Journal of Theoretical and Applied Information Technology
31st May 2025. Vol.103. No.10

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

SELF-RELIANT RESIDUAL NETWORK BASED DEEP LEARNING FRAMEWORK FOR MELANOMA SKIN DISEASE DETECTION

V. RADHIKA[1], A. MUTHUCHUDAR[2], M. LINGARAJ[3]

[1]Principal, [2] Assistant Professor, Associate Profes sor Department of Computer Science,

Sankara College of Science and Commerce, India

E-mail: [1] radhikavin73@yahoo.com, [2]muthuchudara@sankara.ac.in, [3]lingarajm@sankara.ac.in

ABSTRACT

Melanoma is one of the deadliest types of skin cancer and one of the most aggressive that may be if caught late. While traditional approaches may have their limits, an accurate diagnosis is vital for patient survival. Thanks to its capacity to understand intricate patterns from massive datasets, deep learning has evolved as a potential method for automated melanoma diagnosis. The challenges persist, including overfitting, instability during training, and difficulties in handling nonlinearities, which can hinder accurate predictions. To address these challenges, Self-Reliant ResNet (SR-ResNet) has been proposed. This enhanced version of ResNet integrates Zoutendijk's Method, a nonlinear optimization technique, to optimize weight updates and improve convergence. SR-ResNet features a series of residual blocks where Zoutendijk's Method refines the learning process, ensuring stability and efficient training, even in deeper networks. The network's architecture has been designed to enhance performance and generalization. The proposed SR-ResNet has been evaluated using a dataset of 10,000 Melanoma Skin Cancer images. The results demonstrate significant improvements in classification accuracy, achieving a high precision rate with reduced overfitting. SR-ResNet outperforms traditional models, establishing itself as a robust tool for melanoma diagnosis.

Keywords: *Melanoma Skin Cancer, Deep Learning, SR-ResNet, Zoutendijk's Method, Classification Accuracy*

1. INTRODUCTION

Melanoma Skin Cancer has emerged as a critical public health concern, recognized for its aggressive nature and high mortality rate if not diagnosed early. This cancer, originating in melanocytes—the cells responsible for producing melanin—demands early detection and precise diagnosis to improve survival rates[1]. Traditional methods, relying on visual examination and biopsy, have exhibited limitations, potentially leading to delays in treatment. The onset of the COVID-19 pandemic has further exacerbated this issue, disrupting routine healthcare services and causing a gradual increase in Melanoma Skin Cancer cases as patients have missed early screenings. This surge has underscored an urgent need for more reliable and advanced diagnostic tools that enhance accuracy and accessibility in the post-pandemic era[2], [3].

Figure 1a shows a benign skin lesion, such as a mole or benign nevus, which is non-cancerous and does not evolve into melanoma. These lesions typically do not invade surrounding tissues or spread to other parts of the body, making them generally harmless.Figure 1b, on the other hand, depicts a malignant skin lesion, specifically melanoma, a dangerous form of skin cancer[4]. Melanoma can grow aggressively, invade nearby tissues, and metastasize to distant organs. Accurate diagnosis and early intervention are crucial for effective treatment of melanoma[4].

Deep learning has gained prominence as a transformative approach in medical imaging, revolutionising how melanoma is detected and classified. Convolutional Neural Networks (CNNs) have emerged as a leading deep learning architecture, showing significant promise in analyzing complex patterns within medical images[5]. Despite these improvements, melanoma detection using deep learning models has been quite difficult. One of the biggest problems is overfitting, which occurs when a model does very well on training data but doesn't adapt well to new, unknown data. A lack of stability in the training process has also been a significant hurdle, often

resulting in difficulties with convergence in deeper networks. [6], [7].

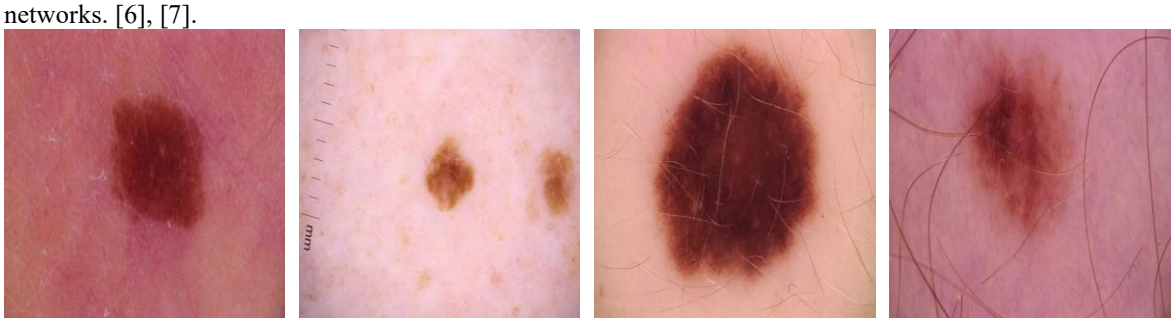

Figure 1a. Benign Skin Lesion

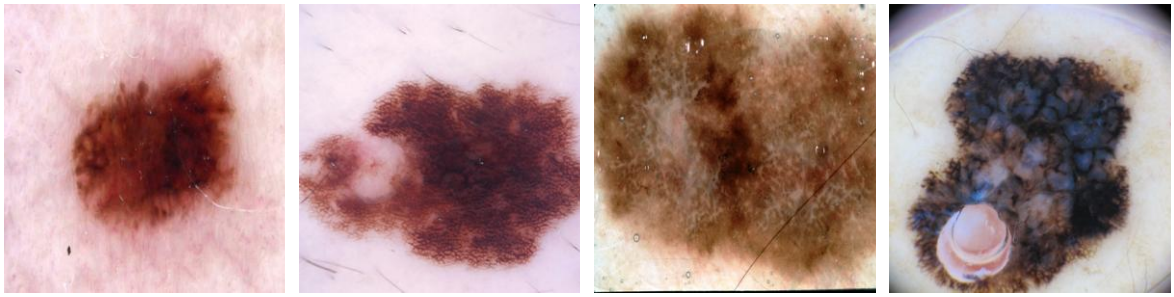

Figure 1b. Malignant Skin Lesion

Self-Reliant Residual Network (SR-ResNet) has been proposed to overcome these challenges, representing an advancement over the traditional ResNet architecture. SR-ResNet integrates Zoutendijk's Method, a nonlinear optimization technique designed to enhance the training process. While ResNet is known for its powerful architecture, which includes residual connections that mitigate the vanishing gradient problem in deep networks, SR-ResNet builds upon this by optimizing weight updates to improve convergence and stability. IntegratingZoutendijk's Method ensures that each training step moves the model toward an optimal solution, effectively addressing issues related to overfitting and instability that have previously hindered deep-learning models in melanoma detection.

By refining the learning process, SR-ResNet has established itself as a robust and reliable tool for melanoma detection, offering significant improvements in accuracy and stability. This model's ability to address the specific challenges posed by melanoma data positions it as a critical advancement in the field, particularly during the COVID-19 pandemic, where the need for early and accurate cancer detection has become increasingly pressing. The healthcare landscape continues to evolve in response to ongoing challenges, with SR-ResNet providing a promising solution for improving patient outcomes in melanoma diagnosis, helping to bridge the gap left by traditional methods and ensuring that more patients receive timely and accurate care.

Despite advancements in melanoma detection, existing models suffer from overfitting, unstable convergence, and limited generalization, especially in deeper networks. Ensemble and transfer learning methods offer improvements but lack optimization efficiency and adaptability. Current approaches often overlook nonlinear optimization within residual blocks. This work introduces SR-ResNet, which integrates Zoutendijk's Method to optimize weight updates, ensuring stable convergence and improved classification accuracy. By addressing a critical gap in convergence-optimized deep learning, SR-ResNet provides a robust solution tailored for melanoma detection, outperforming traditional models and offering a scalable, precise diagnostic framework suitable for clinical deployment.

2. LITERATURE REVIEW

Ensemble Embeddings Approach [8] has explored melanoma detection by integrating an ensemble of machine learning models with deep feature embeddings, demonstrating significant improvements in accuracy. The study highlights the advantages of combining traditional machine learning algorithms with deep learning features, enhancing overall predictive performance. The

proposed ensemble method mitigates overfitting and increases robustness, making it a promising approach for melanoma detection in clinical settings. MuSClD System [9] has introduced Multi-site Cross-organ Calibrated Deep Learning (MuSClD), an automated system for diagnosing non-melanoma skin cancer. The research emphasizes the importance of calibrating deep learning models across multiple sites to improve generalizability. The proposed model leverages cross-organ calibration to enhance the accuracy of diagnosis, addressing the challenges associated with varying data distributions across different clinical settings. Interpretable DL System [10] have developed an interpretable deep learning system for multi-class segmentation and classification of non-melanoma skin cancer. Their approach combines deep learning with interpretability techniques, allowing clinicians to understand the decision-making process of the model. The study demonstrates that interpretable models can achieve high accuracy while providing insights into the model's behavior, which is crucial for clinical adoption.

Grasshopper DL Hybrid [11] have proposed a hybrid Grasshopper optimization algorithm combined with deep learning for skin lesion segmentation and melanoma classification. The research integrates evolutionary algorithms with deep learning to optimize the segmentation process, improving classification accuracy. The hybrid approach addresses the limitations of traditional deep learning models in segmenting complex skin lesions, providing a more reliable tool for melanoma diagnosis. Hyperspectral Signature Learning [12]has been explored to classify actinic keratosis and non-melanoma skin cancers using near-infrared hyperspectral signatures. This research shows that hyperspectral imaging in conjunction with machine learning can effectively differentiate between various skin lesions. Their technology provides an alternative to invasive procedures for early identification of skin cancer, which has the potential to greatly improve diagnostic accuracy.Deep Learning Classifier [13]studied the efficacy of several deep learning architectures on datasets consisting of skin lesion information in order to identify and categorize skin cancers. Findings show that deep learning algorithms can detect skin cancer more accurately than conventional approaches, suggesting that they may eventually replace them. It delves into the difficulties encountered by deep learning models, including overfitting and the requirement for extensive datasets.

