Summer Internship cum Training Program - 2025

PROJECT REPORT

On

Skin Cancer Classification

Submitted By:

Kratik Mudgal (CSINTERN/25/039)

Under the Supervision:

Dr. Panthadeep Bhattacharjee

Department of Computer Science and Engineering

NATIONAL INSTITUTE OF TECHNOLOGY

ROURKELA

July, 2025

DECLARATION

I, Kratik Mudgal, Internship ID: CSINTERN/25/039, hereby declare that the report of the project entitled “Skin Cancer Classification” which is being submitted to the Department of Computer Science and Engineering, in partial fulfillment of the requirements for the Summer Internship cum Training Program - 2025 (CSInternship-25) on “Deep Learning for Healthcare and Cryptography”, is a bonafide report of the work carried out by me. The materials contained in this report have not been submitted to any University or Institution for the award of any degree. Any contribution made to this work by others, with whom I have worked at NIT Rourkela or elsewhere, is explicitly acknowledged in the dissertation. Works of other authors cited in this dissertation have been duly acknowledged under the sections “Reference” or “Bibliography”.

Name:	______________________
Internship ID:	______________________

DATE: July, 2025

ACKNOWLEDGEMENT

This project is prepared in partial fulfillment of the requirements for the Summer Internship cum Training Program - 2025 (CSInternship-25) on “Deep Learning for Healthcare and Cryptography” in Department of Computer Science and Engineering. I owe my deepest gratitude to the Department of Computer Science and Engineering, NIT Rourkela, for providing me with an opportunity to work on the project as a part of the internship program. I would also like to offer my gratitude to my supervisor, (Dr. Panthadeep Bhattacharjee), for his/her guidance. The experience of working on this project will surely enrich my technical knowledge and also gives experience of working on a project.

Name:	______________________
Internship ID:	______________________

Department of Computer Science and Engineering
National Institute of Technology Rourkela, Odisha

SUPERVISOR's CERTIFICATE

Name: Kratik Mudgal
Internship ID : CSINTERN/25/039
Title of Dissertation: Skin Cancer Classification

The undersigned certify that they have read, and recommended for acceptance the project report entitled **'Skin Cancer Classification'** submitted by **Kratik Mudgal** in partial fulfillment of the requirements for the Summer Internship cum Training Program - 2025 (CSInternship-25) on "Deep Learning for Healthcare and Cryptography" in Department of Computer Science and Engineering.

Supervisor: Dr. Panthadeep Bhattacharjee
Department of Computer Science and Engineering
National Institute of Technology Rourkela

DATE: 10/07/2025

ABSTRACT

Skin cancer is one of the most prevalent forms of cancer globally, with early detection being crucial for effective treatment and improved patient outcomes. This project focuses on developing a deep learning-based system for the classification of various skin cancer types from dermatoscopic images. Leveraging the Skin Cancer ISIC The International Skin Imaging Collaboration dataset from Kaggle, a Convolutional Neural Network (CNN) model was designed and trained to distinguish between nine different classes of skin lesions. The project involved comprehensive data preprocessing, including image resizing and normalization, and employed data augmentation techniques to address dataset imbalances and enhance model generalization. A pre-trained DenseNet201 model was utilized as a base, followed by custom layers for classification. The model achieved a notable accuracy of 98.41% on the training set and 97.05% on the testing set. The developed system is deployed as a web application using Streamlit, allowing users to upload skin lesion images for instant classification, providing a preliminary assessment. This prototype aims to assist in the early screening of skin cancer, serving as a supportive tool for healthcare professionals.

Keywords: Skin Cancer, Deep Learning, Convolutional Neural Networks (CNN), DenseNet201, Image Classification, Dermatoscopy, Streamlit, Machine Learning, Early Detection.

