**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Approaching To a Solution for Hair Diseases Classification Base on Deep Learning

by

Nguyen Thi Minh Hien

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

© Copyright by Nguyen Thi Minh Hien 2024

**MINISTRY OF EDUCATION AND TRAINING**

**FPT UNIVERSITY**

Approaching To a Solution for Hair Diseases Classification Base on Deep Learning

by

Nguyen Thi Minh Hien

A thesis submitted in conformity with the requirements  
for the degree of Master of Software Engineering

Supervisor:

Dr. Mai Hoang Bao An

Dr. Dang Ngoc Minh Duc

© Copyright by Nguyen Thi Minh Hien 2024

Approaching To a Solution for Hair Diseases Classification Base on Deep Learning

Nguyen Thi Minh Hien

Degree Master of Software Engineering

FPT University

2024

Abstract

Scalp and hair diseases are common health issues, affecting human health, spirit, and psychology. If scalp and hair diseases are diagnosed accurately and promptly, it will contribute to effective treatment in shortening time. Fortunately, deep learning algorithms have shown enormous potential for medicine, including the diagnosis of scalp and hair diseases. This research was conducted on a novel dataset of 10 hair diseases. Different CNN architectures, which were VGG-16, VGG-19, Inception-v3, Resnet-50, and Resnet-152, were used for transfer learning on the hair disease dataset. VGG-16, VGG-19, and Inception-v3 had high testing accuracy, thus, they were further deployed to the application. The checkpoint was used to save the model when the model was considered the best and the latest best points to help high accuracy for further predictions. Finally, a web application was developed that allows users to input images, and then the application gets predictions from the models and shows the predictions results to the user interface.

Acknowledgments

I would like to express my deepest gratitude to my supervisors, Dr. Mai Hoang Bao An and Dr. Dang Ngoc Minh Duc, who have the qualities of a teacher full of knowledge and enthusiastically guided me to complete my thesis.

I would like to express my deep gratitude to all the teachers who taught me at FSB in the master’s program in software engineering. They have provided me with a lot of interesting and useful knowledge, not only in software engineering but also new knowledge in software project management and a global vision for developing the future of software development. Especially, this course has helped me realize the enormous potential of AI in its application and development trends in various fields of life.

I would like to thank all FSB staff including the admissions department, education department, and accounting…for their enthusiastic attention and support throughout this course.

Finally, I thank the organizers of the FSB course program for designing a very practical course and keeping up with the trend of updating AI knowledge in software engineering.

Table of Contents

Contents

[Acknowledgments 3](#_Toc160098559)

[Table of Contents 4](#_Toc160098560)

[List of Tables 6](#_Toc160098561)

[List of Figures 7](#_Toc160098562)

[List of Appendices 9](#_Toc160098563)

[**Chapter 1 Introduction** 1](#_Toc160098564)

[1 Introduction 1](#_Toc160098565)

[1.1 Introduction 1](#_Toc160098566)

[1.2 Related Work 4](#_Toc160098567)

[1.3 Contributions 8](#_Toc160098568)

[**Chapter 2 Fundamental Theory** 10](#_Toc160098569)

[2 Fundamental Theory 10](#_Toc160098570)

[2.1 Image Classification Using CNNs 10](#_Toc160098571)

[2.2 CNN Architectures 11](#_Toc160098572)

[2.3 Metrics 14](#_Toc160098573)

[2.3.1 Confusion Matrix 14](#_Toc160098574)

[2.3.2 Classification Metrics 15](#_Toc160098575)

[**Chapter 3 Approaching Solution** 17](#_Toc160098576)

[3 Approaching Solution 17](#_Toc160098577)

[3.1 Dataset 19](#_Toc160098578)

[3.2 Proposed CNNs for Classification 24](#_Toc160098579)

[3.2.1 Summary of Proposed CNN Architectures 26](#_Toc160098580)

[3.2.2 Proposed Hyperparameters 29](#_Toc160098581)

[3.2.3 Proposed Metrics for Evaluation 30](#_Toc160098582)

[3.3 Proposed Web Structure 30](#_Toc160098583)

[**Chapter 4 Result** 31](#_Toc160098584)

[4 Result 31](#_Toc160098585)

[4.1 Classification Results 31](#_Toc160098586)

[4.2 Built API 38](#_Toc160098587)

[4.3 Web Application 40](#_Toc160098588)

[4.4 Implementation Structure 43](#_Toc160098589)

[**Chapter 5 Conclusion** 45](#_Toc160098590)

[5 Conclusion 45](#_Toc160098591)

[References or Bibliography 46](#_Toc160098592)

[Appendices 48](#_Toc160098593)

[Copyright Acknowledgements 57](#_Toc160098594)

List of Tables

[Table 1: A brief history of CNNs 12](#_Toc160098595)

[Table 2: Approaching Solution Description 17](#_Toc160098596)

[Table 3: Data Distribution in Train – Validation – Test Dataset 19](#_Toc160098597)

[Table 4: Class Distribution in the Train Dataset 20](#_Toc160098598)

[Table 5: Class Distribution in the Validation Dataset 21](#_Toc160098599)

[Table 6: Class Distribution in the Test Dataset 23](#_Toc160098600)

[Table 7: Hyperparameters used in model training. 29](#_Toc160098601)

[Table 8: Classification Results 31](#_Toc160098602)

[Table 9: Description of Implementation Structure 44](#_Toc160098603)

List of Figures

[Figure 1: Sample of Hair disease images collected by Kaggle. 3](#_Toc160098604)

[Figure 2: Convolution layer with multiple filters. 10](#_Toc160098605)

[Figure 3: Convolution Neural Network (CNN) Architecture 11](#_Toc160098606)

