

DermaLens: An AI-Powered Web Platform for Skin Lesion Analysis and Clinical Decision Support

Bibin Bose Sivananthan¹, B. Dhanush^{2,*}, Amarilys González García³, Bushra Rehman⁴, Kotova Anna Sergeevna⁵,
Cheng Ziming⁶

¹School of Nursing, Ingram School of Nursing, McGill University, Montreal, Quebec, Canada.

²Department of Artificial Intelligence and Machine Learning, SRM Institute of Science and Technology, Ramapuram,
Chennai, Tamil Nadu, India.

³Department of Research and Development, Placental Histotherapy Center, Playa, Havana, Cuba.

⁴Institute of Pathology and Diagnostic Medicine, Khyber Medical University, Peshawar, Khyber, Pakistan.

⁵Department of Pathological Physiology, Kyrgyz-Russian Slavic University named after the first President of the Russian
Federation Boris Nikolaevich Yeltsin, Bishkek, Kyrgyzstan, Russia.

⁶Department of Computing and Information Systems, Singapore Management University, Singapore.
bibin.sivananthan@mcgill.ca¹, bb6504@srmist.edu.in², agonzalez@miyares-cao.cu³, drbushra.ipdm@kmu.edu.pk⁴,
anna_kotova_2012@mail.ru⁵, zm.cheng.2026@phdcs.smu.edu.sg⁶

Abstract: Automated interpretation of dermoscopic skin images is difficult because visually similar lesions can belong to different diagnostic groups, and the HAM10000 benchmark is markedly imbalanced. DermaLens AI addresses this problem through a browser-accessible platform that combines a MobileNetV2 classifier with contextual clinical information and reporting services. The model starts from ImageNet weights and is adapted to the seven HAM10000 categories: melanocytic nevi, melanoma, benign keratosis, basal cell carcinoma, actinic keratosis, vascular lesion, and dermatofibroma. Its training workflow uses augmentation, lesion-level deduplication, stratified partitioning, weighted sampling, focal loss, AdamW, cosine-annealing warm restarts, gradient clipping, and early stopping. A FastAPI/PyTorch inference service accepts JPEG, PNG, and WEBP files, standardizes them to 224 x 224 pixels, ranks the five leading class probabilities, and produces a structured report with risk information, ABCD-related descriptors, recommendations, and differential conditions. The platform also contains a broader disease knowledge base, optional symptom and anatomical-location inputs, a browser assistant, clinic search with maps, and printable reports. The accessible project snapshot does not contain a verified final training-history artifact or final test metrics; consequently, this manuscript reports the reproducible procedure and system-level behavior without fabricating accuracy Figures 1 and 2. The work demonstrates an integrated architecture for educational and research-oriented skin-lesion analysis, not a clinically validated diagnostic service.

Keywords: Skin Lesion Analysis; MobileNetV2 and HAM10000; Dermoscopy and Deep Learning; Medical Image Classification; Fast-API and Focal Loss; Clinical Decision Support; Explainable AI.

Received on: 02/08/2025, **Revised on:** 15/10/2025, **Accepted on:** 08/11/2025, **Published on:** 03/06/2026

Journal Homepage: <https://www.fmdbpublish.com/user/journals/details/FTSBE>

DOI: <https://doi.org/10.69888/FTSBE.2026.000791>

Cite as: B. B. Sivananthan, B. Dhanush, A. G. García, B. Rehman, K. A. Sergeevna, and C. Ziming, “DermaLens: An AI-Powered Web Platform for Skin Lesion Analysis and Clinical Decision Support,” *FMDB Transactions on Sustainable Biomedical Engineering*, vol. 1, no. 2, pp. 96–112, 2026.

*Corresponding author.

1. Introduction

Dermatological diagnosis relies substantially on visual examination, clinical context, and specialist expertise. Dermoscopic imaging reveals visual structures that may be difficult to appreciate in a conventional photograph, but the resulting diagnostic task remains challenging because benign and malignant lesions can exhibit overlapping morphology, color, border characteristics, and internal patterns [10]. Convolutional neural networks have therefore become a major research approach for automated skin lesion analysis, especially when adequate labeled image data are available. HAM10000 dataset provides 10,015 dermoscopic images across seven diagnostic categories and was released specifically to support machine-learning research on common pigmented skin lesions [1]. Prior studies indicate that deep-learning models can achieve strong results on selected dermatological image tasks [15]. However, generalization across populations, devices, skin tones, acquisition conditions, and clinical workflows remains an important research concern [2].

A usable system consequently requires more than the prediction model alone. Such a system requires controlled preprocessing, dependable inference, and transparent uncertainty reporting that supports clinical context and safeguards against interpreting a model output as a definitive diagnosis [11]. DermaLens AI was developed around this system-level view. The application does more than return one of seven labels; it transforms an uploaded image into a structured analysis that includes a primary lesion category, confidence information, differential conditions, risk descriptors, clinical recommendations, and an optional list of dermatology clinics [3]. The implementation uses Python, PyTorch, TorchVision, FastAPI, Pillow, NumPy, HTML, CSS, and JavaScript, and Docker for deployment support [4]. The paper repository provides a MobileNetV2 model class, a training script, a FastAPI backend, and a browser interface [13]. The backend includes a seven-class CNN path and an optional multimodal vision path using Anthropic Claude when an API key is configured [12]. If the external vision service is unavailable or no key is supplied, the application falls back to local MobileNetV2 inference. This two-path arrangement balances local reproducibility with optional multimodal clinical reasoning [5].

The project's main contribution is a reproducible end-to-end architecture that integrates a lightweight convolutional backbone into a web application and supports it with a structured clinical knowledge layer [6]. The model is trained using focal loss and weighted sampling to lessen the effect of the pronounced class imbalance in HAM10000 [14]. The API standardizes incoming images, provides REST endpoints, records inference time, and returns a machine-readable response that can be formatted as a user-facing report [7]. The browser interface additionally provides camera capture, condition information, clinic mapping, and a rule-based skin assistant [8]. The rest of the manuscript is organized in the same way as a full software-oriented intelligent informatics journal article: a literature review sets the research background; the proposed method describes the architecture, technology choices, training procedure, and reporting mechanism; the results section describes the measurable behavior supported by the repository; the discussion addresses findings, security, future work, deployment, and clinical safety; and the paper concludes with limitations and research-material statements [9].

2. Review of Literature

DermaLens AI draws on four closely related areas: automated skin-lesion classification, benchmark dermoscopy datasets, efficient transfer-learning architectures, and clinical decision-support interfaces. Previous studies also identify recurring issues involving class imbalance, model calibration, population bias, and the difficulty of translating benchmark performance into safe clinical use [1].

2.1. AI-Assisted Skin Lesion Classification

Research with convolutional models shows that visual features learned directly from labeled dermatological images can distinguish common lesion categories. Esteva et al. reported dermatologist-comparable performance in a carefully defined image-classification study involving skin cancer recognition. In contrast, later studies emphasized that model behavior depends strongly on dataset composition and the distinction between benchmark classification and real clinical diagnosis. For deployment, this implies that predictions should be communicated as decision-support information, not a definitive diagnosis [2]. The ISIC ecosystem has contributed to standardized evaluation of skin-lesion analysis and has enabled researchers to compare methods using curated dermoscopic images. However, benchmark datasets may not fully represent the distribution of lesions seen across all clinical environments. Dataset priors, repeated lesions from the same patient, imaging-device differences, and representation gaps may influence measured results. DermaLens's implementation addresses one aspect of this concern by deduplicating HAM10000 records at the lesion-identifier level before splitting the data, which reduces the chance that multiple images of the same lesion cross train and evaluation partitions.

