Image Classification System for Bruise Detection Using Convolutional Neural Network

Michaela Angela E. Cailing
College of Computing and Information Technologies
National University - Manila
Sampaloc, Manila
cailingme@students.national-u.edu.ph

Mark Rhey Anthony De Luna
College of Computing and Information Technologies
National University - Manila
Sampaloc, Manila
delunamr@students.national-u.edu.ph

Rodney Lei D. Estrada
College of Computing and Information Technologies
National University - Manila
Sampaloc, Manila
estradar@students.national-u.edu.ph

Danyssa B. Tamayo
College of Computing and Information Technologies
National University - Manila
Sampaloc, Manila
tamayodb@students.national-u.edu.ph

***Abstract*—Bruises are critical indicators of physical trauma and often serve as early signs in cases of abuse, including intimate partner violence and child maltreatment. Traditional bruise detection relies on visual assessment, which can be subjective and inconsistent. To address these limitations, this study presents a Convolutional Neural Network (CNN)-based image classification system for automated bruise detection, deployed via a browser-accessible web application. Trained on a dataset of labeled skin images using TensorFlow, the model achieved a test accuracy of 93%, demonstrating strong performance in binary classification of "Bruise" versus "Normal" skin. Key techniques such as data augmentation, LeakyReLU activation, and dropout regularization were integrated to enhance generalization and prevent overfitting. The system, implemented with Streamlit for user accessibility, shows promise for clinical and field applications. However, current limitations include dataset diversity and real-world generalizability. Future work will focus on expanding the dataset, improving model robustness across skin tones and lighting conditions, and exploring mobile or offline deployment for broader healthcare and forensic utility.**

Keywords - Bruise Detection, Convolutional Neural Networks (CNN), Deep Learning, Medical Image Classification

I. Introduction

Bruises are among the most visible indicators of physical trauma. In both healthcare and forensic settings, the presence of bruises often serves as a critical factor in evaluating injuries, identifying cases of abuse, and enabling timely medical or legal intervention. These visual signs are especially important in sensitive scenarios such as intimate partner violence and child maltreatment. Nearly 1 in 4 women globally have experienced physical violence from a partner, with bruising frequently reported as a consequence [1]. Similarly, bruises are among the most common indicators in child protection cases [2]. As such, accurate and timely detection of bruises is essential not only for individual safety, but also for broader public health and legal processes.

Traditionally, bruise detection relies on visual inspection conducted by medical professionals. However, this method is inherently subjective and may lead to inconsistencies—especially when bruises are faint, patients have darker skin tones, or environmental lighting is imperfect. Studies have shown that even experienced clinicians may fail to detect certain bruises without the aid of specialized tools such as alternate light sources or hyperspectral imaging [3]. Despite their effectiveness, these technologies are often costly and require trained personnel, limiting their practical use in routine medical or fieldwork settings [4].

The rise of deep learning, particularly Convolutional Neural Networks (CNNs), has significantly advanced the capabilities of computer vision in fields such as facial recognition, medical imaging, and object detection. CNNs have demonstrated strong performance in image-based classification tasks, including the identification of skin lesions and the detection of diseases from radiographic images [5]. With these developments, CNNs present a promising opportunity for automating the detection of physical injuries like bruises. Such automation can improve the consistency, accessibility, and speed of detection across various environments.

This study proposes the development of a web-based application that uses CNNs to classify skin images based on the presence or absence of bruises. By allowing users to upload an image through a web interface, the system provides an instant classification result, making bruise detection more accessible in both clinical and remote settings. The decision to implement the solution as a web application ensures platform independence, broader reach, and ease of use—benefitting medical practitioners, forensic investigators, and social workers, especially those working in resource-limited or field environments.

The significance of this research lies in its potential to promote consistent and accessible bruise detection, ultimately aiding early intervention and documentation in cases of abuse or trauma. Additionally, it addresses equity concerns in healthcare access by minimizing reliance on expensive tools and expert-only assessments. By leveraging widely available technology and artificial intelligence, the proposed system supports scalable injury analysis.

CNN models will be trained using a dataset of labeled skin images and deployed in a browser-accessible environment. The model's performance will be evaluated using standard metrics. While the scope of the application focuses on binary classification—bruised skin or normal skin—it sets the foundation for future work in comprehensive, AI-assisted injury detection and digital diagnostics.

II. Review of Related Literature

A Convolutional Neural Network (CNN) is a class of deep learning models primarily designed to process and analyze visual data by mimicking how the human brain interprets images. CNNs are particularly effective for image classification, object detection, and other computer vision tasks because they automatically learn spatial hierarchies of features—from simple edges and textures to complex object parts and entire scenes [13]. This is achieved through layers that perform convolutions, activations, and pooling operations, allowing CNNs to extract and generalize important features across varying input conditions.

CNNs use convolutional layers, which apply filters across local regions to detect patterns like edges or textures. These layers are followed by activation functions such as ReLU or LeakyReLU to introduce non-linearity. LeakyReLU improves on ReLU by addressing the "dying ReLU" issue, allowing a small gradient for negative inputs [14]. Pooling layers (e.g., max pooling) reduce spatial dimensions, lowering computational cost while preserving important features.

Batch normalization is commonly employed to standardize layer inputs during training, improving convergence and generalization. Dropout, another regularization method, randomly deactivates neurons during training to reduce overfitting, especially effective between fully connected layers with a typical dropout rate of 50% [15]. Finally, fully connected (dense) layers at the end of the network synthesize learned features to perform final predictions, such as class probabilities.

Training CNNs effectively requires careful selection of optimizers and learning rate strategies. Traditional Stochastic Gradient Descent (SGD) updates weights in fixed steps and benefits from momentum, while adaptive methods like RMSprop and Adam adjust learning rates dynamically. Adam combines features from AdaGrad and RMSprop, making it suitable for sparse gradients and noisy data [16].

Adam has demonstrated superior performance in practice. One study noted that "Adam, in combination with deep CNNs, achieved the best validation accuracy of 91.1%" compared to Adamax (88.9%) and RMSprop (87.7%) [17]. However, while Adam converges quickly, SGD with momentum often finds flatter minima, which generalize better [18].

Learning rate schedulers also play a crucial role. Fixed learning rates are simple but can stagnate during training. ReduceLROnPlateau, a dynamic scheduler, adjusts the learning rate when validation metrics plateau, improving convergence stability [19]. Empirical evidence shows Adam with a learning rate of 0.001 offers strong performance and generalization, especially when paired with learning rate scheduling.

The number of training epochs significantly affects model performance. Fixed epochs may cause underfitting or overfitting if not carefully set. Early stopping helps mitigate this risk by monitoring validation loss and halting training once performance stops improving. It is especially effective in deep models prone to overfitting, allowing long training schedules without wasting resources [20].

