Skin Cancer Classification Report Using Deep Learning

# Table of Contents

1. Executive Summary

2. Introduction

3. System Architecture

4. Data Description

5. Methodology

6. Implementation

7. Results

8. Challenges and Solutions

9. Future Work

10. References

# 1. Executive Summary

Objective:

The project aims to develop a deep learning solution to classify skin cancer types using medical images of skin lesions.

Outcome:

Achieved an accuracy of 86.25 % in classifying different types of skin cancer, including melanoma and non-melanoma skin cancers.

Importance:

Early diagnosis of cancer types helps improve access to appropriate treatment, reducing the risks of life-threatening outcomes and permanent physical effects.

# 2. Introduction

Background:

Skin cancer is one of the most common types of cancer, with melanoma being one of the most dangerous forms. Doctors rely on medical images of skin lesions for diagnosis.

Objective:

Automating the skin cancer classification process using deep learning techniques for accurate and fast diagnosis.

Scope:

Focus on skin images.

The system covers only certain types of cancer, such as melanoma and non-melanoma skin cancer.

# 3. System Architecture

Workflow Diagram:

Data Collection → Data Preprocessing → Feature Extraction → Model Training → Prediction

Components:

Input: Skin images.

Model: Convolutional Neural Network (CNN) architecture.

Output: Disease classification with confidence scores.

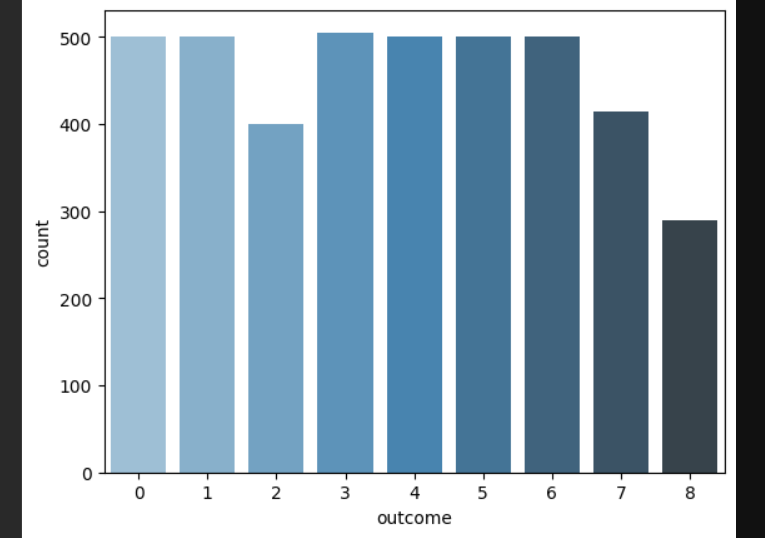
# 4. Data Description

Dataset:

Source: A dataset of skin images containing various types of skin cancer.

Size: 4,100 images across multiple classes.

Features: Images are classified by cancer type and severity.



Preprocessing:

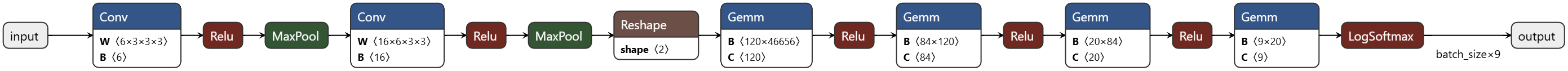
Steps: Resizing, random transformations such as rotation and flipping, normalization.

Tools: PyTorch, OpenCV, Matplotlib.

# 5. Methodology

Model Selection:

The model used in this project is a Convolutional Neural Network (CNN), a well-established architecture in deep learning that performs excellently in image classification tasks.



Training Process:

Framework: We used PyTorch as the framework to train the model.

Parameters:

Learning Rate: 0.001

Batch Size:32

Epochs: 80

Evaluation: We used several metrics to evaluate the model such as:

Accuracy: 86.25 %

F1-Score: 0.8605

Confusion Matrix: To be included in evaluation results.