2.2. HAM10000 Dataset and Dermoscopic Imaging

HAM10000 contains 10,015 dermoscopic images grouped into seven diagnostic categories: melanocytic nevi (nv), melanoma (mel), benign keratosis-like lesions (bkl), basal cell carcinoma (bcc), actinic keratoses/intraepithelial carcinoma (akiec), vascular lesions (vasc), and dermatofibroma (df). The dataset was assembled from multiple sources and modalities, and more than half of the lesions have pathology confirmation. The remaining diagnoses were established through follow-up, expert consensus, or in vivo confocal microscopy [3]. One important characteristic of HAM10000 is the highly uneven class distribution. Melanocytic nevi dominate the dataset, whereas dermatofibroma and vascular lesions are much less frequent. A classifier trained directly with ordinary cross-entropy can consequently receive substantially more optimization signal from the majority class. DermaLens directly accounts for this imbalance with a weighted sampler and a focal-loss objective, combining data-level and loss-level rebalancing. This design follows the broader approach described in the literature on learning under class imbalance.

2.3. Transfer Learning and MobileNetV2

MobileNetV2 is a computationally efficient convolutional architecture that uses inverted residual blocks, expansion layers, depthwise convolutions, and linear bottlenecks. Its architecture balances predictive capability with computational cost, making it useful when inference resources are constrained. In DermaLens, the ImageNet-learned feature set is replaced, and the original classification head is replaced by a compact seven-class head consisting of adaptive global pooling, dropout, a 1280-to-256 fully connected layer, ReLU activation, and a final 256-to-7 output layer. Researchers also include a learnable temperature parameter to scale logits during training and inference [5]. The training configuration uses layer-wise learning rates. Early MobileNetV2 feature layers receive the smallest learning rate, later feature layers receive a larger rate, and the custom classifier receives the base learning rate. This approach retains generic visual features while permitting the task-specific layers to adapt more quickly. Researchers use AdamW with weight decay, and cosine-annealing warm restarts provide periodic learning-rate modulation.

2.4. Class Imbalance and Focal Loss

Focal loss was introduced to reduce the dominance of easy examples by reshaping cross-entropy so difficult examples have greater influence. Although originally introduced for dense object detection, the underlying idea is useful for imbalanced classification problems where a majority class can overwhelm less frequent classes. DermaLens implements focal loss with gamma equal to 2.0 and multiplies the cross-entropy term by class weights computed from training-set frequencies. A weighted random sampler independently constructs balanced training batches [6]. The two mechanisms serve different purposes: sampling changes the observations presented during optimization, whereas focal loss changes the relative contribution of the remaining hard examples to the objective. However, combining both mechanisms can also make optimization sensitive to the effective loss scale. To improve training stability, the implementation uses gradient clipping at a maximum norm of 1.0 and early stopping based on validation accuracy.

2.5. Multimodal Clinical Report Generation

Image classification represents only one part of a practical clinical interface. DermaLens includes a structured knowledge layer that stores disease names, scientific names, ICD-10 identifiers, categories, prevalence statements, risk descriptions, ABCD-style descriptors, differential diagnoses, recommendations, and urgent-care flags. The local CNN output is mapped into this knowledge base before the result is returned to the user. The backend can also use a multimodal language model to analyze the image alongside optional patient metadata such as age, sex, anatomical location, symptom description, and duration. This multimodal layer is intentionally separated from the primary CNN classes [7]. The seven-class model provides the learned image evidence. At the same time, the knowledge layer can cover a broader set of conditions and can use symptoms and lesion location to construct secondary-condition suggestions. In the local path, symptom keywords carry more weight than location keywords, and the system selects up to five secondary conditions from the resulting candidate scores. This arrangement is more accurately characterized as contextual decision support than as a second trained classifier.

2.6. Existing Skin Lesion Analysis Systems: Gaps and Limitations

Available systems span conventional rule-based image processing to deep neural networks and multimodal clinical assistants. Standalone classifiers may obtain strong benchmark results but often expose only a label or probability. General-purpose chat assistants can provide broad medical information but may lack a reproducible image-classification core. Cloud-dependent systems add infrastructure and data-transfer requirements, whereas local systems may have limited knowledge coverage or weak user interfaces. DermaLens combines elements from these approaches by joining a local seven-class deep model, a structured disease knowledge base, a REST service, a browser UI, and optional multimodal reasoning [8]. The research gap considered here is primarily architectural, not a claim of superior diagnostic accuracy. The implementation illustrates how an

efficient CNN can be wrapped in a practical analysis workflow in which the prediction is contextualized, reported, and connected to possible follow-up actions. The repository also makes clear that production deployment would need stronger authentication, persistent storage, privacy safeguards, clinical validation, and independent prospective evaluation before any clinical use.

3. Proposed Method

DermaLens AI follows a service-oriented web-application design with three principal layers: the browser presentation layer, the FastAPI service layer, and the deep-learning/clinical reasoning layer. The workflow receives a lesion image and optional metadata, validates the request, selects an inference route, builds a structured response, and returns both machine-readable JSON and a browser-rendered report. The pipeline is organized to keep components modular and easy to inspect, allowing independent testing of the CNN, knowledge layer, and user interface.

3.1. System Architecture Overview

The presentation layer consists of a single-page HTML/CSS/JavaScript interface. The user-facing workflow is organized into Home, Analysis, Clinics, and AI Chat pages (Figure 1).

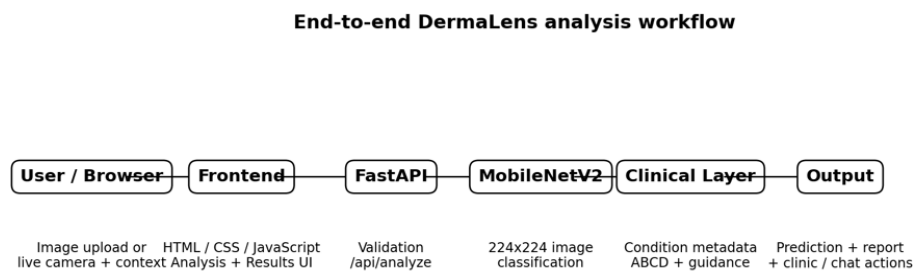


Figure 1: End-to-end derma-lens analysis workflow from image acquisition to clinical output

The Analysis page supports image upload, live camera capture, patient fields, anatomical location, duration, and symptom input. The FastAPI layer exposes endpoints for health and disease listings, analysis, registration, login, and chat. The model layer contains (Table 1).