TABLE OF CONTENTS

List of Figures

List of Tables

1. LIST OF ABBREVIATIONS

AI	Artificial Intelligence
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
ISIC	The International Skin Imaging Collaboration
PIL	Pillow (Python Imaging Library)
ReLU	Rectified Linear Unit
SGD	Stochastic Gradient Descent
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative

2. INTRODUCTION

2.1. Background

Skin cancer is the most common cancer worldwide, with increasing incidence rates. Early and accurate diagnosis is critical for effective treatment and improved patient prognosis. Traditional diagnosis relies heavily on visual inspection by dermatologists, which can be subjective and prone to variations depending on the expert's experience. Deep learning, particularly Convolutional Neural Networks (CNNs), has shown remarkable success in medical image analysis, offering a promising avenue for automated and objective skin cancer detection. This project aims to harness these capabilities to develop a supportive tool for dermatological diagnosis.

2.2. Motivation

The primary motivation behind this project is to leverage the power of deep learning to develop an accessible and efficient tool for preliminary skin cancer classification. By automating part of the diagnostic process, this system can potentially aid healthcare professionals in triaging cases, reducing misdiagnoses, and enabling earlier intervention, especially in regions with limited access to dermatological expertise. The goal is not to replace human experts but to provide a valuable assistive technology.

2.3. Objectives

The main objectives of this project are:

1. To implement and train a deep learning model capable of accurately classifying nine distinct skin cancer types from dermatoscopic images.
2. To develop a user-friendly web application for real-time interaction with the trained classification model.

2.4. Problem Statement

The challenge lies in developing a robust and accurate deep learning model that can reliably distinguish between nine different types of skin lesions, some of which may exhibit subtle visual differences, given the inherent variability in dermatoscopic images. Furthermore, ensuring the model's generalization capability on unseen data and making it accessible through an intuitive interface are key aspects to address.

2.5. Scope of Project

This project focuses on the classification of nine specific skin lesion types as provided in the chosen dataset. The scope includes:

- Data collection and preprocessing.
- Implementation and training of a deep learning model using TensorFlow/Keras with a pre-trained DenseNet201 architecture.
- Evaluation of model performance.
- Development of a Streamlit web application for image upload and real-time inference.

The project provides a preliminary classification and is explicitly stated as not providing medical advice, emphasizing the need for professional medical consultation.

3. LITERATURE REVIEW

The field of medical image analysis has witnessed significant advancements with the advent of deep learning. Specifically, Convolutional Neural Networks (CNNs) have emerged as the state-of-the-art for image classification tasks due to their ability to automatically learn hierarchical features from raw pixel data. Numerous studies have explored CNNs for skin cancer detection.

Early works often utilized simpler CNN architectures or transfer learning from models pre-trained on large natural image datasets like ImageNet. Recent research has shown the effectiveness of deeper architectures and more sophisticated transfer learning approaches. Models like VGG, ResNet, Inception, and DenseNet have been successfully adapted for dermatoscopic image analysis. For instance, DenseNet models are known for their ”dense connectivity” pattern, which helps alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters. This makes them particularly suitable for tasks requiring rich feature extraction from complex medical images.

Data imbalance is a common issue in medical datasets, where certain disease classes are significantly rarer than others. Techniques such as data augmentation (rotation, shifting, flipping, zooming) and resampling have been widely adopted to mitigate this problem and improve model generalization. The use of robust preprocessing methods, including normalization based on training data statistics, is also crucial for consistent model performance.

Deployment of such models into user-friendly interfaces, often through web applications, is a growing trend to make AI-powered tools accessible to healthcare practitioners. Frameworks like Streamlit offer rapid prototyping and deployment capabilities for machine learning models.

4. METHODOLOGY

The methodology adopted for this project involved several key stages, from data acquisition to model deployment. The total lines of code written for this project are 839.

4.1. Data Collection and Preparation

4.1.1. Dataset Description

The dataset used for this project is ”Skin cancer ISIC The International Skin Imaging Collaboration” [1] available on Kaggle. This dataset is organized into ’Train’ and ’Test’ directories, each containing subdirectories for different skin lesion classes. The dataset contains 9 distinct classes of skin lesions.