[Figure 4: Confusion matrix. 14](#_Toc160098607)

[Figure 5: Approaching Solution 17](#_Toc160098608)

[Figure 6: Percentage of Data Distribution in Train – Validation – Test Dataset 19](#_Toc160098609)

[Figure 7: Percentage of Class Distribution in the Train dataset. 21](#_Toc160098610)

[Figure 8: Percentage of Class Distribution in the Validation dataset. 22](#_Toc160098611)

[Figure 9: Percentage of Class Distribution in the Test dataset. 24](#_Toc160098612)

[Figure 10: CNNs for Classification 25](#_Toc160098613)

[Figure 11: Training Process 26](#_Toc160098614)

[Figure 12: VGG-16 architecture. 26](#_Toc160098615)

[Figure 13: VGG-19 architecture. 27](#_Toc160098616)

[Figure 14: Inception-v3 architecture. 28](#_Toc160098617)

[Figure 15: Resnet-50 architecture. 29](#_Toc160098618)

[Figure 16: Web Structure. 30](#_Toc160098619)

[Figure 17: Confusion matrix of VGG-16 33](#_Toc160098620)

[Figure 18: Confusion matrix of VGG-19 34](#_Toc160098621)

[Figure 19: Confusion matrix of Inception-v3. 35](#_Toc160098622)

[Figure 20: Confusion matrix of ResNet50. 36](#_Toc160098623)

[Figure 21: Confusion matrix of ResNet152. 37](#_Toc160098624)

[Figure 22: Load the best-trained models. 38](#_Toc160098625)

[Figure 23: Example of an API created in Python Flask. 39](#_Toc160098626)

[Figure 24: Web UI for input. 40](#_Toc160098627)

[Figure 25: Web UI for output. 40](#_Toc160098628)

[Figure 26: Implementation structure. 43](#_Toc160098629)

List of Appendices

[Appendix A: Image Preprocessing and Data Augmentation. 48](#_Toc160098630)

[Appendix B: Load the Pre-trained Model. 49](#_Toc160098631)

[Appendix C: Compile the Model with Hyperparameters. 49](#_Toc160098632)

[Appendix D: Setup the Checkpoint. 49](#_Toc160098633)

[Appendix E: Train and save the Model. 50](#_Toc160098634)

[Appendix F: Visualize the Trained Model. 51](#_Toc160098635)

[Appendix G: VGG-16 Trained Result. 52](#_Toc160098636)

[Appendix H: VGG-19 Trained Result. 52](#_Toc160098637)

[Appendix I: Inception-v3 Trained Result. 53](#_Toc160098638)

[Appendix J: Resnet-50 Trained Result. 53](#_Toc160098639)

[Appendix K: Resnet-152 Trained Result. 54](#_Toc160098640)

[Appendix L: Evaluate the Model with Test Dataset. 54](#_Toc160098641)

[Appendix M: Print Evaluation Metrics with Test Dataset 55](#_Toc160098642)

[Appendix N: Python Flask loads the Trained Models. 56](#_Toc160098643)

[Appendix O: Python Flask gets the Model’s Predictions and responses to the Web application. 56](#_Toc160098644)

1. **Introduction**

# Introduction

## Introduction

Hair is an important part of the human body. Hair not only protects the scalp from ultraviolet rays but also hair conveys sensory and gender information, affecting body image and confidence as research was introduced by Takagi et al. in 2023 [1]. Common scalp and hair diseases can cause symptoms of peeling, itching, redness, and hair loss, even seriously affecting the patient’s psychology. Sometimes some hair diseases become complicated, causing infections that seriously affect the patient's appearance and health. Early and highly accurate diagnosis of the diseases is one of the decisive factors in the patient's effective treatment.

In recent years, AI has tended to be strongly applied in medicine such as disease diagnosis, research for drug development, and optimization for individual treatment. In particular, the applications based on deep learning for disease diagnosis have the potential to achieve great progress in automatic disease diagnosis, making diagnosis faster, cheaper, and more accessible. It is like students spend at least 10 years studying medicine then going to the hospital to identify diseases and remember them until going to work and seeing similar patients again, the memory is used to recognize the disease. Meanwhile, computers “learn” faster, and “remember” more accurately and the amount of data to remember is almost endless, so an automatic diagnosis application is no less good at diagnosing than top experts and it can be copied worldwide quickly and inexpensively.

This research is performed on a novel dataset to find the best solution for hair disease classification based on deep learning. There are different classes of hair diseases: Alopecia Areata, Contact Dermatitis, Folliculitis, Head Lice, Lichen Planus, Male Pattern Baldness, Psoriasis, Seborrheic Dermatitis, Telogen Effluvium and Tinea Capitis are shown in below Figure 1. If an application that automatically classifies hair diseases is used, not only will it assist health experts in diagnosing earlier and with higher accuracy for effective treatment, but also be more accessible and lower cost with minor personnel in spas or beauty institutes.

The structure of this research includes two parts:

* Part 1: training models with different architectures then compare them with each other and select models having reliable performance.
* Part 2: based on the reliable performance-trained models creating APIs and developing an application that allows users to input images, then the application calls the API to determine the hair disease class name and return the prediction result to the user interface.

|  |  |  |  |
| --- | --- | --- | --- |
| A close-up of a person's head  Description automatically generated |  |  |  |
| **Alopecia Areata** | **Contact Dermatitis** | **Folliculitis** | **Head Lice** |
|  |  |  |  |
| **Lichen Planus** | **Male Pattern Baldness** | **Psoriasis** | **Seborrheic Dermatitis** |
|  |  |  |  |
| **Telogen Effluvium** | **Tinea Capitis** |  |  |
|  | | | |

Figure 1: Sample of Hair disease images collected by Kaggle.