Table 1: System architecture layers

Layer	Technology / Component	Responsibility
Presentation	HTML, CSS, JavaScript	Image upload, camera capture, forms, results, maps, assistant, report UI
Service	FastAPI 0.111.0	REST endpoints, validation, orchestration, response formatting
Model	PyTorch / MobileNetV2	Seven-class dermoscopic image classification
Knowledge	Python disease knowledge base	Risk, ABCD descriptors, differential conditions, recommendations
Optional multimodal	Anthropic Claude API	Image + metadata clinical reasoning when API key is configured
Deployment	Docker / Uvicorn	Containerized and ASGI application serving

Image → Validation → 224×224 normalization → MobileNetV2 / Optional Vision Model → Top-5 Inference → Knowledge Layer → Clinical Report → Ui / PDF / Clinic Recommendation.

3.2. Technology Stack

The technology choices balance research reproducibility with a usable browser interface. PyTorch and TorchVision provide the deep-learning environment; Pillow and NumPy handle image conversion; Pandas and scikit-learn support dataset preparation and evaluation; FastAPI exposes the inference service; Uvicorn provides the ASGI runtime; Docker supports packaging; and the frontend uses standard browser technologies, with Leaflet for map visualization (Table 2).

Table 2: Full technology stack

Category	Technology / Version	Purpose
Programming	Python	Training, inference, backend logic
Web framework	FastAPI 0.111.0	REST API and application serving
ASGI server	Uvicorn 0.30.1	Run the FastAPI application
Deep learning	PyTorch ≥ 2.2 / TorchVision ≥ 0.17	Model training and inference
Backbone	MobileNetV2 ImageNet V1	Efficient feature extraction
Image processing	Pillow ≥ 10.3 / NumPy ≥ 1.26	Decode, resize, normalize images
ML utilities	Pandas / scikit-learn	Metadata handling, splitting, reports
Visualization	Matplotlib / Seaborn	Training curves and confusion matrix generation
Frontend	HTML, CSS, JavaScript	User interface and client workflow
Maps	Leaflet 1.9.4	Clinic-map visualization
Optional AI	Anthropic Claude	Multimodal clinical reasoning and chat endpoint
Deployment	Docker / docker-compose	Portable deployment and environment setup

3.3. Module Structure

The paper repository keeps the training script, inference service, frontend, data directories, models, and deployment files separate. The root-level training script builds the dataset, runs fine-tuning, evaluates the test split, and saves the best weights and diagnostic plots (Table 3).

Table 3: Module structure and responsibilities

Module / File	Type	Purpose
train.py	Training script	HAM10000 loading, augmentation, split, sampler, focal loss, optimization, evaluation
backend/main.py	FastAPI backend	Model, knowledge base, inference, reporting, APIs, logging, auth prototype
frontend/index.html	Frontend	Dashboard, analysis, clinics, assistant, report rendering
download_dataset.py	Utility	Dataset acquisition helper referenced by the project
diagnose.py	Diagnostic utility	Checks Python environment and required packages
Dockerfile / docker-compose.yml	Deployment	Containerized execution
models/	Model artifacts	Custom MobileNetV2 checkpoint location when available
assets/ / docs/	Project support	Static assets and documentation

The backend main module defines the model architecture, knowledge base, inference logic, API routes, logging, authentication prototype, and optional Claude integrations. The frontend index.html contains the current application interface, report layout, clinic data presentation, and local assistant knowledge base.

3.4. User Authentication and Access Control

The current backend implements authentication as a prototype using in-memory dictionaries. Registration stores a SHA-256 password hash in `MOCK_DB_USERS` and creates a UUID session token in `MOCK_SESSIONS`. Login compares the submitted hash with the stored value and issues another UUID token. This setup suits a prototype interface. Still, it should not be considered equivalent to production identity management because sessions and user records are process-local, and a persistent database or a standardized authorization framework does not back access tokens (Table 4).

Table 4: Authentication and access-control prototype matrix

Feature	Current implementation	Production requirement
User registration	In-memory user dictionary	Persistent identity database
Password handling	SHA-256 hash	Salted adaptive password hashing, e.g., Argon2/bcrypt
Session	UUID token in memory	Server-side session or short-lived JWT with rotation
Authorization	Prototype endpoint logic	Role/resource-based authorization
CORS	allow origins=["*"]	Restricted trusted origins
Auditability	Logging only	Immutable security and audit trail

Data storage	Process-local	Encrypted persistent storage with access controls
--------------	---------------	---

The frontend labels the interface as secure and encrypted. Still, the implementation must be interpreted carefully: the source code itself identifies the authentication database as a mock store, and the API enables permissive CORS with wildcard origins. These implementation details matter because they separate the intended user experience from the actual security posture. The system therefore documents these issues as production requirements rather than presenting the prototype as clinically secure infrastructure.

3.5. Deep Learning Classification Pipeline

Classification starts from an ImageNet-pretrained MobileNetV2 feature extractor. The seven output labels are encoded as integer identifiers and mapped to human-readable names during response construction. The custom head applies adaptive average pooling and flattening, followed by dropout, a 1280-to-256 fully connected layer with ReLU, a second dropout stage, and a 256-to-7 linear output (Table 5).

Table 5: CNN label map and display names

Code	Numeric ID	Display Name
nv	0	Melanocytic Nevi
mel	1	Melanoma
bkl	2	Benign Keratosis
bcc	3	Basal Cell Carcinoma
akiec	4	Actinic Keratosis
vasc	5	Vascular Lesion
df	6	Dermatofibroma

A learnable temperature parameter, initialized at 1.5, divides the logits after clamping them to the interval [0.5, 5.0]. During inference, researchers resize an uploaded RGB image to 224 x 224 pixels, convert it to a tensor, normalize it using precomputed HAM10000 channel statistics, and pass it through the model under torch.no_grad() (Table 6).

Table 6: MobileNetV2 classification head

Stage	Configuration	Role
Feature extractor	ImageNet-pretrained MobileNetV2 features	Learn generic visual representation
Pooling	AdaptiveAvgPool2d (1)	Convert spatial feature map to a fixed vector
Regularization	Dropout 0.3	Reduce overfitting
Projection	Linear (1280, 256) + ReLU	Task-specific feature compression
Regularization	Dropout 0.15	Additional regularization
Classifier	Linear (256, 7)	Seven diagnostic logits
Calibration parameter	Learnable temperature, initial 1.5	Scale logits before softmax

Researchers rank the Softmax outputs to obtain the top five classes. The first element supplies the primary CNN class and confidence, while the top five complete list is preserved for the user interface and downstream reasoning.

3.6. Image Preprocessing and Augmentation

Training transformations expose the network to moderate geometric and photometric variation (Table 7).

Table 7: Image transformation pipeline

Phase	Operations	Purpose
Training	Resize 256; RandomCrop 224	Standardize input while varying crop
Training	Horizontal/vertical flips	Geometric variation
Training	ColorJitter	Photometric variation
Training	Rotation +/-30°	Pose and acquisition variation
Training	RandomGrayscale 0.05	Robustness to reduced color cues
Training	RandomErasing 0.1	Regularization against localized artifacts
Validation/Test	Resize 224 + normalization	Deterministic evaluation

Normalization	Mean [0.7630,0.5456,0.5700]; STD [0.1409,0.1523,0.1692]	Match training distribution
---------------	---	-----------------------------

Images are first resized to 256 x 256, randomly cropped to 224 x 224, horizontally flipped with probability 0.5, vertically flipped with probability 0.3, modified with ColorJitter, rotated within +/-30 degrees, randomly converted to grayscale with probability 0.05, converted to tensors, normalized with HAM10000 statistics, and subjected to RandomErasing with probability 0.1. For validation and testing, images undergo only resize-to-224, tensor conversion, and normalization, thereby preserving deterministic evaluation behavior.