4.1.2. Data Loading and Initial Exploration

The first step involved loading image paths and their corresponding labels from the ’Train’ and ’Test’ directories into Pandas DataFrames.

```
train_dir = '/kaggle/input/skin-cancer9-classesisic/Skin␣cancer␣ISIC␣The␣
    International␣Skin␣Imaging␣Collaboration/Train'
test_dir = '/kaggle/input/skin-cancer9-classesisic/Skin␣cancer␣ISIC␣The␣
    International␣Skin␣Imaging␣Collaboration/Test'
```

A combined DataFrame ‘df‘ was then created. The initial distribution of labels was visualized using a pie chart to understand class representation. The ‘label$_{m}$*ap‘dictionarywascreatedtomapnum*

4.1.3. Image Preprocessing and Resizing

To ensure uniformity and compatibility with the chosen model architecture, all images were resized to a dimension of (100, 75) pixels (width, height). This resizing was performed using ‘PIL.Image.resize‘ with ‘Image.LANCZOS‘ for high-quality downsampling. To expedite this process, ‘concurrent.futures.ThreadPoolExecutor‘ was utilized for parallel processing of image resizing.

4.1.4. Data Augmentation

To address class imbalance and prevent overfitting, data augmentation techniques were extensively applied. An ‘ImageDataGenerator‘ from Keras was configured with various transformations:

- ‘rotation$_{r}ange = 20$“$width_{s}hift_{r}ange = 0.2$‘
- ‘height$_{s}hift_{r}ange = 0.2$“$shear_{r}ange = 0.2$‘
- ‘zoom$_{r}ange = 0.2$“$horizontal_{f}lip = True$‘
- ‘fill$_{m}ode =' nearest'$‘ The goal was to bring each class up to a ‘max$_{i}mages_{p}er_{c}lass$‘$of 2500.Thisensureda$

4.1.5. Data Normalization

After resizing and augmentation, the image pixel values were normalized. This crucial step involves scaling pixel intensities to a standard range, which helps in faster convergence and better performance of deep learning models. The normalization was performed using the mean and standard deviation calculated from the training data:

```
x_train_mean = np.mean(x_train)
x_train_std = np.std(x_train)
x_train = (x_train - x_train_mean)/x_train_std
x_test = (x_test - x_test_mean)/x_test_std
```

This ensures that the model sees data within a consistent scale during training.

4.1.6. Label Encoding

The categorical labels were converted into a one-hot encoded format using ‘keras.utils.np$_{u}tils.to_{c}ategoric$
$classclassificationproblemsindeeplearning, asitallowsthemodeltointerpretclassprobabilitiesdirectly.$

4.1.7. Train-Validation-Test Split

The dataset was split into training, validation, and testing sets to evaluate the model's generalization capabilities. Initially, the combined ‘df‘ was split into features (image array) and target (label). Then, the data was split into ‘x$_{t}rain$‘, ‘$x_{t}est$‘, ‘$y_{t}rain$‘, ‘$y_{t}est$‘$witha$‘$test_{s}ize =$

0.20‘$.Finally,thetrainingsetwasfurtherdividedintotrainingandvalidationsets: ‘x_{t}rain‘, ‘x_{v}alidate‘, ‘y_{t}$
0.2‘$.Thishierarchicalsplitensuresthatthemodel'sperformanceonunseendata(testset)isareliableindicat$
$worldeffectiveness.$

4.2. Model Architecture

The chosen model architecture is based on transfer learning using the DenseNet201 pre-trained on ImageNet. This approach leverages the powerful feature extraction capabilities learned by DenseNet201 from a vast dataset. The architecture consists of:

  - A DenseNet201 base model, with ‘include$_{t}op = False‘toremoveitsoriginalclassificationhead, an$
    $imagenet'‘tousethepre-trainedweights.The‘input_{s}hape‘wassetto(75,100,3)tomatchthepreproc$