Optimized DL Segmentation [14]highlights the importance of accurate segmentation for better diagnosis accuracy by optimizing deep learning methods for skin lesion segmentation and skin cancer detection. Their research incorporates advanced deep-learning architectures tailored to handle the unique challenges posed by skin lesion images. The study demonstrates that optimized deep-learning models can achieve superior performance in skin cancer detection, particularly in challenging cases. Hybrid DL Framework [15] have introduced a hybrid deep learning framework for predicting skin cancer, combining various deep learning techniques to improve accuracy. The framework integrates convolutional neural networks with other deep learning methods, resulting in a robust model capable of handling diverse skin lesion images. Their approach offers a comprehensive solution for skin cancer prediction, enhancing the model's generalizability across different datasets. Polarimetric ML Classifier [16] has used polarimetric imaging with machine learning to classify non-melanoma skin cancer in mice tissues. The study leverages optical parameters derived from polarimetric imaging to improve classification accuracy. Their findings suggest combining polarimetric imaging with machine learning provides a novel approach to skin cancer diagnosis, potentially offering more accurate results than traditional imaging methods. Transfer Learning Hybrid[17]identify melanoma skin cancer by using transfer learning for segmentation in conjunction with hybrid classification. This study aims to improve melanoma detection systems by utilizing pre-trained models, especially in cases when labeled data is few. The work shows that melanoma diagnosis may be much improved using transfer learning and hybrid classification algorithms, which makes it a viable tool for clinical situations.

Ensemble Transfer Learning (ETL) [18]is utilizing deep transfer learning and ensemble machine learning in a combined effort to classify melanoma. In order to improve classification accuracy, the study investigates using deep learning in conjunction with several machine learning models. The ensemble approach addresses the challenges of overfitting and model robustness, providing a more reliable method for melanoma detection. Their findings indicate combining traditional and deep learning methods can lead to

superior classification performance.DL Meta-analysis (DLMA)[19]have reviewed and analyzed deep learning algorithms for dermoscopy-based melanoma diagnosis in a systematic way. In order to assess deep learning's efficacy in melanoma diagnosis, their study combines results from several investigations. While deep learning models have demonstrated promising results in increasing diagnosis accuracy, the study notes that there is still a need for more research into issues like data quality and model interpretability.Bio-inspired Optimization plays a significant role in different research to achieve the best result [20]-[53]. Even in skin cancer detection also, bio-inspired optimization can be applied.

3. SELF-RELIANT RESIDUAL NETWORK

Self-Reliant ResNet (SR-ResNet) represents a substantial advancement over traditional ResNet architectures by integratingZoutendijk's Method, a sophisticated nonlinear optimization technique. By leveraging Zoutendijk's Method, SR-ResNet optimizes the weight update process during training, significantly improving convergence stability and computational efficiency. This integration effectively mitigates issues such as gradient vanishing and overfitting, which are common challenges in deep learning models, particularly in complex tasks like melanoma detection. The enhanced optimization process enables SR-ResNet to achieve more precise and reliable predictions for melanoma. It is a powerful tool in applications that demand high accuracy and robustness in detecting this aggressive form of skin cancer.

3.1. Input Layer

The first step in SR-ResNet involves the input layer, where the raw data, typically an image, is fed into the network. The input image can be represented as a tensor X of dimensions $H \times W \times C$, where H represents the height, W represents the width and denotes the number of channels, usually corresponding to the color channels in an RGB image $(C = 3)$. The mathematical formulation of the input tensor can be described as:

$$X = [x_{ijk}], \text{ for } i = 1,2,\dots H; j = 1,2,\dots,W; k = 1,2,\dots,C \quad (1)$$

In the initial convolutional layer, the input tensor X undergoes a convolution operation with a set of filters of kernels, denoted as W_f, where W_f is a four-dimensional tensor of dimensions $F_h \times F_w \times C \times N$. Here, F_h and F_w denote the height and width of the filter, respectively, and N represents the number of filters applied. The convolution operation between$X$ and W_f can be mathematically represented as:

$$Y_{mn} = \sum_{i=1}^{F_h} \sum_{j=1}^{F_w} \sum_{k=1}^{C} X_{(m+i-1)(n+j-1)k} W_{ijkn}^{f} \quad (2)$$

where $Y_{mn}$is the output feature map resulting from the convolution operation at position (m,n). The above equation computes the weighted sum of the input pixel values and the corresponding filter weights, producing an output feature map of reduced dimensions, which captures the essential features from the input image.

A nonlinearity is introduced into the model after the convolution process by applying an activation function σ element-wise to the output feature map Y. The mathematical expression of the most used activation function, the Rectified Linear Unit (ReLU):

$$Z_{mn} = \sigma(Y_{mn}) = \max(0, Y_{mn}) \quad (3)$$

where $Z_{mn}$is the activated output at position (m,n). This operation retains positive values and sets all negative values to zero, which helps to prevent the vanishing gradient problem.

In specific architectures, a max-pooling operation follows the activation function, reducing the spatial dimensions of the feature maps by selecting the maximum value from non-overlapping subregions within the feature map. This operation can be defined as:

$$P_{ij} = max\left(Z_{(i+a-1)(j+b-1)}\right), \text{ for } a = 1,2,\dots,p_h; b = 1,2,\dots,p_w \quad (4)$$

where P_{ij} is the pooled output, $p_h \times p_w$represents the size of the pooling window, and Z is the activated feature map. This pooling process reduces the computational complexity and provides spatial invariance, thereby setting up the initial feature map P for further processing in the subsequent layers of SR-ResNet.

3.2. Initial Convolution and Pooling

The initial feature map undergoes further processing through convolutional layers within the residual block. The initial feature map P, obtained after the first convolutional and pooling operations, is passed through a series of convolutional layers designed to extract deeper features. Let P have dimensions.$H' \times W' \times C'$.The first convolutional

layer within this block applies a set of filters.W_1, with dimensions $F_h \times F_w \times C' \times N_1$, where N_1 represents the number of filters. The operation is defined as:

$$Y_1^{(l)}(m,n) = \sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{C'} P_{(m+i-1)(n+j-1)k} W_{ijk}^{(1)} \quad (5)$$

where $Y_1^{(l)}(m,n)$ represents the output feature map after the first convolutional operation in layer l.

A batch normalization step is applied to the output feature map.$Y_1^{(l)}$, which normalizes the output to have a mean of zero and a variance of one. Let $\mu_{Y_1^{(l)}}$and $\sigma^2_{Y_1^{(l)}}$ represent the mean and variance, respectively:

$$\hat{Y}_1^{(l)}(m,n) = \frac{Y_1^{(l)}(m,n) - \mu_{Y_1^{(l)}}}{\sqrt{\sigma^2_{Y_1^{(l)}} - \epsilon}} \quad (6)$$

where ϵ is a small constant added for numerical stability.

Following batch normalization, the activation function σ, typically ReLU, is applied:

$$Z_1^{(l)}(m,n) = \sigma\left(\hat{Y}_1^{(l)}(m,n)\right) = max\left(0, \hat{Y}_1^{(l)}(m, \right. \quad (7)$$

This process is repeated in subsequent convolutional layers within the same residual block. Consider the next convolutional layer with filters.W_2 of dimensions $F_h \times F_w \times N_1 \times N_2$, where N_2 is the number of filters in the second convolutional layer. The output of this layer is:

$$Y_2^{(l)}(m,n) = \sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{N_1} Z_1^{(l)}{}_{((m+i-1)(n+j-1)k)} W \quad (8)$$

Batch normalization is again applied to $Y_2^{(l)}$:

$$\hat{Y}_2^{(l)}(m,n) = \frac{Y_2^{(l)}(m,n) - \mu_{Y_2^{(l)}}}{\sqrt{\sigma^2_{Y_2^{(l)}} + \epsilon}} \quad (9)$$

Following normalization, the ReLU activation function is applied:

$$Z_2^{(l)}(m,n) = \sigma\left(\hat{Y}_2^{(l)}(m,n)\right) = max\left(0, \hat{Y}_2^{(l)}(m, \right. \quad (10)$$

In a standard residual block, these operations are followed by adding the original input P to the final output of the convolutional layers. In SR-ResNet, this step incorporates Zoutendijk's Method for optimization.The residual function is denoted as:

$$R^{(l)} = Z_2^{(l)}(m,n) \quad (11)$$

The input P is added to the residual function.$R^{(l)}$, yielding the output $F^{(l)}$ of the residual block:

$$F^{(l)}(m,n) = P(m,n) + R^{(l)}(m,n) \quad (12)$$

Zoutendijk's Method is then applied to optimize $F^{(l)}(m,n)$, where the feasible direction $d^{(l)}$ and step size $\alpha^{(l)}$ are calculated. The update step for the parameters $\theta^{(l)}$ involves moving in the direction $d^{(l)}$ by a step size $\alpha^{(l)}$:

$$\theta^{(l+1)} = \theta^{(l)} - \alpha^{(l)} d^{(l)} \quad (13)$$

The optimized output $F^{(l)}(m,n)$ becomes the input to the next block in the SR-ResNet, ensuring efficient training and enhanced convergence throughout the network.