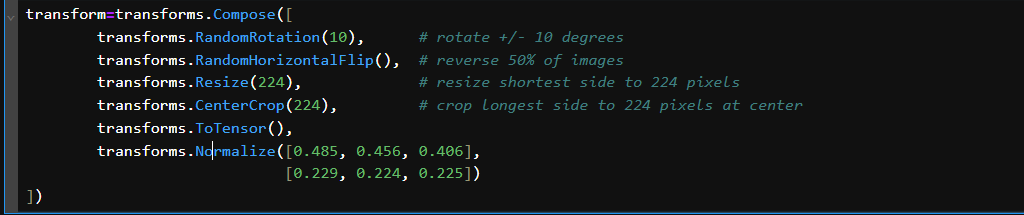
# 6. Implementation

Code Steps:

1. Data Loading:

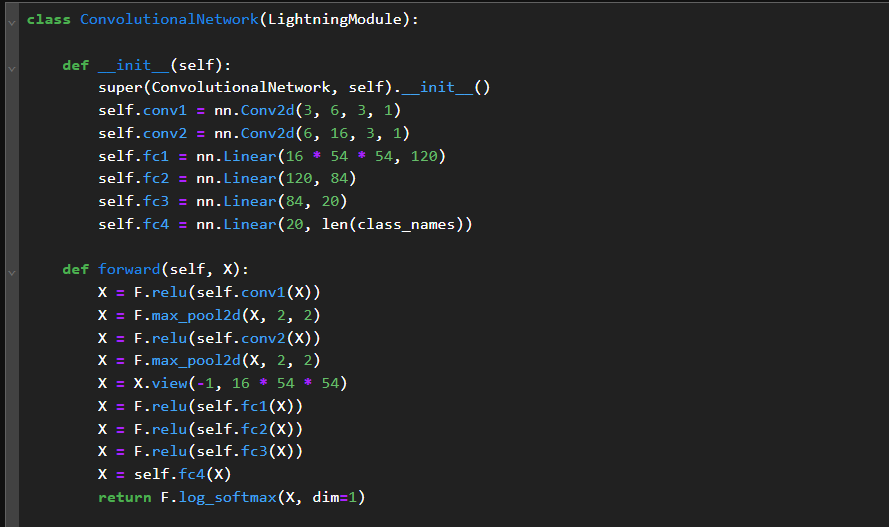
Initially, we load the dataset using DataLoader from PyTorch, where data is loaded in batches with parameters such as batch size.

In the code, data is loaded with transformations like rotation and normalization using transforms.Compose():



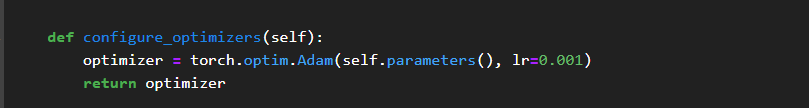
2. Model Definition:

The Convolutional Neural Network is defined using Conv2d and Linear layers, with two convolutional layers and dense layers for classification.



3. Training Process:

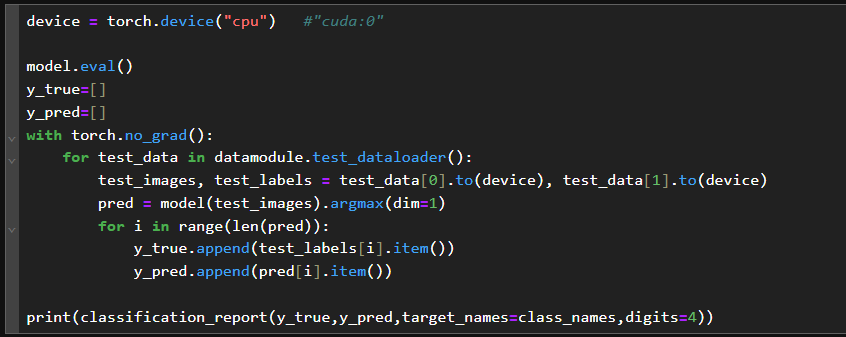
The model is trained using the Trainer from PyTorch Lightning. The cross\_entropy function is used to compute the loss and the optimizer is Adam.



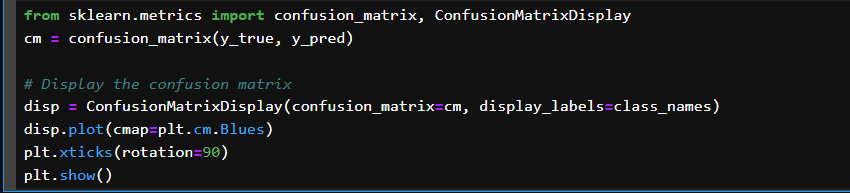
4. Evaluation:

After training, the model is evaluated using test data through classification\_report to show performance across different classes, alongside a confusion matrix (confusion\_matrix).

classification\_report:



confusion\_matrix :



# 7. Results

Performance Metrics:

