Visvesvaraya Technological University

Belagavi, Karnataka-590 014

A Mini-Project Report on

"Burn Classification Using Deep Learning"

submitted in partial fulfilment of the requirement for the award of the degree of

Bachelor of Engineering in Medical Electronics Engineering

Submitted by:

RITU BHAT [1MS22MD030]

SAGAR S R [1MS22MD032]

SANGEET BHATI [1MS22MD035]

VIBHA N D [1MS22MD048]

Under the Guidance of

Dr. Basavaraj Hiremath, Assistant Professor

Department of Medical Electronics Engineering

Ramaiah Institute of Technology, Bangalore 560054

2024-25

RAMAIAH Institute of Technology

CERTIFICATE

Certified that the project work entitled "**Burn Classification Using Deep Learning"** carried out by Mr. Sagar S R [1MS22MD032], Ms. Ritu Bhat [1MS22MD030], Mr. Sangeet Bhati [1MS22MD035], Ms. Vibha N D [1MS22MD048] bonafide students of Ramaiah Institute of Technology, in partial fulfilment for the award of Bachelor of Engineering in Medical Electronics Engineering Department of the Visvesvaraya Technological University, Belagavi during the year 2024-2025. It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the Report deposited in the departmental library. The project report has been approved as it satisfies the academic requirements with respect to the Project work prescribed for the said Degree.

Dr. Basavaraj Hiremath	**Dr. C.K. Narayanappa**	**Dr. N V R Naidu**
Supervisor	HOD	Principal

External Viva Examiner — Signature with date

1.

2.

DECLARATION

We, the students of the 6th semester of Medical Electronics Engineering, Ramaiah Institute of Technology, Bangalore-560054 declare that the work entitled "Burn Classification Using Deep Learning" has been successfully completed under the guidance Dr. Basavaraj Hiremath, Medical Electronics Department, Ramaiah Institute of Technology, Bangalore. This dissertation work is submitted to Visvesvaraya Technological University in partial fulfilment of the requirements for the award of Degree of Bachelor of Engineering in Medical Electronics Engineering during the academic year 2024 - 2025. Further, the matter embodied in the project report has not been submitted previously by anybody for the award of any degree or diploma to any university.

Place:

Date:

Team members:

1. **RITU BHAT [1MS22MD030]**
2. **SAGAR S R [1MS22MD032]**
3. **SANGEET BHATI [1MS22MD035]**
4. **VIBHA N D [1MS22MD048]**

ACKNOWLEDGEMENT

We express our sincere thanks to Dr. N.V.R. Naidu, Principal, RIT for providing us with the necessary platform and administrative support.

We take immense pleasure in thanking HOD Dr. C.K. Narayanappa, Department of Medical Electronics Engineering, for inspiring us and allowing us to work on this project. We thank him for all his support and encouragement.

We heartfully thank our project supervisor Dr. Basavaraj Hiremath, Assistant Professor, Department of Medical Electronics Engineering, for his continuous guidance and encouragement to complete this project.

We would like to thank the Department of Medical Electronics Engineering, RIT, for providing us with the facilities to complete our project.

We are grateful to all the authors and writers for their books, e-resources, and scientific papers references for our project.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Burn injuries usually require accurate classification for timely medical treatment, yet traditional methods can be subjective and inconsistent. This project entails the development of an AI-based system to classify burn wounds into first, second, and third degrees utilizing deep learning. A Multi-layered Convolution Neural Network (CNN) model based on MobileNetV3 was trained using an augmented Kaggle dataset and achieved 97% accuracy. The model was deployed into a light-weight Flutter mobile application to support remote diagnosis. The system was effective for on-device inference and provided consistency to non-specialists in low resource or emergency health-care settings.

CHAPTER 1: INTRODUCTION

1.1 Burn Injuries

Burn injuries pose a notable public health problem particularly in low- and middle-income countries with limited medical resources. Burns may be caused by thermal, electrical, chemical, radiation or friction injuries and can range in severity from minor discomfort to life-threatening injury. The World Health Organization (WHO) estimates there are approximately 180,000 burn-related deaths annually, with the most occurring in regions with limited resources and less access to timely and specialized treatment.

An early diagnosis of burn injury severity is particularly important to identify circumstances for appropriate medical intervention and treatment of the burn injury. Generally, burn wounds can be classified into four degrees depending on the tissue depth:

First-degree burns (superficial) where the integrity of the epidermis or outermost layer of skin is impaired, which may be reddened, and may cause minor pain and discomfort.

Second-degree burns (partial thickness) through the dermis, which may produce painful blisters, and may cause some scarring.

Third-degree burns (full thickness) which extends through the dermis and may also damage underlying tissue, may require surgical procedures for repair (skin graft).

Fourth-degree burns are life-threatening injuries that extend through tissue to involve muscle and/or bone, requiring critical intervention immediately!

Traditional diagnosis assessing burn injury often relies on clinician judgment and visual assessment. This may include subjectivity and personnel case in the minimal early stage of injury, or in regions where access is limited.

1.2 Motivation and Problem Statement

In an emergency situation, medical personnel must classify the type of burn they have dealt with as quickly as possible to ensure proper treatment. However, there are many factors, including availability and experience level of the clinician, that can lead to different assessments of burn classification, ultimately risking patient harm or treatment errors.

Since most technological devices used to classify and diagnose burns such as laser Doppler imaging, ultrasound, or even visual skin imaging are not available in rural or disaster areas, assessing and classifying burn injuries will require consistent, precise, efficient, and accurate augmentation solutions, with appropriately trained or no prior burn clinical experience available to patients anywhere work is being done.

There is an ample need for automated, accurate, reliable, and accessible methods to help clinicians classify and triage burns. The combination of deep learning and app-based mobile solutions could provide an opportunity for all clinicians working within the bounds of their expertise to assess and classify burns in a reliable manner. With modern approaches such as Convolutional Neural Networks, it is feasible to consider automated image-based classification of burns as an alternative to visual inspection. If the AI-based classifier is deployed as an application on smart-phone capable devices, the volunteer first reported to the health emergency could access the mechanism through the app to the nurse depending on the assessment and need, or potentially even the patient in remote areas, of their burn for classification, allowing for early assessment and intervention.

1.3 Goals of the Project

The main aim of this project is to successfully design, train, and implement a small deep learning model that can classify burn images into three degrees of burn - first-degree, second-degree, and third-degree burn. In order to accomplish this goal, our project will implement a sophisticated deep learning pipeline using MobileNetV3 a convolutional neural network architecture designed for efficiency on mobile and edge devices. The training of our model will use a carefully curated and augmented dataset of over 12,000 labeled burn images below of three distinct and significant diversity from the Kaggle dataset, along with supplementary data ensuring the sample size was enough for accurate representation of the range of burn categories.