3.3. Residual Block with Zoutendijk's Optimization

The process continues with stacking multiple residual blocks, each incorporating the enhanced optimization technique inspired by Zoutendijk's Method. The input to each subsequent residual block is the output from the previous block, optimized to improve training stability and convergence. Let $F^{(l)}$ represent the output of the l*th*residual block, which serves as the input to the next block.The input $F^{(l)}$ is first passed through a convolutional layer with filters W_3, defined by the dimensions $F_h \times F_w \times C'' \times N_3$, where C'' is the number of channels in $F^{(l)}$, and N_3 is the number of filters. The operation is expressed as:

$$Y_3^{(l)}(m,n) = \sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{C''} F_{(m+i-1)(n+j-1)k}^{(l)} W_{ijk}^{(3)} \quad (14)$$

Following this convolution, the batch normalization step is applied to the output feature map.$Y_3^{(l)}$, with the mean $\mu_{Y_3^{(l)}}$ and variance $\sigma^2_{Y_3^{(l)}}$ calculated as:

$$\hat{Y}_3^{(l)}(m,n) = \frac{Y_3^{(l)}(m,n) - \mu_{Y_3^{(l)}}}{\sqrt{\sigma^2_{Y_3^{(l)}} + \epsilon}} \quad (15)$$

The output $\hat{Y}_3^{(l)}$is then passed through the activation function, typically ReLU:

$$Z_3^{(l)}(m,n) = \sigma\left(\hat{Y}_3^{(l)}(m,n)\right) = max\left(0, \hat{Y}_3^{(l)}(m \right. \quad (16)$$

The process continues with another convolutional layer, this time using filters.W_4 with dimensions $F_h \times F_w \times N_3 \times N_4$, where N_4 represents the number of filters in the next layer. The convolution operation is given by:

$$Y_4^{(l)}(m,n) = \sum_{i=1}^{F_h}\sum_{j=1}^{F_w}\sum_{k=1}^{C''} Z_3^{(l)}((m+i-1),(n+ \quad (17)$$

Batch normalization is again applied:

$$\hat{Y}_4^{(l)}(m,n) = \frac{Y_4^{(l)}(m,n) - \mu_{Y_4^{(l)}}}{\sqrt{\sigma^2_{Y_4^{(l)}} + \epsilon}} \quad (18)$$

The activated output is then:

$$Z_4^{(l)}(m,n) = \sigma\left(\hat{Y}_4^{(l)}(m,n)\right) = max\left(0, \hat{Y}_4^{(l)}(m \quad (19)$$

This output $Z_4^{(l)}$ serves as the residual function $R^{(l+1)}$ for the next layer. To maintain continuity, the original input $F^{(l)}$ is added to this residual function, forming the input to the next block:

$$F^{(l+1)}(m,n) = F^{(l)}(m,n) + R^{(l+1)}(m,n) \quad (20)$$

At this stage, Zoutendijk's Method is applied again. The objective function $L\left(F^{(l+1)}\right)$, representing the loss, is optimized by determining a feasible direction $d^{(l+1)}$ and an optimal step size $\alpha^{(l+1)}$:

$$\theta^{(l+1)} = \theta^{(l)} - \alpha^{(l+1)} d^{(l+1)} \quad (21)$$

The optimized set of parameters $\theta^{(l+1)}$ updates the weights within the block, ensuring that the network continues to learn efficiently. The output $F^{(l+1)}$ becomes the input for the subsequent residual block, preserving the gradient flow and enhancing the overall network performance.By repeating this process, SR-ResNet constructs a deep network where each residual block is optimized for improved convergence, leading to better model accuracy and efficiency in learning complex patterns.

3.4. Stacking SR-ResNet Blocks

The output from the series of residual blocks is processed through global average pooling, fully connected layers, and the final optimization steps to produce the network's output. The input to this step is the output from the last residual block, denoted as $F^{(L)}$, where L is the index of the final residual block in the network.The global average pooling operation is applied to $F^{(L)}$, which has dimensions $H'' \times W'' \times C''''$.The feature maps are aggregated into a single value using the global average pooling method, which averages all the spatial components.This operation is mathematically expressed as:

$$G(c) = \frac{1}{H'' \times W''}\sum_{m=1}^{H''}\sum_{n=1}^{W''} F^{(L)}(m,n,c) \quad (22)$$

where $G(c)$ is the pooled output corresponding to the cth feature map, and $c = 1,2,\ldots C''''$. The result is a vector G of length.C''''.

This pooled vector G is then passed to a fully connected layer with a weight matrix W_{fc} and bias vector b_{fc}. The fully connected layer computes the output as:

$$h_i = \sum_{c=1}^{C''''} G(c).W_{fc}(c,i) + b_{fc}(i) \quad (23)$$

where h_i is the output of the ith neuron in the fully connected layer, and $i = 1,2,\ldots N_{fc}$, withN_{fc} being the number of neurons in the fully connected layer.

After the fully connected layer, the outputs are transformed into probabilities using a softmax activation function. The softmax function for the ith class is defined as:

$$p_i = \frac{exp(h_i)}{\sum_{j=1}^{N_{fc}} exp(h_j)} \quad (24)$$

where p_i represents the predicted probability of the ith class.

The loss function $L(\theta)$is determined during training by using cross-entropy loss, which quantifies the discrepancy between the actual labels and the anticipated probabilities p_i.The cross-entropy loss is given by:

$$L(\theta) = -\sum_{i=1}^{F_{fc}} y_i log(p_i) \quad (25)$$

where y_i is the ground truth label for the ithclass.

To optimize the network, Zoutendijk's Method is applied, which involves determining the feasible direction d and the optimal step size α for minimizing the loss function. The weight updates for the fully connected layer are performed as:

$$\theta_{fc}^{(t+1)} = \theta_{fc}^{(t)} - \alpha^{(t)} d_{fc}^{(t)} \quad (26)$$

where $\theta_{fc}^{(t)}$ represents the weights at iteration t, and $d_{fc}^{(t)}$ is the feasible direction determined by Zoutendijk's Method.

The back propagation process updates the parameters throughout the network, including the weights in the convolutional layers and fully connected layers, ensuring the optimization of the overall objective function $L(\theta)$. The final output of SR-ResNet is the class with the highest probability from the softmax layer, representing the network's prediction.

3.5. Bottleneck Blocks

The primary focus is optimizing the entire network using Zoutendijk's Method, which is integrated with the backpropagation algorithm. The optimization process begins with calculating gradients for all network parameters, followed by determining feasible directions and optimal step sizes for weight updates.To initiate the optimization, the loss function $L(\theta)$, where θ represents the entire set of network parameters, is computed. Each parameter θ_k is used to represent the gradient of the loss function.

$$\nabla_{\theta_k} L(\theta) = \frac{\partial L(\theta)}{\partial \theta_k} \quad (27)$$

The chain rule is used to compute the gradient.$\nabla_{\theta^{(l)}} L(\theta)$for every layer$l$ in the network. To minimize the loss, the parameters might be modified in the direction indicated by this gradient.The gradient at layer l for weight $W^{(l)}$ is given by:

$$\nabla_{W^{(l)}} = \frac{\partial L(\theta)}{\partial h^{(l)}} \cdot \frac{\partial h^{(l)}}{\partial W^{(l)}} \quad (28)$$

The output of the layer before adding an activation function is represented by $h^{(l)}$.Once the gradients are calculated, Zoutendijk's Method is applied to determine a feasible direction.$d^{(l)}$ for each parameter update. This involves solving the optimization problem:

$$minimize \nabla_{\theta^{(l)}} L(\theta)^{\mathrm{T}} d^{(l)} \quad (29)$$

$$subject to C(\theta^{(l)} + \alpha^{(l)} d^{(l)}) \leq 0 \quad (30)$$

where $C(\theta^{(l)})$represents any constraints on the parameters and $\alpha^{(l)}$ is the step size. The direction $d^{(l)}$is selected to ensure that the loss decreases while maintaining feasibility concerning the constraints. The optimal step size $\alpha^{(l)}$is determined by minimizing the loss along the direction $d^{(l)}$:

$$\alpha^{(l)} = \arg\min_{\alpha} L(\theta^{(l)} + \alpha d^{(l)}) \quad (31)$$

The parameters $\theta^{(l)}$ for layer,l are then updated using the feasible direction $d^{(l)}$ and the calculated step size $\alpha^{(l)}$ as follows:

$$\theta^{(l+1)} = \theta^{(l)} - \alpha^{(l)} d^{(l)} \quad (32)$$

This update rule is applied to all layers, ensuring that each set of parameters is optimized according to Zoutendijk's Method. The process is iterative, with gradients recalculated after each update until convergence is achieved, i.e., when the loss function $L(\theta)$ stabilizes at a minimum or near-minimum value.The back propagation process, in conjunction with Zoutendijk's Method, ensures that the network's parameters are updated in the direction that reduces the loss and optimally within the feasible region defined by any constraints. This results in a robust and efficient learning process for the entire SR-ResNet architecture.

3.6. Downsampling

The focus is on the iterative process of refining the model through repeated application of Zoutendijk's Method across multiple training epochs. The process ensures that the model progressively moves closer to the global or local minimum of the loss function $L(\theta)$ while respecting any constraints imposed on the parameters. Given the parameter set $\theta^{(t)}$ at epoch t, the objective is to further minimize the loss function by updating the parameters through a series of iterations. To get the loss function's gradient concerning the parameters, one uses the following formula:

$$\nabla_{\theta^{(t)}} L(\theta) = \frac{\partial L(\theta)}{\partial \theta^{(t)}} \quad (33)$$

Zoutendijk's Method is then applied to determine the feasible direction.$d^{(t)}$that minimizes the objective function while ensuring the parameters remain within the feasible region. This optimization problem can be expressed as:

$$minimize \nabla_{\theta^{(t)}} L(\theta)^{\mathrm{T}} d^{(t)} \quad (34)$$

$$subject to C(\theta^{(t)} + \alpha^{(t)} d^{(t)}) \leq 0 \quad (35)$$

where $C(\theta)$ represents any constraints, such as non-negativity or boundedness, that must be satisfied by the parameters θ.