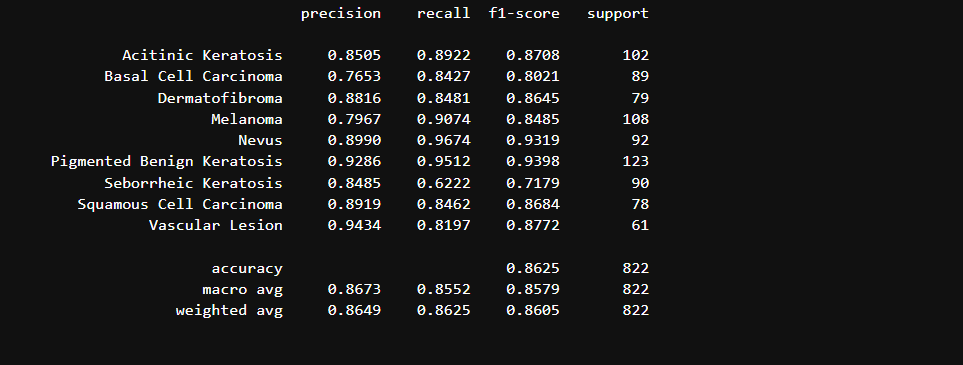
We used classification\_report and confusion\_matrix to measure the model's accuracy and effectiveness across different categories.

Example:

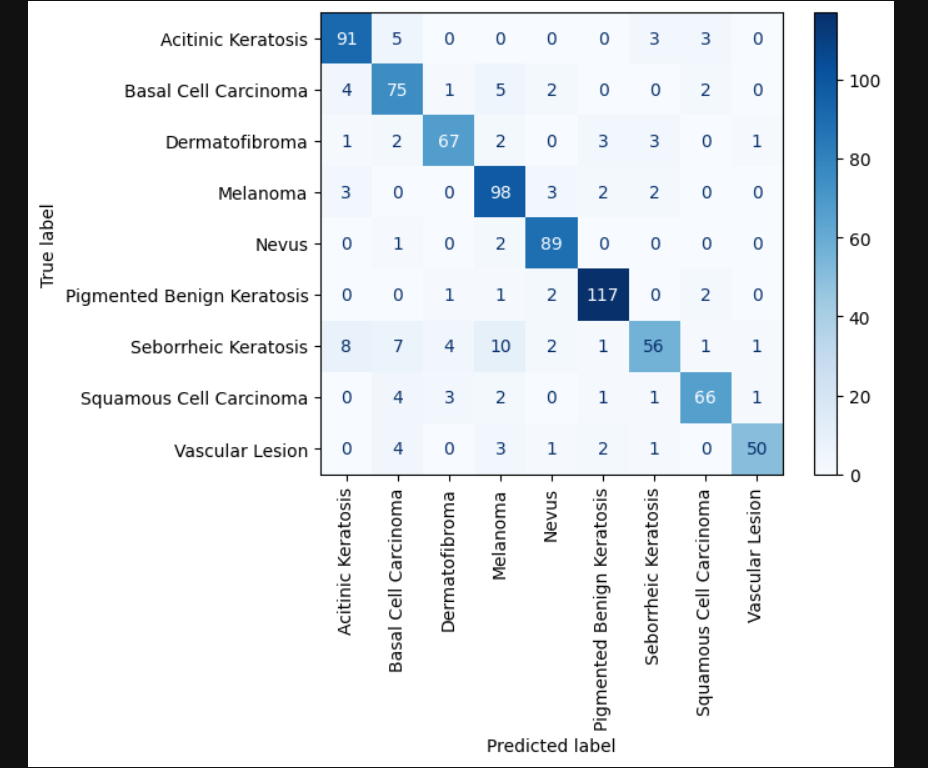
Input Image: [Image appears here]

Classification: Melanoma (Confidence: 86.25 %)

classification\_report:



confusion\_matrix:



# 8. Challenges and Solutions

Challenge:

Data Imbalance: The imbalance in classes may cause issues in training the model on some classes.

Solution: Use WeightedRandomSampler to balance the dataset and train the model more fairly.

Challenge:

Memory Consumption: Storing extracted features and processing data might require large amounts of memory.

Solution: Use memory optimization techniques such as PCA to extract important features and reduce data size.

Challenge:

Model Tuning: Optimizing hyperparameters such as learning rate and batch size.

Solution: Experiment with different values to optimize performance.

# 9. Future Work

Expand detection to include other types of cancers.

Develop a mobile application for direct skin cancer detection.

Explore Transformer-based models to improve accuracy.

# 10. References

1. Skin Cancer Dataset.

2. PyTorch Documentation.

3. Kaggle.