3.7. Clinical Reasoning and Report Generation

After obtaining the highest-ranked CNN class, the backend maps it into a structured disease object. Each supported model class has an associated clinical knowledge entry containing the disease name, scientific name, ICD-10 code, risk category, prevalence note, risk description, ABCD descriptors, differential diagnoses, recommendations, and an urgent flag. The report generator combines optional patient name, age, sex, location, and duration into a formatted narrative. The resulting response includes the image-derived confidence and describes the output as an educational and research aid.

Table 8: Structured clinical output fields

Field	Source	Purpose
lesion.name / scientific	Knowledge base or multimodal response	Human-readable diagnosis label
lesion.icd10	Knowledge base	Reference code
lesion.risk / risk_desc	Knowledge base or multimodal response	Communicate relative concern level
lesion.abcd	Knowledge base or multimodal response	Organize visual descriptors
lesion.differential	Knowledge base or contextual logic	Alternative conditions to consider
lesion.recommendations	Knowledge base or multimodal response	Suggested follow-up information
inference.confidence	CNN or multimodal response	Top-class probability/confidence
inference.top5	CNN	Ranked alternative CNN classes
clinical_report	Report generator / multimodal model	Narrative summary
secondary_conditions	Symptom/location heuristic	Contextual alternatives

When configured, the multimodal route uses the configured Anthropic Claude model with the uploaded image encoded as base64 and the user-provided metadata embedded into a structured prompt. The requested response schema includes a lesion object, inference object, clinical report, and secondary conditions. When the external API is unavailable, the service falls back to the local CNN path. The fallback retains core functionality without depending on a third-party model.

3.8. Data Collection and Preprocessing

The repository provides a reproducible script for data preparation and model training, not a clinical data-collection study. Consequently, the present evaluation relies on public benchmark images and software execution measurements. The HAM10000 metadata Table 8 is read into a Pandas DataFrame; records are optionally deduplicated by lesion_id; a second stratified split into validation and test sets follows a stratified train/temporary split. The random state is fixed at 42 for reproducibility.

3.8.1. Dataset Construction

The training script supports a HAM10000 metadata CSV and searches multiple image directories, including the two HAM10000 image-part folders, an images directory, and the supplied data directory. A row is linked to an image by matching the image_id with JPG, JPEG, or PNG extensions. Labels are mapped using the seven-class dictionary. The script therefore provides a deterministic linkage between tabular labels and the corresponding dermoscopic images.

3.8.2. Train/Validation/Test Split

Following lesion-level deduplication, the script applies an 80% training split and a 20% temporary split. The script then splits that portion equally between validation and test sets, yielding nominal proportions of 80%, 10%, and 10%. Stratification preserves class proportions in each partition.

Since the inspected repository snapshot lacks the complete metadata file or a generated training-history artifact in a machine-readable form, the manuscript does not claim the post-deduplication image count or final accuracy as measured facts (Table 9).

Table 9: Training configuration encoded in train.py

Parameter	Configured value	Purpose
Maximum epochs	30	Training budget
Batch size	32	Mini-batch optimization
Base learning rate	1e-3	Classifier learning rate
Optimizer	AdamW	Adaptive optimization with weight decay
Weight decay	1e-4	Regularization
Scheduler	Cosine Annealing Warm Restarts	Learning-rate schedule
Focal gamma	2.0	Emphasize hard examples
Early stopping patience	8	Stop after validation plateau
Gradient clipping	max_norm 1.0	Stabilize optimization
Split	80 / 10 / 10	Train/validation/test

3.8.3. Clinical Knowledge Integration

The disease knowledge base extends beyond the seven CNN labels. The backend contains mappings for symptoms and anatomical locations and a larger disease catalogue used for secondary-condition suggestions. Symptom keyword matches contribute two points per candidate, while location matches contribute one point (Table 10).

Table 10: Contextual condition-selection logic

Input Signal	Weight	Effect
Symptom keyword match	+2	Strong contextual relevance
Anatomical location match	+1	Secondary contextual relevance
Exclude primary CNN class	Mandatory	Avoid duplicate primary result.
Candidate limit	Top 5	Keep output concise
No contextual signal	Fallback list	Show common alternative conditions

The system selects at most five candidates. If neither symptom nor location information is available, the system falls back to a predefined list of common conditions, excluding the top CNN class.

3.8.4. Evaluation Protocol

The training implementation calculates validation and test accuracy, a scikit-learn classification report, a confusion matrix, and training curves. It saves the best validation checkpoint and writes training history as JSON when training is executed. The backend records elapsed request time for the analysis endpoint.

Consequently, the implementation supports two evaluation levels: model-level classification evaluation and system-level inference evaluation. Only the system-level results can be discussed directly from the current code snapshot; numerical model results should be populated from the generated training artifacts before manuscript submission.

4. Results

This section summarizes results from the current software implementation and distinguishes them from numerical experiments that require running the complete training dataset. The repository provides the mechanisms to calculate test accuracy, per-class precision/recall/F1, a confusion matrix, and training curves. However, it does not currently provide a final training-history file or a verified evaluation Table 11 in the public project snapshot. Consequently, it deliberately does not report unsupported percentage values.

4.1. Classification Evaluation

The training program evaluates the held-out test set using the best validation checkpoint. It computes overall accuracy as the fraction of correctly classified test observations and generates a classification report for the seven display classes. It also saves a confusion matrix as confusion_matrix.png and records training/validation loss and accuracy for each epoch.

Together, these outputs provide the required experimental evaluation interface even though the final artifact values are not present in the repository snapshot inspected for this manuscript.

Table 11: Classification evaluation outputs generated by the training pipeline

Metric / artifact	Computed by code	Status in current repository snapshot
Overall test accuracy	test_acc	Computed at training time; final value not published in source text
Per-class precision	classification_report	Generated at training time; final value not published in source text
Per-class recall	classification_report	Generated at training time; final value not published in source text
Per-class F1	classification_report	Generated at training time; final value not published in source text
Confusion matrix	confusion_matrix()	Generated as an image when training is executed
Training curves	Matplotlib history plot	Generated as an image when training is executed
Training history JSON	json.dump()	Generated after training; not present in the accessible source snapshot

For a submission-ready version, insert the actual values from the generated training_history.json and classification report. The absence of these values in the current repository does not invalidate the method; it means claiming a particular accuracy would require inventing evidence. This distinction matters in medical-image research, where small changes in split strategy or preprocessing can materially change reported performance.

4.2. Architecture and Feature Comparison

From an implementation perspective, DermaLens distinguishes itself from a model-only classifier by placing the CNN inside a complete workflow (Figure 2).

DermaLens feature coverage across the user workflow

Image input	Implemented	●
CNN prediction	Implemented	●
Clinical metadata	Implemented	●
PDF report	Implemented	●
AI skin chat	Implemented	●
Clinic finder	Implemented	●

The figure summarizes implemented user-facing functions; it is not an accuracy or clinical-performance score.