## Related Work

There have been many studies using deep learning to determine disease type such as a study using a hybrid deep learning for automated skin disease prognosis was introduced by Rao et al. in 2023 [2]. The study suggested using a deep convolution neural networks (DCNN) algorithm to categorize the skin disease type, and then the binary butterfly optimization approach (BBOA) algorithm is applied to improve the method performance. The dataset is used from Kaggle HAM10000 which includes 10,000 images. The model’s performance reached a 91.02 accuracy rate and took 99.67 seconds to train across 20 epochs.

In 2022, Ahammed et al. proposed a research based on machine learning to approach for detection and classification of skin diseases using image segmentation [3]. The paper introduced a method using morphological filtering for digital hair removal and then de-blur or denoise the images by applying Gaussian filtering. The Gray Level Co-occurrence Matrix (GLCM) and statistical features techniques are used to extract the input patterns from skin images. Three machine learning algorithms Decision Tree (DT), Support Vector Machine (SVM), and K-Nearest are used to extract features and effectively classify skin diseases as melanoma, melanocytic, basal cell carcinoma, nevus, benign keratosis, actinic keratosis, vascular lesion, dermatofibroma, and Squamous cell carcinoma. Two datasets ISIC 2019 challenge and HAM10000 are used for models’ validation. The research recommended SVM performs slightly better than the others with an average accuracy of 95% and 97%.

Rafay et al. recommended an Efficient Net model for the classification of 31 skin diseases on a large manually curated dataset in 2023 [4]. This study was performed on a novel dataset of 31 skin diseases by merging two different datasets. Three different CNN architectures (EfficientNet, RestNet, VGG) were used for transfer learning. Among the architectures, EfficientNet had the highest testing accuracy of 71%, so it was further trained to find fine turning. After the 70/30 data split comprising 3,423 samples was further augmented, the model performance was improved to an accuracy rate of 72%. Then the experiment was re-performed with a train/ test data split of 80/20 and the accuracy score was improved to 74%. Augmentations were applied to the data and the accuracy score was able to reach 87.15%. Finally, the best accuracy-trained model was deployed on a web server.

A proposed smartphone-based application for early skin disease prognosis of Shahin et al. was published in 2023 [5]. This paper suggested a set of 16 distinctive CNN models based on deep learning, to conduct over 45,000 images from the HAM10000 dataset. Data augmentations were applied to increase the sample size and balance the number of images in each class. The authors have compared their models against one another to evaluate their efficiency and effectiveness. by using performance measurements such as accuracy, sensitivity, precision, F-score, G-Mean, and specificity. The results proved that most applied models reached up to an accuracy of 99%.

In 2023, Maqsood et al. suggested a unified Computer-aided diagnostics model based on deep learning for skin lesion segmentation and classification [6]. In the proposed methodology, a contrast enhancement-based modified bio-inspired multiple exposure fusion to pre-process the input thermoscopic images. In the second stage, a custom 26-layered CNN framework is applied for lesion segmentation. In the third stage, four pre-trained CNN models (Xception, ResNet-50, ResNet-101, and VGG16) are modified to retrieve deep features and trained using transfer learning on the segmented lesion images. Following that, the features are fused using the convolutional sparse image decomposition fusion scheme, and the univariate measurement and Poisson distribution approaches are utilized for best feature selection. Finally, the selected features are fed to the multi-class support vector machine (MC-SVM) to classify. This approach was performed on HAM10000, ISIC2018, ISIC2019, and PH2 datasets and the result achieved accuracy rates of 98.57%, 98.62%, 93.47%, and 98.98%. The limitation of this proposed methodology is handling the increased dimensionality of features after the fusion process.

As research was introduced by Karthik et al. in 2022 [7], a model named Eff2Net was designed on EfficientNetV2 in combination with the Efficient Channel Attention (ECA). The research used a methodology that replaces the Excitation (SE) block and standard Squeeze in the EfficientNetV2 architecture with the ECA block. By using that method, the total number of trainable parameters was significantly decreased. The suggested CNN learned around 16 M parameters of skin disease classification. The research collected 4,930 images from various sources, then data augmentation was used, and generated 17,329 images for the dataset. The skin diseases were classified into four classes: acne, actinic keratosis (AK), melanoma, and psoriasis. The model reaches a testing accuracy of 84.70%.

A system that can detect skin lesions and categorize them as normal or benign was developed in MATLAB by Hatem and proposed in 2022 [8]. K-nearest neighbor (KNN) architecture was used effectively for approaching the difference between malignant skin lesions and normal skin. The system achieved 98% test accuracy in skin lesion classification. The disadvantage of the system is that the prediction accuracy is not high anymore on a large dataset.

Relating to skin lesion classification, another study was conducted by Dhivyaa et al. and published in 2020 [9]. The study was performed by using decision trees combined with random forest technique and then comparing them on different datasets. High-resolution feature maps can be built with the proposed method, so the spatial details of the image can be preserved. As compared with the other existing algorithms, the proposed method works more accurately. The authors found the results on the ISIC and HAM datasets reached 97% accurate, 99% specific, and 87% sensitive for skin lesion classifications.