To refine the parameters, the optimal step size $\alpha^{(t)}$is determined by minimizing the loss along the direction $d^{(t)}$:

$$\alpha^{(t)} = \arg\min_{\alpha} L\left(\theta^{(t)} + \alpha d^{(t)}\right) \quad (36)$$

The parameters are then updated for the next iteration $t+1$ using the calculated direction and step size:

$$\theta^{(t+1)} = \theta^{(t)} - \alpha^{(t)} d^{(t)} \quad (37)$$

As training progresses through multiple epochs, the gradient $\nabla_{\theta^{(t+1)}} L(\theta)$ is recalculated after each parameter update, ensuring that the parameters continually move in the direction most effectively reduces the loss function.The iteration process continues with updated gradients.$\nabla_{\theta^{(t+1)}} L(\theta)$ and directions $d^{(t+1)}$ for each subsequent epoch:

$$\nabla_{\theta^{(t+1)}} L(\theta) = \frac{\partial L(\theta)}{\partial \theta^{(t+1)}} \quad (38)$$

$$d^{(t+1)} = -\nabla_{\theta^{(t+1)}} L(\theta) \quad (39)$$

The refinement process also involves ensuring that the constraints $C\left(\theta^{(t+1)}\right)$ remain satisfied. This requires recalculating the feasible region at each iteration and adjusting the step size.$\alpha^{(t+1)}$accordingly to maintain feasibility:

$$C\left(\theta^{(t+1)}\right) \leq 0 \quad (40)$$

This iterative process continues across multiple epochs until the loss function $L(\theta)$ reaches a stable minimum, indicating that the model parameters have converged to optimal values. The repeated application of Zoutendijk's Method throughout this process ensures that the parameter updates are effective in reducing the loss and robust against potential constraint violations, leading to a well-optimized model in the SR-ResNet framework.

3.7. Global Average Pooling

The focus is on evaluating and adjusting the model during the training process to ensure the convergence and stability of the optimization. This step involves monitoring the loss function, changing the learning rate, and applying regularization techniques to enhance the model's generalization capabilities.The primary objective in this step is to minimize the loss function $L(\theta)$ by refining the parameters θ while ensuring that the model does not overfit the training data. The process begins by evaluating the loss function after each training epoch. The loss function at epoch t is denoted as:

$$L^{(t)}(\theta) = -\frac{1}{N}[y_i log p_i(\theta) + (1-y_i) log(1- \quad (41)$$

where N represents the number of training samples, y_i is the true label, and $p_i(\theta)$ is predicted probability for the ith sample.

The loss function's gradient concerning the parameters θ may be determined using the following formula:

$$\nabla_\theta L^{(t)}(\theta) = -\frac{1}{N}\sum_{i=1}^{N}\left[\frac{y_i - p_i(\theta)}{p_i(\theta)(1-p_i(\theta))} \cdot \frac{\partial p_i(t}{\partial \theta} \quad (42)$$

Applying regularization methods, such as L2 regularization, helps avoid overfitting and enhances generalization.The L2 regularization term $\lambda\|\theta\|_2^2$ is added to the loss function:

$$L_{reg}^{(t)}(\theta) = L^{(t)}(\theta) + \lambda\|\theta\|_2^2 \quad (43)$$

where λ controls the penalty's intensity as a regularization parameter.

The total loss with regularization is minimized by updating the parameters using the gradient descent method with a learning rate.$\eta^{(t)}$ that is potentially adjusted during training:

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \nabla_\theta L_{reg}^{(t)}(\theta) \quad (44)$$

To ensure stable convergence, the learning rate $\eta^{(t)}$ may be adjusted based on the evaluation of the loss function over successive epochs. If the loss does not decrease as expected, the learning rate is reduced:

$$\eta^{(t+1)} = \gamma\eta^{(t)}, \text{with } \gamma \in (0,1) \quad (45)$$

This step also involves monitoring the model's performance on a validation set. The validation loss $L_{val}^{(t)}(\theta)$ is calculated similarly to the training loss. If the validation loss starts increasing while the training loss decreases, early stopping criteria may be applied to prevent overfitting:

If $L_{val}^{(t)}(\theta) > L_{val}^{(t-1)}(\theta)$, then stop training (46)

Continuous evaluation of the loss function, adjustment of the learning rate, and application of regularization ensure that SR-ResNet converges to a well-generalized solution that effectively performs on training and unseen data.

3.8. Fully Connected Layer

In Fully Connected Layer, the process involves final adjustments and the evaluation of the trained model's performance on the test data. This step ensures the model generalizes well to new, unseen data and evaluates its robustness across different scenarios.The trained model parameters θ^*, obtained after the training process, are used to make predictions on the test dataset. For each test sample x_i^{test}, the model computes the predicted probability $p_i^{test}(\theta^*)$ for the target class using the softmax function:

$$p_i^{test}(\theta^*) = \frac{exp\left(h_i(\theta^*)\right)}{\sum_{j=1}^{C} exp\left(h_j(\theta^*)\right)} \quad (47)$$

where C denotes the number of classes and $h_i(\theta^*)$ represents the output of the final layer for the ith test sample.

The predicted class label $\hat{y}_i^{test}$ for each test sample is determined by selecting the class with the highest predicted probability:

$$\hat{y}_i^{test} = \arg\min_c p_i^{test}(\theta^*) \quad (48)$$

Next, measures like F1-score, recall, accuracy, and precision assess the model's overall performance. Using the number of test samples divided by the number of properly predicted labels, we can determine the accuracy:

$$Accuracy = \frac{1}{N_{test}} \sum_{i=1}^{N_{test}} 1(\hat{y}_i^{test} = y_i^{test}) \quad (49)$$

The indicator function $1(\cdot)$ is defined as 1 when the predicted label is identical to the real label and 0 otherwise, with N_{test} being the total number of test samples.

Precision, recall, and F1-score are calculated to provide a more detailed analysis of the model's performance, particularly in imbalanced datasets. Precision P for a class c is defined as:

$$P_c = \frac{TP_c}{TP_c + FP_c} \quad (50)$$

where TP_c and FP_cdenote the true positives and false positives for class c, respectively. Recall R for class c is defined as:

$$R_c = \frac{TP_c}{TP_c + FN_c} \quad (51)$$

where FN_cdenotes the false negatives for class c. The F1-score for class c is the harmonic mean of precision and recall:

$$F1_c = \frac{2.P_c.R_c}{P_c + R_c} \quad (52)$$

To assess the overall performance across all classes, the macro-averaged F1-score is calculated:

$$MacroF1 - score = \frac{1}{C}\sum_{c=1}^{C} F1_c \quad (53)$$

After evaluating the performance on the test set, if the model's performance metrics meet the desired criteria, the model is considered ready for deployment. However, if the performance is suboptimal, the training process may be revisited, and adjustments to the learning rate, regularization, or architecture may be made.

3.9. Softmax Activation

SR-ResNet undergoes refinement through fine-tuning, which aims to optimize the network's performance on a specific task or dataset. Fine-tuning involves retraining some or all network layers using a lower learning rate while potentially adjusting other hyper parameters to enhance performance. The fine-tuning process begins by re-evaluating the loss function $L(\theta)$ on the target dataset, where θ represents the current set of network parameters. A new calculation is made for the loss function's gradient concerning these parameters, and it is:

$$\nabla_\theta L(\theta) = \frac{\partial L(\theta)}{\partial \theta} \quad (54)$$

In fine-tuning, the learning rate η is typically reduced compared to the initial training phase to allow for more precise weight adjustments. The learning rate at iteration t during fine-tuning is expressed as:

$$\eta^{(t)} = \eta_0.\gamma^t \quad (55)$$

where $\eta_0$is the initial learning rate used during fine-tuning, and γ is a decay factor such that $0 < \gamma < 1$.

The updated learning rate $\eta^{(t)}$Controls the step size in the gradient descent process during fine-tuning.

The parameter update rule during fine-tuning is given by:

$$\theta^{(t+1)} = \theta^{(t)} - \eta^{(t)} \nabla_\theta L\left(\theta^{(t)}\right) \quad (56)$$

Regularization techniques, such as L2 regularization, continue to be applied during fine-tuning to prevent overfitting. The regularized loss function $L_{reg}(\theta)$ during fine-tuning is expressed as:

$$L_{reg}(\theta) = L(\theta) + \lambda \|\theta\|_2^2 \quad (57)$$

where λ is the regularization parameter that controls the strength of the penalty applied to the model weights.

During fine-tuning, certain layers of the network, particularly the earlier layers, may be frozen to preserve the learned low-level features. In comparison, only the higher or fully connected layers are updated. The frozen layers are denoted by θ_{frozen}, and their gradients are set to zero:

$$\nabla_{\theta_{frozen}} L(\theta) = 0 \quad (58)$$

The gradients for the layers that are fine-tuned, denoted by θ_{tuned}, are computed as usual:

$$\nabla_{\theta_{tuned}} L(\theta) = \frac{\partial L(\theta)}{\partial \theta_{tuned}} \quad (59)$$

The fine-tuning process continues until the loss function.$L_{reg}(\theta)$ converges or achieves a minimum value on the target dataset. The final parameters θ^* obtained after fine-tuning represents the optimized model tailored to the specific task.

3.10. Backpropagation with Zoutendijk's Optimization

In this step the model's final evaluation and calibration occur, ensuring robustness and accuracy before deployment. This step involves using calibration techniques to adjust the model's probability estimates and further validation across diverse datasets to confirm reliability. The calibrated probabilities are obtained by adjusting the softmax outputs.$p_i(\theta^*)$ of the model to better reflect the true likelihood of the classes. One common method for calibration is temperature scaling, which involves introducing a scalar temperature parameter T into the softmax function:

$$picalθ*,T=exphi(θ*)/Tj=1Cexphj(θ*)/T \quad (60)$$

where $h_i(\theta^*)$ represents the logits for the ith class, and $T > 0$ is the temperature parameter.

Using a validation set, we minimize the negative log-likelihood to find the optimal temperature parameter:

$$T^* = \arg\min_T -\frac{1}{N_{val}} \sum_{i=1}^{N_{val}} logp_{y_i}^{cal}(\theta^*, T) \quad (61)$$

where N_{val} is the number of validation samples and y_i is the true label for the ith validation sample.

