APPLYING DEEP LEARNING TECHNIQUES TO SEGMENTIZE AND CLASSIFY TONGUE REGIONS FOR TRADITIONAL AND COMPLEMENTARY MEDICINE (TCM) DIAGNOSIS

BY

YEAP CHUN HONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

APPLYING DEEP LEARNING TECHNIQUES TO SEGMENTIZE AND CLASSIFY TONGUE REGIONS FOR TRADITIONAL AND COMPLEMENTARY MEDICINE (TCM) DIAGNOSIS

BY

YEAP CHUN HONG

A REPORT

SUBMITTED TO

Universiti Tunku Abdul Rahman

in partial fulfillment of the requirements

for the degree of

BACHELOR OF COMPUTER SCIENCE (HONOURS)

Faculty of Information and Communication Technology

(Kampar Campus)

JUNE 2025

ACKNOWLEDGEMENTS

I would like to express my deepest gratitude to my supervisor, Ts. Dr. Lee Wai Kong, for their invaluable guidance, support, and encouragement throughout my deep learning image segmentation projects. The insightful advice, technical expertise, and constructive feedback have been instrumental in shaping my understanding of deep learning techniques, particularly in tongue image analysis. I truly appreciate the patience, motivation, and dedication, which have greatly contributed to the success of this research. I would also like to extend my sincere thanks to Ts. Dr. Ahmad Zaffry Hadi Bin Mohd Juffry, the moderator of this project for the support and involvement which have been greatly appreciated throughout the project. Thank you both for your unwavering support and mentorship.

I would also like to thank my friend, Mr. Tan Yi Jian, for his help during this research. Thank you for the encouragement and support that have been a constant source of motivation.

ABSTRACT

Tongue diagnosis is a fundamental component of Traditional and Complementary Medicine (TCM), yet manual inspection remains subjective and inconsistent. This study proposes a deep learning framework to enhance tongue image analysis through segmentation, classification, and explainable artificial intelligence (XAI). A Mobile U-Net model was proposed and developed for efficient and accurate tongue region segmentation. Classification tasks were conducted for both binary (stained vs. non-stained) and multi-class pathological coatings, covering clinically relevant categories. Lightweight architectures, including the proposed Efficient-ResNet, achieved competitive accuracy with minimal computational cost, demonstrating strong potential for deployment in resource-constrained environments. Grad-CAM was integrated to provide visual explanations of model decisions, improving transparency and clinical trust. Experimental results show that ResNet50 and LECA-EfficientNetV2-S achieved the highest accuracy of 99% in binary classification, while EfficientNetV2-B3 and -S excelled in multi-class tasks. Efficient-ResNet maintained strong accuracy (98.5%) with only 0.31M parameters. The findings highlight the framework's balance of efficiency, accuracy, and interpretability, offering a practical solution to standardize and modernize tongue diagnosis in TCM for both clinical and telemedicine applications.

Area of Study (Minimum 1 and Maximum 2): Medical Image Analysis, Deep Learning

Keywords (Minimum 5 and Maximum 10): Tongue Diagnosis, Traditional and Complementary Medicine (TCM), Image Segmentation, Classification, Explainable AI, Mobile U-Net, Efficient-ResNet

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

AI	Artificial Intelligence
AUC	Area Under Curve
CBAM	Convolutional Block Attention Module
CBWCE	Contour-Based Weighted Cross-Entropy
CNN	Convolutional Neural Network
DCNN	Deep Convolutional Neural Network
FBMV	Fusion Based Majority Voting
Faster R-CNN	Faster Region-Based Convolutional Neural Network
GCYTD	GELU-CA-YOLO Tongue Detection
Grad-CAM	Gradient-weighted Class Activation Mapping
HAAS	Heatmap Assisted Accuracy Score
IDE	Integrated Development Environment
IoU	Intersection over Union
JGP	Joint Photographic Experts Group
LECA	Lightweight Efficient Channel Attention
LR-ASPP	Lite Reduced Atrous Spatial Pyramid Pooling
MDSC	Mean Dice Similarity Coefficient
MIOU	Mean Intersection over Union
MPA	Mean Pixel Accuracy
NAFLD	Non-Alcoholic Fatty Liver Disease
PNG	Portable Network Graphics
PR	Precision-Recall
REF	Receiver Operating Characteristic
RGB	Red, Green, Blue
ROC	Region-based and Edge-based fusion
SAM	Segment Anything Model
SE	Squeeze-and-Excitation
TCM	Traditional and Complementary Medicine
ViT	Vision Transformer
XAI	Explainable Artificial Intelligence

Chapter 1
Introduction

The problem statement, motivation and objectives of research, project scopes, contributions to the tongue image classification fields and the overall report organization will be discussed in this chapter.