- A ‘Dropout(0.5)‘ layer for regularization, to prevent overfitting.
- A ‘Dense(512, activation=’relu’)‘ layer for further feature learning.
- A final ‘Dense(num$_{c}lasses, activation =' softmax')‘layerformulti-classclassification, outputtingpro$
  The model summary is as follows:

```
Model: "sequential"
_________________________________________________________________
 Layer (type) Output Shape Param #
=================================================================
 densenet201 (Functional) (None, 2, 3, 1920) 18,321,984

 flatten (Flatten) (None, 11520) 0

 dropout (Dropout) (None, 11520) 0

 dense (Dense) (None, 512) 5,898,752

 dense_1 (Dense) (None, 9) 4,617

=================================================================
Total params: 24,225,353
Trainable params: 24,142,329
Non-trainable params: 83,024
_________________________________________________________________
```

4.3. Model Training

The model was compiled using the Stochastic Gradient Descent (SGD) optimizer with a
‘learning$_{r}ate = 0.001$‘*and*‘*momentum* $= 0.9$‘.*Thelossfunctionwassetto*‘*categorical*$_{c}$*rossentropy*‘, *whic*
classclassificationwithone − *hotencodedlabels*, *andaccuracywasusedastheprimarymetric*.*A*‘*ReduceL*
0.5‘*ifnoimprovementisobservedfor*‘*patience* $= 3$‘*epochs*, *witha*‘*min*$_{l}$*r* $= 0.00001$‘.*Thishelpsinfine* −
tuningthemodelandachievingbetterconvergence.*Themodelwastrainedfor*30*epochswitha*‘*batch*$_{s}$*ize* =
32‘.*Thetrainingprocessinvolvedfittingthemodeltothe*‘x_{t}*rain*‘*and*‘y_{t}*rain*‘*data*, *withvalidationperform*

4.4. Model Evaluation

After training, the model’s performance was evaluated on both the training and testing datasets. Standard classification metrics were used:

- **Accuracy:** The proportion of correctly classified instances.
- **Loss:** The value of the loss function, indicating the error of the model.
- **Precision:** The ratio of correctly predicted positive observations to the total predicted positives.
- **Recall (Sensitivity):** The ratio of correctly predicted positive observations to all observations in actual class.
- **F1-score:** The weighted average of Precision and Recall.
- **Cohen’s Kappa Score:** A robust statistic used to assess inter-rater reliability, but also useful for classifier agreement beyond chance.
- **Confusion Matrix:** A table that summarizes the performance of a classification algorithm.

5. RESULTS AND DISCUSSION

5.1. Training and Validation Performance

The training history was plotted to visualize the model's learning progress in terms of accuracy. The "Training vs Validation Accuracy" plot indicates how well the model learned from the training data and generalized to unseen validation data over epochs. A converging and closely aligned trend between training and validation accuracy suggests good model stability and generalization.

Figure 5.1: Training vs Validation Accuracy Plot

5.2. Model Evaluation Metrics

The final evaluation on the training and testing sets yielded the following results:

- Train Accuracy: 0.9841 (98.41%)
- Train Loss: 0.0460
- Testing Accuracy: 0.9705 (97.05%)
- Testing Loss: 0.0881

Further evaluation metrics on the test set were calculated: These metrics indicate a strong

Table 5.1: Model Evaluation Metrics

Metric	Value
Accuracy	0.9705
Precision (macro avg)	0.9708
Recall (macro avg)	0.9705
F1-score (macro avg)	0.9706
Kappa score	0.9672

performance of the model, with high accuracy and well-balanced precision and recall across classes, suggesting its effectiveness in classifying skin lesions.

5.3. Confusion Matrix Analysis

A confusion matrix was generated to provide a detailed breakdown of the model's predictions versus the actual labels on the test set. The heatmap visualization of the confusion matrix allows for easy identification of which classes are being correctly classified and where the model might be making errors (e.g., confusing one type of lesion for another). High values along the diagonal indicate correct predictions for each class.

Figure 5.2: Confusion Matrix

6. WEB APPLICATION DEVELOPMENT

6.1. Streamlit Framework

The web application was developed using Streamlit, a Python library that enables rapid creation of interactive data applications. Streamlit was chosen for its simplicity, ease of use, and quick deployment capabilities, allowing for a seamless integration of the trained deep learning model into a user-friendly interface.