After obtaining the optimal temperature T^*, the model's calibrated probabilities are used to assess its performance. Metrics such as Expected Calibration Error (ECE) and Maximum Calibration Error (MCE) are computed to quantify the calibration quality:

$$ECE = \sum_{m=1}^{M} \frac{|B_m|}{N_{val}} |acc(B_m) - conf(B_m)| \quad (62)$$

$$MCE = \max_{m \in \{1,\dots,M\}} |acc(B_m) - conf(B_m)| \quad (63)$$

where $B_m$denotes the set of samples whose predicted confidence falls into bin $m, acc(B_m)$is the accuracy within the bin $B_m, conf(B_m)$ is the average confidence within the bin B_m, and M is the number of bins.

To further ensure robustness, the calibrated model is evaluated across multiple test datasets, which may vary in domain, distribution, or noise levels. The test loss $L_{test}(\theta^*, T^*)$ is computed as:

$$L_{test}(\theta^*, T^*) = -\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} logp_{y_i}^{cal}(\theta^*, T^*) \quad (64)$$

where $N_{test}$is the number of test samples, and y_i represents the true labels for the test dataset.

A model undergoes adversarial testing to guarantee it is robust against input data perturbations of a modest magnitude. The model's resilience to adversarial examples x_i^{adv}, generated using methods like the Fast Gradient Sign Method (FGSM), is measured by evaluating the adversarial loss:

$$x_i^{adv} = x_i + \epsilon . sign\left(\nabla_{x_i} L_{test}(\theta^*, T^*)\right) \quad (65)$$

$$L_{adv}(\theta^*, T^*) = -\frac{1}{N_{test}} \sum_{i=1}^{N_{test}} log p_{y_i}^{cal}(x_i^{adv}, \theta^*, T^*) \quad (66)$$

where ϵ is a small perturbation applied to the input x_i.

This step culminates in assessing the model's generalization capabilities, ensuring high performance across varied conditions. The combination of calibration and rigorous testing confirms the model's readiness for deployment, providing confidence in its ability to make reliable predictions in real-world applications.

3.11. Output Layer

The Output Layer focuses on the deployment phase, where the model's predictions are integrated into a real-world system. This step involves converting the trained model into an efficient format, optimizing inference speed, and implementing the model in a production environment.The first task is to convert the trained model θ^*into a format suitable for deployment. This conversion may involve quantization, where the model weights and activations are reduced from 32-bit floating-point to lower precision, such as 8-bit integers. Quantization is mathematically expressed as:

$$\hat{\theta}_q = round\left(\frac{\theta^* - min(\theta^*)}{max(\theta^*) - min(\theta^*)} \times (2^b - 1\right) \quad (67)$$

where $\hat{\theta}_q$ represents the quantized weights, b is the number of bits used for quantization, and $min(\theta^*)$ and $max(\theta^*)$denote the minimum and maximum values of the weights before quantization.

After quantization, the inference speed is optimized by applying techniques such as pruning and compression. Pruning involves removing weights in the network that have minimal impact on the output. The pruned weights θ_{pruned} are determined by setting a threshold τ:

$$\theta_{pruned} = \begin{cases} \theta^* if |\theta^*| \geq \tau \\ 0 \quad if |\theta^*| < \tau \end{cases} \quad (68)$$

where τ is selected based on the desired trade-off between model size and accuracy.

Compression further reduces the model size by encoding the pruned weights. Huffman coding or similar techniques are used to achieve this, where the weights are compressed as:

$$C(\theta_{pruned}) = -\sum_i p(\theta_{pruned}^i) log_2 p(\theta_{prune} \quad (69)$$

where $C(\theta_{pruned})$ represents the compressed model, and $p(\theta_{pruned}^i)$ is the probability distribution on the pruned weights.

Once the model is optimized, it is integrated into the production environment. The model receives input data.x_{prod} in real-time, which is passed through the SR-ResNet layers to compute the output logits $h_{prod}(\theta_{pruned})$ The softmax function is applied to obtain the predicted probabilities:

$$p_{prod}(x_{prod}, \theta_{pruned}) = \frac{exp\left(h_{prod}^j(\theta_{prune}\right.}{\sum_{k=1}^{C} exp\left(h_{prod}^k(\theta_{pr}\right.} \quad (70)$$

where $p_{prod}(x_{prod}, \theta_{pruned})$represents the probability distribution over the classes, and j indicates the class index.

The class with the highest probability is selected as the final prediction. $\hat{y}_{prod}$:

$$\hat{y}_{prod} = \arg\min_j p_{prod}(x_{prod}, \theta_{pruned}) \quad (71)$$

The deployment environment is monitored to ensure the model performs well and responds appropriately to inputs. The output predictions are logged, and any misclassifications are analyzed to identify potential areas for model improvement. The deployed model may undergo periodic retraining to adapt to new data, ensuring continued relevance and accuracy. This retraining uses a similar process as the original training, where the updated weightsθ_{update} are obtained by minimizing the loss function on newly collected data:

$$\theta_{update} = \arg\min_\theta L_{new}(\theta) \quad (72)$$

where $L_{new}(\theta)$represents the loss on the new dataset. The model is then redeployed with these updated weights, ensuring continuous learning and adaptation in the production environment.

3.12. Evaluation

This step focuses on shifts with continuous monitoring, maintenance, and updating of the deployed model to ensure long-term performance and adaptability. This step involves periodic evaluation, model updating, and adaptation based on real-world data to maintain the model's effectiveness over time. The first aspect of this step is the ongoing evaluation of the model's performance. The deployed model's predictions$\hat{y}_{prod}$ are continuously compared against true labels y_{true} collected in the production

environment. The accuracy of the model is recalculated at regular intervals:

$$Accuracy_{prod} = \frac{1}{N_{prod}} \sum_{i=1}^{N_{prod}} 1(\hat{y}_{prod,i} = y_{tru} \quad (73)$$

The number of samples analyzed is denoted by N_{prod}, and the indicator function $1(\cdot)$ gives one if the prediction matches the real label and 0 otherwise.To ensure the model remains relevant, drift detection mechanisms are employed to monitor for changes in data distribution. The distribution of new datax_{new} is compared to the original training data distribution x_{train} using statistical tests, such as the kullback-Leiber (KL) divergence:

$$KL(P_{train}||P_{new}) = \sum_{i} P_{train}(x_i) log \frac{P_{train}(x}{P_{new}(x_i} \quad (74)$$

where $P_{train}(x_i)$ and $P_{new}(x_i)$represent the probability distributions of the training and new data, respectively. A significant increase in KL divergence may indicate a data shift, necessitating model updates.

In response to detected drift or deteriorating performance, the model undergoes retraining. The retraining process involves collecting a new dataset. D_{new} and updating the model parameters θ by minimizing the loss function on this new dataset:

$$\theta_{new} = \arg\min_{\theta} L_{new}(\theta) \quad (75)$$

where $L_{new}(\theta)$ represents the loss on the new dataset D_{new}.

Regularization is applied during retraining to prevent overfitting to the new data, using techniques like L2 regularization:

$$L_{new,reg}(\theta) = L_{new}(\theta) + \lambda\|\theta\|_2^2 \quad (76)$$

The updated parameters θ_{new} are then used to redeploy the model, ensuring it remains effective under the current data distribution.To further enhance robustness, ensemble learning techniques may be employed,$\theta^{(1)}, \theta^{(2)}, \ldots, \theta^{(K)}$ are trained on different subsets of data or with different initializations. The ensemble model's prediction $\hat{y}_{ensemble}$ is calculated as:
where $p_c^{(k)}(x_{prod}, \theta^{(k)})$ is the probability output for

$$\hat{y}_{ensemble} = \arg\min_{c} \sum_{k=1}^{K} p_c^{(k)}(x_{prod}, \theta^{(k)}) \quad (77)$$

class c by the k-th model.

The ensemble approach helps mitigate the risk of performance degradation due to model drift or changes in data distribution.Threats to validity, including data bias and overfitting, were addressed via augmentation and regularization. Evaluation metrics accuracy, precision, recall, and F1-scorewere chosen for their relevance in clinical diagnostics, ensuring balanced assessment of model performance and robustness in melanoma detection.

Table 1. Attribute and Description of Melanoma Skin Cancer Dataset

Attribute	Description
Total Images	10,000
Image Dimensions	Varies (Standardized to a standard resolution before model input)
Image Format	JPEG
Classes	2 (Malignant Melanoma, Benign Melanoma)
Training Images	9,600
Testing Images	1,000
Class Distribution	Balanced (Ensures equal representation of malignant and benign images in both training and testing)
Data Splitting	80/20 split for training and testing
Potential Use Cases	Melanoma detection, binary image classification, transfer learning, fine-tuning of pre-trained models
Preprocessing Steps	Includes resizing, normalization, and potential data augmentation
Labelling	Binary labels (0 for benign, 1 for malignant)

4. DATASET

Melanoma Skin Cancer Dataset [54]contains 10,000 images categorized into malignant and benign melanoma. This dataset is crucial in developing deep learning models for accurate melanoma classification, potentially aiding in early detection and treatment. It includes 9,600 images designated for training and 1,000 images reserved for model evaluation. With its comprehensive collection of high-quality images,

this dataset provides a valuable resource for researchers and developers aiming to create models that can precisely distinguish between benign and malignant melanoma, ultimately contributing to better diagnostic tools in the fight against skin cancer.Researchers have relied on this dataset to create models to overcome common challenges like gradient vanishing and overfitting, particularly in complex tasks like melanoma classification. By providing a large and well-annotated dataset, the Melanoma Skin Cancer Dataset has ensured that models trained on it are accurate and robust, making it an essential resource in the ongoing efforts to improve melanoma detection and treatment outcomes.The dataset has also facilitated applying advanced optimization techniques, such as Zoutendijk's Method, which has been integrated into models to enhance training efficiency and stability.