The trained model will be evaluated on several performance metrics such as accuracy, precision, recall, F1 score, and confusion matrices to ensure it can effectively classify varying burns. When we have reached an acceptable level of performance we will export and optimize the model into a TorchScript Lite format, for quick efficient inference on mobile devices. Finally, we will integrate this model into a Flutter mobile application that can take an input image through the gallery or camera. This application will perform a

real-time classification of the input image, as well as leveraging geolocation services to show nearby hospitals, and provide users with immediate first aid instructions. The system will be assessed in a simulation of real.

1.4 Background and Technical Foundations

Deep learning, and specifically Convolutional Neural Networks (CNNs), have made significant advances, and shown much promise in a variety of medical imaging tasks. Some examples include the early detection of skin cancer, screening for diabetic retinopathy, and classifying pneumonia from chest X-rays. The primary feature of CNNs is the ability to automatically learn spatial hierarchies of features from the raw image pixels and eliminate the need for feature engineering.

For this project, we are using MobileNetV3- Large, known for its lightweight architecture and demonstrated accuracy and computational efficiencies. We employ transfer learning, which allows us to initialise the model with ImageNet weights, this enables faster convergence in training and better generalisation in testing. We also added additional fully connected layers and dropout regularisation to help fine-tune the model for the burn classification task.

Data augmentation is an important tool for modelling robustness; for this project we were able to build some advanced data augmentation strategies with the Albumentations library. Geometric augmentations (flips, shifts, rotations), colour (brightness, saturation, hue) and noise (Gaussian noise, coarse dropout) variables were used to simulate real-world environmental variations in skin tone, lighting and obstructions.

The training and evaluation pipeline was built using PyTorch. Note, the final mobile app will integrate the trained model using the PyTorch Lite framework and the programming language Flutter and Dart. Other functional features were embedded to enhance usability such as geolocation mapping with Google Maps, triggering emergency contacts, and guidance first aid in case of burns.

1.5 Need for the Study

This study is based on the need for reliable, accessible, and timely diagnostic tools in burn care as we see a growing need for viable diagnostic options in regions that lack medical

specialist skill sets and diagnostic infrastructures. Traditional methods of diagnosing burn injuries are valuable in a clinical setting but do not offer the same reliability in a consistent diagnosis when the assessment is performed by a non-specialist. Diagnostic options from a clinical setting are impractical to implement in the rural space or in emergency situations within a known disaster. In addition, we also see the rising impact of burn injuries from domestic accidents, industrial accidents, natural disasters, and many variances between. By identifying a better way to conduct early triaging and classification of burn injuries, in which this project will explore, we are also helping to fill a gap of service in that population.

The integration of artificial intelligence with mobile devices may allow a frontline healthcare worker, paramedic; or even patient; to utilize a decision-support tool to provide guidance when there are no specialists or burn facilities available to them. Accompanying evidenced based practice of deep posing analysis within an app on their mobile device may create a trained diagnostic tool that can be shared; and can allow a non-specialist to perform the early initial assessment as indicated by the available clinical framework to which deep learning and clinical diagnostic options mirror one another. This approach and objectives of this study have the potential to offer a one-to-one scalable, low-cost service that can potentially fill a gap between a burn patient's first contact; or assessment; and eventual treatment by a specialist. The project will produce a prototype system that not only is efficient at diagnosing burn injuries but also will provide efficiency in resource allocation and potentially faster patient response times and eventually better outcomes for those who are underserved within those populations.

1.6 Purpose and Objectives

1.6.1 Purpose

To create an artificial intelligence-based approach for the accurate classification of skin burn injuries by applying deep learning and image processing, and to turn this into a mobile application that can assess injuries early-on and remotely triage burn injuries, using and upon a trained model deployed in mobile platform for early injury diagnosis. The application will be enabled with a multi-layer enhancement MobileNetV3 architecture to ensure lightweight deployment upon mobile stamina, whilst retaining weight training and accuracy.

1.6.2 Objectives

The main objectives of this project are to:

To build a deep learning model based on the MobileNetV3 architecture that can accurately classify skin burns into first, second and third degrees of burns, using image preprocessing techniques including resizing, normalization and augmentation.

To build the trained model into a mobile application using Flutter and TensorFlow Lite, enabling the diagnoses of burns in real-time and mapping the location of appropriate treatment nearby.

To appraise and validate the performance of the system using metrics of accuracy, confusion matrix and classification report for performance, and to ensure the solution is light and reliable for deployment in low resource settings.

CHAPTER 2: LITERATURE REVIEW

Deep learning has quickly become a revolutionary technology in the healthcare field, particularly for image-based diagnosis. Utilizing large datasets with little or no prior knowledge allows deep learning to learn hierarchical features and patterns, subsequently enabling new advancements in medical imaging tasks, ranging from cancer detection to organ segmentation. In the case of burns, which are time-sensitive and often diagnosed visually, deep learning can be used to automate and standardize burnt injury classification equally in any number of scenarios, including remote or inaccessible regions.

This chapter consists of a literature review of important deep learning literature related to burns diagnosis, mobile medical imaging, lightweight deployment models, and augmentation methods. The reviewed studies illustrate the progression of CNN-based applications in healthcare, technology, methods and architectures that have bearing on the construction of this project.

2.1 Deep Learning in Burn Depth Prediction

Cirillo et al. (2019) researched a time-invariant model that predicts burn depth using deep convolutional neural networks. The study utilized a number of CNN architectures, including the VGG-16, GoogleNet, ResNet-50, and ResNet-101, and assessed the models' performance on burn images acquired very early on. The authors found that ResNet-101 achieved the highest classification accuracy of 90.5%, thus confirming the viability of deep learning for a timely, automated diagnosis of burn depth. The authors highlighted how developing machine learning models can help clinicians, especially in high-stakes burn care, enable rapid, more consistent decisions.

2.2 Deep Learning in Radiology and Medical Imaging

Mazurowski et al. (2018) reported that deep learning in radiology has become prevalent, particularly in MRI and CT modalities, with CNNs being used for localization, classification, and segmentation tasks for numerous diseases. Statistical analysis showed AI models performed as accurately, or more accurately, as human diagnosticians. This gives weight to the value proposition of using deep learning in visually dependent fields

like radiology—and in turn, burn classification—where accurate visual interpretation is essential for patient diagnosis and treatment plans.

2.3 Implementing Image Classification Models on Mobile Devices

Muneeb et al. (2022) introduced a pipeline which successfully deployed trained deep learning models on mobile devices using TensorFlow Lite and Flutter. The models, trained on data such as chest X-rays for COVID-19 and scans for cancers, also had low-latency requirements for real-time inference. Their study dealt with real-world situations involving model compression, data format and inference speed. Their deployment approach directly informs this project's work of integrating burn classification as a mobile application for use in field work and emergency situations.

