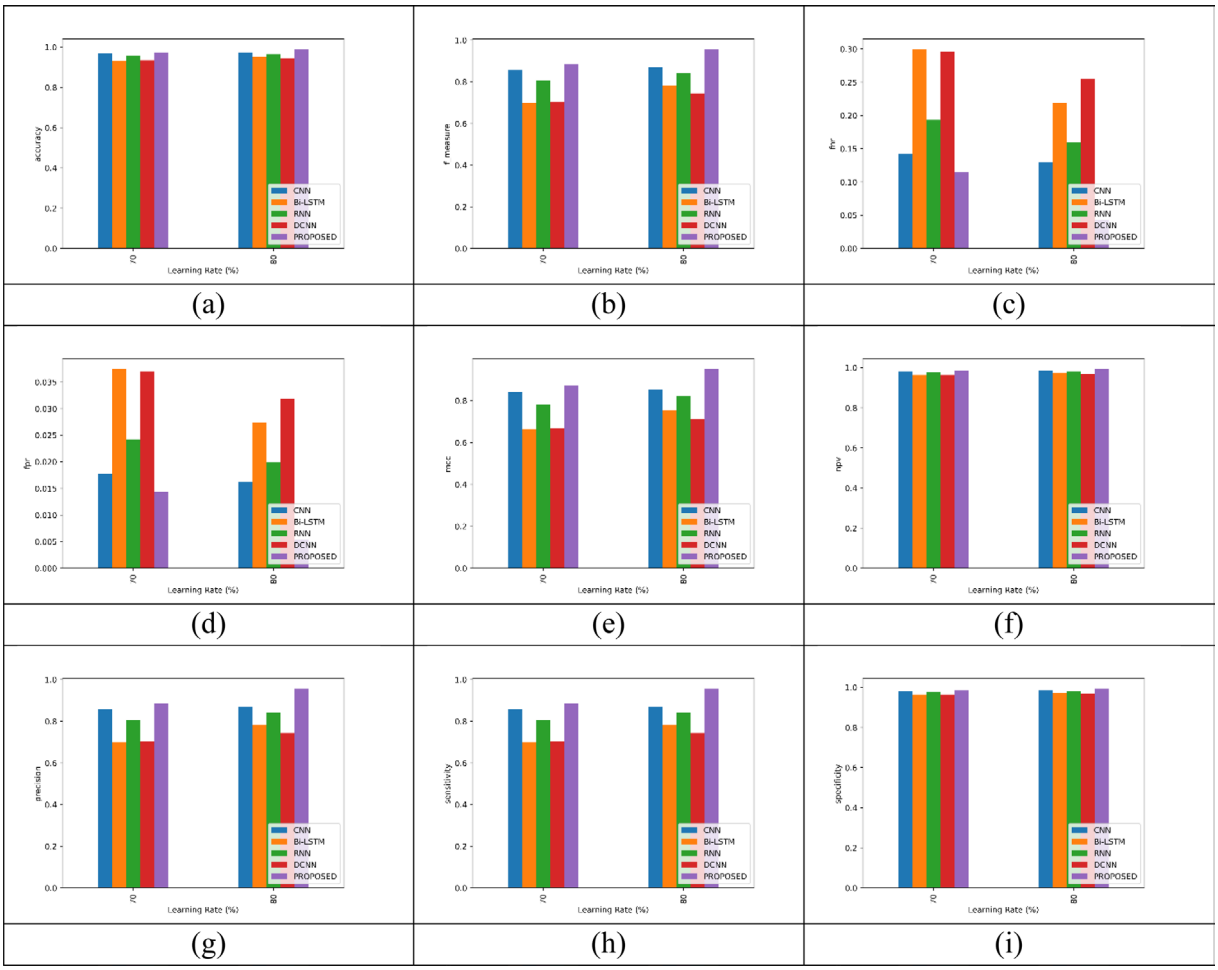

Fig. 7. Classifier performance Analysis for (a) accuracy, (b) F_measure, (c) FNR, (d) FPR, (e) MCC, (f) NPV, (g) Precision, (h) Sensitivity, (i) Specificity.

dependencies. This integration allows the model to capture intricate patterns and variations in ceramic tile images more comprehensively than single-network approaches.