5.. RESULTS AND DISCUSSION

The analysis of performance metrics for ETL, DLMA, and SR-ResNet has highlighted their effectiveness in melanoma detection. SR-ResNet has demonstrated the highest efficacy, showcasing exceptional accuracy in correctly identifying malignant and benign melanoma cases. The model has also minimized errors, reflecting its robustness in reducing type I and II errors.DLMA has shown moderate performance, with a greater likelihood of misclassification than SR-ResNet. ETL has displayed significant limitations in classification accuracy.SR-ResNet has exhibited superior classification capabilities, making it the most reliable model among the three for accurate melanoma detection. Detailed values for true positives, true negatives, false positives, and false negatives are provided in Table 2.

Table 2. TP, FP, TN, FN Values.

Variables of Performance Metrics	ETL	DLMA	SR-ResNet
TP	2542	3161	4772
TN	2595	3625	4707
FP	2309	1723	354
FN	2554	1491	167

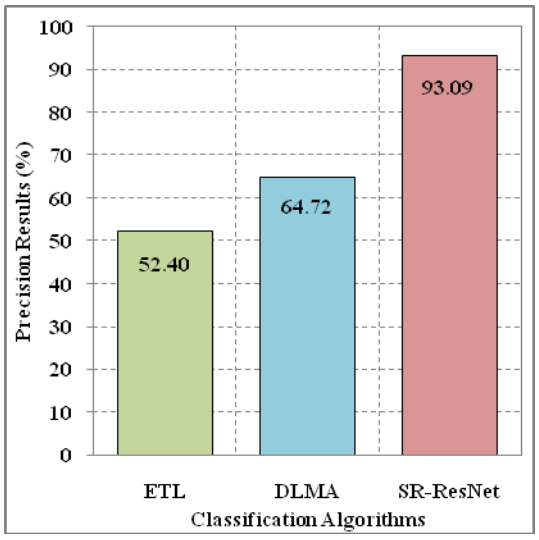

Figure 2. Precision Results

5.1. Precision

The precision analysis of the melanoma detection models has provided critical insights into their diagnostic accuracy. Precision, a key performance metric, quantifies the proportion of true positives (correctly identified malignant cases) relative to the total number of positive predictions (true positives plus false positives). This metric is fundamental in medical diagnostics, where minimizing false positives is essential to avoid unnecessary treatments and interventions. In the comparative analysis, SR-ResNet has exhibited a precision of 93.09%, substantially surpassing the other models. This high precision indicates SR-ResNet's superior ability to accurately identify malignant melanoma cases while significantly reducing the occurrence of false positives. The model's integration of Zoutendijk's Method for nonlinear optimization has likely contributed to this enhanced precision, enabling more reliable differentiation between malignant and benign lesions.

DLMA, with a precision of 64.72%, has shown moderate performance, reflecting a higher false positive rate that could compromise its clinical applicability. ETL has recorded the lowest precision at 52.40%, highlighting a considerable risk of misclassification. Such a performance could lead to a higher incidence of false positives, which may result in overdiagnosis and overtreatment in a clinical setting.The precision metrics underscore SR-ResNet's effectiveness in achieving highly accurate melanoma detection, positioning it as a

more reliable tool for clinical diagnostics than DLMA and ETL.

5.2. Recall

The recall rates of melanoma detection models reflect their underlying working mechanisms and effectiveness in identifying true melanoma cases. Recall, or sensitivity, measures how well a model detects true positives, a crucial factor in clinical accuracy.SR-ResNet, incorporatingZoutendijk's Method for nonlinear optimization, has achieved a recall rate of 96.62%. This high recall results from SR-ResNet's ability to optimize weight updates efficiently, enhancing convergence and reducing errors during training. By refining the learning process and stabilizing deep network training, SR-ResNet accurately identifies most melanoma cases, making it highly reliable in detecting malignancies.

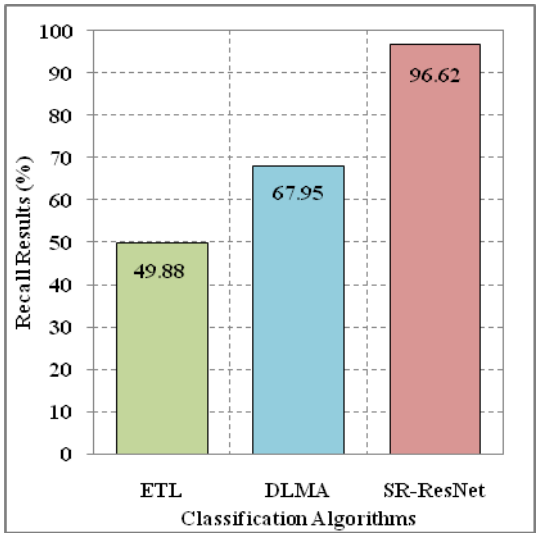

Figure 3. Recall Results

DLMA, utilizing deep learning meta-analysis, has achieved a recall of 67.95%. DLMA synthesizes insights from multiple deep-learning models, which helps in improving generalization across diverse datasets. The meta-analysis approachcan still miss some cases due to its reliance on aggregated findings, leading to a moderate recall rate.ETL, which combines ensemble techniques with transfer learning, has recorded a recall of 49.88%. While ensemble methods generally improve robustness by combining multiple models, ETL's lower recall suggests that integrating pre-trained models may not fully capture the complexity of melanoma cases, leading to a significant number of missed diagnoses.

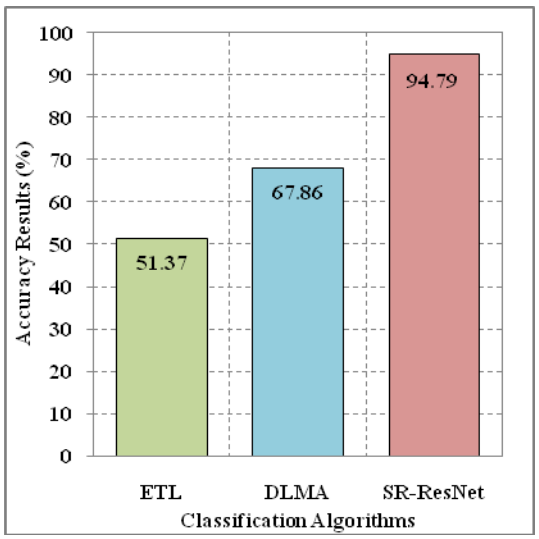

Figure 4. Classification Accuracy Results

5.3. Classification Accuracy

The accuracy rates of the melanoma detection models, as shown in the table, clearly indicate their overall performance in classifying malignant and benign cases. Accuracy, a critical metric, represents the proportion of correctly identified cases—true positives and true negatives—relative to the total number of cases. SR-ResNet has demonstrated an impressive accuracy of 94.79%, surpassing the other models. This high accuracy reflects SR-ResNet's robust design, which integrates Zoutendijk's Method to optimize the training process. The method improves convergence stability and minimizes errors such as overfitting, leading to a model that can reliably differentiate between malignant and benign lesions across diverse and complex datasets. The architecture's depth and the refinement in its residual blocks ensure that SR-ResNet consistently makes accurate predictions, making it highly suitable for clinical deployment.

DLMA, with an accuracy of 67.86%, performs moderately well, balancing its predictions across different cases but still falling short in capturing the full complexity of melanoma diagnosis. DLMA's meta-analysis approach, which aggregates results from various deep learning models, enhances generalization but may introduce variability that affects overall accuracy.ETL, with an accuracy of 51.37%, presents the lowest performance among the models. Although

ensemble techniques combined with transfer learning generally aim to enhance robustness, ETL's lower accuracy suggests that its model integration has not fully addressed the intricacies of melanoma detection, resulting in a higher rate of misclassifications.

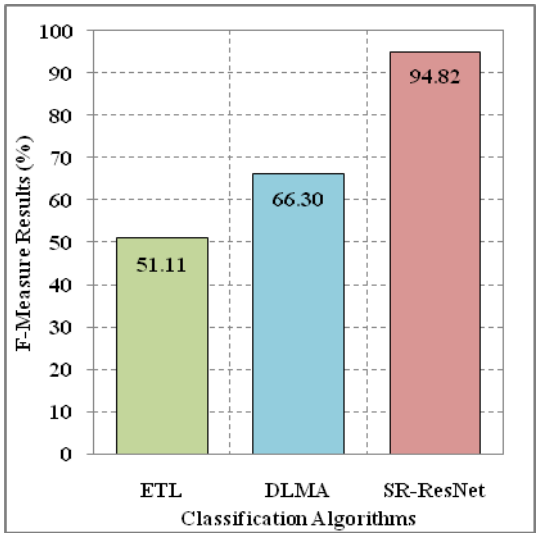

Figure 5. F-Measure Results

5.4. F-Measure

The F-Measure results comprehensively show how well each melanoma detection model balances precision and recall. This is crucial for assessing overall performance and identifying true cases while minimizing errors.With an F-Measure of 94.82%, SR-ResNet demonstrates its advanced capability to deliver reliable results. The high F-Measure reflects SR-ResNet's ability to maintain a substantial equilibrium between correctly identifying melanoma and reducing misclassifications. This success can be attributed to SR-ResNet's sophisticated use of residual connections and deep architecture, allowing it to capture complex patterns in melanoma data effectively. The depth of the network, coupled with precise optimization of weight updates, ensures that the model remains robust against common challenges like overfitting and gradient vanishing, resulting in fewer false positives and negatives.

DLMA, achieving an F-Measure of 66.30%, shows moderate effectiveness. The model employs deep learning meta-analysis, which leverages insights from multiple algorithms. However, while this method enhances overall generalization, it may introduce variability that affects the delicate balance between precision and recall, leading to a lower F-Measure.ETL, with an F-Measure of 51.11%, reveals the limitations of its ensemble transfer learning approach. Although combining multiple models generally aims to enhance performance, ETL's approach may lack the depth and fine-tuned integration needed to capture the nuances of melanoma detection fully. The ensemble method might struggle with consistency across different data types, leading to a higher misclassification rate.