2.4 MobileUNetV3 for Lightweight Medical Segmentation

A hybrid architecture combining the lightweight aspect of MobileNetV3 and the the segmentation capability of UNet was introduced by Alsenan et al. (2022): MobileUNetV3, was used to segment spinal cord gray matter and achieved a Dice Similarity Coefficient of 0.87 and specificity of 99.9%, outperforming dense traditional models. Their findings show the capability of a lightweight network to perform a dense medical imaging task and offer an excellent architectural example when adapting MobileNetV3 for burn classification, as this problem requires multi-level feature extraction and multi-level analysis.

2.5 Color Image Augmentation in Medical Imaging

A 2024 systematic review published in Healthcare Analytics highlighted the role of data augmentation for improving medical imaging models and more importantly, the role color-based transformations such as brightness, hue, and saturation shifts could have on improving resilience to different lighting conditions and/or skin tones. This is useful information for burn classification tasks, where there is major variability in image quality based on the camera used, lighting and the patient's skin color. The review supported using augmentation schemes in this project that were based on Albumentations.

SL NO	STUDY	TECHNIQUE	DATASET/ DOMAIN	RESULT	RELEVANCE
1	Cirillo et al. (2019)	ResNet-based CNNs	Burn images (TiVi dataset)	Accuracy: 90.5%	Early burn depth detection using deep CNN
2	Mazurowski et al. (2018)	CNNs in radiology	MRI & CT imaging	AI ≈ Human acc.	Validates CNN in visual diagnosis
3	Muneeb et al. (2022)	DL pipeline, TFLite	COVID, cancer datasets	On-device ML	Relevant for mobile burn classification
4	Alsenan et al. (2022)	MobileUNetV3	Spinal MRI segmentation	DSC: 0.87	MobileNetV3 benchmark in medical imaging
5	Healthcare Analytics Review (2024)	Color augmentation	General medical imaging	↑ Robustness	Supports burn image augmentation techniques

Table 2.1: Literature Review

CHAPTER 3: METHODOLOGY

3.1 Dataset Description

We use the Skin Burns Dataset from Kaggle for this project. The original dataset had more than 45,000 labeled burn images. We drew from a curated dataset of approximately 12,000 images, in order to ensure quality while providing enough diversity for the model to learn. The set was separated into:

Training set: 9073 images
Validation set: 1500 images
Test set: 1500 images

All labeled images were in three classes: 1st-degree, 2nd-degree, and 3rd-degree burns. This arrangement allowed for a valid supervised training design, as well as a strict evaluation of model performance.

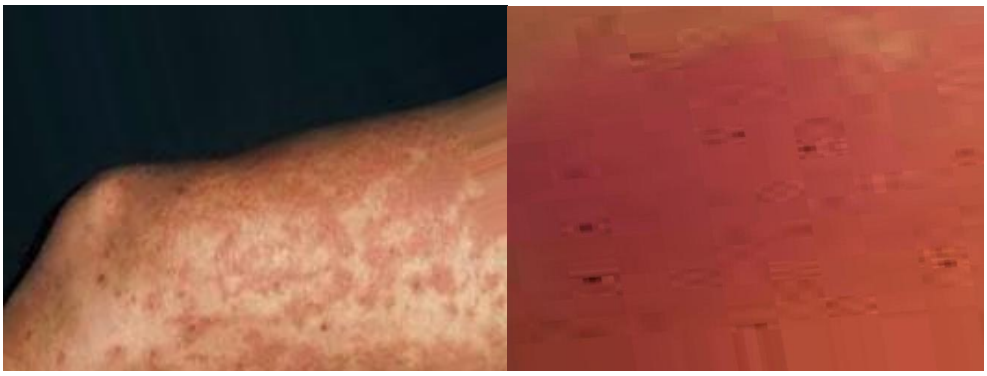

Figure 3.1.1 1st Degree Images Sample

Figure 3.1.2 2nd Degree Images Sample

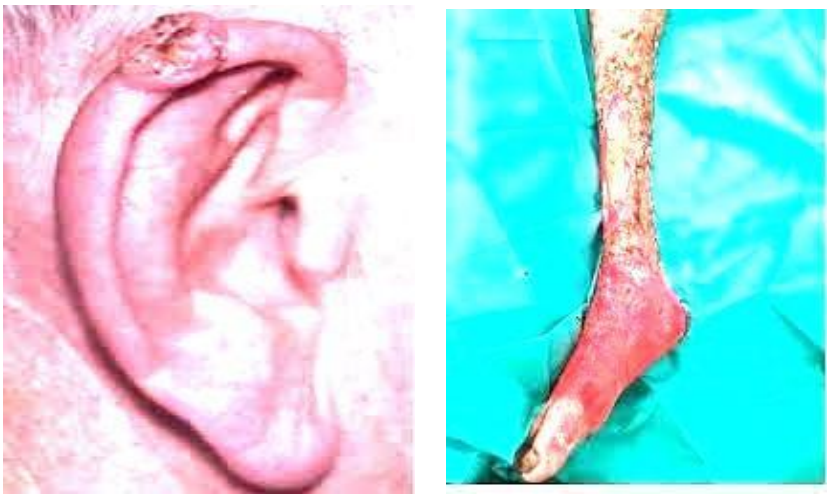

Figure 3.1.3 3rd Degree Images Sample

3.2 Image Preprocessing and Augmentation

3.2.1 Image Resizing and Normalization

All images within the dataset are uniformly resized to a dimension of 224×224 pixels. This standardization ensures consistent input dimensions for the Convolutional Neural Network (CNN), which is crucial for efficient model processing. Following resizing, the images undergo normalization using mean (0.485,0.456,0.406) and standard deviation (0.229,0.224,0.225) values derived from the ImageNet dataset. This normalization step is critical for accelerating the convergence of the training process and achieving improved overall model performance by scaling pixel values to a consistent range.

3.2.2 Data Augmentation Techniques

To make the model more adaptable and reliable in real-world conditions, a variety of data augmentation techniques were applied specifically to the training images. These techniques were implemented using the Albumentations library, which is known for its flexibility and effectiveness in image preprocessing. The main goal of augmentation was to introduce controlled variability into the training data so that the model could learn to recognize burn injuries even when they appear in different orientations, under varying lighting conditions, or with partial obstructions.

One of the simplest yet powerful transformations used was horizontal flipping, applied with a 50% chance. This helped the model become less sensitive to the direction of the image, improving its ability to correctly classify burns whether they appear on the left or right side of the body. Vertical flipping, though less common in real scenarios, was also included at a lower probability to account for cases where the orientation might differ due to how the image was captured.