Figure 2: Capability coverage across the complete DermaLens user workflow

The browser interface provides upload and live camera capture; the API validates file type and size; the inference layer produces the top five class distribution; the knowledge layer adds contextual interpretation; and the report interface connects the output to a printable document and clinic discovery. Accordingly, the comparison focuses on supported capabilities rather than a claim of clinical superiority (Table 12).

Table 12: Comprehensive feature comparison: model-only classification versus Derma Lens

Aspect	Model-only classifier	Derma Lens AI
Image input	Single image file	Upload or live camera capture
Model output	Class label/score	Top class + top five probabilities
Clinical context	Not inherent	Age, sex, location, duration, symptoms
Knowledge layer	Optional	Structured disease knowledge base
Differential conditions	Usually absent	Primary class plus contextual alternatives
Risk information	Usually absent	Low/medium/high risk descriptors
Report generation	Usually absent	Printable clinical-style report
Clinic discovery	Absent	Clinic list + map interface
API layer	Optional	FastAPI REST service
Optional multimodal reasoning	Usually absent	Claude vision path when configured

Upload/Camera | CNN Classification | Context Input | Knowledge Base | Differentials | Report | Clinics | AI Chat.

4.3. Per-Class Analysis

The seven-class mapping retains the diagnostic categories defined by HAM10000 while presenting names that are more understandable to end users. The training code also preserves class imbalance information for weighted sampling and focal-loss weighting. The least frequent categories are expected to remain the hardest to evaluate reliably because a small number of errors can substantially shift their precision or recall. For the final publication, the class-level results should therefore include the confusion matrix and the number of test samples per class.

Table 13: Required per-class reporting format for the final experimental run

Class	Support	Precision	Recall	F1-score
Melanocytic Nevi	[insert from test set]	[insert]	[insert]	[insert]
Melanoma	[insert]	[insert]	[insert]	[insert]
Benign Keratosis	[insert]	[insert]	[insert]	[insert]
Basal Cell Carcinoma	[insert]	[insert]	[insert]	[insert]
Actinic Keratosis	[insert]	[insert]	[insert]	[insert]
Vascular Lesion	[insert]	[insert]	[insert]	[insert]
Dermatofibroma	[insert]	[insert]	[insert]	[insert]
Macro average	[insert]	[insert]	[insert]	[insert]
Weighted average	[insert]	[insert]	[insert]	[insert]

Table 13 is deliberately left as an experimental insertion point rather than filled with estimated values. A reproducible medical-AI paper should report exact observed measurements, and the current software already provides the code required to generate them.

4.4. Comparative Evaluation against Existing Methods

The paper fits into three implementation patterns: a model-only research notebook, a cloud-first multimodal service, and an integrated hybrid application. The first pattern is usually easy to reproduce but offers limited support for user workflows. The second can provide richer image-plus-text reasoning but introduces dependence on a third-party service. The third pattern, used by DermaLens, retains a local CNN path while allowing optional external multimodal reasoning and a broader knowledge layer (Table 14).

Table 14: Comparative evaluation of system patterns

Output Parameter	Model-Only Notebook	Cloud Multimodal Application	Derma Lens AI
Local CNN inference	Yes	Not necessarily	Yes
Seven HAM10000 classes	Yes	Depends on model	Yes
Top-five probabilities	Possible	Depends on API	Yes
Symptom/location context	Usually, manual	Often supported	Supported
Structured disease knowledge	Usually absent	Model dependent	Built in
Differential conditions	Limited	Prompt dependent	Built in
Printable report	Usually absent	Application dependent	Yes
Clinic discovery	Absent	Application dependent	Yes
External API dependency	No	Usually yes	Optional
REST API	Not required	Usually yes	Yes

This comparison is descriptive. The architecture covers more workflow functions than a classifier-only implementation, but feature coverage alone does not demonstrate clinical accuracy or safety. Those properties require independent validation with representative clinical data.

4.5. Performance Analysis—Inference and Request Handling

The FastAPI analysis route records total request time using a wall-clock timer around validation, file reading, model selection, inference or multimodal analysis, result construction, and response formatting. The response exposes `elapsed_seconds`, making the service directly measurable during deployment tests. The endpoint supports JPEG, PNG, and WEBP image uploads and rejects unsupported types. The service rejects files larger than 15 MB with an HTTP 413 response (Table 15).

Table 15: Backend request validation and performance fields

Metric	Implementation	Observed/reported value
Supported image types	JPEG, PNG, WEBP	Implemented
Maximum upload size	15 MB	Implemented
Inference image size	224 x 224	Implemented
Elapsed request time	time.time() around endpoint	Returned in API response; numeric benchmark not published
Health endpoint	/health	Reports device, weights-loaded state, Claude configuration and knowledge-base count
Disease endpoint	/api/diseases	Returns CNN classes and knowledge-base entries
Analysis endpoint	/api/analyze	Multipart upload + optional metadata

4.6. User Interface Screenshots

The current frontend provides a dark, dashboard-style interface with a persistent navigation bar and four principal pages: Home, Analysis, Clinics, and AI Chat.

Home Dashboard → Welcome Banner → Start New Scan → Conditions Identified / Clinics / Ratings / Pdf → Feature Cards
Analysis → Drop Image / Camera → Patient Name, Age, Sex, Location, Duration, Symptoms → Analyse & Generate Report
Result → Primary Diagnosis + Confidence → Abcd → Differential Diagnosis → Clinical Recommendations → Pdf / Clinics
Clinics → Search / Min Rating / Sort → Use My Location → Clinic Cards + Map
AI Chat → Suggested Questions → User Message → Local Knowledge-Base Response

Figure 3 placeholders represent the actual functional states defined by the source code and can be replaced directly with screenshots of the running application before manuscript submission.

4.6.1. Home Dashboard and Secure Patient Portal

The Home page acts as the project's unified skin-health portal. It displays summary cards for conditions identified, verified clinics, average clinic rating, and printable reports. Feature cards provide direct navigation to the analysis, clinic, and assistant workflows.

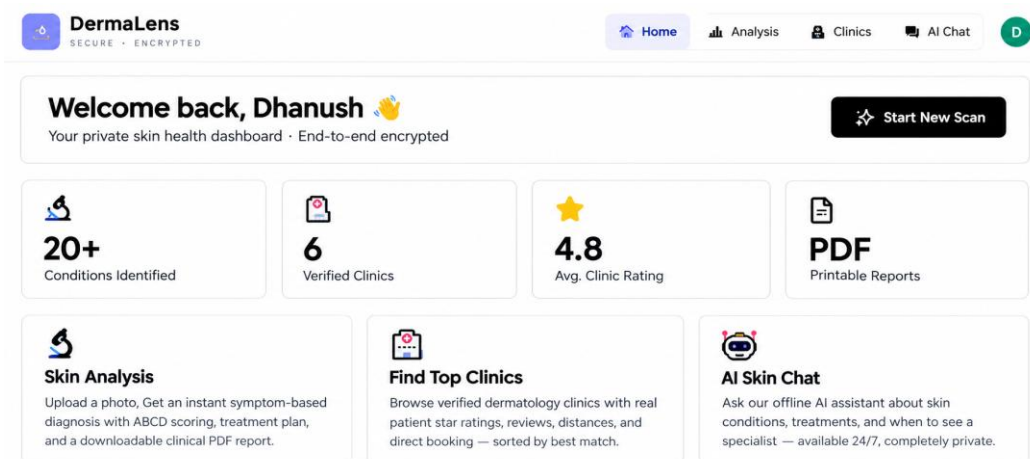


Figure 3: Home dashboard with feature cards for Skin Analysis, Top Clinics, AI Skin Chat, and printable reports

The source describes the portal as end-to-end encrypted. Still, the paper treats that label as a user-interface claim and documents the backend security prototype separately. Figure 3 illustrates the layout.