- Hybrid Optimization for Feature Selection: The hybrid optimization approach, which combines arithmetic optimization algorithms and dingo optimization, ensures that the most relevant features are selected for defect detection. This optimization strategy enhances the model's ability to discriminate between defective and non-defective tiles by focusing on the most informative features while minimizing the risk of overfitting.
- Superior Performance Metrics: The experimental results demonstrate that the proposed defectfuse classifier consistently outperforms existing classifiers across various performance metrics, including accuracy, precision, sensitivity, specificity, MCC, and NPV. These metrics serve as objective measures of the model's effectiveness in accurately identifying defects while minimizing false positives and false negatives.
- Reduced Error Rates: Compared to existing classifiers, the proposed model exhibits lower false positive and false negative rates (FPR and FNR), indicating its ability to minimize both type I and type II errors. This reduction in error rates is crucial for ensuring that defective tiles are accurately identified and appropriate actions are taken to maintain product quality.
- Comprehensive Validation: The validation of the defectfuse classifier using a robust experimental setup, including training and testing on separate datasets, enhances the reliability and reproducibility of the results. The model's consistent performance across different training rates further validates its effectiveness across varying data scenarios.

In summary, the proposed defectfuse classifier offers a holistic solution to the challenges of ceramic tile surface defect detection by leveraging advanced deep learning architectures and innovative optimization techniques. Its superior performance metrics, coupled with reduced error rates and comprehensive validation, justify its superiority over existing methods in enhancing quality control processes in the ceramic tile manufacturing industry.

Conclusion

This research work has introduced a new deep learning approach for ceramic tile surface defect detection. (a) Data Acquisition (b) Pre-processing (c) Feature Extraction (d) Feature selection (e) Deep-learning based defect detection. Firstly, defect detection was conducted using images from the raw images of the sample. The collected images were pre-processed through median filtering and image augmentation. The processed images were endowed with features like HOG, LBP, Color Histograms, Gabor Filters, and Edge Detector. Among these features, the optimal features were determined using the new ADOA (combined arithmetic optimization algorithms and dingo optimization). Subsequently, the defect was detected with the proposed model using a new Defectfuse classifier model. The proposed Defectfuse classifier model comprised deep learning classifications such as CNN, Bi-LSTM, and RNN, respectively. Finally, the results for voting classifiers were validated. Python was used to performed in the system.