Moreover, CNNs have become foundational in medical image processing due to their ability to learn hierarchical features directly from raw visual data. These networks excel in classifying, detecting, and segmenting medical images across radiological modalities, such as MRI, CT, and X-rays, often outperforming traditional diagnostic methods in accuracy and speed [21]. In clinical and forensic applications, CNNs are increasingly employed for bruise detection, providing an objective alternative to traditional visual assessments. These systems analyze skin images to identify bruises related to trauma, abuse, or injury—tasks often subject to human error and variability. CNN-based models have shown promising results in distinguishing bruise presence, estimating age, and classifying severity, especially in pediatric and elder care contexts [22]. Studies have explored both binary and multiclass classification frameworks, achieving high sensitivity and specificity by training on annotated skin images with varying tones, lighting conditions, and bruise stages [23].

To address challenges like dataset imbalance and variability in skin tone, recent approaches use transfer learning, ensemble methods, and image enhancement techniques. Mobile-integrated CNN systems are also emerging, enabling point-of-care or field assessments for telemedicine and forensic reporting [24]. Still, challenges remain, including limited annotated datasets, generalizability across diverse skin types, and the need for greater interpretability.

III. Methodology

A. Data Collection

The study utilized the Wound Classification Dataset created by Ibrahim Fateen, which is publicly available on Kaggle. (https://www.kaggle.com/datasets/ibrahimfateen/wound-classification). The study dataset was originally compiled for medical image analysis purposes, with contributions from various medical institutions and research facilities. For our specific study, we focused exclusively on the bruise category to develop a binary classification system.

The study uses 242 images for bruises, and 200 images for normal images(without bruises). The dataset was last updated in 2022 with license CC0: Public Domain.The images have various formats of JPG, JPEG, and PNG, and have different sizes which must be standardized later for the CNN models.

B. Data Pre-Processing

All images in the dataset are made the same size by being resized to 224×224 pixels. The number of dimensions was set to maintain both the important details in bruises and avoid making the algorithm too slow. By standardizing the data, the neural network can be sure each training input has consistent dimensions without losing important details for sensing features. This stage is very important, since input images might come from diverse sources with varying image resolutions.

Normalizing Color Space. At this point, all images go through RGB normalization which makes sure every channel has a different value. Consistent lighting helps doctors see the difference between bruises and normal skin by focusing on color. Using RGB format allows the rich detail of bruising to stay present and makes the data consistent throughout. Pixel Value Normalization. The final preprocessing step in Pixel Intensity Standardization (PIS) is to normalize the pixel values by dividing them by 255 and make them fall within [0,1]. It is very important to normalize the data in a neural network to train it more efficiently and with fewer issues. It brings all the input features together so they aren't easily noticed just because their values are much larger than the rest. Processing Data Validation and working on Handling Errors. It uses strong error handling methods to look after common problems: only JPEG, PNG and JPG files are processed; any corrupt images are found and noted; any problem with converting colors is resolved smoothly; and loading and processing images efficiently uses little memory.Dataset Organization. The processed images are organized into a structured format suitable for deep learning: Features (X): Normalized RGB images with shape (224, 224, 3) Labels (y): Binary classification (0 for normal skin, 1 for bruise) Data is partitioned into training (70

The preprocessing pipeline ensures data consistency and quality while maintaining the essential characteristics needed for effective bruise detection. Will serve as the standard preprocessing for all experiments.

C. Experimental Setup

Python 3.11 and TensorFlow 2.x are used as the main deep learning framework and they are implemented in Visual Studio Code via integrated Jupyter notebooks. Adding the important libraries NumPy for working with numbers, Pillow for handling images and Scikit-learn to manage datasets. To graphical representations and scrutiny, the Matplotlib and Seaborn modules are necessary. The web app will be simulated using Streamlit.

Version control and easy access to all models are possible because they are in their own Google Drive repository. The study application will make use of metrics such as accuracy, precision, recall and F1-score alongside detailed performance analysis by using confusion matrices.

D. Algorithm Selection

- **Model Architecture**
  Convolutional Neural Network. A CNN that is built for the specific purpose of finding bruises.The researchers fundamentally rely on the work by LeCun et al. (2015) which made CNNs the leading architecture for classification in images. The network we used is carefully designed with 32 convolutional filters and 3×3 kernels based on the guidelines from the VGG network series (Simonyan Zisserman, 2014).

  In the convolutional layer, each of the 32 filters learns to spot different types of patterns with the help of backpropagation. Filters work by carrying out multiplication and summation on each pixel of the input image which helps produce maps that highlight features such as edges, textures and changes in color. Experience-based evidence showed that the 3×3 kernel has the best performance and computational efficiency and since it is considered the standard by many experts, it will be the comparison point for the rest of the models.

- **Regularization Techniques**
  L2 Regularization. The researchers chose L2 regularization which is built on principles introduced by Tikhonov (1963), to improve generalization by making the loss function include weighted squared weights.

  $$L = L\_0 + \lambda \| \| w \| \| \quad (1)$$

  Here, L _0 describes the initial loss function, λ (lambda) is a regularization parameter, and——w—— means the L2 norm of the weights. Using this method stops the model from overfitting by limiting big weights which encourages general features in learning.

  Dropout. This method involves Dropout, created by Srivastava et al. (2014) which deactivates neurons at random during training and this occurs with a probability of typically 0.3 or 0.5. This method is similar to running multiple neural networks and it is scientifically explained by looking at it as a Bayesian approach to account for uncertainty in models (Gal & Ghahramani, 2016). Using the method will make it easier for researchers to prevent co-adaptation and help their models adapt to different tasks.

- **Activation Functions**
  Rectified Linear Unit, usually written as ReLU in shorthand, stands for Rectified Linear Unit. The activation function chosen for neurons by Nair and Hinton in 2010 is usually rectified linear unit (RUL). ReLU was preferred for deep learning activation function since it successfully addresses the problem of vanishing gradients and is considered to work efficiently. Because it's not in a single direction, the network can learn difficult information. Because finding the sample mean is quick, students develop their understanding faster.

Leaky ReLU. An examination of LeakyReLU was carried out according to Maas et al.'s (2013) paper. Choosing a small negative coefficient (such as 0.01) makes Leaky ReLUs useful again.It was found from the experiments that learning can take place even if negative brain signals occur as a result of the adjustment.

- **Evaluation Metrics**

Precision (Positive Predictive Value). Tells us how often a test is correct in knowing if a trait is actually present. Precision shows how accurate positive predictions are by looking at how often they match the actual positive outcomes.

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{2}$$

The metric quantifies the model's ability to make reliable positive predictions. A high precision indicates minimal false positive predictions, while a low precision suggests the model frequently makes incorrect positive predictions.

Recall (Sensitivity). Recall (sensitivity) measures how often your prediction is right for confirmed positive cases by using the number of accurate predictions over the number of actual positives:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{3}$$

It shows how good the model is at finding all cases where the result is positive in the dataset. When recall is high, the model does not often miss out on true positive cases, but when recall is low, there are often more false negatives.

F1-Score. F1-score is the harmonic mean of precision and recall, giving us a single score that shows how well both measures go together:

$$F_1 = \frac{2 \cdot (\text{Precision} \cdot \text{Recall})}{\text{Precision} + \text{Recall}} \tag{4}$$

The resulting values lie between 0 and 1 and 0 means the worst and 1 means the best precision and recall. Usually, the better the model is, the greater the F1-score.