Hossain et al. proposed a method exploring convolutional neural networks and transfer learning to diagnose Lyme disease from skin lesion images in 2022 [10]. First, an erythema migrants (EM) dataset was created with the help of expert dermatologists. Second, the dataset was used to train and test twenty-three CNN algorithms customized from ResNet, VGG, DenseNet, Xception, MobileNet, NASNet, and EfficientNet then benchmarked the efficiency of these models in terms of computational complexity, predictive performance, and statistical significance. Third, custom transfer learning was used from ImageNet pre-trained models as well as pre-trained the CNNs with the dataset HAM1000 of skin lesions. Fourth, utilizing Gradient-weighted Class Activation Mapping for the input-regions visualization that are important to CNNs for predictions. Fifth, the researchers proposed a methodology for model selection based on computational complexity and predictive performance. The result showed that the customized ResNet50 algorithm reached the best classification accuracy of 84.42% ±1.36, precision of 83.1% ±2.49, AUC of 0.9189 ±0.0115, specificity of 80.65% ±3.59 and sensitivity of 87.93% ±1.47.

In 2022, a study that proposed a methodology of deep learning-based hair removal for improved skin disease diagnosis was introduced by EI-Shafai et al. [11]. The study suggested using convolutional neural networks (CNN) combined with generative adversarial networks (GAN) to remove hair in images. For the proposed model evaluation, a dataset containing both hairless and hair-covered images is required. A new dataset Modified-HAM 10000 (M-HAM 10000) was introduced. The result showed that the proposed technique improved performance on the used dataset. Moreover, the proposed method achieved high efficiency in hair removal, increasing the accuracy of diagnosing skin diseases after comparing it to other existing methods.

In general, all existing methods based on various deep learning architectures such as DCNN combined with BBOA, SVM, EfficientNet, MC-SVM, or Eff2Net for skin diseases classification model trained as well as used KNN, decision trees combined with random forest technique or ResNet50 architecture for skin lesion classification, and they achieved efficient performance on skin diseases images dataset. Some studies have a disadvantage in that the prediction accuracy is not high anymore on a large dataset. The presented studies were conducted on skin disease datasets, and there has been no research performed on this hair disease dataset.

## Contributions

The main contributions of this research are as below:

* Collecting dataset from two parts: a collection of Hair Diseases dataset that contains 12,000 images from Kaggle and 530 images of hair diseases collected from the internet. Labeled images into 10 categories.
* Preprocessing and augmenting data.
* Training the dataset by many different model architectures based on convolution neural networks such as VGG-16, VGG-19, Inception-v3, ResNet50, and ResNet152 to classify hair diseases into 10 classes.
* Comparing the performed models with each other, then among them selecting models having reliable performance.
* Building APIs from the selected good models and developing an application that allows users to input images into the system then the system calls APIs to get the models’ classification result and returns the diagnostic result to the client’s web UI.
* Creating a demo of the application.

1. **Fundamental Theory**

# Fundamental Theory

This chapter presents some theoretical frameworks of techniques that will be applied in the following chapters.

## Image Classification Using CNNs

Image Classification involves the feature extraction from the image to observe some patterns in the dataset. If ANNs are used to classify images, the trainable parameters will be extremely large, leading to extremely expensive computational costs.

CNNs can learn to recognize patterns and extract features in images by using convolutional layers that apply a set of filters to the input images to detect specific patterns (below Figure 2 shows the filters using the convolution layer). Image classification includes assigning labels to input images. It is a supervised learning type where a model is trained on labeled image data to predict the class of unlabeled images.

A group of squares with different colors

Description automatically generated

Figure 2: Convolution layer with multiple filters.

CNNs are particularly for image classification since they can learn the spatial hierarchical features such as textures, edges, and shapes, and can accurately recognize objects in images. Learning hierarchical features is especially suitable for image classification, where an image’s visual features can vary widely. 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 (please see below Figure 3 for a CNN architecture).

A diagram of a computer

Description automatically generated

**Figure 3: Convolution Neural Network (CNN) Architecture**

## CNN Architectures

Before GPUs were produced and put into use, computers could not process enormous amounts of images within a reasonable amount of time. That is why neural networks did not spark until 2010. Since GPUs came into use and the ImageNet Large Scale Visual Recognition Challenge which is a competition of object recognition and image classification first held in 2010, it prompted the creation and development of the CNN architectures. A brief history of CNNs from 2010 is described in Table 1 below.

**Table 1: A brief history of CNNs**

|  |  |  |
| --- | --- | --- |
| **Architecture** | **Year** | **Description** |
| AlexNet | 2012 | It is one of the most influential achievements of the computer vision field because it has prompted more studies using CNN and GPUs to speed up machine learning. In addition, the computations were split on multiple GPUs which makes training faster. It won the competition of 2012 ILSVRC. |
| VGG-nets | 2014 | This architecture was introduced by the Visual Geometry Group of Oxford University and won the second prize in the classification task of the ImageNet competition in 2014. It is seen as a deeper version of AlexNet. |
| GoogLeNet | 2014 | VGG-nets were defeated by GoogLeNet in a classification task of the 2014 ImageNet competition. GoogLeNet won the championship. |
| Inception-v3 | 2015 | It was introduced during the ImageNet Recognition Challenge. The design of Inception-v3 allows deeper networks. It uses techniques like factorizing larger convolutions to smaller convolutions and asymmetric factorizations. These factorizations are done to reduce the number of parameters being used at every inception module. |
| ResNet | 2015 | It won the 2015 ILSVRC competition. ResNet suggests the residual module which makes training much deeper networks easier. |
| DenseNet | 2016 | It is a neural network with dense connections. It suggested a direct connection between any two layers. It makes the input of each layer the union outputs of all previous layers, and the feature learned by the layer is directly transmitted to all layers afterward. So, this network architecture minimizes the problem of vanishing gradient, encourages feature reuse, enhances feature propagation, and significantly reduces the number of parameters. |
| ResNeXt | 2017 | Based on GoogLeNet and ResNet, it was built by combining Inception modules between skip connections. |
| EfficientNet | 2019 | The EfficientNet was introduced by a team of researchers at Google AI in 2019, then it became a go-to architecture for many challenging tasks including image segmentation, object recognition, and even language processing. Its success lies in its ability to balance two key factors: computational efficiency and model performance. |
|  | | |

## Metrics

### Confusion Matrix

A confusion matrix is a matrix that shows the number of correct and incorrect predictions compared with the actual classifications in the test set. This matrix describes the performance of a classification model on test data for the true value known. It is a matrix with dimensions n\*n where n is the number of classes (see below Figure 4 for a confusion matrix).