Beyond flipping, more advanced geometric transformations were introduced through shifting, scaling, and rotating the images. These changes allowed the model to better handle differences in how close or far, straight or tilted the burns appeared in the photos. Color adjustments were another important part of augmentation. Using the ColorJitter technique, slight changes were made to brightness, contrast, saturation, and hue. These adjustments helped simulate different lighting environments and skin tones, making the model more inclusive and effective across diverse users.

To further enhance the model's ability to handle imperfections, Gaussian noise was added to some images. This mimicked the kind of grainy or low-quality photos that might come from older cameras or poor lighting. Additionally, a method called CoarseDropout was used to block out small parts of the image at random. This forced the model to look at multiple areas of the image rather than relying on just one region, teaching it to make decisions based on a more complete understanding.

Overall, these augmentation techniques worked together to create a more varied and realistic training environment, helping the model perform more confidently and accurately when used outside the lab or in real-time mobile applications.

3.3 Model Architecture: BurnClassifier

3.3.1 MobileNetV3-Large as Feature Extractor

MobileNetV3-Large was selected as the feature extractor. MobileNetV3-Large is a lightweight and efficient CNN model, which was previously trained on ImageNet. MobileNetV3-Large employs depthwise separable convolutions and incorporates squeeze-and-excitation blocks and hard-swish activations, meaning it is optimal for devices with fewer resources, like smartphones.

Due to the use of transfer learning, the model has remained with end-to-end generalized visual understanding from ImageNet, but it is able to adapt to the burn classification task with a relatively small dataset.

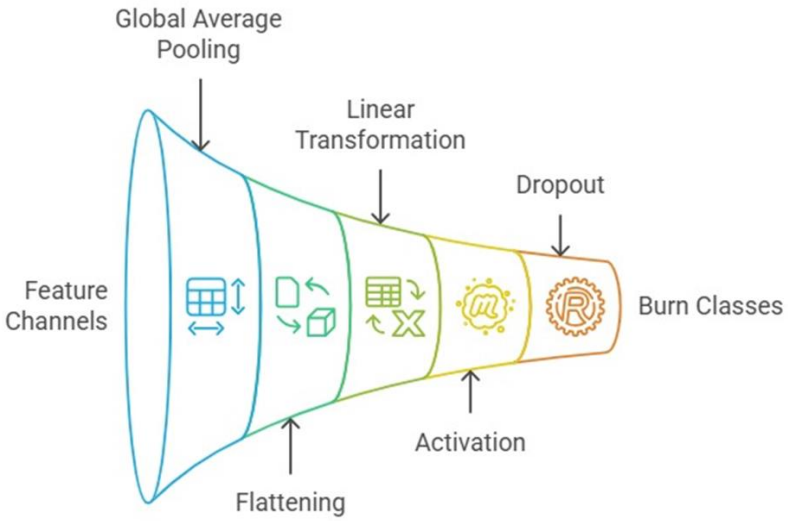

Figure 3.3.1 Model Architecture: BurnClassifier

The core of the classification system is the BurnClassifier model, built upon a pre-trained MobileNetV3-Large backbone.

3.3.2 Custom Classification Head

To accurately determine the severity of burns from images, a specialized classification head was added to the MobileNetV3-Large feature extractor. This component plays a crucial role in converting the complex patterns identified by the backbone into meaningful predictions—specifically, whether a burn is first, second, or third degree. The process starts with an adaptive average pooling layer that condenses each feature map into a single, representative value. This step simplifies the high-dimensional data while still preserving essential spatial details.

Once the feature maps are pooled, they are flattened into a one-dimensional vector so they can be processed by fully connected layers. The classification head itself is composed of multiple dense layers, each designed to gradually distill and refine the information extracted from the image. SiLU activation functions, also known as Swish, are used between these layers to introduce non-linearity and help the network learn more complex patterns. To make the model more robust and reduce the risk of overfitting, dropout layers are also

included after each dense layer, randomly disabling some neurons during training to encourage broader learning.

The first dense layer reduces the feature size from 960 to 512 units and applies a 40% dropout rate. The second layer brings it down to 256 units with a 30% dropout, followed by a third layer that reduces it further to 128 units with a 20% dropout. Finally, the last layer outputs three values—one for each burn category—based on what the model has learned. This streamlined yet powerful classification head ensures that the model can confidently and accurately interpret the complex features of burn wounds, making it highly effective in real-world medical and emergency scenarios.

3.4 Training Strategy

The model is trained using a well-defined strategy involving specific optimization techniques, learning rate scheduling, and early stopping.

3.4.1 Loss Function and Optimizer

We use the AdamW optimizer for training with an initial learning rate of 0.0001 and a weight decay of 1 × 10-4. AdamW is widely recognized as an appealing option for training deep learning models, and weight decay encourages overfitting prevention. In standard deep learning applications of classification, we use a Cross-Entropy Loss as a criterion (specifically, nn.CrossEntropyLoss) which is standard for multi-class classification approaches.

3.4.2 Learning Rate Scheduling

A CosineAnnealingLR scheduler was applied to enable learning rate adjustment during the training process. The CosineAnnealingLR scheduler specifies a learning rate adjustment that follows a cosine annealing schedule, starting from the initial learning rate and reducing to a minimum eta_min (1e-6) over T_max epochs. This approach can result in better convergence and potentially escaping local minima.

3.4.3 Early Stopping Mechanism

To mitigate overfitting, and promote the best performing model, an EarlyStopping mechanism was applied during training. The EarlyStopping mechanism checks for the

validation loss and stops if there is no meaningful change in validation loss for a specified number of epochs. In this case, a patience value of 7 epochs was assigned, so the training would stop if there was no meaningful changes in validation loss for a period of seven epochs. The delta was defined as 0.0005, meaning that any change in validation loss less than 0.0005 was not considered to be an improvement. The verbose option was enabled to provide logs of the early stopping to allow real-time monitoring. When the best model was determined, the best weights were saved to path /content/drive/MyDrive/final1.pth, so the model could be accessed later without the need to retrain.

3.5 Model Evaluation

After training, the model's performance is rigorously evaluated on the unseen test dataset.

3.5.1 Classification Report

The classification report measures how well the model performed across all three classes of burns using many key metrics. Precision refers to the number of true positives made for a particular class over all positive predictions made for that class which indicates how often the model is correctly labeling the different classes and not mistaking it for one another. Recall determines how well the model can predict everything that is a positive class, this gives the model's performance sensitivity. The F1- score is a way of averaging precision and recall by using a harmonic mean which allows for a balance of the two classifications and gives you an optimal measure of performance. Support simply tells you how many true classes there were in the test dataset. In addition, macro average and weighted average values are both provided to allow for an assessment of the model's performance in all classes, providing an overall assessment of how the model performed and identify if there were any classes that the model had significant weaknesses in.