Accuracy. Accuracy represents the overall correctness of predictions across all classes:

$$\text{Accuracy} = \frac{\text{True Positives} + \text{True Negatives}}{\text{Total Samples}} \tag{5}$$

Metrics will appear as plots and charts such as ROC curves, precision-recall curves and confusion matrices so you can see how the model behaves in a range of situations.

- **Experimental Phases**

1) **Baseline CNN Model.**
At the beginning, the main focus is on creating a base model and working out its key performance measures. The basic CNN applies a straightforward yet efficient structure, where at the input level are 224x224x3 RGB images and then a convolutional layer with 32 filters and 3x3 kernels. It includes a MaxPooling layer to reduce the number of parameters, then flatten and dense layers and finally, the output results in a binary classification. This part of the process is used as a control and subsequent changes are assessed using basic training parameters, an Adam optimizer and binary cross-entropy loss.

2) **Experimental Comparison**
Model Optimization and Analysis Regularization Experimentation. Experimenting on model optimization, analysis and experimentation with regularization is very important. Regularization techniques are evaluated in a well-organized way at the start of the experiment. Higher dropout rates (0.3 or 0.5) are used to avoid overfitting and L2 regularization is applied as an extra way to avoid the same issue. It considers the benefits of using both approaches and tests the performance by measuring validation metrics and how well the system generalizes. It is designed to determine the ideal parameters for regularization in the detection of bruises.

Data Augmentation Exploration. Investigations on Data Augmentation. Robustness in models is improved in the optimization stream by using data augmentation. Comparison was made between the performance of the original images and images that were handled using techniques like image horizontal flipping, rotation (±20 degrees), zooming(±20%) and combination of all in one photo. Every method is independently evaluated to measure its influence on how well the model works, focusing on how well it deals with different bruise types.

Architecture Optimization. Researchers tried out various CNN configurations, comparing shallow networks (only 2 convolutional layers) to deeper ones (up to 5 layers). It evaluates how the size of the kernel (either 3x3 or 5x5) can affect the results and also studies adding batch normalization. Furthermore, a number of activation functions are compared (e.g. ReLU against LeakyReLU) to select the ideal network design for bruise detection.

Training Efficiency Enhancement. The component looks at how different optimization algorithms work and their training progress. It tests the consequences of using different optimizers (Adam, RMSprop and SGD) and a variety of learning rates (0.001 and 0.0001) with different learning rate decay strategies. It also investigates the changes in performance and speed of convergence when different epoch numbers and early stopping are used.

3) **Integration and Deployment.**
The last phase brings together the important points from the other optimization streams. Improved architecture is developed by merging successful methods and their configurations. The researchers final structure is Input level 224x224x3 RGB images, Conv layer with 32 filters, 3x3 kernel, LeakyReLU(alpha=0.01), Conv layer with 32 filters, 3x3 kernel, LeakyReLU(alpha=0.01), MaxPooling2D(2,2), Flatten(), Layer Dense(64), LeakyReLU(alpha=0.01), Layer Dropout(0.5), and Layer Dense(1, sigmoid). Optimizers of Adam(learning rate of 0.001), and loss is binary_crossentropy. During this phase, a Streamlit-based interface gets developed, making sure the system runs quickly and correctly when used by users. Special emphasis is given to managing how sophisticated the model is while maintaining usefulness which leads to a working bruise detection system.

IV. Results and Discussions

This chapter presents the comprehensive results obtained from the three-phase experimental approach to bruise detection using Convolutional Neural Networks (CNNs). The results are organized according to the experimental phases: baseline model development, experimental comparison and optimization, and integration and deployment. Each phase's findings are presented with detailed analysis, performance metrics, and critical discussion of the outcomes.

1) **Baseline CNN Model**
The starting CNN was made based on a simple architecture that will form the basis of the performance foundation of further optimization work. The model took a 224x224x3 RGB images and utilized a single convolutional layer consisting of 32 filters with 3x3 kernels, and MaxPooling to reduce the dimension, flatten out the data and use dense layers to produce a final binary output. The training was performed on both the Adam optimizer and binary cross-entropy loss functions which are par for course when performing the binary classification task.

The baseline model had accuracy of 89% coupled by training loss of 13% and validation loss of 36% respectively. The measurements of precision, recall and F1-score type were in 90%, 89%, 89% allowing to have an idea of how the model classifies the data with

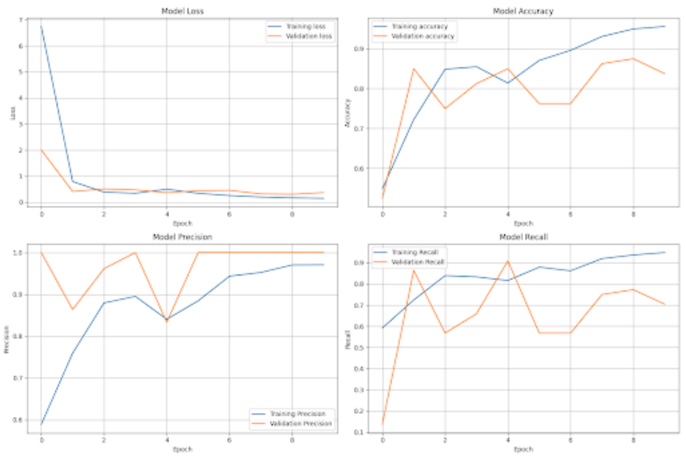

Fig. 1. Baseline CNN Model Metrics

standard metrics. It used 10 epochs to converge, which indicates reasonableness of computation efficiency of the simple architecture.

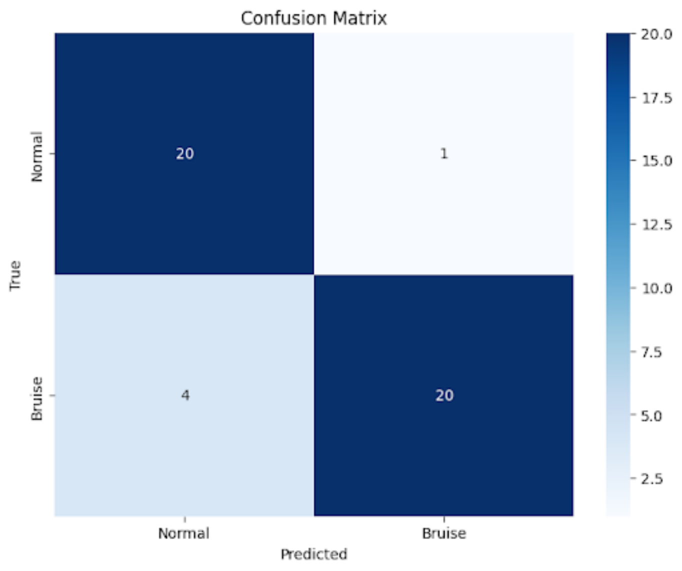

Fig. 2. Baseline CNN Model Confusion Matrix

The baseline model was analyzed and some fundamental characteristics were identified which became the guiding principle on future optimization methods. The model showed plausible performance between being able to differentiate between bruised and non-bruised, as well as, initial notes showed a possible problem of overfitting with the difference between training and validation accuracy. The learning curves were characteristic of any simple CNN architecture, i.e., quickly learning and converging. Confusion matrix showed certain patterns of misclassification, especially, the issues with some types of bruises or image conditions. These results had put in concrete directions of getting the points boosted by regularization means, architectural manipulation, and training optimization plans.