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | |  | | Predicted Value | |
|  | |  | | Class 0 | Class 1 |
| Actual Value | | Class 0 | | True  Negative | False  Positive |
| Class 1 | | False  Negative | True  Positive |
|  |  | |  | | |

Figure 4: Confusion matrix.

Four outcomes can occur while predicting classification:

* True Positives (TP): number of outcomes that are actually positive and are predicted positive.
* True Negative (TN): number of outcomes that are actually negative and are predicted negative.
* False Positive (FP): number of outcomes that are actually negative but predicted positive.
* False Negative (FN): number of outcomes that are actually positive but predicted negative.

### Classification Metrics

Classification metrics are used to measure classification performance. There are many ways to measure performance, some of the most popular metrics are accuracy, log-loss, AUC-ROC, precision, recall, and F1-score.

**Accuracy:**

Accuracy explains how often the classifier correctly predicts. It is defined as the ratio of the number of correct predictions and the total number of predictions.

**Precision:**

Precision measures to determine how many of the correctly predicted cases actually turned out to be possible.

**Recall:**

Recall is defined as the number of true positives divided by the total number of actual positives.

**F1-score:**

F1-score is the harmonic mean of precision and recall.

**Log loss:**

*logloss*(N=1) = *y* log(*p*) + (1 - *y*) log(1-*p*)

Log loss is one of the important metrics to assess the performance of a classification model.

1. **Approaching Solution**

# Approaching Solution

This study proposes steps as below in Figure 5 to approach the best solution for hair disease diagnosis.

A diagram of a model

Description automatically generated

Figure 5: Approaching Solution

Detailed steps are described below table (see Table 2)

Table 2: Approaching Solution Description

|  |  |  |
| --- | --- | --- |
|  | **Step** | **Description** |
| 1 | Collect dataset | Collect image data. |
| 2 | Preprocessing images | Label images and resize images to a size of 224x224. |
| 3 | Data Augmentation | * Rescale images 1. /255 to transform every pixel value from the range [0,255] -> [0,1]. * Using shear range to apply shearing transformations. * Using zoom range to zoom inside the picture. * Using width shift range, and height shift range to translate pictures vertical or horizontal. * Using horizontal flip to make pictures flip half of the images horizontally. * Using rotation range to rotate pictures. |
| 4 | Classification (Training, Validation) | After preprocessing and augmentation, images are inputted into many different CNN architectures to classify hair diseases through training and validation. |
| 5 | Model evaluation | Let the model predict on test set then report evaluation metrics: accuracy, precision, recall, f1-score, and support. |
| 6 | Compare and select models that have reliable performance | Based on evaluation metrics report to compare and select models having reliable performance. |
| 7 | Build APIs from good models | Build APIs from the good models. |
| 8 | Build a website/ application | Build a website/ application that calls APIs of the selected models for hair disease diagnostic. |

## Dataset

The dataset is collected in two parts: using a Kaggle dataset that contains 12,000 images of hair diseases and 530 images collected from the internet. Image data are labeled into 10 classes of hair diseases. The dataset is split into train-validation-test sets with ratio [0.8, 0.1, 0.1]. Below Table 3 and Figure 6 describe data distribution in Train – Validation – Test dataset.

Table 3: Data Distribution in Train – Validation – Test Dataset

|  |  |  |
| --- | --- | --- |
| **Train** | **Validation** | **Test** |
| 10024 | 1253 | 1253 |

Figure 6: Percentage of Data Distribution in Train – Validation – Test Dataset

Below Table 4 and Figure 7 describe class distribution in the Train dataset.

Table 4: Class Distribution in the Train Dataset

|  |  |  |  |
| --- | --- | --- | --- |
|  | **From Kaggle** | **From Others** | **Total** |
| **Alopecia Areata** | 960 | 96 | 1056 |
| **Contact Dermatitis** | 960 | 32 | 992 |
| **Folliculitis** | 960 | 32 | 992 |
| **Head Lice** | 960 | 40 | 1000 |
| **Lichen Planus** | 960 | 32 | 992 |
| **Male Pattern Baldness** | 960 | 40 | 1000 |
| **Psoriasis** | 960 | 32 | 992 |
| **Seborrheic Dermatitis** | 960 | 48 | 1008 |
| **Telogen Effluvium** | 960 | 32 | 992 |
| **Tinea Capitis** | 960 | 40 | 1000 |

Figure 7: Percentage of Class Distribution in the Train dataset.

Below Table 5 and Figure 8 describe class distribution in the Validation dataset.

Table 5: Class Distribution in the Validation Dataset

|  |  |  |  |
| --- | --- | --- | --- |
|  | **From Kaggle** | **From Others** | **Total** |
| **Alopecia Areata** | 120 | 12 | 132 |
| **Contact Dermatitis** | 120 | 4 | 124 |
| **Folliculitis** | 120 | 4 | 124 |
| **Head Lice** | 120 | 5 | 125 |
| **Lichen Planus** | 120 | 4 | 124 |
| **Male Pattern Baldness** | 120 | 5 | 125 |
| **Psoriasis** | 120 | 4 | 124 |
| **Seborrheic Dermatitis** | 120 | 6 | 126 |
| **Telogen Effluvium** | 120 | 4 | 124 |
| **Tinea Capitis** | 120 | 5 | 125 |

Figure 8: Percentage of Class Distribution in the Validation dataset.

