Skin Cancer Classification App Documentation

Habiba Tarek Nassar

September 17, 2025

Contents

1 Introduction

1.1 Overview

The Skin Cancer Classification App is a user-friendly web application designed to assist in the preliminary classification of skin lesions as benign or malignant. It leverages a pre-trained deep learning model (TensorFlow/Keras) to analyze uploaded skin images and provide predictions. The app is built using Streamlit for the frontend and backend, making it accessible via a web browser. The project is intended for medical and research purposes, offering a simple interface for users to upload images and view predictions.

1.2 Goals and Objectives

- **Goal 1**: Provide an accessible tool for preliminary skin cancer screening.
- **Goal 2**: Ensure accurate image preprocessing to match the model's input requirements.
- **Goal 3**: Deliver a professional, user-friendly interface suitable for medical applications.
- **Goal 4**: Support extensibility for future enhancements (e.g., multi-class classification, additional model integration).

1.3 Scope

- **In Scope**: Image upload, preprocessing (resizing, RGB conversion, normalization), model prediction, and result display.
- **Out of Scope**: Training the deep learning model, real-time image capture, and integration with medical record systems (planned for future iterations).

2 Features

Feature	Description	Status
Image Upload	Supports JPG, JPEG, and PNG formats.	Implemented
Automatic Preprocessing	Converts images to RGB, resizes to model input shape (224x224), and normalizes pixel values.	Implemented
Model Prediction	Uses a pre-trained Keras model to classify skin lesions as benign or malignant.	Implemented
User Interface	Streamlit-based interface with a light blue theme, suitable for medical contexts.	Implemented
Result Display	Shows prediction probabilities and class (if applicable).	Implemented
Caching	Uses Streamlit's caching to load the model efficiently.	Implemented

Table 1: Key Features of the Skin Cancer Classification App

3 Requirements

3.1 System Requirements

- **Operating System**: Windows 10+, macOS 10.15+, Linux (e.g., Ubuntu 20.04+)
- **Hardware**: Minimum 4GB RAM, 2GHz CPU (GPU recommended for faster model inference)
- **Storage**: Approximately 500MB for dependencies and model file
- **Internet**: Required for initial dependency installation

3.2 Software Dependencies

The project relies on the following Python packages:

- `tensorflow==2.20.0`: For loading and running the Keras model
- `streamlit==1.49.1`: For the web application framework
- `pillow==11.3.0`: For image processing
- `numpy==2.3.3`: For numerical operations
- Additional dependencies: See `requirements.txt` for the full list.

3.3 Model Requirements

- The pre-trained model (`myModel.keras`) expects input images of shape `(224, 224, 3)` (RGB format).
- The model file must be present in the project root directory.

4 Installation

4.1 Prerequisites

- **Python**: Version 3.8 or higher (`python -version`)
- **Git**: For cloning the repository (`git -version`)
- **Virtual Environment**: Recommended for dependency isolation

4.2 Step-by-Step Guide

1. **Clone the Repository**:

```
git clone https://github.com/habeba-tarek/skin-cancer-classifier
   .git
cd skin-cancer-classifier
```

2. **Create a Virtual Environment** (optional but recommended):

```
python -m venv venv
source venv/bin/activate  # Linux/Mac
venv\Scripts\activate     # Windows
```

3. **Install Dependencies**: Ensure `requirements.txt` is in the project root, then run:

```
pip install -r requirements.txt
```

4. **Verify Model File**: Ensure `myModel.keras` is in the project root.
5. **Run the Application**:

```
streamlit run app.py
```

This will start a local server, typically at `http://localhost:8501`.

4.3 Docker Installation (Optional)

1. Create a `Dockerfile`:

```
FROM python:3.8-slim
WORKDIR /app
COPY . .
RUN pip install -r requirements.txt
EXPOSE 8501
CMD ["streamlit", "run", "app.py", "--server.port=8501", "--
   server.address=0.0.0.0"]
```

2. Build and run the Docker image:

```
docker build -t skin-cancer-classifier .
docker run -p 8501:8501 skin-cancer-classifier
```

5 Usage

5.1 Running the Application

1. Start the Streamlit app:

```
streamlit run app.py
```

2. Open a web browser and navigate to `http://localhost:8501`.
3. Upload a skin image (JPG, JPEG, or PNG).
4. View the prediction results displayed below the uploaded image.

5.2 Example Workflow

1. **Upload Image**: Select an image file using the file uploader.
2. **View Image**: The uploaded image is displayed for confirmation.

3. **Prediction**: The app processes the image and shows:
    - For binary classification: Probability of the positive class (e.g., malignant).
    - For multi-class classification: Predicted class index and probabilities.

5.3 Example Output

For a binary classification model:

```
Probability of positive class: 0.8732
```

For a multi-class model:

```
Class: 1 | Probabilities: [[0.12, 0.78, 0.10]]
```

6 Configuration

6.1 Environment Variables

- `MODEL`$_{PATH}$: $Pathto the Keras model file(default :$ `myModel.keras`$)$`PORT` : $Streamlit server port(d$ `8501`$)$ Example `.env` file:

```
MODEL_PATH=./myModel.keras
PORT=8501
```

6.2 Streamlit Configuration

Customize settings in `.streamlit/config.toml`:

```
[server]
port = 8501
headless = false

[theme]
primaryColor = "#4682b4"
backgroundColor = "#f0f8ff"
```

6.3 Model Configuration

- The model expects images of shape `(224, 224, 3)`.