References

1. Q. Lu, J. Lin, L. Luo, Y. Zhang, and W. Zhu, Adv. Eng. Inform. 53[8] (2022) 101692.
2. Y. Sun, T. Wu, Y. Bao, Y. Li, D. Wan, K. Li, and L. He, Int. J. Appl. Ceram. Technol. 19[1] (2022) 604-611.
3. C. Wang, S. Wang, X. Li, Y. Liu, X. Zhang, Q. Chang, and Y. Wang, Int. J. Appl. Ceram. Technol. 18[3] (2021) 1052-1062.
4. G. Wan, H. Fang, D. Wang, J. Yan, and B. Xie, Ceram. Int. 48[8] (2022) 11085-11093.
5. R. Alamsyah and A.D. Wiranata, Sci. Res. J. VII[4] (2019) 41-45.
6. B. Zorić, T. Matić, and Ž. Hocenski, ISA Trans. 125[6] (2022) 400-414.
7. X. Yu, Q. Yu, Q. Mu, Z. Hu, and J. Xie, Appl. Sci. 13[21] (2023) 12057.
8. O. Stephen, U.J. Maduh, and M. Sain, Electronics 11[1] (2021) 55.
9. S. Ye and L. Sun, J. Phys.: Conf. Ser. 1650[3] (2020) 032045.
10. K.B. Kim, D.H. Song, and H.J. Park, Indones. J. Electr. Eng. Comput. Sci. 19[3] (2020) 1505-1511.
11. N.T. Huynh, D.D. Ho, and H.N. Nguyen, Sustainability 15[6] (2023) 5455.
12. Z. Zhou, Q. Lu, Z. Wang, and H. Huang, Sensors 19[22] (2019) 5000.
13. K. Wang, Z. Li, and X. Wang, Appl. Sci. 12[3] (2022) 1249.
14. M.F. Ohemu, Z.O. Zubair, and R.E. Donatus, ATBU J. Sci. Technol. Educ. 10[2] (2022) 170-180.
15. A. Ravendran and S. Rianmora, J. Comput. Appl. Res. Mech. Eng. 10[2] (2021) 391-404.
16. J. Wang, G. Yi, S. Zhang, and Y. Wang, Appl. Sci. 11[1] (2020) 283.
17. H.S. Nogay, T.C. Akinci, and M. Yilmaz, Neural Comput. Appl. 34[2] (2022) 1423-1432.
18. J. Wang, Z. Wu, W. Hu, C. Song, X. Guo, P. Cao, L. Yang, and Z. Zhu, Investigation on distributed rescheduling with cutting tool maintenance based on nsga-iii in large-scale panel furniture intelligent manufacturing. Journal of Manufacturing Processes, 112(Feb.) (2024) 214-224.
19. F. Sušac, T. Matić, I. Aleksi, and T. Keser, Bull. Pol. Acad. Sci. Tech. Sci. 69[3] (2021) e137121.
20. Oglakci, B., et al. "The Effect of Different Polishing Systems on the Surface Roughness of Nanocomposites: Contact Profilometry and SEM Analyses." Operative Dentistry 46[2] (2021) 173-187.
21. S. Boovaneswari, S. Nirmal, V. Meyappan, and B. Nandakumar, Semicond. Optoelectron. 42[1] (2023) 1082-1095.
22. W. Qian, X. Yu, W. Xie, L. Zhang, and W. Wu, J. Phys.:

Conf. Ser. 2044[1] (2021) 012105.
23. X. Le, J. Mei, H. Zhang, B. Zhou, and J. Xi, Neurocomputing 408[9] (2020) 112-120.
24. G.S. Junior, J. Ferreira, C. Millán-Arias, R. Daniel, A.C. Junior, and B.J. Fernandes, Appl. Sci. 11[13] (2021) 6017.
25. H.R. Ravikumar, Y.K. Sharma, and S. Raghav, Int. J. Innov. Sci. Eng. Technol. 7[2] (2020) 252-263.
26. K. Avazov, M.K. Jamil, B. Muminov, A.B. Abdusalomov, and Y.I. Cho, Sensors 23[16] (2023) 7078.
27. S. Gundu and H. Syed, Sensors 23[5] (2023) 2569.
28. P. Banerjee, A.K. Bhunia, A. Bhattacharyya, P.P. Roy, and S. Murala, Expert Syst. Appl. 113[12] (2018) 100-115.
29. F. Bianconi, A. Fernández, F. Smeraldi, and G. Pascoletti, J. Imaging 7[11] (2021) 245.
30. H. Kim, Y. Arisato, and J. Inoue, Unsupervised segmentation of microstructural images of steel using data mining methods. Computational Materials Science. 201[5] (2022) 110855.
31. G. Lambard, K. Yamazaki, and M. Demura, Generation of highly realistic microstructural images of alloys from limited data with a style-based generative adversarial network. Scientific Reports 13[1] (2023) 566.
32. L. Abualigah, A. Diabat, S. Mirjalili, M. Abd Elaziz, and A.H. Gandomi, Comput. Methods Appl. Mech. Eng. 376[4] (2021) 113609.
33. D. Bhatt, C. Patel, H. Talsania, J. Patel, R. Vaghela, S. Pandya, K. Modi, and H. Ghayvat, Electronics 10[20] (2021) 2470.
34. E. Cumbajin, N. Rodrigues, P. Costa, R. Miragaia, L. Frazão, N. Costa, A. Fernández-Caballero, J. Carneiro, L.H. Buruberri, and A. Pereira, J. Imaging 9[10] (2023) 193.