Below Table 6 and Figure 9 describe class distribution in the Test dataset.

Table 6: Class Distribution in the Test Dataset

|  |  |  |  |
| --- | --- | --- | --- |
|  | **From Kaggle** | **From Others** | **Total** |
| **Alopecia Areata** | 120 | 12 | 132 |
| **Contact Dermatitis** | 120 | 4 | 124 |
| **Folliculitis** | 120 | 4 | 124 |
| **Head Lice** | 120 | 5 | 125 |
| **Lichen Planus** | 120 | 4 | 124 |
| **Male Pattern Baldness** | 120 | 5 | 125 |
| **Psoriasis** | 120 | 4 | 124 |
| **Seborrheic Dermatitis** | 120 | 6 | 126 |
| **Telogen Effluvium** | 120 | 4 | 124 |
| **Tinea Capitis** | 120 | 5 | 125 |

Figure 9: Percentage of Class Distribution in the Test dataset.

## Proposed CNNs for Classification

Using many different CNN architectures such as VGG-16, VGG-19, Inception-v3, Resnet-50, and Resnet-152 to classify hair diseases. Below Figure 10 describes classification detail.

The classification result is categorized into 10 classes: Alopecia Areata, Contact Dermatitis, Folliculitis, Head Lice, Lichen Planus, Male Pattern Baldness, Psoriasis, Seborrheic Dermatitis, Telogen Effluvium, Tinea Capitis.

A diagram of a computer

Description automatically generated

Figure 10: CNNs for Classification

The main idea here is to train a model by loading a pre-trained model from Keras (Keras is a high-level, deep learning API developed by Google for implementing neural networks) and then compiling the model with hyper-parameters like optimizer and learning rate. To save the model at the points when the model is considered the best and the latest best model, using Checkpoint in the model with save best only argument is true (see the training process in Figure 11).

A diagram of a model

Description automatically generated with medium confidence

Figure 11: Training Process

### Summary of Proposed CNN Architectures

This section is a brief description of proposed CNNS used in this research such as VGG-16, VGG-19, Inception-v3, Resnet-50, and Resnet-152.

**VGG-16:**

The VGG-16 architecture consists of 16 layers that include 13 convolutional layers and 3 fully connected layers. Each convolution layer uses a 3x3 filter size and a stride size of 1 pixel. It uses the pooling layers to reduce the size of the input and down the feature maps. Based on extracted features the fully connected layer helps to classify the images. The input of VGG is an RGB image of 224x224 size. Below Figure 12 is an illustration of the VGG-16 architecture.

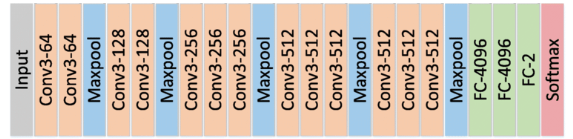


Figure 12: VGG-16 architecture.

**VGG-19:**

The VGG-19 is a deeper network compared to VGG-16. The network of the VGG-19 architecture has 19 layers, including 16 convolutional layers and 3 fully connected layers. Each convolution layer uses a 3x3 filter size and a stride size of 1 pixel. It uses the pooling layers to reduce the size of the input and down the feature maps. Based on extracted features the fully connected layer helps to classify the images. The input of VGG is an RGB image of 224x224 size. The following Figure 13 is an illustration of the VGG-19 architecture.

A group of text on a white background

Description automatically generated with medium confidence

Figure 13: VGG-19 architecture.

**Inception-v3:**

The Inception-v3 consists of 42 layers and has a lower error rate than its predecessor. The below Figure 14 shows the architecture of the Inception-v3.

A diagram of a machine

Description automatically generated

Figure 14: Inception-v3 architecture.

**Resnet-50:**

The Resnet-50 is a powerful model for image classification that can be trained on a large dataset. It consists of 50 layers divided into 16 residual blocks, with each block consisting of some convolutional layers with residual connections. With residual connection, the network can learn the residual function instead of directly learning to the underly mapping which leads to better learning and performance improvement. It also includes pooling layers, fully connected layers, and a SoftMax output layer for classification. The below Figure 15 illustrates the Resnet-50 architecture.

A diagram of a block diagram

Description automatically generated

Figure 15: Resnet-50 architecture.

**Resnet-152:**

The Resnet-152 is a deeper network than the Resnet-50. It contains 152 layers organized into 50 residual blocks. Each block consists of some convolutional layers with residual connections. The Resnet-152 also includes pooling layers, fully connected layers, and a SoftMax output layer for classification.

### Proposed Hyperparameters

The hyperparameters are used in model training as described in Table 7 below.

Table 7: Hyperparameters used in model training.

|  |  |
| --- | --- |
| **Hyperparameter** | **Value** |
| Learning rate | 0.001 |
| Batch size | 128 |
| Optimizer | Using optimize Adam to compile with the model. |
| Number of epochs | 20 |

### Proposed Metrics for Evaluation

After the models are trained, the trained models are compared with each other based on the following evaluation metrics: Test Accuracy (%), Loss (%), Precision, Recall, F1-score, Support.

## Proposed Web Structure

Python loads the trained models, and Python Flask is used to build REST API that receives requests from web applications, then REST API returns a response to the web application. See the following Figure 16 for the web structure illustration.

Diagram of a cloud with text and a cloud

Description automatically generated

Figure 16: Web Structure.

1. **Result**

# Result

## Classification Results

The dataset is pre-processed, augmented, and trained by CNN architectures VGG-16, VGG-19, Inception-v3, Resnet-50, and Resnet-152. All models are compiled with optimizer Adam and a learning ratio of 0.01. After training and testing, the classification results on the dataset are illustrated in Table 8.