2) **Experimentation Comparison**

Data Augmentation Exploration

Five strategies of data augmentation, baseline (no augmentation), horizontal flip, rotation (±20deg), zoom (±20%), and combined augmentation were tried and tested in a systematic way. Experiments applied the same CNN structure so that the comparison across augmentation methods would be fair.

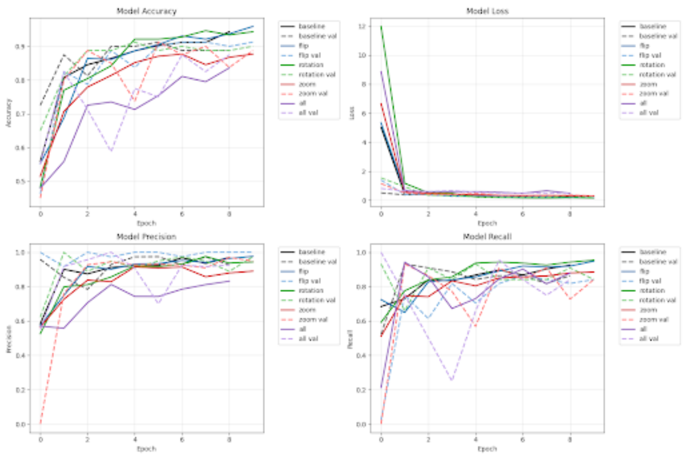

Fig. 3. Data Augmentation CNN Model Graph Metrics (Training vs. Validation)

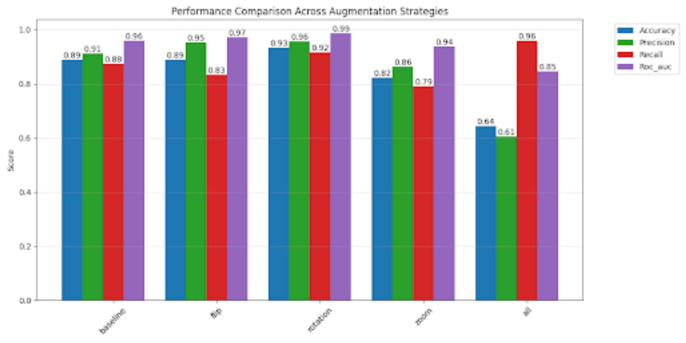

Fig. 4. Data Augmentation CNN Model Metrics

Among the examined data augmentation strategies, horizontal flip and rotation can be discussed as particularly useful methods of enhancing the generalization and robustness of the CNN-based bruise detection model. The process of horizontal flip augmentation is effective in dealing with the symmetry of the anatomy that is largely frequent in cases of bruise incurring as far as the injuries can be found on either part of the body. Reflecting the images in the course of training, this method sharpens the model to understand the appearance of bruises in various parts of the body without making artificial distortions, which would have interfered with the diagnosis. On the same note, rotation augmentation (±20) also performed well as it helped the model to cope with the change in the direction of images which was inherent since images are taken in varying angles of cameral or patient positioning. The rotational limits made the transformations to be realistic and at the same time avoid distortion of morphological variations that would disorient the model.

Compared to flipping or rotation, zoom augmentation was not as effective but it increases the model capacity to find bruises of different sizes and photographic distances, especially the recall rates of unusual size ratios. Nevertheless, it exhibited a decreasing value relative to the other procedures perhaps because of the distortion of features through scaling.

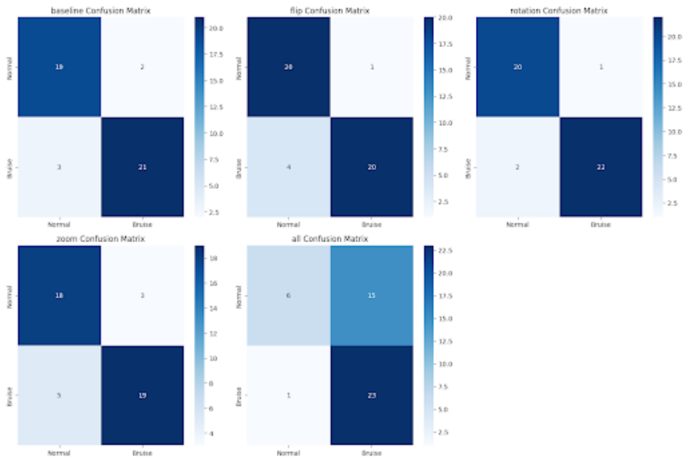

Fig. 5. Data Augmentation CNN Confusion Matrix

When multiple augmentation techniques were applied simultaneously (horizontal flip + rotation + zoom), the model showed the worst performance among all tested approaches. Possible reasons for this stems from excessive distortion, overlapping augmentations that created conflicting visual cues, and small distortions created given that the zoom already showed not a great performance if some sort of it affects the model. Data Augmentation in nature is not bad and a powerful tool where it introduces generalization among the pictures to be trained at the model. But these compounded effects demonstrate that, in medical image analysis, more augmentation is not always better, finding the right balance between it is the best. Although zoom modification really affects the whole downgrade of the model.

Architecture Optimization.

A total of 12 distinct CNN architectures, categorized into shallow and deep models, were evaluated on a bruise

TABLE I
SUMMARY TABLE OF METRICS

Model	Acc	Prc		F.000	AUC	
shallow_k3_no_bn_relu	12.9000.886	1.000	0.815	0.898	0.937	93.900
shallow_k3_bn_relu	12.9000.386	0.000	0.000	0.000	0.500	139.900
shallow_k5_no_bn_relu	12.9000.909	0.960	0.889	0.923	0.950	149.400
shallow_k5_bn_relu	12.9000.386	0.000	0.000	0.000	0.889	198.100
shallow_k3_no_bn_leaky	12.9000.955	0.963	0.963	0.963	0.991	100.600
shallow_k3_bn_leaky	12.9000.386	0.000	0.000	0.000	0.741	154.700
deep_k3_no_bn_relu	1.6400.886	1.000	0.815	0.898	0.919	145.400
deep_k3_bn_relu	1.6500.386	0.000	0.000	0.000	0.392	217.700
deep_k5_no_bn_relu	4.4300.864	1.000	0.778	0.875	0.963	284.500
deep_k5_bn_relu	4.4300.386	0.000	0.000	0.000	0.994	351.700
deep_k3_no_bn_leaky	1.6400.909	1.000	0.852	0.920	0.961	154.700
deep_k3_bn_leaky	1.6500.386	0.000	0.000	0.000	0.540	223.400

detection dataset. Shallow CNNs consisted of 8 to 10 layers, including two convolutional layers, max pooling, and a small dense network. Deep CNNs, on the other hand, comprised 15 to 20 layers, with five convolutional blocks, deeper fully connected layers, and optional batch normalization to enhance learning stability. Architectural variations included:

- Kernel sizes: Comparing 3×3 vs. 5×5 filters.
- Activation functions: ReLU vs. LeakyReLU.
- Batch normalization: Evaluated for training stability and generalization.