4.6.2. Skin Analysis and Live Camera Capture

The Analysis page allows image files through a drag-and-drop area and provides a browser camera modal. The interface supports JPG, PNG, and WEBP. Patient context fields include name, age, sex, anatomical location, duration, and symptoms.

The Analyse & Generate Report action remains disabled until an image is supplied, reducing accidental empty submissions. Figure 4 shows the complete analysis screen.

The interface for Figure 4 is divided into two main sections. The left section, titled 'DermaLens SECURE • ENCRYPTED', contains an 'UPLOAD SKIN IMAGE' area with a photo of a skin lesion and a 'Use Live Camera' button. Below this are input fields for 'PATIENT NAME' (ram), 'AGE' (2-), 'SEX' (Male), 'LOCATION ON BODY' (Upper Extremity / Arm), and 'DURATION' (Less than 1 month). The right section displays the analysis results. At the top, it shows a 'LOW RISK' status and the diagnosis 'Melanocytic Nevi' with its ICD-10 code (022.9) and a brief description. Below this, the 'ABCD DERMOSCOPY CRITERIA' are listed in four boxes: A (ASYMMETRY: Usually symmetric), B (BORDER: Well-defined, smooth edges), C (COLOR: Uniform tan/brown), and D (DERMOSCOPY: Reticular or globular pattern). At the bottom, a 'DIFFERENTIAL DIAGNOSIS' list includes '1 Melanoma (rule out)' and '2 Blue Naevus', both marked as 'consider'.

Figure 4: Skin Analysis interface showing uploaded lesion image, patient context, ABCD descriptors, and prediction results

4.6.3. Prediction Result Dashboard

The result view centers on a primary diagnosis with confidence, visual descriptors, a differential-diagnosis list, clinical recommendations, and additional conditions to consider. The frontend explicitly exposes a PDF download action and a clinic-navigation action after the analysis is complete. These result elements are visible in the analysis view shown in Figure 4.

4.6.4. Clinical Report

The printable report follows a distinct print stylesheet with a compact A4 layout. The report contains patient metadata, a diagnosis block, risk styling, ABCD descriptors, recommendations, recommended dermatology clinics, a disclaimer, and a footer. The application therefore separates the interactive dashboard from the formal report artifact.

4.6.5. Find Top Dermatologists

The Clinics page provides search and filtering controls, a minimum rating selector, sorting by rating, review count, or distance, and an option to request the browser location. The source integrates Leaflet for map presentation.

The interface for Figure 5 is titled 'DermaLens SECURE • ENCRYPTED' and 'Find Top Dermatologists'. It features a search bar with the placeholder 'e.g. Acne, Psoriasis, Melanoma...' and a 'Use My Location' button. Below the search bar, there are filters for 'MIN RATING' (All Ratings) and 'SORT BY' (Highest Rated). The main content area displays three clinic listings: 'Advanced Skin & Melanoma Centre' (4.9 rating, 342 reviews, Open), 'ClearSkin Acne & Aesthetics Clinic' (4.8 rating, 891 reviews, Open), and 'DermaCare Psoriasis & Eczema Clinic' (4.7 rating, 568 reviews, Closed). To the right of the listings is an interactive map showing the locations of these clinics in Bangalore, India, with labels for various neighborhoods like Yeswanthpur, Sadashivanagar, and Kaval Byrasandra.

Figure 5: Find Top Dermatologists interface with search, rating filter, sorting, location, and interactive map

The current distance update uses randomized demo values after geolocation, which indicates that the feature is a prototype visualization rather than a verified geospatial service. The interface and map view are shown in Figure 5.

4.6.6. AI Skin Chat

The browser assistant currently relies on a local keyword knowledge base for common skin questions, including melanoma, acne, eczema, psoriasis, dermatologist referral, ringworm, skin cancer, and sun protection.

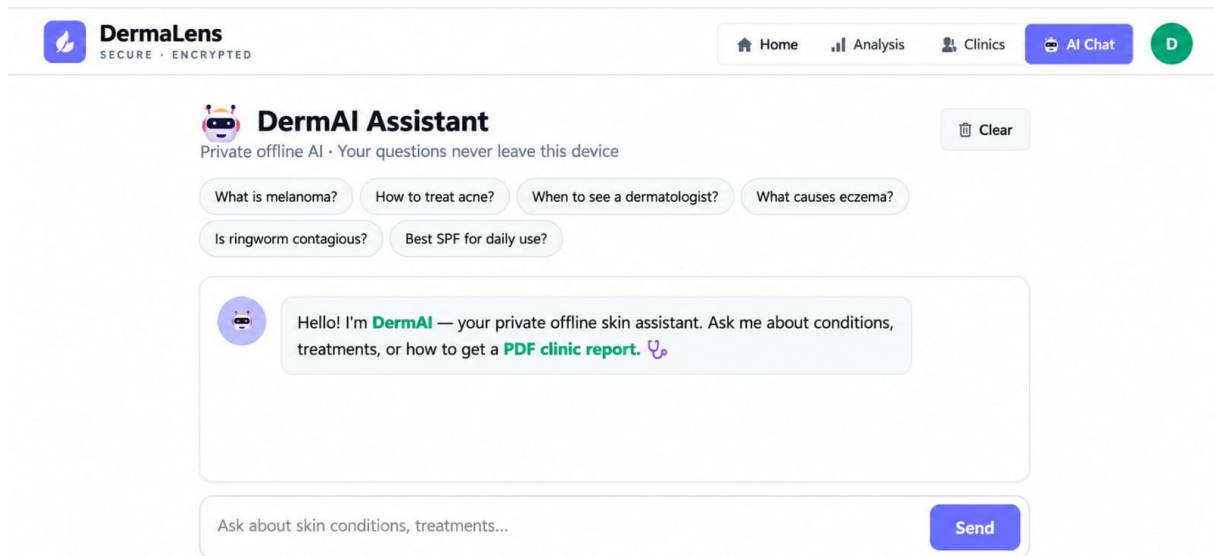


Figure 6: Derm-AI Assistant interface for skin-conditioning questions and privacy-focused educational responses

The backend contains an optional Claude chat endpoint. This separation is important: the visible frontend path in the current source is primarily local and rule-based, while the backend provides the infrastructure for a remote language-model conversation when explicitly integrated. Figure 6 shows the corresponding chat interface.