Table 8: Classification Results

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| Classification method | Statistical measures | | | | | | |
| Test Accuracy (%) | Loss  (%) | Precision | Recall | F1-score | Support | Epoch |
| VGG-16 | **96.81** | **20.13** | **0.97** | **0.97** | **0.97** | 1253 | 20 |
| VGG-19 | **96.73** | **19.94** | **0.97** | **0.97** | **0.97** | 1253 | 20 |
| Inception-v3 | **95.13** | 100.74 | **0.95** | **0.95** | **0.95** | 1253 | 20 |
| Resnet-50 | 49.16 | 151.75 | 0.71 | 0.49 | 0.45 | 1253 | 25 |
| Resnet-152 | 36.87 | 181.54 | 0.72 | 0.37 | 0.37 | 1253 | 10 |

The models VGG-16 and VGG-19 classifications produced the best results. VGG-16 classification result reached an accuracy rate of 96.81%, and log loss was 20.13%. VGG-19 classification produced an accuracy rate of 96.73% and a log loss of 19.94%. Inception-V3 shows a good accuracy rate at 95.13% while log loss is not good. All these three models also achieved a good classifier Precision, Recall, and F1-score that is over 0.95.

The above classification result of each model can be explained that each model’s effectiveness depends on a combination of factors consisting of the model architecture, dataset characteristics, and hyperparameters turning. With the same dataset and same hyperparameters turning when training the models, different models archive different performances as the following:

* VGG-16 and VGG-19 archived the best performance because the dataset might be well-suited to VGG architectures and images had similar scale and complexity to the ImageNet dataset which the VGG models were originally trained on.
* Inception-v3 showed good testing accuracy but the extremely high log loss suggests that the model might not be well calibrated, and it may be overconfident in its predictions. The dataset used for training in this research may be so different from the dataset characteristic that Inception-v3 was originally trained on.
* Resnet-50 and Resnet-152 archived an extremely low testing accuracy and extremely high log loss. The dataset might be too small for Resnet architectures to perform effectively, leading to inferior performance.

The Confusion matrix of models is illustrated in the following images Figure 17, Figure 18, Figure 19, Figure 20, Figure 21.

A graph with blue squares

Description automatically generated

Figure 17: Confusion matrix of VGG-16

A graph with blue squares and white text

Description automatically generated

Figure 18: Confusion matrix of VGG-19

A graph with numbers and names

Description automatically generated with medium confidence

Figure 19: Confusion matrix of Inception-v3.

A screenshot of a graph

Description automatically generated

Figure 20: Confusion matrix of ResNet50.

A blue and white graph with numbers and a bar chart

Description automatically generated with medium confidence

Figure 21: Confusion matrix of ResNet152.

## Built API

APIs were built by using the Python Flask framework. They loaded the trained models of the three best models VGG-16, VGG-19, and Inception-v3 for hair diseases classification. The APIs received the Get/ Post/ Input/ Delete request from the client and responded to relevant suggestions. The following images Figure 22 and Figure 23 show examples of loading the three best-trained models and creating REST APIs by using Python Flask.

A computer screen shot of a program

Description automatically generated

Figure 22: Load the best-trained models.

A screenshot of a computer program

Description automatically generated

Figure 23: Example of an API created in Python Flask.

## Web Application

A web application was built to display hair disease classification results from the best three models. The web application UI was designed as below in Figure 24 and Figure 25.

A screenshot of a social media post

Description automatically generated

Figure 24: Web UI for input.

A screenshot of a computer

Description automatically generated

Figure 25: Web UI for output.

The web application responded to below use cases:

|  |  |
| --- | --- |
| **Use Case 1** | The user inputs images and review |
| **Actor** | Doctor |
| **Overview** | The system allows the user to input one or multiple image files then the user can review and remove some of the input images. |
| **Trigger** | N/A |
| **Precondition** | N/A |

|  |  |
| --- | --- |
| **Use Case 2** | Diagnosis |
| **Actor** | Doctor |
| **Overview** | The user clicks the diagnosis button to ask the system to classify hair diseases from the input image files. The web system will return and display the diagnostic result from the models’ prediction. |
| **Trigger** | N/A |
| **Precondition** | Image files are input. |

|  |  |
| --- | --- |
| **Use Case 3** | The user fills diagnostic conclusion and saves each image |
| **Actor** | Doctor |
| **Overview** | The system allows the user to fill diagnostic and save for each image. The image will be saved with the label as the diagnostic conclusion.  The folder to save the image is c:\DiagnosticDownImages. |
| **Trigger** | N/A |
| **Precondition** | User-filled diagnostic conclusion. |

|  |  |
| --- | --- |
| **Use Case 4** | The user saves all images with the diagnostic conclusion |
| **Actor** | Doctor |
| **Overview** | The system allows the user to save all images at one time. The images will be saved with the label as the diagnostic conclusion.  The folder to save the images is c:\DiagnosticDownImages. |
| **Trigger** | N/A |
| **Precondition** | User-filled diagnostic conclusion. |

## Implementation Structure

Implementation is structured in directories as the following Figure 26.

A screenshot of a computer program

Description automatically generated

Figure 26: Implementation structure.

See detailed description in the following Table 9.