3.5.2 Confusion Matrix

The confusion matrix is generated using sklearn.metrics.confusion_matrix. A confusion matrix shows the overall performance of the model in terms of number of true positives, true negatives, false positives, and false negatives for each class. The confusion matrix is helpful to understand where the model is making mistakes.

3.6 Model Export and Mobile Deployment

3.6.1 TorchScript Tracing

As the initial step in enabling the model for deployment on mobile, the working, trained PyTorch model first became a TorchScript model. This was done using the torch.jit.trace() function, which traces the working model using a dummy input tensor. The tensor acts as a reasonable representation of a standard image by being shaped as torch.rand(1, 3, 224, 224). By using a dummy input, it also enables the tracing function to analyse the model's architecture and pattern of execution (Bai, 2019).

TorchScript is a statically typed, serializable version of the model; thus enabling deployment to platforms without dependencies on Python. The framework assists in exporting the model in a manner that makes it portable, for use by the PyTorch Lite runtime on mobile.

3.6.2 Adjustment for Mobile

After tracing, the TorchScript model passed through torch.utils.mobile_optimizer.optimize_for_mobile(). This command runs the TorchScript model through mobile optimizations to improve inference speed and reduce resources. Some of the optimizations include:

Operator fusion which fuses sequential Operators into optimized single Operator.

Dead code removal where unnecessary portions of the computation graph are removed.

Memory layout optimized to maximize tensor operations on mobile architectures.

These optimizations are useful for performance-oriented applications, especially real-time applications requiring fast response. With these optimizations, the model is much lighter and more efficient to run on constrained devices such as smartphones.

3.6.3 Model Saving

The optimized model was saved in the .ptl format (PyTorch Lite) with the name burnaidmodel.pt. This is a file format used specifically for mobile applications using PyTorch Lite interpreter. The model was placed in the assets/models/ folder of the Flutter

project, to enable the app to locally load the model with the pytorch_lite Flutter plugin. This allows all predictions to run locally on their devices without an internet connection, thus requiring no backend or server-side support. On-device inference not only provides faster results and lower latency, but also stronger privacy and the ability for offline capability, which are ideal for emergency and remote healthcare applications.

3.7 Mobile App Integration

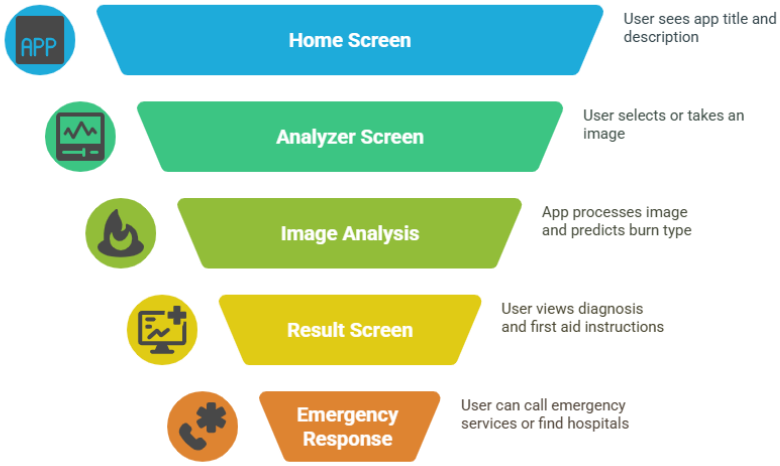

Figure 3.7.1 Mobile app integration

The mobile application, which is developed in this project, is built in Flutter, a modern, cross-platform UI toolkit powered by Dart. The development of the mobile application happened in Android Studio. Android Studio served as a stable environment for UI design and the integration of all the backend logic. The AI integration is based on the PyTorch Lite plugin, which enables the usage of TorchScript models in a Flutter app, meaning that real-time inference can happen offline.

3.7.1 Development Environment

The mobile app was developed using Flutter, an open-source UI framework created by Google, which allowed for quick development and responsive cross-platform development of the mobile app. Dart was the programming language used to implement both the user interface and application logic. The app was built and tested in Android Studio in an integrated development environment, which provided a reliable place to manage and use

emulators and real devices. As for the deployment of the deep learning model, the PyTorch Lite plugin was used, in which the converted .ptl model ran seamlessly on mobile devices. The trained model was exported in the TorchScript Lite format so that it could be set up to run on the Flutter platform and conduct inference on the device without needing internet connectivity.

3.7.2 Application Architecture

The application was designed with three main screens: HomeScreen, AnalyzerScreen and ResultScreen. When the user opens the app, they see the HomeScreen, which has the app title, an icon, a short description, and a button that takes them to the AnalyzerScreen; this means that the HomeScreen is the place to start for the user to begin the process of diagnosing the burn.

In the AnalyzerScreen, the user can take a new picture from the camera or select an option from the gallery. The picking feature was implemented through the image_picker package. After the user chooses the image, that image is displayed on the screen, and the user can click the "Analyze" button to complete the burn analysis. At this point, the app automatically loads the burnaidmodel.pt model through the PytorchLite.loadClassificationModel method; the image will be resized to 224×224 shape before being passed into the model. The model returns raw prediction scores (logits) which are passed through the softmax function to produce probabilities and the model determines the highest probability score as the predicted burn type (first degree, second degree, or third degree).

3.7.3 Result Display and First Aid Integration

When a prediction is made, the application moves to the ResultScreen. where the image of a burn is shown, in addition to their result. The result will identify the burn (burn type) and will report the classification result as a probability percentage (e.g, "This appears to be a first-degree burn with an 87% probability"). The ResultScreen also shows the first aid instructions based on the application's predicted severity of the burn type. A first-degree burn might say "cool the burn with some water, and you can apply aloe vera" whereas a third-degree burn would direct the user to "seek emergency medical assistance immediately."

There is also an emergency response option on the ResultScreen. If the model predicts a third-degree burn there is a button active that will open the phone dialer with the emergency number (112) pre-filled, so the user can instantly call for emergency services. The emergency response feature is done using the url_launcher plugin. The app will also show a button that can locate nearby burn treating facilities or hospitals. When pressed, it will call the geolocator package, create a GPS location, and then open Google Maps with a query for "burn treatment centers," for quick access to medical treatment.

3.7.4 Thematic Design and Navigation Flow

This application utilizes Flutter's navigation system. The main.dart file is responsible for defining the root of the application and initiating the BurnAId class. The theme is built around a teal color scheme to represent calmness and healthcare undertones. The HomeScreen is defined as the initial screen shown, and this was accomplished through routing to the AnalyzerScreen and from there to the ResultScreen by using MaterialPageRoute with the provided route. Text styling, padding, and button shapes are defined as needed throughout the application to establish fluid visual consistency and improve usability.

