A blue and white logo

Description automatically generated**INTERNATIONAL UNIVERSITY - VIETNAM NATIONAL UNIVERSITY**

**SCHOOL OF COMPUTER SCIENCE AND ENGINEERING**

**Bald Classification Using CNN**

By

Nguyen Vo Nhat Anh

**Lecturer:** Mai Hoang Bao An

Ho Chi Minh City, Vietnam

Year 2023

*Abstract: Bald can significantly impact an individual's confidence and self-esteem, often with delayed awareness and exploration of treatment options. This project explores the use of Convolutional Neural Networks for predicting baldness in facial images. A big amount of data, encompassing many photos of bald and non-bald people, has been retrieved from Kaggle.*

*Keyword: CNN, bald detection*

**Github Link:** [**GitHub**](https://github.com/nhatanh1112/BaldClassificationProject)

**Streamlit Link:** [**Streamlit**](https://baldclassification.streamlit.app/)

# Introduction

Hair loss is a pervasive concern affecting individuals across diverse demographics and geographic regions. Beyond its aesthetic implications, hair loss can be indicative of underlying health conditions, making its early detection crucial for timely intervention and improved outcomes. The advent of artificial intelligence (AI) has opened unprecedented avenues for innovative solutions in healthcare, and our research focuses on harnessing the power of deep learning to revolutionize the detection of hair loss.

The project aims to develop an AI model tailored for the detection(prediction) of hair loss. The model will be trained on a large dataset comprising images of bald and non-bald people. By analyzing intricate patterns and features within the images, the AI system will able to predict the baldness percentage of a person.

# Related Concept

Convolutional Neural Network  
A Convolutional Neural Network (CNN) is a specialized type of artificial neural network designed for processing and analyzing visual data, such as images and videos. CNNs have proven to be highly effective in tasks like image recognition, object detection, and facial recognition, among others. What sets CNNs apart from traditional neural networks is their unique architecture, which includes convolutional layers that automatically and adaptively learn hierarchical representations of the input data.

A diagram of a network

Description automatically generated

Figure 1 CNN Architecture

CNNs use convolutional operations to scan the input data with filters or kernels, extracting relevant features and capturing spatial hierarchies. This allows CNNs to effectively recognize patterns and complex structures within the visual information, making them particularly well-suited for tasks involving images and spatial relationships. Additionally, CNNs often include pooling layers to down sample the spatial dimensions of the data, reducing computational complexity while retaining important features. The combination of convolutional and pooling layers, along with fully connected layers, forms the basic architecture of a CNN.

A diagram of a graph

Description automatically generated

Figure 2 CNN Operation

CNNs have become a cornerstone in computer vision applications, demonstrating state-of-the-art performance in various domains, including image classification, segmentation, and even tasks beyond visual data processing. Their ability to automatically learn and extract relevant features from raw data makes CNNs a powerful tool for a wide range of applications in artificial intelligence and machine learning.

# Bald Classification System

## Dataset (Retrieved from Kaggle)

**Folder Structure:**

* Bald Folder: This folder contains images where individuals with all kinds of baldness from having little hair to completely bald.
* Not Bald Folder: This folder contains images where individuals having all kinds of hairstyles.

**Image Data:**

* Format: The images in both folders are all the same in the same format: PNG
* Resolution: All images have the same resolution
* Name: All images are organized by numbers from 000000 – 999999

The total amount of images in Kaggle is about 200K images. But I don’t have the resources to train all of it. Plus, the trained model is going to be too big to upload to GitHub. So, I sorted out around 5000 images of Bald people and 10000 images of non-bald people to work on this project.

## Classification model

A deep learning classification model is a sort of artificial intelligence (AI) model that is meant to classify incoming data into specified classes or categories. The basic objective is to learn how to map input properties to a certain output class, allowing the model to make predictions or choices on previously unknown data. Here are some of the key components of PyTorch:

1. **Model Definition:** In this section, a built-in model is instantiated using the **torchvision** library. The model is designed for image classification with two output classes (Bald or Not Bald).

A computer screen with white text

Description automatically generated

Figure 3 Built-in Model

1. **Data Loading and Transformation:** Data transformations are applied to augment the dataset and enhance model generalization. These transformations include random cropping, horizontal and vertical flips, converting images to tensors, and normalizing pixel values.

A screen shot of a computer program

Description automatically generated

Figure 4 Data Transforming

1. **Training Loop:** The training loop iterates over the dataset for a specified number of epochs, updating the model parameters to minimize the defined loss function. For each epoch, the training and testing accuracies, as well as losses, are calculated and logged. This loop is crucial for the iterative learning process, allowing the model to adapt to the training data and generalize well to unseen examples.

A computer code on a black background

Description automatically generated

Figure 5 Training

1. **Model Saving:** After training, the model's state dictionary is saved to a file (classification{model\_name}.pth). This step is essential for preserving the learned weights and biases, enabling future use, deployment, or further fine-tuning without retraining the model from scratch.

A black screen with white text

Description automatically generated

Figure 6 Model Saving

## Face detection for the input

Since most of our images in the dataset are the face of a person with hair. So, in order to make sure the input in the application is correct to the train model. We make a face detection class with a little bit of margin to the top, left and right margin to make sure that we get the face and the hair of the input image.

A computer screen shot of a program code

Description automatically generated

Figure 7 Face Detection Using facenet

The method iterates over the detected face bounding boxes, clips them to fit within the image boundaries, adds margins to the face region, and crops the face image accordingly.

# Instruction for Usage

You can use the product [here](https://baldclassification.streamlit.app/)

A screenshot of a computer

Description automatically generated

Figure 8 Streamlit UI

You can upload image(s) using the Browse files button and change the model using the drop-down box below.

A screenshot of a computer

Description automatically generated

Figure 9 Demo Usage

# Result

**RegNet:** train\_loss 0.1707, train\_acc 0.928, test\_loss 0.1910, test\_acc 0.9173

**MobileNetV3\_small:** train\_loss 0.1831, train\_acc 0.9237, test\_loss 0.1921, test\_acc 0.9182

**ShuffeNetv2x1.0:** train\_loss 0.1842, train\_acc 0.9209, test\_loss 0.198, test\_acc 0.9173

A screenshot of a computer

Description automatically generated

Figure 10 Model Result

We create the classification model using three models: MobileNetV3, ShuffleNetV2 and RegNet. The results show that all three models have nearly the same results, however RegNet is somewhat better than the other two models. This shows that RegNet's architecture is suitable for our dataset.