Table 9: Description of Implementation Structure

|  |  |
| --- | --- |
| Main.py | It is a main Python file for loading the trained models and building REST APIs. |
| template | This folder contains html files of the web application. |
| static | It contains CSS, JavaScript, temporary folder of uploaded images for the web page UI. |
| AI\_models | This folder contains Jupiter notebook files of trained models and hdf5 files of the saved models training. |
| requirements.txt | This file contains libraries that need to be installed for the application can build and run. |

1. **Conclusion**

# Conclusion

This research suggested models, which are based on CNNs, have succeeded in categorizing hair diseases with checkpoints used to save points when the models were the latest best models. The latest best model points were loaded to predict images and return the prediction result having the highest accuracy. Moreover, the research also built a web application to show the prediction of the models. The limitation of this research is that the models were mainly trained on a dataset taken from Kaggle. Therefore, when the application is put into practical use, there will be a possibility that the prediction results will no longer be highly accurate on diverse images are input into the application. However, the application supports the function of saving new images that will be labeled by medical experts. It will help later when using more diverse data to train models and upgrade the system.

References or Bibliography

[1] M. Takagi *et al.*, “Invention of automated numerical algorithm adopting binarization for the evaluation of scalp hair coverage: An image analysis providing a substitute for phototrichogram and global photography assessment for hair diseases,” *Journal of Dermatological Science*, p. S0923181123001949, Sep. 2023, doi: 10.1016/j.jdermsci.2023.09.003.

[2] G. M. Rao, D. Ramesh, P. Gantela, and K. Srinivas, “A hybrid deep learning strategy for image based automated prognosis of skin disease,” *Soft Comput*, Mar. 2023, doi: 10.1007/s00500-023-08007-0.

[3] M. Ahammed, Md. A. Mamun, and M. S. Uddin, “A machine learning approach for skin disease detection and classification using image segmentation,” *Healthcare Analytics*, vol. 2, p. 100122, Nov. 2022, doi: 10.1016/j.health.2022.100122.

[4] A. Rafay and W. Hussain, “EfficientSkinDis: An EfficientNet-based classification model for a large manually curated dataset of 31 skin diseases,” *Biomedical Signal Processing and Control*, vol. 85, p. 104869, Aug. 2023, doi: 10.1016/j.bspc.2023.104869.

[5] M. Shahin, F. F. Chen, A. Hosseinzadeh, H. Khodadadi Koodiani, A. Shahin, and O. Ali Nafi, “A smartphone-based application for an early skin disease prognosis: Towards a lean healthcare system via computer-based vision,” *Advanced Engineering Informatics*, vol. 57, p. 102036, Aug. 2023, doi: 10.1016/j.aei.2023.102036.

[6] S. Maqsood and R. Damaševičius, “Multiclass skin lesion localization and classification using deep learning-based features fusion and selection framework for smart healthcare,” *Neural Networks*, vol. 160, pp. 238–258, Mar. 2023, doi: 10.1016/j.neunet.2023.01.022.

[7] R. Karthik, T. S. Vaichole, S. K. Kulkarni, O. Yadav, and F. Khan, “Eff2Net: An efficient channel attention-based convolutional neural network for skin disease classification,” *Biomedical Signal Processing and Control*, vol. 73, p. 103406, Mar. 2022, doi: 10.1016/j.bspc.2021.103406.

[8] M. Q. Hatem, “Skin lesion classification system using a K-nearest neighbor algorithm,” *Vis. Comput. Ind. Biomed. Art*, vol. 5, no. 1, p. 7, Dec. 2022, doi: 10.1186/s42492-022-00103-6.

[9] C. R. Dhivyaa, K. Sangeetha, M. Balamurugan, S. Amaran, T. Vetriselvi, and P. Johnpaul, “Skin lesion classification using decision trees and random forest algorithms,” *J Ambient Intell Human Comput*, Nov. 2020, doi: 10.1007/s12652-020-02675-8.

[10] S. I. Hossain *et al.*, “Exploring convolutional neural networks with transfer learning for diagnosing Lyme disease from skin lesion images,” *Computer Methods and Programs in Biomedicine*, vol. 215, p. 106624, Mar. 2022, doi: 10.1016/j.cmpb.2022.106624.

[11] W. El-Shafai, I. A. El-Fattah, and T. E. Taha, “Deep learning-based hair removal for improved diagnostics of skin diseases,” *Multimed Tools Appl*, Sep. 2023, doi: 10.1007/s11042-023-16646-6.

Appendices

Appendix A: Image Preprocessing and Data Augmentation.

A screen shot of a computer program

Description automatically generated

Appendix B: Load the Pre-trained Model.

A screen shot of a computer program

Description automatically generated

Appendix C: Compile the Model with Hyperparameters.

A screen shot of a computer program

Description automatically generated

Appendix D: Setup the Checkpoint.

A computer screen shot of a black background

Description automatically generated

Appendix E: Train and save the Model.

A computer screen shot of a program

Description automatically generated

Appendix F: Visualize the Trained Model.

A screen shot of a computer program

Description automatically generated

Appendix G: VGG-16 Trained Result.

A screenshot of a graph

Description automatically generated

Appendix H: VGG-19 Trained Result.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Appendix I: Inception-v3 Trained Result.

A graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of a graph of

Description automatically generated

Appendix J: Resnet-50 Trained Result.

A graph of a graph of a training and a training

Description automatically generated with medium confidence

Appendix K: Resnet-152 Trained Result.

A graph of a line and a line

Description automatically generated with medium confidence

Appendix L: Evaluate the Model with Test Dataset.

A screen shot of a computer program

Description automatically generated

Appendix M: Print Evaluation Metrics with Test Dataset

A screenshot of a computer

Description automatically generated

Appendix N: Python Flask loads the Trained Models.

A screen shot of a computer program

Description automatically generated

Appendix O: Python Flask gets the Model’s Predictions and responses to the Web application.

A screen shot of a computer program

Description automatically generated

Copyright Acknowledgements

© Copyright by Nguyen Thi Minh Hien 2024