6.2. Application Features

The Streamlit web application (`Webapp.py`) [2] provides the following key features:

- **User-Friendly Interface:** A clean and intuitive layout for easy navigation.
- **Image Upload:** Users can upload up to 5 skin lesion images (JPG, JPEG, PNG formats). A ”Clear Uploads” button is provided to reset the uploader.
- **Model Loading:** The pre-trained skin cancer classification model (`skin cancer using resnet50.h5`) is loaded efficiently using `st.cache_resource` to prevent reloading on every rerun, optimizing performance.
- **Image Preprocessing:** Uploaded images are automatically preprocessed (resized to (100, 75) and normalized using the training mean and standard deviation) to match the model's input requirements.
- **Real-time Classification:** Upon clicking the ”Classify Images” button, the model predicts the class of each uploaded image, displaying the predicted lesion type and confidence score.
- **Visual Feedback:** Uploaded images are displayed with captions, and classification results are presented clearly for each image.
- **Disclaimer:** A prominent disclaimer is included, emphasizing that the application is for demonstration purposes only and does not provide medical advice, urging users to consult healthcare professionals.
- **Banner Image:** An external image is fetched and processed to serve as a visual banner for the application, enhancing its aesthetic appeal.

7. CONCLUSION

This internship project successfully developed a deep learning-based system for the classification of skin cancer types from dermatoscopic images. By leveraging a pre-trained DenseNet201 model and robust data preprocessing techniques, including extensive data augmentation and normalization, the model achieved high accuracy on both training (98.41%) and testing (97.05%) datasets. The deployment of the model via a Streamlit web application makes the prototype easily accessible and demonstrates the practical applicability of AI in medical image analysis. This tool serves as a valuable preliminary screening aid, potentially assisting in the early detection of skin cancer.

8. Future Enhancements

Several avenues for future work could further enhance this project:

- **Integration of External Data:** Incorporating more diverse and larger datasets from various clinical settings could improve the model's robustness and generalization.
- **Advanced Architectures:** Experimenting with other state-of-the-art CNN architectures or ensemble methods could potentially yield higher accuracy.
- **Interpretability (XAI):** Implementing explainable AI (XAI) techniques (e.g., Grad-CAM) to visualize salient regions of the image that the model focuses on, providing insights into its decision-making process.
- **Severity Assessment:** Extending the model to not only classify but also assess the severity or stage of the cancer.
- **Doctor-Patient Integration:** Developing features for dermatologists to upload cases, receive AI predictions, and then provide their confirmed diagnosis, potentially building a feedback loop for continuous model improvement.
- **Edge Deployment:** Exploring deployment on edge devices for faster, on-site analysis in clinical settings.

A. Code Snippets

(Key Functions)

- `preprocess_image_for_model` function from `Webapp.py`
- Model Architecture Definition from `skin-cancer-classification-kratik.py`
- Data Augmentation Configuration from `skin-cancer-classification-kratik.py`

B. Project Environment and Dependencies

- Python Version: Python 3.11
- Key Libraries and their versions:
  - `TensorFlow==2.12.0`
  - `Streamlit`
  - `NumPy`
  - `Pandas`
  - `Pillow` (Python Imaging Library)
  - `scikit-learn`
  - `Matplotlib`
  - `Seaborn`

The dataset used for this project was sourced from Kaggle's International Skin Imaging Collaboration [1]. The core model architecture is defined in the 'skin-cancer-classification-kratik.py' file [3], while the user interface functionalities are handled by 'Webapp.py' [2]. Further details and the complete source code are available in the project repository [4].

References

[1] Kaggle, “Skin cancer ISIC The International Skin Imaging Collaboration,” https://www.kaggle.com/datasets/nodoubttome/skin-cancer9-classesisic/suggestions, 2024, accessed: [Current Date, e.g., July 8, 2025].

[2] Kratik Mudgal, “Webapp.py (local project file),” 2025, source code available in project repository.

[3] Kratik, “skin-cancer-classification-kratik.py (local project file),” 2025, source code available in project repository.

[4] moonboyknm, “Skin-Cancer-Detection,” https://github.com/moonboyknm/Skin-Cancer-Detection, 2025, accessed: July 8, 2025.