- To use a different model, update the `load`$_{m}y_{m}odel() function in$`app.py`.

7 API Reference

7.1 Functions

- **`load`**$_{m}y_{m}odel()$:
    - **Description: Loads the pre-trained Keras model using Streamlit's caching.**

- **Returns: TensorFlow/Keras model object.**

- **Caching: Uses `@st.cache`$_{r}esource$.**

  Image Preprocessing:

  - Convert image to RGB: `Image.open().convert("RGB")`

- Resize to model input shape: `img.resize(input`$_{s}hape)Converttoarrayandnormalize$: `np.expand`$_{d}ims(img_{a}rray, axis = 0)/255.0$

- **Prediction**:

  - **Input**: Preprocessed image array.
  - **Output**: Model prediction (array of probabilities).

7.2 Streamlit Components

- `st.file`$_{u}ploader$: $Acceptsimageuploads(JPG, JPEG, PNG)$.`st.image` : $Displaystheupload$
- `st.write`: Shows prediction results.

8 Architecture

8.1 File Structure

```
skin-cancer-classifier/
 app.py                # Main Streamlit application
 myModel.keras         # Pre-trained Keras model
 .gitignore            # Git ignore file
 requirements.txt      # Python dependencies
 README.md             # Project description
```

8.2 Components

- **Frontend**: Streamlit handles the UI, including image upload and result display.
- **Backend**: TensorFlow/Keras for model inference, Pillow/Numpy for image preprocessing.
- **Model**: A pre-trained convolutional neural network (CNN) saved as `myModel.keras`.

- **Caching**: Streamlit's `@st.cache`$_{r}esourceensuresef ficientmodelloading$.

8.3 Data Flow

1. User uploads an image via the Streamlit interface.
2. The image is preprocessed (converted to RGB, resized, normalized).
3. The pre-trained model predicts the class/probability.
4. Results are displayed in the UI.

9 Testing

9.1 Unit Tests

To add tests, create a `tests/` directory and use `pytest`:

```
import pytest
from PIL import Image
import numpy as np

def test_image_preprocessing():
    img = Image.new("RGB", (300, 300))
    img_array = image.img_to_array(img.resize((224, 224)))
    img_array = np.expand_dims(img_array, axis=0) / 255.0
    assert img_array.shape == (1, 224, 224, 3)
    assert img_array.max() <= 1.0
```

9.2 Manual Testing

- Upload various image formats (JPG, PNG) to ensure compatibility.
- Test with images of different sizes to verify resizing.
- Check prediction output for binary and multi-class models.

10 Deployment

10.1 Local Deployment

```
streamlit run app.py
```

10.2 Cloud Deployment

- **Streamlit Community Cloud**:
  - Push the repository to GitHub.
  - Connect to Streamlit Community Cloud and select the repository.
  - Specify `app.py` as the entry point.
- **Heroku**:
  - Create a `Procfile`:

    ```
    web: streamlit run app.py --server.port=$PORT --server.
       address=0.0.0.0
    ```

  - Deploy using Heroku CLI:

    ```
    heroku create
    git push heroku main
    ```

10.3 Docker Deployment

See the Docker setup in the Installation section.

11 Troubleshooting

11.1 Common Issues

- **Error: "No module named 'tensorflow'"**:
  - Ensure `tensorflow` is installed (`pip install tensorflow==2.20.0`).
  - Verify the virtual environment is activated.
- **Error: "Model file not found"**:
  - Ensure `myModel.keras` is in the project root.
- **Image Upload Fails**:
  - Check file format (only JPG, JPEG, PNG supported).
  - Ensure the file is not corrupted.

11.2 Debugging

- Set `logLevel` in Streamlit config to `debug`.
- Check TensorFlow logs for model inference issues.

12 Contributing

1. Fork the repository.
2. Create a feature branch: `git checkout -b feature-name`.
3. Commit changes: `git commit -m "Add feature"`.
4. Push to the branch: `git push origin feature-name`.
5. Open a pull request on GitHub.

13 Changelog

13.1 v1.0.0 (2025-09-17)

- Initial release with image upload, preprocessing, and prediction.
- Streamlit UI with light blue theme.
- Support for binary and multi-class model outputs.

14 FAQ

- **Q: What image formats are supported?**
  A: JPG, JPEG, and PNG.
- **Q: Can I use a different model?**
  A: Yes, replace `myModel.keras` with your model and update `app.py` if needed.

15 Acknowledgments

- **TensorFlow/Keras**: For the deep learning framework.
- **Streamlit**: For the web application framework.
- **Pillow/Numpy**: For image processing and numerical operations.

16 Contact

For questions or support, contact:

- Habiba Tarek Nassar (habibatarek898@gmail.com)

- GitHub Issues: `https://github.com/habeba-tarek/skin-cancer-classifier/issues`

17 License