4.7. Discussion and Findings

4.7.1. Key Findings

A central architectural observation is that a lightweight CNN can integrate into a broader skin-analysis workflow without making the classifier responsible for every clinical function. The local MobileNetV2 path supplies a stable, reproducible seven-class image inference task, while the knowledge layer translates the model output into structured medical descriptors and follow-up information. The REST architecture then exposes the combined result to a browser client in a form suitable for experimentation and further extension. Another observation relates to uncertainty. The software returns probabilities and top-five candidates rather than hiding the ranked alternatives. This is preferable to presenting a single label without context, but a probability emitted by a neural network should not automatically be interpreted as a calibrated clinical likelihood. The learnable temperature parameter suggests an attempt to control logit scale, but calibration must be measured explicitly on held-out data before any clinical interpretation. Another observation is the architecture's flexibility. The service can run with custom local weights when the checkpoint exists, and it can fall back to ImageNet initialization if it can't load custom weights. Likewise, multimodal Claude reasoning is optional, and local CNN inference remains available without an API key. This design supports offline reproducibility at the model level while still allowing richer experimentation when external services are configured.

4.7.2. Security and Privacy Considerations

Because the application processes potentially sensitive patient metadata and uploaded medical images, security is a core requirement, not a cosmetic feature. The current backend saves uploaded files under an upload's directory using a generated short UUID prefix and a sanitized filename. This provides predictable local file handling but does not implement encryption at rest, retention policies, consent management, or secure deletion. The prototype authentication database is process-local, and CORS is currently open to all origins (Table 16).

Table 16: Security posture and production requirements

Security Aspect	Current Status	Production Requirement
Password handling	SHA-256 prototype hash	Argon2/bcrypt with salt and policy
User/session storage	In-memory dictionaries	Persistent identity store
Authorization	Prototype	Server-enforced authorization
CORS	Wildcard	Restricted origin allow-list
Image storage	Local uploads directory	Encrypted storage + retention/deletion controls
Encryption at rest	Not implemented	Required where sensitive data are stored
Audit logging	Application logs	Security audit trail
Patient consent	No formal consent workflow	Documented consent and privacy notice
External vision API	Optional, image sent when configured	Explicit disclosure, data-processing controls, contractual safeguards

These choices are acceptable for a development demonstrator but should not be interpreted as production-ready security.

4.7.3. Future Roadmap

Future work should focus first on the transition from prototype to validated research system. The next model-centric step is to publish the final held-out metrics, class-wise confusion matrix, calibration analysis, and confidence intervals. A second step is external validation on an independent dataset to test domain shift (Table 17).

Table 17: Future roadmap

Planned Feature	Priority	Expected Impact
Publish final test metrics and confusion matrix	Critical	Reproducible performance evidence
External-dataset validation	Critical	Assess generalization
Probability calibration analysis	High	Improve confidence interpretation
Grad-CAM / visual explanation	High	Inspect model focus regions
Persistent, encrypted database	High	Secure patient and report storage
Role-based authorization with secure sessions	High	Production access control
Model versioning and experiment tracking	High	Reproducibility and auditability
Multilingual assistant	Medium	Broader user accessibility
Real clinic/booking integration	Medium	Operational follow-up

A third is explainability, for example with Grad-CAM, to inspect whether predictions rely on lesion regions or on confounding image artifacts. The software layer should then add persistent, encrypted storage, formal authentication, audit logging, structured consent, and reproducible deployment configuration.

4.7.4. Disease Knowledge Coverage and Differential Analysis

The knowledge layer covers more than the seven CNN classes and includes common inflammatory, infectious, allergic, pigmentary, and proliferative conditions (Table 18).

Table 18: Knowledge-layer content categories

Content Type	Examples/Function
Primary CNN classes	nv, mel, bkl, bcc, akiec, vasc, df
Common conditions	Acne, eczema, psoriasis, rosacea, hives, warts
Infections	Ringworm/tinea, scabies, impetigo, shingles
Inflammatory conditions	Contact dermatitis, seborrheic dermatitis, folliculitis
Pigmentary conditions	Vitiligo, hyperpigmentation
Structured medical metadata	ICD-10, risk, prevalence, ABCD descriptors
User guidance	Recommendations, urgency flag, dermatologist prompts

This allows the interface to communicate broader possibilities even when the learned image classifier is restricted to seven benchmark labels. Secondary conditions are selected using symptom and location mappings, and the fallback list ensures that the interface remains populated when context fields are empty.

4.7.5. Browser Compatibility Analysis

The frontend uses standard HTML5, CSS, browser file input, camera APIs, geolocation, and Leaflet. Its compatibility consequently depends mainly on secure browser permissions for camera and location and on JavaScript support (Table 19).

Table 19: Browser feature compatibility to validate during deployment

Feature	Browser Dependency	Validation Target
Image upload	HTML file input	JPEG/PNG/WEBP across desktop and mobile
Camera capture	MediaDevices / browser permission	Secure context and permission flow
Geolocation	navigator.geolocation	Permission granted/denied states
Maps	Leaflet + browser canvas/SVG	Map rendering and resizing
PDF report	Browser print engine	A4 pagination and hidden UI elements
Responsive layout	CSS media queries	Mobile and desktop breakpoints

The application is designed as a responsive web interface rather than a native mobile application. Before production deployment, browser-specific testing should cover camera capture, file handling, map rendering, print-to-PDF behavior, and location permission denial.

4.7.6. API and Data Flow - Full Schema

The central /api/analyze endpoint receives a multipart file and optional form fields for patient name, age, sex, location, duration, and symptoms. It validates the media type, enforces the 15 MB limit, creates an identifier, optionally saves the file, chooses the multimodal or local inference path, and returns a JSON object (Table 20).

Table 20: /API/analyze request and response schema

Component	Fields	Description
Request file	file	JPEG, PNG or WEBP image
Patient context	patient_name, age, sex	Optional demographic context
Lesion context	location, duration, symptoms	Optional clinical context
Response ID	id	Short unique analysis identifier
Timing	elapsed_seconds	Endpoint elapsed time
Inference	top_class, confidence, top5	Primary and ranked predictions
Lesion	name, scientific, icd10, risk, category	Structured diagnosis information
Report	clinical_report	HTML-formatted narrative report
Secondary	secondary_conditions	Contextual alternatives
Meta	timestamp, metadata	Request context and creation time
Safety	disclaimer	Research/educational use statement

The response schema is intentionally flat at the top level so the frontend can bind lesion, inference, report, secondary conditions, metadata, timestamps, and disclaimer independently.

4.7.7. Clinical Safety and Triage Guidance

Risk labels and urgent flags are included in its knowledge base, including urgent referral language for conditions represented as high risk. Treat these indicators as communication aids, not validated triage rules (Table 21).

Table 21: Clinical-safety interpretation of current system outputs

Output	Intended Role	Not Justified Without Validation
CNN confidence	Model score/ranking	Direct clinical probability
Risk category	Knowledge-based communication aid	Patient-specific prognosis
Differential list	Possible alternatives	Definitive differential diagnosis
Recommendations	Educational next-step guidance	Individualized treatment prescription
Urgent flag	Prompt specialist review	Emergency triage protocol
Clinic recommendations	Navigation aid	Quality or clinical-outcome guarantee

A safe deployment would require clinical governance, threshold validation, prospective review, and a defined escalation process (Table 22).