A consolidated results table was generated to compare the CNN architectures based on key metrics. The table includes the following columns: Model Name, Accuracy, Precision, Recall, F1-Score, AUC, Training Time (in seconds), and Total Parameters.

The selected evaluation metrics include accuracy, precision, recall, F1-score, and Area Under the ROC Curve (AUC), providing a multifaceted view of each model's predictive power. In addition, training time and the number of trainable parameters were considered to gauge the computational efficiency and complexity of each architecture.

Among all the models evaluated, the shallow_k3_no_bn_leaky architecture achieved the best overall performance, exhibiting a balanced and high accuracy of 0.9545, precision and recall both at 0.9630, an F1-score of 0.9630, and an AUC of 0.9913.

In contrast, models employing LeakyReLU without batch normalization consistently outperformed their ReLU-based and batch-normalized counterparts. These models achieved high precision and recall values, which translated into superior F1-scores, highlighting LeakyReLU's effectiveness in maintaining neuron activity during training. Adjusting the kernel size from 3×3 to 5×5 offered slight improvements in some ReLU-based shallow models (e.g., shallow_k5_no_bn_relu), though these improvements were insufficient to surpass the performance of LeakyReLU-based networks. This suggests that kernel size alone was not the primary determinant of model effectiveness.

Shallow CNNs have significantly more parameters (12M) compared to deep CNNs (2.5M–4.4M), likely due to larger dense layers after convolution blocks. Despite having fewer parameters, deep CNNs do not show clear superiority in accuracy, indicating shallow networks can be more parameter-efficient for this task.

Comparing deep and shallow models, shallow networks—particularly those without batch normalization—tended to generalize better, likely due to reduced model complexity and overfitting. Although some deep models, such as deep_k5_no_bn_relu, achieved perfect precision (1.0000), they suffered from lower recall, indicating a tendency to miss actual positive cases. This trade-off between precision and recall is suboptimal in sensitive detection tasks like bruise identification. Finally, the shallow models not only offered superior classification metrics but also trained significantly faster, reinforcing their suitability for real-time or resource-constrained deployment scenarios.

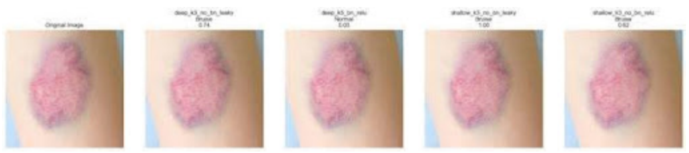

Fig. 6. A sample prediction output using the best performing CNN models

A sample prediction output is shown in Figure 1, where best performing CNN model configurations were evaluated on the same input image. The original image (far left) depicts a visible skin bruise. The prediction results from various models are displayed alongside, indicating the predicted class label (Bruise or Normal) and the associated confidence score. Notably, the shallow_k3_no_bn_leaky model confidently and correctly predicted the presence of a bruise with a perfect score of 1.00, while the deep_k5_bn_relu model failed to detect the bruise, misclassifying it as normal with 0.00 confidence. This qualitative example reinforces earlier quantitative findings that shallow models without batch normalization and with LeakyReLU activation outperform deeper or batch-normalized counterparts in both accuracy and reliability.

Training Efficiency Enhancement.

This experiment evaluated the performance of a shallow convolutional neural network architecture, shallow_k3_no_bn_leaky.The target was to classify bruise presence from input data, and the model was trained under various combinations of optimizers, learning rates, and training strategies.

Two optimizers were tested—Adam and SGD—each with learning rates of 0.001 and 0.0001. For each setting, the model was trained using either early stopping or a

fixed number of 15 epochs. The training objective was binary cross-entropy loss, and the model's performance was evaluated using accuracy and F1-score.
The final performance results are summarized in the table below:

TABLE II
PERFORMANCE COMPARISON ACROSS OPTIMIZER, LEARNING RATE, AND TRAINING STRATEGY CONFIGURATIONS

Cnfg	Acc	F1
ADAM_lr0.001_early_stop	0.921	0.921
ADAM_lr0.001_fixed_15	0.876	0.876
ADAM_lr0.0001_early_stop	0.910	0.910
ADAM_lr0.0001_fixed_15	0.865	0.865
SGD_lr0.001_early_stop	0.607	0.540
SGD_lr0.001_fixed_15	0.809	0.808
SGD_lr0.0001_early_stop	0.618	0.515
SGD_lr0.0001_fixed_15	0.573	0.387

From the results, the Adam optimizer with a learning rate of 0.001 and early stopping achieved the highest accuracy (92.13%) and F1-score (0.9210). This configuration demonstrated strong convergence behavior and generalization capability, highlighting Adam's efficiency in adaptive learning.

In contrast, the SGD optimizer exhibited lower overall performance, particularly when combined with early stopping and lower learning rates. This can be attributed to SGD's reliance on fixed gradient steps, which can limit convergence speed and increase sensitivity to hyperparameters in shallow networks.

Interestingly, SGD with a learning rate of 0.001 and fixed 15 epochs performed better than expected (Accuracy = 80.90%, F1 = 0.8081), suggesting that with sufficient training iterations, SGD can still be viable under certain conditions.

Overall, the results affirm that:

- Adam is more suitable for optimizing this shallow CNN architecture in binary classification tasks.
- Early stopping generally improves model generalization, especially when paired with the Adam optimizer and an appropriate learning rate.
- Learning rate sensitivity is more pronounced with SGD, which requires more careful tuning.

These findings can guide future model selection and training strategies for real-time bruise detection or similar binary image classification problems.

Model Optimization and Analysis Regularization Experimentation.
This section evaluates six neural network models, each using a different regularization strategy, to determine which configuration best reduces overfitting while preserving high classification performance.

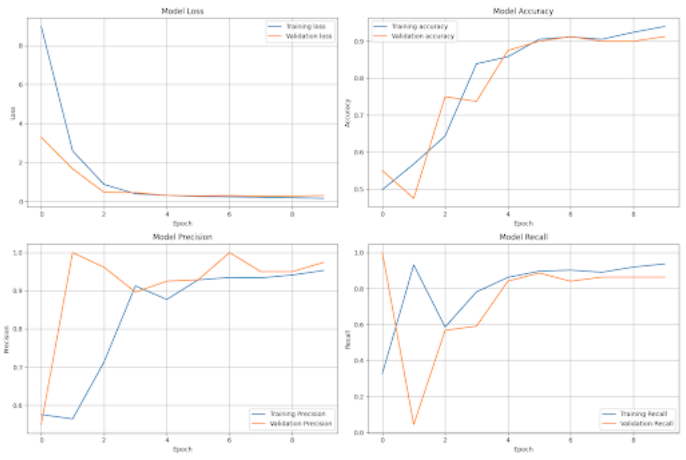

Fig. 7. Training History Curves for the Baseline Model

Baseline Model (No Regularization)

The baseline model demonstrated clear signs of overfitting. As seen in the figure n, the training loss decreased sharply across epochs, while the validation loss plateaued and eventually diverged, indicating that the model began to memorize the training data rather than generalizing from it. This was further evidenced by a widening gap between training and validation accuracy, which became prominent after epoch 5. Despite achieving high test performance (AUC = 0.97 and accuracy of 88.89%), this success is misleading. The stability in validation precision and recall curves (with F1-scores around 0.88–0.89) suggests the model learned meaningful patterns, but the divergence in training dynamics points to overfitting during training. Therefore, while numerically strong, the model lacks robustness and reliability when exposed to unseen data.