3.8 Comparison with Alternatives

Model	Accuracy	Mobile Compatibility	Inference Speed
Custom CNN	97%	Medium	Fast
MobileNetV3	96%	High	Fast
EfficientNet-Lite	82%	High	Moderate

Table 3.8.1 Comparison of Models

MobileNetV3 was chosen as our base model over other architectures for its excellent trade-off between accuracy, speed, and low memory footprint, making it perfect for mobile applications.

3.9 Software and Execution Requirements

The burn classification system included both software tools for training models and mobile app development. The model was written in Python 3.10 or higher using PyTorch version 2.x as the primary deep learning framework and relied on additional packages such as Albumentations for data augmentation, Matplotlib for plotting visuals, and Scikit-learn for model performance evaluation in the various stages of pre-processing, training and validation. For mobile development, the app was developed in Android Studio and employed the Flutter SDK with the Dart programming language, Dart version 3.2 or greater.

In terms of hardware, development of the model and the Android app compilation occurred on a system with at least an Intel i5 equivalent processor, and a minimum of 8 GB RAM. Once finished, the mobile application was tested on Android devices running Android version 8.0 (Oreo) or higher to allow it to be compatible with a wide range of smartphones. Model training and experimentation was done using Google Colab where the cloud-based GPU accelerated training was beneficial to training cycles. The final version of the app was deployed and tested on the Android emulator and then with physical devices to confirm performance and usability in real-world scenarios.

CHAPTER 4: RESULTS AND CONCLUSION

4.1 Final Architecture

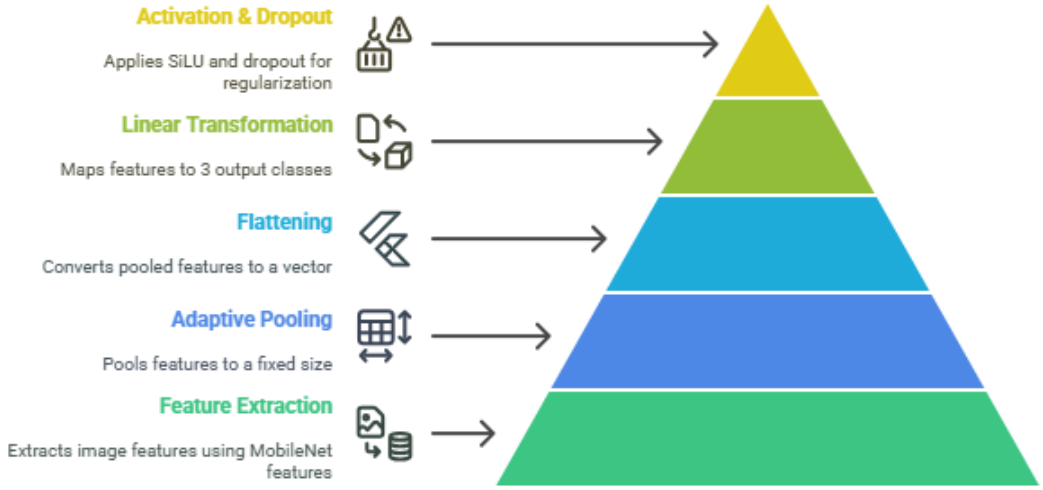

Figure 4.1: Multi layered CNN Architecture Pyramid

The final model architecture adopted in this project's implementation was based off of an efficient, small footprint, convolutional neural network called MobileNetV3-Large that was built for low-level deployment on mobile and edge devices. In this case, the pre-trained (on ImageNet) MobileNetV3 model provided the backbone for the project; transfer learning was utilized—fine-tuning the MobileNetV3 model to discriminate first, second, and third-degree burns using the curated burn dataset. While using transfer learning meant the model could benefit from previous general learned visual features, it was focused on "learning" the specific discriminative differences in the types of burn severity.

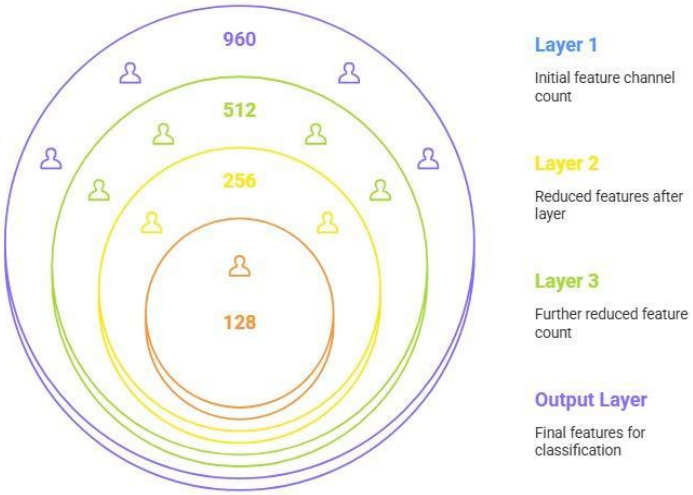

Figure 4.2: Feature Reduction in Custom Classifier

In order to fit the burn classification, MobileNetV3 required a custom classification head added. The custom classification head composed of 3 layers of fully connected neural networks progressively decreased dimensionality of the feature map outputted by the backbone network (from 960-dimensions to 512 → 256 → 128) - employing SiLU activation and a regularizer (Dropout). The last output layer will give the final class probabilities of the burn image. A progressive approach to the reduction of dimensions was adopted to filter out over-fitted data, and intended to allow the model to focus on the discriminative feature patterns of each burn (class distance). The final model successfully replicated a very high classification accuracy percentage at minimal processing load, permitting sufficient lead time for deployment on mobile-based applications with real-time requirements.

4.2 Performance Metrics

4.2.1 Accuracy and Loss Study

During the training phase, the MobileNetV3 model displayed an ongoing trend of improvement for both training and validation accuracies, indicating learning was successfully occuring. The accuracy curve approached to 97% and continued to move upward, while training loss continued to decline—indicating minimal classification errors. Validation loss showed some variance, which was expected given the limited validation dataset available and potential noise associated 3.38with medical images. Of importance was observing no evident spikes in loss value or divergence in training/testing loss value,

indicating that the model was not overfitting, and maintained good generalization properties.