6. CONCLUSION

The research has conclusively demonstrated that the Self-Reliant Residual Network (SR-ResNet) significantly enhances melanoma skin disease detection compared to traditional deep learning models. By integrating Zoutendijk's Method, SR-ResNet optimizes the training process, resulting in superior convergence stability, reduced overfitting, and heightened classification accuracy, achieving an impressive accuracy rate of 94.79% on the Melanoma Skin Cancer Dataset. The model's advanced architecture, including optimized residual connections and precise weight updates, allows it to manage the complexities of melanoma data adeptly, ensuring consistent and accurate performance. SR-ResNet's robust and reliable diagnostic capabilities position it as a transformative tool in medical imaging, offering a more precise and efficient approach to early melanoma detection. This advancement improves patient outcomes through timely interventions and sets a new standard for future innovations in medical diagnostics. The model's success underscores the critical importance of integrating sophisticated optimization techniques in deep learning, opening the door to further advancements in detecting and diagnosing other complex diseases. As SR-ResNet continues to evolve, future enhancements could involve refining its architecture to handle even more diverse datasets and extending its application to other forms of cancer or complex medical conditions, paving the way for more comprehensive and scalable diagnostic tools. This study introduces SR-ResNet, combining residual learning with Zoutendijk's Method to enhance convergence, accuracy, and robustness in melanoma detection. The approach effectively mitigates overfitting and improves classification reliability. However, the use of a single dataset limits generalizability across diverse clinical environments. Future work will focus on testing SR-ResNet with multimodal and real-time data, improving interpretability through explainable

AI, and optimizing deployment for clinical integration.

REFERENCES

[1] K. Sethanan *et al.*, "Double AMIS-ensemble deep learning for skin cancer classification," *Expert Syst. Appl.*, vol. 234, p. 121047, 2023, doi: https://doi.org/10.1016/j.eswa.2023.121047.

[2] W. Abbes and D. Sellami, "Deep Neural Networks for Melanoma Detection from Optical Standard Images using Transfer Learning," *Procedia Comput. Sci.*, vol. 192, pp. 1304–1312, 2021, doi: https://doi.org/10.1016/j.procs.2021.08.134.

[3] F. Alenezi, A. Armghan, and K. Polat, "A multi-stage melanoma recognition framework with deep residual neural network and hyperparameter optimization-based decision support in dermoscopy images," *Expert Syst. Appl.*, vol. 215, p. 119352, 2023, doi: https://doi.org/10.1016/j.eswa.2022.119352.

[4] S. Kadry, E. Verdú, R. Damasevicius, L. Abualigah, V. Singh, and V. Rajinikanth, "CNN segmentation of skin melanoma in pre-processed dermoscopy images," *Procedia Comput. Sci.*, vol. 235, pp. 2775–2782, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.262.

[5] Q. Liu, H. Kawashima, and A. Rezaei sofla, "An optimal method for melanoma detection from dermoscopy images using reinforcement learning and support vector machine optimized by enhanced fish migration optimization algorithm," *Heliyon*, vol. 9, no. 10, p. e21118, 2023, doi: https://doi.org/10.1016/j.heliyon.2023.e21118.

[6] L. Andersson *et al.*, "Poly(ethylene glycol)-poly(ester-carbonate) block copolymers carrying PEG-peptidyl-doxorubicin pendant side chains: Synthesis and evaluation as anticancer conjugates," *Biomacromolecules*, vol. 6, no. 2, pp. 914–926, 2005, doi: 10.1021/bm049381p.

[7] Ç. Suiçmez, H. Tolga Kahraman, A. Suiçmez, C. Yılmaz, and F. Balcı, "Detection of melanoma with hybrid learning method by removing hair from dermoscopic images using image processing techniques and wavelet transform," *Biomed. Signal Process. Control*, vol. 84, p. 104729, 2023, doi: 10.1016/j.bspc.2023.104729.

[8] S. Ghosh, S. Dhar, R. Yoddha, S. Kumar, A. K. Thakur, and N. D. Jana, "Melanoma Skin Cancer Detection Using Ensemble of Machine Learning Models Considering Deep Feature Embeddings," *Procedia Comput. Sci.*, vol. 235, pp. 3007–3015, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.284.

[9] Y. Zhou *et al.*, "Multi-site cross-organ calibrated deep learning (MuSClD): Automated diagnosis of non-melanoma skin cancer," *Med. Image Anal.*, vol. 84, p. 102702, 2023, doi: https://doi.org/10.1016/j.media.2022.102702.

[10] S. M. Thomas, J. G. Lefevre, G. Baxter, and N. A. Hamilton, "Interpretable deep learning systems for multi-class segmentation and classification of non-melanoma skin cancer," *Med. Image Anal.*, vol. 68, p. 101915, 2021, doi: https://doi.org/10.1016/j.media.2020.101915.

[11] P. Thapar, M. Rakhra, M. Alsaadi, A. Quraishi, A. Deka, and J. V. Naga Ramesh, "A hybrid Grasshopper optimization algorithm for skin lesion segmentation and melanoma classification using deep learning," *Healthc. Anal.*, vol. 5, p. 100326, 2024, doi: https://doi.org/10.1016/j.health.2024.100326.

[12] L. A. Courtenay, I. Barbero-García, S. Martínez-Lastras, S. Del Pozo, M. Corral, and D. González-Aguilera, "Using computational learning for non-melanoma skin cancer and actinic keratosis near-infrared hyperspectral signature classification," *Photodiagnosis Photodyn. Ther.*, vol. 49, p. 104269, 2024, doi: https://doi.org/10.1016/j.pdpdt.2024.104269.

[13] C. Kavitha, S. Priyanka, M. P. Kumar, and V. Kusuma, "Skin Cancer Detection and Classification using Deep Learning Techniques," *Procedia Comput. Sci.*, vol. 235, pp. 2793–2802, 2024, doi: https://doi.org/10.1016/j.procs.2024.04.264.

[14] R. R. Babu and F. M. Philip, "Optimized deep learning for skin lesion segmentation and skin cancer detection," *Biomed. Signal Process. Control*, vol. 95, p. 106292, 2024, doi: https://doi.org/10.1016/j.bspc.2024.106292.

[15] E. Farea, R. A. A. Saleh, H. AbuAlkebash, A. A. R. Farea, and M. A. Al-antari, "A hybrid deep learning skin cancer prediction framework," *Eng. Sci. Technol. an Int. J.*, vol. 57, p. 101818, 2024, doi: https://doi.org/10.1016/j.jestch.2024.101818.

[16] T.-T.-H. Pham, T.-N. Luu, T.-V. Nguyen, N.-T. Huynh, Q.-H. Phan, and T.-H. Le, "Polarimetric imaging combining optical parameters for classification of mice non-melanoma skin cancer tissue using machine

learning," *Heliyon*, vol. 9, no. 11, p. e22081, 2023, doi: https://doi.org/10.1016/j.heliyon.2023.e22081.

[17] R. Dandu, M. Vinayaka Murthy, and Y. B. Ravi Kumar, "Transfer learning for segmentation with hybrid classification to Detect Melanoma Skin Cancer," *Heliyon*, vol. 9, no. 4, p. e15416, 2023, doi: https://doi.org/10.1016/j.heliyon.2023.e15416.

[18] M. Roshni Thanka *et al.*, "A hybrid approach for melanoma classification using ensemble machine learning techniques with deep transfer learning," *Comput. Methods Programs Biomed. Updat.*, vol. 3, p. 100103, 2023, doi: https://doi.org/10.1016/j.cmpbup.2023.100103.

[19] Z. Ye *et al.*, "Deep learning algorithms for melanoma detection using dermoscopic images: A systematic review and meta-analysis," *Artif. Intell. Med.*, vol. 155, p. 102934, 2024, doi: https://doi.org/10.1016/j.artmed.2024.102934.

[20] B. Suchitra, J. Ramkumar, and R. Karthikeyan, "Frog Leap Inspired Optimization-Based Extreme Learning Machine For Accurate Classification Of Latent Autoimmune Diabetes In Adults (LADA)," *J. Theor. Appl. Inf. Technol.*, vol. 103, no. 2, pp. 472–494, 2025, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85217140979&partnerID=40&md5=9540433c16d5ff0f6c2de4b8c43a4812

[21] J. Ramkumar, V. Valarmathi, and R. Karthikeyan, "Optimizing Quality of Service and Energy Efficiency in Hazardous Drone Ad-Hoc Networks (DANET) Using Kingfisher Routing Protocol (KRP)," *Int. J. Eng. Trends Technol.*, vol. 73, no. 1, pp. 410–430, 2025, doi: 10.14445/22315381/IJETT-V73I1P135.

[22] R. Jaganathan, S. Mehta, and R. Krishan, "Preface," *Bio-Inspired Intell. Smart Decis.*, pp. xix–xx, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85195725049&partnerID=40&md5=7a2aa7adc005662eebc12ef82e3bd19f

[23] J. Ramkumar, B. Varun, V. Valarmathi, D. R. Medhunhashini, and R. Karthikeyan, "Jaguar-Based Routing Protocol (Jrp) For Improved Reliability And Reduced Packet Loss In Drone Ad-Hoc Networks (DANET)," *J. Theor. Appl. Inf. Technol.*, vol. 103, no. 2, pp. 696–713, 2025, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85217213044&partnerID=40&md5=e38a375e46cf43c95d6702a3585a7073

[24] J. Ramkumar and R. Vadivel, "Multi-Adaptive Routing Protocol for Internet of Things based Ad-hoc Networks," *Wirel. Pers. Commun.*, vol. 120, no. 2, pp. 887–909, Apr. 2021, doi: 10.1007/s11277-021-08495-z.

[25] A. Senthilkumar, J. Ramkumar, M. Lingaraj, D. Jayaraj, and B. Sureshkumar, "Minimizing Energy Consumption in Vehicular Sensor Networks Using Relentless Particle Swarm Optimization Routing," *Int. J. Comput. Networks Appl.*, vol. 10, no. 2, pp. 217–230, 2023, doi: 10.22247/ijcna/2023/220737.