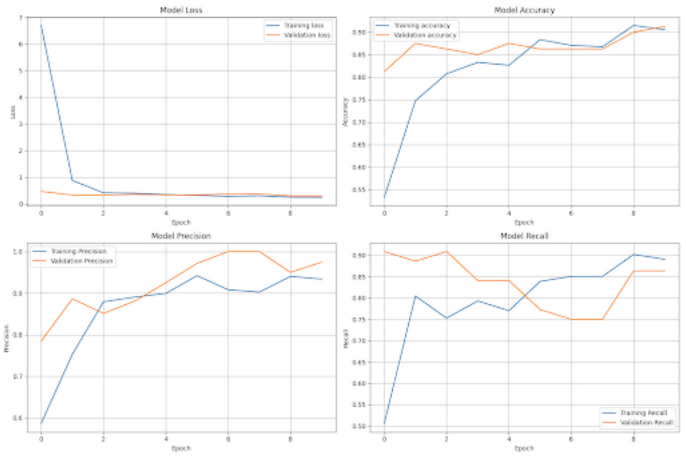

Fig. 8. Training History Curves for the Dropout 0.3 Model

Dropout 0.3

As seen in figure n, training and validation loss curves followed each other very closely, showing no divergence even in later epochs. Accuracy curves were nearly identical throughout training, with both converging to 91%. Precision and recall curves remained smooth and parallel, with validation precision even occasionally surpassing training precision — a rare but ideal sign of generalization. The model also maintained strong class performance in its confusion matrix and classification report (F1-scores 0.89 for both classes, AUC = 0.96, confirming strong generalization). Importantly, it did this without any additional regularization complexity (e.g., L2). The stability of metrics, the alignment of curves, and the absence of a generalization gap all support the conclusion that this model generalized exceptionally well and did not overfit. As such, Dropout 0.3 is the most effective standalone regularization strategy in this study.

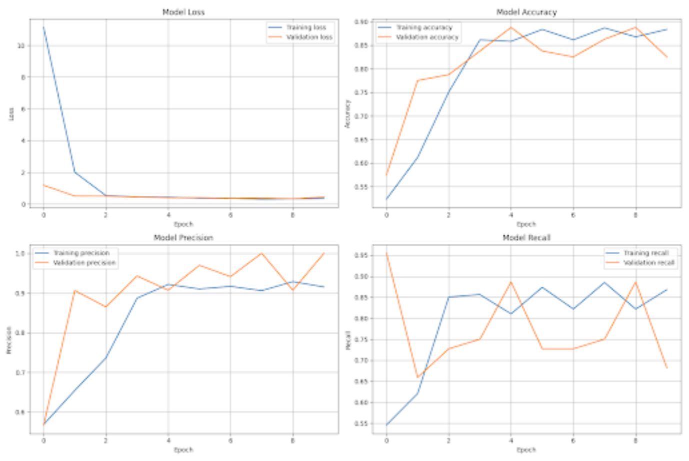

Fig. 9. Training History Curves for the Dropout 0.5 Model

Dropout 0.5

The model with Dropout 0.5 avoided overfitting, as evidenced in Figure n by the closely aligned training and validation loss and accuracy curves. Despite stable training behavior, its overall performance was slightly lower compared to the Dropout 0.3 model. The recall for the "Bruise" class was significantly lower, indicating that the model failed to consistently detect positive cases. While validation loss remained stable, the model became overly conservative in predicting the Bruise class, as reflected by a high precision (0.95) but lower recall (0.83). This resulted in four Bruise samples being misclassified as Normal in the confusion matrix, representing a higher false negative rate. The AUC for this model was 0.92 — the lowest among all tested configurations — suggesting reduced ability to distinguish between classes. These outcomes imply that, although the model avoided overfitting, the high dropout rate (50%) may have overly constrained the network's capacity to learn detailed class features, leading to underfitting, especially for the Bruise class. This model is less effective in scenarios where recall is critical, despite its regularization strength.

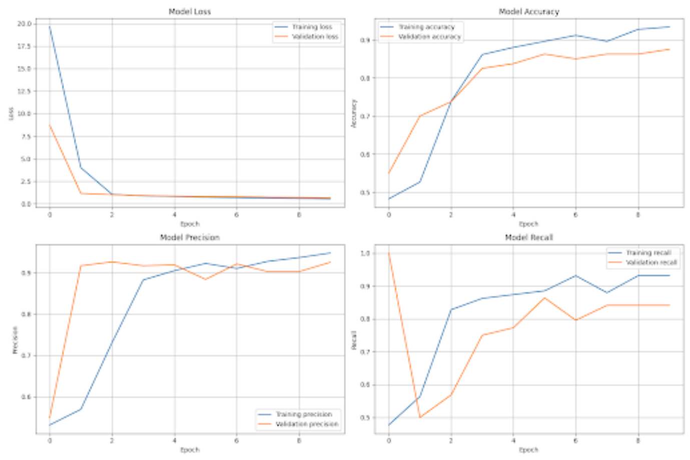

Fig. 10. Training history curves for the L2 regularization only ($\lambda = 0.001$) model.

L2 Regularization Only ($\lambda = 0.001$)

This model used L2 weight regularization with $\lambda = 0.001$, which penalizes large weight magnitudes during training. It moderately reduced overfitting compared to the baseline. As seen in figure n, while training and validation loss curves were better aligned, a persistent but small gap remained throughout training. Accuracy curves also revealed a consistent advantage in training performance over validation. Although the AUC remained respectable at 0.95, the model's F1-scores (both 0.82) and precision-recall trends suggested a limited ability to generalize fully. Thus, while L2 with $\lambda = 0.001$ helped stabilize the model, it was not as effective as dropout-based methods in mitigating overfitting in this case.

