Skin Cancer Detection — Research Report

Title

Automated Skin Lesion Classification (HAM10000): A CNN & Transfer Learning Approach

Authors

Your Name

Abstract

This report documents a complete workflow for building a skin lesion classifier using the HAM10000 dataset. The work covers data collection, preprocessing, model design (from-scratch CNN and transfer learning with MobileNetV2), training, evaluation (accuracy, precision, recall, ROC AUC), interpretability (Grad-CAM), deployment steps, and ethical considerations. The pipeline is implemented and tested in Google Colab with GPU acceleration.

Link to project brief / reference

The project follows the steps described in the AI Major Projects brief included with this conversation: AI Major Projects PDF

1. Introduction

Skin cancer is one of the most common cancers worldwide. Early detection improves outcomes. This project demonstrates how convolutional neural networks (CNNs) can assist dermatologists by flagging suspicious lesions for further clinical review. The scope of this report is to provide a reproducible pipeline from raw data to an evaluated model and guidance for deployment and ethics.

2. Dataset

HAM10000 (Skin Cancer MNIST) — a curated collection of dermatoscopic images (≈10,015 images) covering seven diagnostic categories. For this project we convert the 7-class dataset into a binary classification task (malignant vs benign) where: `akiec`, `bcc`, and `mel` are treated as malignant; the remaining classes are benign.

Files used: - `HAM10000_metadata.csv` - image files (`ISIC_*.jpg`) extracted into a folder for processing

3. Preprocessing

Goals: normalize images, handle class imbalance, split data, and produce a directory structure compatible with Keras `ImageDataGenerator`.

Steps performed: - Read metadata CSV and inspected columns: `lesion_id`, `image_id`, `dx`, `dx_type`, `age`, `sex`, `localization`. - Created binary label column: `label = 1` if `dx` in ["akiec","bcc","mel"] else `0`. - Performed stratified split into Train (70%), Validation (15%), Test (15%) to keep class proportions consistent. - Organized processed images into `processed_data/{train,val,test}/{benign,malignant}` for generator-based loading.

Notes on best practice: Use patient- or lesion-aware splitting (group by `lesion_id`) for final evaluation to avoid leakage if multiple images from the same lesion exist.

4. Model Design

Two model tracks are presented:

4.1 From-scratch CNN (educational)

A compact CNN architecture used to teach concepts: - Conv2D(32) → MaxPool - Conv2D(64) → MaxPool - Conv2D(128) → MaxPool - Flatten → Dense(128) → Dropout(0.5) → Dense(1, sigmoid)

This model helps explain convolutions, pooling, dense layers, dropout, and binary cross-entropy.

4.2 Transfer Learning (production-ready)

MobileNetV2 (pretrained on ImageNet) is used as a feature extractor: - Input resizing to 224×224 - `mobilenet_v2.preprocess_input()` applied - Base model with `include_top=False`, `weights='imagenet'` and frozen layers initially - GlobalAveragePooling2D → Dropout → Dense(1, sigmoid) - Optionally unfreeze top layers of the base model and fine-tune with a lower learning rate.

Why transfer learning? Better generalization on small/medium medical datasets and faster convergence.

5. Training

Training strategy: - Use `ImageDataGenerator` for real-time rescaling and augmentation for the training set (rotation, shifts, flips, zoom). - Use class weights to handle imbalance or experiment with oversampling / focal loss. - Use callbacks: `ModelCheckpoint` (save best), `EarlyStopping` (restore best), `ReduceLROnPlateau`.

Example hyperparameters: - Batch size: 16–32 - Initial LR: 1e-4 for classifier head, 1e-5 for fine-tuning - Epochs: 10–30 (use early stopping)

6. Evaluation

Metrics used: accuracy, precision, recall (sensitivity), F1-score, ROC AUC, confusion matrix. For clinical tasks, sensitivity for malignant detection is emphasized.

Sample evaluation steps (to run in Colab): - `y_prob = model.predict(test_generator)` - `y_pred = (y_prob > 0.5).astype(int)` - `classification_report(y_true, y_pred)` - `roc_auc_score(y_true, y_prob)` - Plot ROC curve and confusion matrix (matplotlib)

Recommendation: When reporting results, include patient-aware split performance and if possible an external hold-out dataset.

7. Explainability (Grad-CAM)

Grad-CAM highlights image regions that influence predictions and helps build clinician trust. The report includes instructions to generate Grad-CAM overlays using either `tf-keras-vis` or `tf-explain` on individual test images.

8. Saving & Exporting the Model

Save the trained Keras model:

```
model.save("skin_cancer_model.h5")
```

Then download from Colab using `google.colab.files.download("skin_cancer_model.h5")` or convert to TFLite for mobile deployment.

9. Testing with New Images

Steps to test on user-provided images: - Upload image(s) to Colab (or to a web app) - Preprocess (resize to 224×224, rescale to [0,1] or use `preprocess_input`) - `pred = model.predict(img_batch)` and interpret probability > 0.5 as malignant

Provide tips on threshold tuning depending on precision/recall tradeoffs.

10. Ethics and Limitations

- This model is **not** a medical device and must not replace clinical judgement.
- Dataset bias, imaging-device variation, and population differences can reduce generalizability.
- Include a clear disclaimer when demonstrating the model and prefer clinician-in-the-loop deployment.

11. Future Work

- Use lesion segmentation (ISIC masks) to crop lesions first.
- Ensemble multiple architectures (EfficientNet, ResNet, MobileNet) for robustness.
- Evaluate on external datasets and on different skin types.
- Prototype a clinician-facing UI (Streamlit / Flask) with Grad-CAM overlays and upload/download options.

12. Appendices

A — Key scripts (filenames)

- `scripts/check_dataset.py` — verify files & metadata
- `scripts/preprocess_data.py` — create binary labels and create train/val/test folders
- `scripts/build_model.py` — CNN from-scratch architecture
- `scripts/transfer_train.py` — MobileNetV2 transfer learning training script
- `scripts/evaluate.py` — evaluation & ROC/Confusion matrix code
- `scripts/gradcam.py` — produce Grad-CAM overlays

B — Reproducible Colab steps

Include the Colab cells for installing dependencies, uploading `kaggle.json`, downloading the dataset, extracting, preprocessing, training (both CNN and transfer learning), evaluating, Grad-CAM, saving, and downloading models.

C — How to export to PDF

1. Open this report in the Canvas (the document created alongside this conversation).
2. Use the three-dot menu to export or copy the content into Google Docs / Word and save as PDF.

If you want, I can now: - Export this report into a ready-to-download PDF and provide a link, or - Generate a PowerPoint presentation summarizing the report (slides ready to download), or - Insert actual model training results and Grad-CAM images into the report if you upload the model outputs and figures.

Which would you like next? (I can create the PDF or make the PowerPoint.)