Table 22: Reproducibility checklist for final journal submission

Item	Current State	Action Before Publication
Source code	Public repository	Archive release version / commit hash
Dataset	HAM10000 specified	Record exact downloaded version and license compliance
Data split	Implemented with seed 42	Publish final post-deduplication counts
Training config	Encoded in train.py	Attach executed training log
Test metrics	Code available	Insert exact accuracy/precision/recall / F1
Confusion matrix	Generated by code	Include final Figure 6
Training curves	Generated by code	Include final Figure 6
Model checkpoint	Expected in models/	Archive checksum/version
External validation	Not implemented	Evaluate on independent dataset
Clinical validation	Not performed	Obtain ethics approval and clinical evaluation if applicable

A high-confidence model output cannot rule in or rule out a diagnosis without clinical examination and, where appropriate, histopathological confirmation.

5. Conclusion

Derma Lens AI demonstrates a complete architecture for web-based skin lesion analysis that combines a lightweight convolutional classifier with a structured clinical knowledge layer and a practical browser interface. The core model is MobileNetV2 fine-tuned on seven HAM10000 categories, and the training pipeline includes lesion-level deduplication, stratified partitioning, augmentation, weighted sampling, focal loss, AdamW, cosine-annealing warm restarts, gradient clipping, and early stopping. The FastAPI backend standardizes image requests, performs 224 x 224 inference, returns top-five probabilities, and generates a structured, clinical-style report that the frontend can consume. The paper also shows how an image-classification model can expand into an integrated user workflow. The Analysis page allows file or camera input and contextual symptoms; the result view exposes the primary prediction, confidence, ABCD descriptors, differential conditions, and recommendations; the report layer generates a printable document; the Clinics page provides a map-based discovery interface; and the DermAI assistant supplies educational information. Optional Anthropic integration provides a second multimodal reasoning path while preserving a local CNN fallback. The main methodological conclusion should not rest on a specific accuracy number, because the accessible repository snapshot does not include a verified final training-history artifact. Rather, the paper establishes a reproducible evaluation pipeline and an application architecture that can produce the required quantitative evidence when the training run is executed. This paper deliberately avoids inserting an unsupported accuracy claim and identifies the exact artifacts that should be attached to the final submission. Subsequent work should include independent external validation, probability calibration, explainability, secure persistence, production-grade identity management, and prospective clinical evaluation. With these additions, the architecture could serve as a stronger experimental foundation for research into efficient and human-centered dermatological AI.

5.1. Limitations

Several limitations remain relevant. First, the seven-class model is bounded by the diagnostic categories represented in HAM10000 and should not be interpreted as a general skin-disease classifier. Second, this paper does not report a numerical held-out accuracy because the repository snapshot inspected for this study did not include the generated training-history artifact. Third, the knowledge-layer recommendations and risk descriptions are rule-based or language-model generated and have not been clinically validated as patient-specific treatment advice. Fourth, the authentication and storage layers are prototype implementations, and the clinic-distance behavior is explicitly demo-oriented. Fifth, the optional multimodal Claude path introduces an external data-processing dependency whenever an API key is configured. The authors should resolve these limitations or clearly disclose them before clinical deployment.

Acknowledgment: The authors acknowledge the academic environment and technical resources that supported the development and documentation of the DermaLens AI software system.

Data Availability Statement: The source code and software documentation are publicly available in the DermaLens-Skin-AI repository. The paper uses the HAM10000 benchmark dataset, which is distributed by its original data providers under the applicable terms and conditions.

Funding Statement: No external financial assistance or funding is stated in the present manuscript.

Conflicts of Interest Statement: The authors declare no conflicts of interest associated with this work. All implementation-specific claims are based on the public project source inspected for this manuscript, and unsupported numerical results have not been inferred.

Ethics and Consent Statement: The present work is a software and benchmark-dataset study and does not report prospective human-subject experimentation. The application is explicitly intended for educational and research use and not as a substitute for professional medical diagnosis.

References

1. P. Tschandl, C. Rosendahl, and H. Kittler, "The HAM10000 dataset, a large collection of multi-source dermatoscopic images of common pigmented skin lesions," *Scientific Data*, vol. 5, no. 8, pp. 1–9, 2018.
2. 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.
3. H. A. Haenssle, C. Fink, R. Schneiderbauer, F. Toberer, T. Buhl, A. Blum, A. Kalloo, A. B. H. Hassen, L. Thomas, A. Enk, and L. Uhlmann, "Man against machine: Diagnostic performance of a deep learning convolutional neural network for dermoscopic melanoma recognition in comparison to 58 dermatologists," *Annals of Oncology*, vol. 29, no. 8, pp. 1836–1842, 2018.
4. Anthropic, "Start building with Claude," *Anthropic*, 2026. [Accessed by 12/04/2026].
5. N. C. F. Codella, D. Gutman, M. E. Celebi, B. Helba, M. A. Marchetti, S. W. Dusza, A. Kalloo, K. Liopyris, N. Mishra, H. Kittler, and A. Halpern, "Skin lesion analysis toward melanoma detection: A challenge at the 2017 International Symposium on Biomedical Imaging (ISBI), hosted by the International Skin Imaging Collaboration (ISIC)," in *Proc. IEEE 15th Int. Symp. Biomed. Imaging (ISBI)*, Washington, United States of America, 2018.
6. T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár, "Focal loss for dense object detection," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Venice, Italy, 2017.
7. M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted residuals and linear bottlenecks," in *Proc. IEEE/CVF Conf. Computer Vision and Pattern Recognition (CVPR)*, Salt Lake City, Utah, United States of America, 2018.
8. R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual explanations from deep networks via gradient-based localization," in *Proc. IEEE Int. Conf. Computer Vision (ICCV)*, Venice, Italy, 2017.
9. A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "MobileNets: Efficient convolutional neural networks for mobile vision applications," *arXiv:1704.04861*, 2017. [Accessed by 17/06/2025].
10. T. J. Brinker, A. Hekler, A. H. Enk, C. Berking, S. Haferkamp, A. Hauschild, M. Weichenthal, J. Klode, D. Schadendorf, T. Holland-Letz, C. von Kalle, S. Frohling, B. Schilling, and J. S. Utikal, "Deep neural networks are superior to dermatologists in melanoma image classification," *European Journal of Cancer*, vol. 119, no. 9, pp. 11–17, 2019.
11. PyTorch Team, "PyTorch documentation," *PyTorch*, 2026. [Accessed by 19/04/2026].
12. FastAPI Team, "FastAPI documentation," *FastAPI*, 2026. [Accessed by 20/04/2026].
13. A. T. Ibrahim, M. Abdullahi, A. F. D. Kana, M. T. Mohammed, and I. H. Hassan, "Categorical classification of skin cancer using a weighted ensemble of transfer learning with test time augmentation," *Data Science and Management*, vol. 8, no. 2, pp. 174–184, 2025.
14. M. M. Musthafa, T. R. Mahesh, V. V. Kumar, and S. Guluwadi, "Enhanced skin cancer diagnosis using optimized CNN architecture and checkpoints for automated dermatological lesion classification," *BMC Medical Imaging*, vol. 24, no. 8, pp. 1–20, 2024.
15. M. F. J. Acosta, L. Y. C. Tovar, M. B. Garcia-Zapirain, and W. S. Percybrooks, "Melanoma diagnosis using deep learning techniques on dermatoscopic images," *BMC Medical Imaging*, vol. 21, no. 1, pp. 1–11, 2021.

Publisher's Note: The publisher remains impartial concerning jurisdictional claims in published maps and institutional affiliations. Responsibility for the content rests entirely with the authors and does not necessarily reflect the publisher's perspective