Dropout 0.3 + L2 Regularization (λ = 0.001)

This combined regularization model integrated Dropout at 0.3 and L2 with $\lambda = 0.001$. It performed similarly to the Dropout 0.3 model alone, with tight alignment between training and validation curves as seen in figure n, and consistent accuracy near 89%. However, an early dip in recall during the first few epochs in the figure n indicated initial training instability, likely caused by the interaction of both regularization techniques. Final test metrics were strong (AUC = 0.97, accuracy = 88.89%), and the model ultimately stabilized. Since it added complexity and introduced brief instability

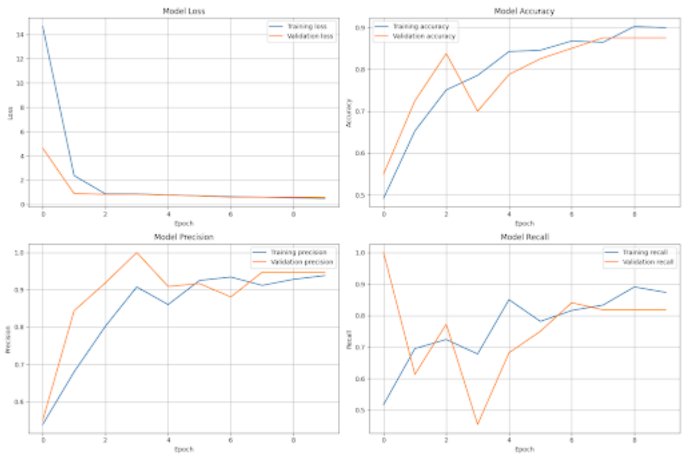

Fig. 11. Training History Curves for the Dropout 0.3 + L2 Regularization (λ = 0.001) Model

without improving performance beyond Dropout 0.3 alone, this configuration was considered redundant rather than advantageous.

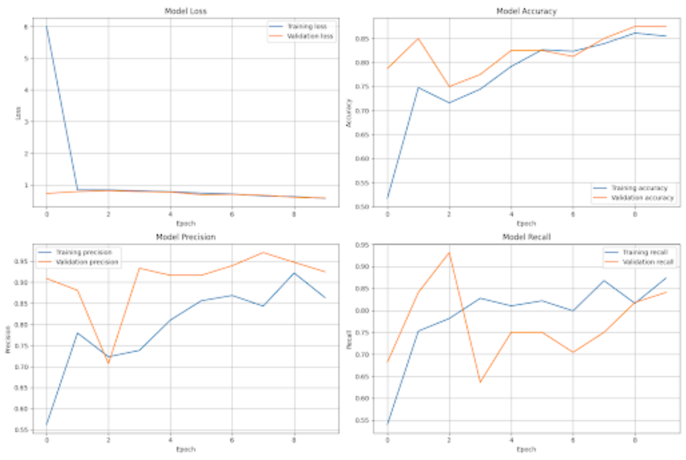

Fig. 12. Training History Curves for the Dropout 0.5 + L2 Regularization (λ = 0.001) Model

Dropout 0.5 + L2 Regularization (λ = 0.001)

The final model combined high Dropout (0.5) and L2 regularization with λ = 0.001. This dual strategy resulted in underfitting, as reflected by flat and lower accuracy and recall curves, and stagnating loss curves in the figure n. Training and validation metrics were closely aligned — not due to balance, but due to limited learning. Both precision and recall were weaker than all other configurations (F1-score = 0.82, AUC = 0.95). The model's inability to extract strong features was likely due to over-regularization, making this configuration the least effective in both learning and generalization.

Among all tested methods, Dropout 0.3 offered the best trade-off between simplicity, performance, and generalization. It avoided overfitting entirely, produced smooth and stable training behavior, and achieved strong test results — all without the need for extra parameters like L2 weight decay.

3) **Final Model**

Given with the phase 2 complete, and now incorporating the hyperparameters and optimizing techniques into one. The final model structure is incorporated by 32 & 64 Convolutional Layer with 3x3 Kernel and activation of LeakyReLU, adding Layer Dropout(0.5).

TABLE III
CLASSIFICATION REPORT FOR THE BRUISE DETECTION MODEL

Class	Precision	Recall	F1-score	Support
Normal	0.910	0.950	0.930	21.000
Bruise	0.960	0.920	0.940	24.000
Accuracy			0.930	45.000
Macro avg	0.930	0.930	0.930	45.000
Weighted avg	0.930	0.930	0.930	45.000

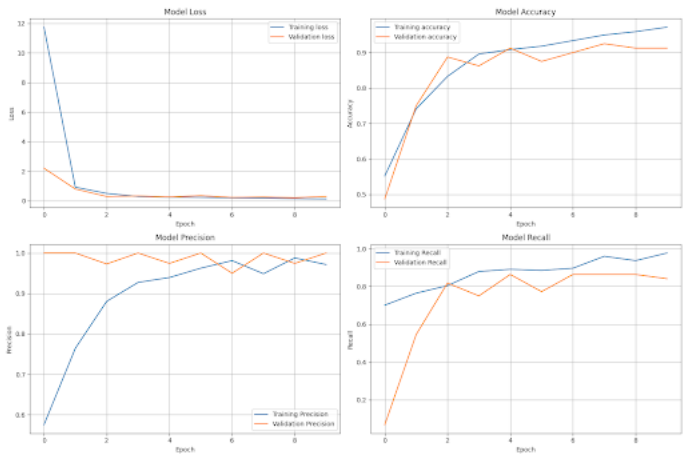

Fig. 13. Final Model Evaluation Metrics Graph

The trained CNN model proved to be very effective in the classification of bruised skin and normal skin with a general accuracy of 93 percent with the test set. Both the classes had a high value of both precision and recall 0.93 in normal and 0.93 in bruise, bringing the F1-score values extremely close to each other. Such outcomes show that not only did the model perform efficiently in the proper detection of bruises but also failed to issue false-positives, which is of crucial significance in practical terms where either over-marking and unjustly spreading on healthy skin or failing to recognize real bruises can have a tangible ramification.

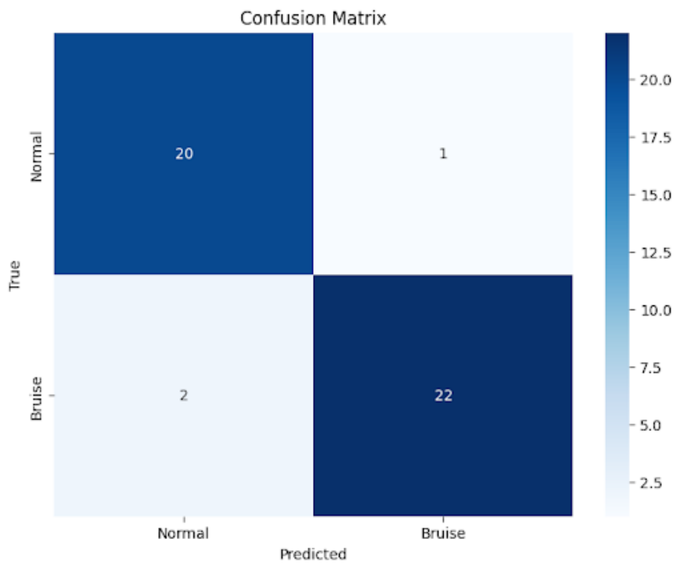

Fig. 14. Final Model Confusion Matrix

The fact that the model showed balanced performance on all classes leads to the conclusion that the model generalized well considering the relative small size of the available data. This level of generalization could have been possible with the adoption of simple data augmentation (which was simply the random horizontal flipping and rotation), which could have created variable data and therefore more resilient features to be learned by the model. The fact that the training and validation curves were consistent further echoes that overfitting did not happen in the model.

All in all, architectural simplicity, augmentation, and regularizational applications significantly helps to generalize and increasing of accuracy of the model.