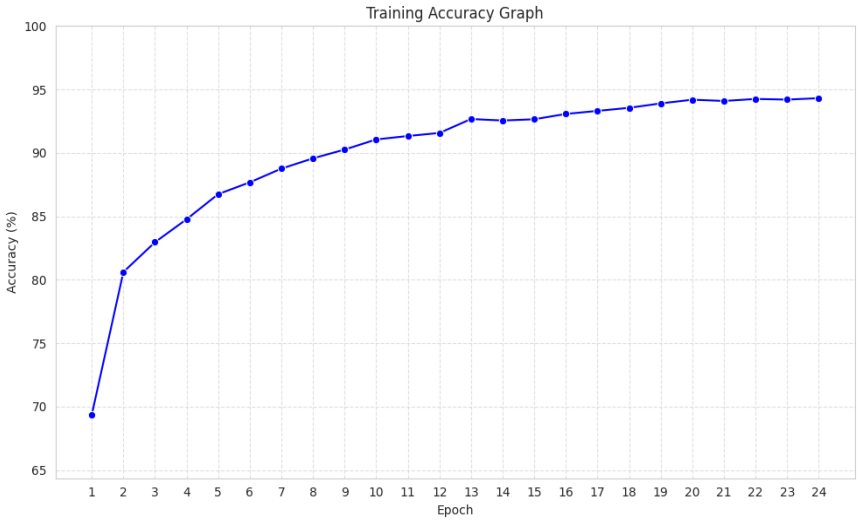

Figure 4.3: Training Accuracy Graph

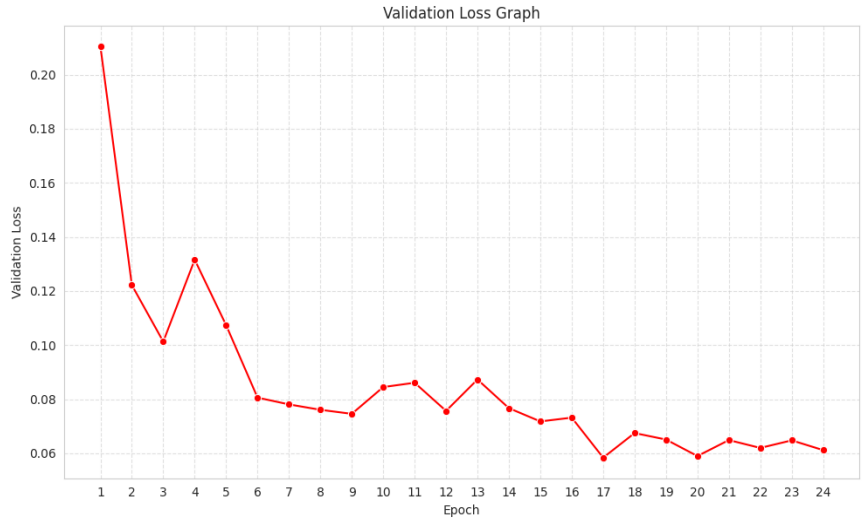

Figure 4.4: Validation Loss Graph

4.2.2 Confusion Matrix

To better understand the performance of the model, confusion matrices were produced for three models: a custom Multilayered CNN, MobileNetV3, and EfficientNet Lite.

The custom CNN performed the best, accurately classifying burn categories, with few misclassification errors.

MobileNetV3 performed just below the custom CNN but offers a trade-off between performance and ease of use due to its lightweight deployment. Note that the accuracy of MobileNetV3 was slightly decreased compared to the custom CNN, however, the loss of predictive ability is still strong.

The EfficientNet Lite model was able to inference quickly but lacked precision and accuracy when classifying the burn categories, particularly '2nd-Degree Burn', the distinguishing features were less pronounced and on average required a more convoluted architecture.

The results confirmed that MobileNetV3 could be used for mobile deployment, while the custom CNN could be used for model benchmarking.

4.2.3 Precision, Recall, and F1-Score

The classification report for the final deployed MobileNetV3 model showed balanced performance across the metrics. Each class maintained a precision, recall, and F1-score all around 0.97. The macro-avg, and weighted-avg metrics averaged all around 0.97 as well, indicating the model is robust in all three classes of burn severity.

The confusion matrix associated with this classification report demonstrated that the model performed well with regards to 3rd degree burns- even though they are the most severe and visually complicated of the classes and it is important cause in real-world application the patient needs real-time treatment.

Metric	Precision	Recall	F1-score	Support
1st Degree	0.97	0.98	0.98	500
2nd Degree	0.97	0.97	0.97	500
3rd Degree	0.98	0.96	0.97	500
Accuracy			**0.97**	**1500**
Macro avg	0.97	0.97	0.97	1500
Weighted avg	0.97	0.97	0.97	1500

Table 4.1: Classification Report of Custom CNN Model

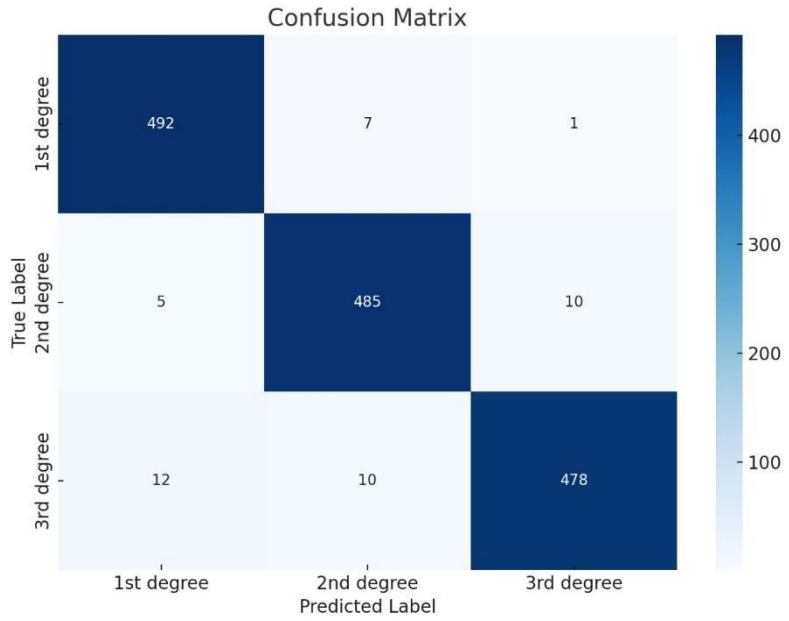

Figure 4.5: Confusion Matrix of Custom CNN Model

Metric	Precision	Recall	F1-score	Support
1st Degree	0.94	0.99	0.97	500
2nd Degree	0.97	0.96	0.96	500
3rd Degree	0.98	0.94	0.96	500
Accuracy			**0.96**	**1500**
Macro avg	0.96	0.96	0.96	1500
Weighted avg	0.96	0.96	0.96	1500