[26] S. P. Priyadharshini and J. Ramkumar, "Mappings Of Plithogenic Cubic Sets," *Neutrosophic Sets Syst.*, vol. 79, pp. 669–685, 2025, doi: 10.5281/zenodo.14607210.

[27] J. Ramkumar and R. Vadivel, "Improved Wolf prey inspired protocol for routing in cognitive radio Ad Hoc networks," *Int. J. Comput. Networks Appl.*, vol. 7, no. 5, pp. 126–136, 2020, doi: 10.22247/ijcna/2020/202977.

[28] R. Jaganathan and R. Vadivel, "Intelligent Fish Swarm Inspired Protocol (IFSIP) for Dynamic Ideal Routing in Cognitive Radio Ad-Hoc Networks," *Int. J. Comput. Digit. Syst.*, vol. 10, no. 1, pp. 1063–1074, 2021, doi: 10.12785/ijcds/100196.

[29] J. Ramkumar and R. Vadivel, "CSIP—cuckoo search inspired protocol for routing in cognitive radio ad hoc networks," in *Advances in Intelligent Systems and Computing*, Springer Verlag, 2017, pp. 145–153. doi: 10.1007/978-981-10-3874-7_14.

[30] S. P. Geetha, N. M. S. Sundari, J. Ramkumar, and R. Karthikeyan, "Energy Efficient Routing In Quantum Flying Ad Hoc Network (Q-Fanet) Using Mamdani Fuzzy Inference Enhanced Dijkstra's Algorithm (MFI-EDA)," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 9, pp. 3708–3724, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85197297302&partnerID=40&md5=72d51668bee6239f09a59d2694df67d6

[31] M. P. Swapna, J. Ramkumar, and R. Karthikeyan, "Energy-Aware Reliable Routing with Blockchain Security for Heterogeneous Wireless Sensor Networks," in *Lecture Notes in Networks and Systems*, V. Goar, M. Kuri, R. Kumar, and T. Senjyu, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 713–723. doi: 10.1007/978-981-97-6106-7_43.

[32] J. Ramkumar, K. S. Jeen Marseline, and D. R. Medhunhashini, "Relentless Firefly Optimization-Based Routing Protocol (RFORP) for Securing Fintech Data in IoT-Based Ad-Hoc Networks," *Int. J. Comput. Networks Appl.*, vol. 10, no. 4, pp. 668–687, 2023, doi: 10.22247/ijcna/2023/223319.

[33] K. S. J. Marseline, J. Ramkumar, and D. R. Medhunhashini, "Sophisticated Kalman Filtering-Based Neural Network for Analyzing Sentiments in Online Courses," in *Smart Innovation, Systems and Technologies*, A. K. Somani, A. Mundra, R. K. Gupta, S. Bhattacharya, and A. P. Mazumdar, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 345–358. doi: 10.1007/978-981-97-3690-4_26.

[34] J. Ramkumar and R. Vadivel, "Whale optimization routing protocol for minimizing energy consumption in cognitive radio wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 4, pp. 455–464, 2021, doi: 10.22247/ijcna/2021/209711.

[35] M. P. Swapna and J. Ramkumar, "Multiple Memory Image Instances Stratagem to Detect Fileless Malware," in *Communications in Computer and Information Science*, S. Rajagopal, K. Popat, D. Meva, and S. Bajeja, Eds., Springer Science and Business Media Deutschland GmbH, 2024, pp. 131–140. doi: 10.1007/978-3-031-59100-6_11.

[36] D. Jayaraj, J. Ramkumar, M. Lingaraj, and B. Sureshkumar, "AFSORP: Adaptive Fish Swarm Optimization-Based Routing Protocol for Mobility Enabled Wireless Sensor Network," *Int. J. Comput. Networks Appl.*, vol. 10, no. 1, pp. 119–129, 2023, doi: 10.22247/ijcna/2023/218516.

[37] M. Lingaraj, T. N. Sugumar, C. S. Felix, and J. Ramkumar, "Query aware routing protocol for mobility enabled wireless sensor network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 3, pp. 258–267, 2021, doi: 10.22247/ijcna/2021/209192.

[38] R. Jaganathan, S. Mehta, and R. Krishan, *Bio-Inspired intelligence for smart decision-making*. IGI Global, 2024. doi: 10.4018/9798369352762.

[39] R. Vadivel and J. Ramkumar, "QoS-enabled improved cuckoo search-inspired protocol (ICSIP) for IoT-based healthcare applications," in *Incorporating the Internet of Things in Healthcare Applications and Wearable Devices*, IGI Global, 2019, pp. 109–121. doi: 10.4018/978-1-7998-1090-2.ch006.

[40] J. Ramkumar, R. Karthikeyan, and M. Lingaraj, "Optimizing IoT-Based Quantum Wireless Sensor Networks Using NM-TEEN Fusion of Energy Efficiency and Systematic Governance," in *Lecture Notes in Electrical Engineering*, V. Shrivastava, J. C. Bansal, and B. K. Panigrahi, Eds., Springer Science and Business Media Deutschland GmbH, 2025, pp. 141–153. doi: 10.1007/978-981-97-6710-6_12.

[41] J. Ramkumar, R. Karthikeyan, and V. Valarmathi, "Alpine Swift Routing Protocol (ASRP) for Strategic Adaptive Connectivity Enhancement and Boosted Quality of Service in Drone Ad Hoc Network (DANET)," *Int. J. Comput. Networks Appl.*, vol. 11, no. 5, pp. 726–748, 2024, doi: 10.22247/ijcna/2024/45.

[42] P. Menakadevi and J. Ramkumar, "Robust Optimization Based Extreme Learning Machine for Sentiment Analysis in Big Data," in *2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022*, Institute of Electrical and Electronics Engineers Inc., 2022. doi: 10.1109/ICACTA54488.2022.9753203.

[43] J. Ramkumar, A. Senthilkumar, M. Lingaraj, R. Karthikeyan, and L. Santhi, "OPTIMAL APPROACH FOR MINIMIZING DELAYS IN IOT-BASED QUANTUM WIRELESS SENSOR NETWORKS USING NM-LEACH ROUTING PROTOCOL," *J. Theor. Appl. Inf. Technol.*, vol. 102, no. 3, pp. 1099–1111, 2024, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85185481011&partnerID=40&md5=bf0ff974ceabc0ad58e589b28797c684

[44] J. Ramkumar and R. Vadivel, "Improved frog leap inspired protocol (IFLIP) – for routing in cognitive radio ad hoc networks (CRAHN)," *World J. Eng.*, vol. 15, no. 2, pp. 306–311, 2018, doi: 10.1108/WJE-08-2017-0260.

[45] R. Jaganathan, S. Mehta, and R. Krishan, *Intelligent Decision Making Through Bio-Inspired Optimization*. IGI Global, 2024. doi: 10.4018/979-8-3693-2073-0.

[46] R. Jaganathan and V. Ramasamy, "Performance modeling of bio-inspired routing protocols in Cognitive Radio Ad Hoc Network to reduce end-to-end delay," *Int. J. Intell. Eng. Syst.*, vol. 12, no. 1, pp. 221–231, 2019, doi: 10.22266/IJIES2019.0228.22.

[47] R. Karthikeyan and R. Vadivel, "Boosted Mutated Corona Virus Optimization Routing Protocol (BMCVORP) for Reliable Data Transmission with Efficient Energy

Utilization," *Wirel. Pers. Commun.*, vol. 135, no. 4, pp. 2281–2301, 2024, doi: 10.1007/s11277-024-11155-7.

[48] R. Karthikeyan and R. Vadivel, "Proficient Dazzling Crow Optimization Routing Protocol (PDCORP) for Effective Energy Administration in Wireless Sensor Networks," in *IEEE International Conference on Electrical, Electronics, Communication and Computers, ELEXCOM 2023*, 2023, pp. 1–6. doi: 10.1109/ELEXCOM58812.2023.10370559.

[49] N. K. Ojha, A. Pandita, and J. Ramkumar, "Cyber security challenges and dark side of AI: Review and current status," in *Demystifying the Dark Side of AI in Business*, 2024, pp. 117–137. doi: 10.4018/979-8-3693-0724-3.ch007.

[50] J. Ramkumar, S. S. Dinakaran, M. Lingaraj, S. Boopalan, and B. Narasimhan, "IoT-Based Kalman Filtering and Particle Swarm Optimization for Detecting Skin Lesion," in *Lecture Notes in Electrical Engineering*, K. Murari, N. Prasad Padhy, and S. Kamalasadan, Eds., Singapore: Springer Nature Singapore, 2023, pp. 17–27. doi: 10.1007/978-981-19-8353-5_2.

[51] J. Ramkumar, C. Kumuthini, B. Narasimhan, and S. Boopalan, "Energy Consumption Minimization in Cognitive Radio Mobile Ad-Hoc Networks using Enriched Ad-hoc On-demand Distance Vector Protocol," in *2022 International Conference on Advanced Computing Technologies and Applications, ICACTA 2022*, 2022. doi: 10.1109/ICACTA54488.2022.9752899.

[52] J. Ramkumar, R. Vadivel, and B. Narasimhan, "Constrained Cuckoo Search Optimization Based Protocol for Routing in Cloud Network," *Int. J. Comput. Networks Appl.*, vol. 8, no. 6, pp. 795–803, 2021, doi: 10.22247/ijcna/2021/210727.

[53] L. Mani, S. Arumugam, and R. Jaganathan, "Performance Enhancement of Wireless Sensor Network Using Feisty Particle Swarm Optimization Protocol," *ACM Int. Conf. Proceeding Ser.*, pp. 1–5, Dec. 2022, doi: 10.1145/3590837.3590907.

[54] M. H. Javid, "Melanoma Skin Cancer Dataset of 10000 Images." Kaggle, 2022. doi: 10.34740/KAGGLE/DSV/3376422.