V. Conclusions

In this study, the researchers developed an image classification model using the TensorFlow library to detect the presence of bruises on human skin. With the use of Convolutional Neural Networks (CNN), the model was able to classify images as either "Bruise" or "Normal" by learning from a dataset of skin images. TensorFlow provided essential functions and modules that simplified the implementation of CNN architectures and training processes.

The model proved to be effective in identifying bruises, achieving a final accuracy of 93% on the test set. The use of convolutional layers allowed the system to automatically extract visual features from the skin images, enabling accurate classification without manual inspection. The final model incorporated data augmentation, LeakyReLU activation, and regularization techniques such as dropout to improve generalization and prevent overfitting. The web-based application, built using Streamlit, also demonstrated the system's potential for practical use in both clinical and field settings.

However, the researchers acknowledge that the dataset used in the study had limitations in terms of quantity and diversity. Some of the images were augmented, and the dataset did not fully represent real-life variations in skin tones, lighting conditions, and bruise types. This may cause the model to misclassify new or unseen images in real-world scenarios. To address this limitation, future work should include collecting a larger and more diverse dataset with more representative samples.

The model may also require more convolutional layers or advanced architectures if expanded to classify different types or stages of bruises, or if extended to detect other forms of physical trauma. Lastly, this system could be adapted for use in mobile or offline environments to assist healthcare providers, social workers, and forensic professionals, especially in remote areas, in the timely and accurate detection of injuries for better medical and legal outcomes.

References

[1] World Health Organization, "Violence against women prevalence estimates, 2018," 2021. [Online]. Available: https://www.who.int/publications/i/item/9789240022256
[2] Child Welfare Information Gateway, "Recognizing child abuse and neglect: Signs and symptoms," U.S. Department of Health and Human Services, 2022. [Online]. Available: https://www.childwelfare.gov/pubPDFs/signs.pdf
[3] K. Scafide, D. J. Sheridan, and J. C. Campbell, "Alternate light five times more effective in detecting bruises on victims of color," George Mason University, 2020. [Online]. Available: https://nursing.gmu.edu/news/2020-04/alternate-light-five-times-more-effective-detecting-bruises-victims-color
[4] National Institute of Justice, "Improving bruise detection with alternate light," 2021. [Online]. Available: https://nij.ojp.gov/topics/articles/improving-bruise-detection-alternate-light
[5] A. Esteva, B. Kuprel, R. A. Novoa, J. Ko, S. M. Swetter, H. M. Blau, and S. Thrun, "Dermatologist-level classification of skin cancer with deep neural networks," *Nature*, vol. 542, no. 7639, pp. 115–118, 2017. doi: 10.1038/nature21056

[6] I. Fateen, “Wound classification dataset,” 2022. [Online]. Available: https://www.kaggle.com/datasets/ibrahimfateen/wound-classification

[7] Y. LeCun, Y. Bengio, and G. Hinton, “Deep learning,” *Nature*, vol. 521, no. 7553, pp. 436–444, 2015. doi: 10.1038/nature14539

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” 2014. [Online]. Available: https://arxiv.org/abs/1409.1556

[9] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: A simple way to prevent neural networks from overfitting,” *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[10] Y. Gal and Z. Ghahramani, “Dropout as a Bayesian approximation: Representing model uncertainty in deep learning,” in *Proc. of the International Conference on Machine Learning (ICML)*, pp. 1050–1059, 2016.

[11] A. L. Maas, A. Y. Hannun, and A. Y. Ng, “Rectifier nonlinearities improve neural network acoustic models,” in *Proc. ICML*, 2013. [Online]. Available: https://web.stanford.edu/ awni/papers/relu$_h ybrid_i cml2013_f inal.pdf$

[12] A. N. Tikhonov and V. Y. Arsenin, *Solutions of Ill-posed Problems*, Washington, D.C.: Winston and Sons, 1963.

[13] S. Madhavan, “Introduction to convolutional neural networks,” IBM Developer, 2021. [Online]. Available: https://developer.ibm.com/articles/introduction-to-convolutional-neural-networks/

[14] Ultralytics, “Leaky ReLU – Discover the power of Leaky ReLU activation for AI and ML,” 2025. [Online]. Available: https://www.ultralytics.com/glossary/leaky-relu

[15] S. Ioffe and C. Szegedy, “Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift,” 2015. [Online]. Available: https://arxiv.org/pdf/1502.03167

[16] O. Hospodarskyy, V. Martsenyuk, N. Kukharska, A. Hospodarskyy, and S. Sverstiuk, “Understanding the Adam Optimization Algorithm in Machine Learning,” 2024. [Online]. Available: https://ceur-ws.org/Vol-3742/paper17.pdf

[17] MDPI, “Adam vs. Adamax vs. RMSprop performance comparison in CNNs,” 2022.

[18] P. Zhou, J. Feng, C. Ma, C. Xiong, and S. Hoi, “Towards Theoretically Understanding Why SGD Generalizes Better Than ADAM in Deep Learning,” in *Proc. NeurIPS*, 2020. [Online]. Available: https://proceedings.neurips.cc/paper/2020/file/f3f27a324736617f20abbf2ffd806f6d-Paper.pdf

[19] A. Al-Kababji, F. Bensaali, and S. P. Dakua, “Scheduling Techniques for Liver Segmentation: ReduceLRonPlateau vs OneCycleLR,” in *Springer*, pp. 204–212, 2022. doi: 10.1007/978-3-031-08277-1$_1$7

[20] B. M. Hussein and S. M. Shareef, “An Empirical Study on the Correlation between Early Stopping Patience and Epochs in Deep Learning,” *ITM Web of Conferences*, vol. 64, p. 01003, 2024. doi: 10.1051/itmconf/20246401003

[21] R. Yamashita, M. Nishio, R. K. G. Do, and K. Togashi, “Convolutional Neural Networks: An Overview and Application in Radiology,” *Insights into Imaging*, vol. 9, no. 4, pp. 611–629, 2018. doi: 10.1007/s13244-018-0639-9

[22] C. Lei, Y. Jiang, K. Xu, S. Liu, H. Cao, and C. Wang, “Convolutional Neural Network Models for Visual Classification of Pressure Ulcer Stages: Cross-Sectional Study,” *JMIR Medical Informatics*, vol. 13, p. e62774, 2025. doi: 10.2196/62774

[23] A. A. Mahmud, S. Azam, I. U. Khan, S. Montaha, A. Karim, A. Haque, M. Z. Hasan, M. Brady, R. Biswas, and M. Jonkman, “SkinNet-14: A Deep Learning Framework for Accurate Skin Cancer Classification Using Low-Resolution Dermoscopy Images with Optimized Training Time,” *Neural Computing and Applications*, vol. 36, no. 30, pp. 18935–18959, 2024. doi: 10.1007/s00521-024-10225-y

[24] B. Diallo, T. Urruty, P. Bourdon, and C. Fernandez-Maloigne, “Robust Forgery Detection for Compressed Images Using CNN Supervision,” *Forensic Science International: Reports*, vol. 2, p. 100112, 2020. doi: 10.1016/j.fsir.2020.100112