Table 4.2: Classification Report of MobileNetV3 Model

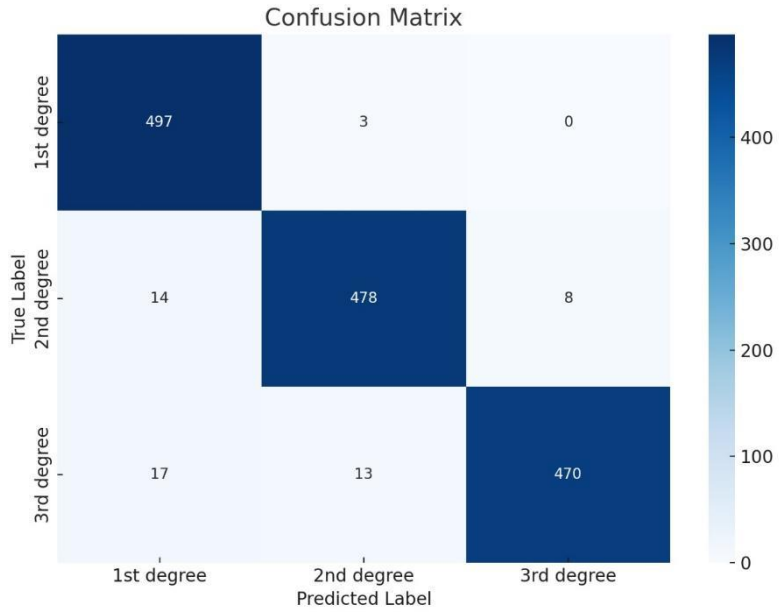

Figure 4.6: Confusion Matrix of MobileNetV3 Model

Burn Degree	Precision	Recall	F1-Score	Support
1st Degree	0.91	0.86	0.88	500
2nd Degree	0.75	0.82	0.78	500
3rd Degree	0.81	0.78	0.80	500
Accuracy			**0.82**	**1500**
Macro Avg	0.82	0.82	0.82	1500
Weighted Avg	0.82	0.82	0.82	1500

Table 4.3: Classification Report of EfficientNet-Lite Model

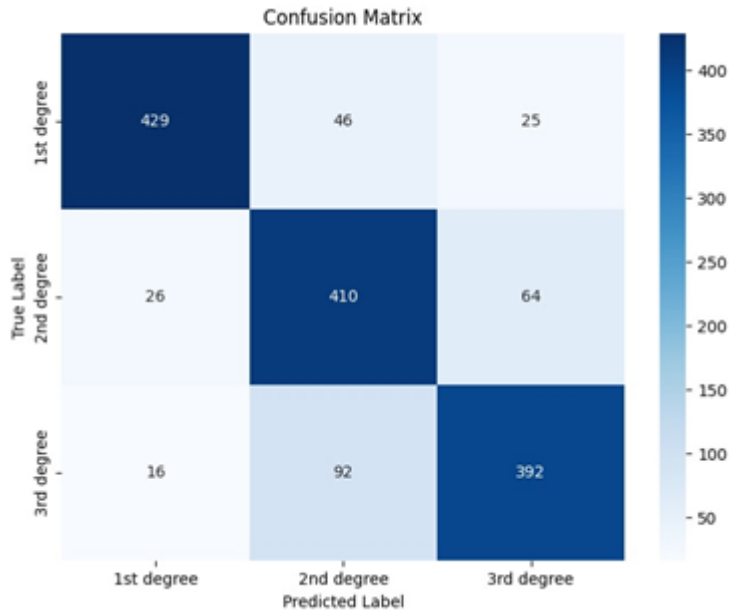

Figure 4.7: Confusion Matrix of EfficientNet-Lite Model

4.3 Conclusion

This project has shown how deep learning and mobile technologies can be combined to solve authentic medical classification challenges. Through training and evaluating three unique models (Multilayered CNN, MobileNetV3, EfficientNet Lite), this project engaged with an overall model that was both accurate and appropriate for mobile use. Although the custom CNN section of this project had the most accuracy, MobileNetV3 served as a good balance between performance and mobile usability.

The Flutter app deployed returns an instant classification result along with first aid, as soon as the user can click or upload a burn image. In case a third-degree burn is found the app does not only instantly provide the option to call emergency services but also locates nearest hospitals through geolocation with quick access links. This app effectively makes the AI model a full digital healthcare assistant ready to readily use in real time in high-stakes environments like remote clinics, emergency response teams, and rural health camps.

4.4 Future Work

The future enhancements of this system can include reaching out to the hospital databases and expanding the dataset with clinically verified real world burn images as the dataset

will improve the accuracy and generalizability of the model. Additionally, the dataset can be enhanced by making it temporal by using sequenced images that track the burn as it progresses over time, which will also include monitoring the healing process. Visual explainability methods (i.e., Grad-CAM), can further be added to provide clinicians and users interpretability and transparency about why the AI made the predictions it made.

Language localization and offline capabilities can improve accessibility for users in resource limited, underserved regions. Finally, formalizing the deployment through pilot testing with hospitals to deliver real time feedback and evaluate clinical usability may allow this system to be embedded into healthcare workflows formally.

REFERENCES

1. Cirillo, M.D., Garnavi, R. (2019). *Time-Independent Prediction of Burn Depth using Deep Convolutional Neural Networks*. Sensors, 19(15), 3269. https://doi.org/10.3390/s19153269
2. Mazurowski, M.A., Buda, M., Saha, A., Bashir, M.R. (2018). *Deep Learning in Radiology: An Overview of the Concepts and a Survey of the State of the Art with Focus on MRI*. Journal of Magnetic Resonance Imaging, 49(4), 939–954. https://doi.org/10.1002/jmri.26534
3. Muneeb, M., Irfan, M., Aziz, A., Ahmad, N., Rahman, Z. (2022). *Deep Learning-Based Image Classification Pipeline on Mobile Devices*. Journal of Computer Science, 18(8), 663–670.
4. Alsenan, S., Farag, W., Fadel, A. (2022). *MobileUNetV3: A Combined UNet and MobileNetV3 Architecture for Spinal Cord Gray Matter Segmentation*. Computers in Biology and Medicine, 146, 105601. https://doi.org/10.1016/j.compbiomed.2022.105601
5. Li, Y., Chen, R., Wang, M., Zhao, L. (2024). *A Systematic Review of Deep Learning Data Augmentation in Medical Imaging*. Healthcare Analytics, 4(1), 100135.
6. Howard, A., Sandler, M., Chu, G., Chen, L.C., Chen, B., Tan, M., Le, Q.V., Adam, H. (2019). *Searching for MobileNetV3*. Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 1314–1324.
7. TensorFlow. (2022). *TensorFlow Lite Guide*. Available at: https://www.tensorflow.org/lite
8. Google Developers. (2023). *Flutter: Build Apps for Any Screen*. https://flutter.dev
9. Kingma, D.P., Ba, J. (2014). *Adam: A Method for Stochastic Optimization*. arXiv preprint arXiv:1412.6980.
10. Kermany, D.S., Goldbaum, M., Cai, W., Valentim, C.C.S., Liang, H., Baxter, S.L., McKeown, A., et al. (2018). *Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning*. Cell, 172(5), 1122–1131.