1.1 Problem Statement and Motivation

Tongue diagnosis is a noninvasive and visual method for assessing human health status. It forms a fundamental part of syndrome identification and treatment in Traditional and Complementary Medicine diagnosis [1]. Therefore, the practitioners commonly examine the features of the tongue to identify early signs of internal imbalances or diseases as a routine component for health check-ups [2]. According to [3], studies show that characteristics such as tongue colour, coating colour and thickness, moisture level, fissures and stains can all serve as the key indicators in evaluating a person’s health status. However, the visual tongue diagnosis done by practitioners is subjective and prone to inconsistencies due to external factors such as lighting, tongue staining and experience of the practitioner. A recent study demonstrated that even experienced TCM practitioners achieved only 69% accuracy in distinguishing stained from pathological tongue coatings, highlighting the variability and inaccuracy of manual visual tongue assessments [4].

The challenge is further compounded by the lack of publicly available and annotated datasets that comprehensively represent stained tongue images. Researchers frequently use self-collected tongue images for model training, which is a time-consuming and tedious procedure that yields relatively small training and validation datasets [5]. Besides that, severe class imbalance in existing datasets is also a frequent challenge in medical imaging applications [6]. For example, white coating tongues may dominate while yellow coating tongues are rare in the datasets. The imbalance will skew the model training and lead to bias classifiers that favor the majority class and fail to generalize across underrepresented but clinically significant categories.

The growing integration of artificial intelligence into medical diagnostics presents a significant opportunity to advance the field of TCM in assisting the practitioners. Although the existing deep learning models have demonstrated promising results, many of the models are computationally intensive and complex in classifying tongue coatings [7]. This limitation motivates the need to develop lightweight models that maintain high diagnostic accuracy while significantly reducing the computational cost so that it is suitable for real world clinical adoption.

Besides that, deep learning models are often treated as black boxes due to the model offering little transparency into how the decisions are made [8]. The lack of explainability is a barrier to trust in the healthcare industry where the practitioner insight and interpretive reasoning are integral to diagnosis. Therefore, there is a strong motivation to incorporate explainable AI (XAI) techniques into the model pipeline to visualize the key decision-making features. The practitioners can have a better understanding of AI outputs so that it bridges the gap between traditional diagnostic expertise and modern machine learning tools.

In summary, this study is motivated by the ambition to apply and improve upon the current state-of-the-art technology in tongue classification by addressing challenges such as class imbalance, limited datasets and the need for robust generalization. The focus is on building a lightweight and efficient deep learning model that is suitable for real-world deployment and integrating explainable AI techniques to enhance transparency and clinical interpretability of the model's predictions.

1.2 Objectives

The main objective of the thesis is to develop a deep learning framework that integrates tongue image segmentation, classification and explainable AI to support Traditional and Complementary Medicine diagnostic processes. The main objective is achieved by first achieving the sub-objective as stated below:

1. To develop a Mobile U-Net model that leverages U-Net's spatial localization capability with MobileNet's lightweight and efficient feature extraction to enable fast and accurate tongue region segmentation.
2. To develop deep learning models for binary classification to distinguish stained and non-stained tongue coatings as well as multi-class classification to categorize pathological tongue coatings into specific types.

3. To design a lightweight model that balances diagnostic accuracy with computational efficiency, thus improving applicability in real-world clinical workflows.
4. To incorporate explainable AI methods for both classification tasks, providing visual interpretability to enhance the transparency and trust of the model.

1.3 Project Scope and Direction

1.3.1 Segmentation

The segmentation component of this research is focused on developing an efficient deep learning model to accurately segment tongue regions from clinical images. A Mobile U-Net architecture will be implemented, which integrates the spatial localization capability of U-Net with the lightweight and computationally efficient backbone of MobileNet. The model is expected to provide precise segmentation results while maintaining low computational complexity so that it is suitable for practical use in clinical settings where efficiency is as important as accuracy. The dataset used for this work consists of tongue images paired with manually annotated ground truth masks and will be divided into training, validation and testing subsets. Preprocessing steps such as normalization will be applied to improve model generalization and robustness given the limited dataset size.

The scope also encompasses model training and evaluation using segmentation-specific loss functions such as Dice loss. Model performance will be measured using well-established metrics including Dice Coefficient, Intersection-over-Union, precision, recall and segmentation accuracy. In addition, the Mobile U-Net will be bench against conventional segmentation architecture such as U-Net and DeepLabV3+ in order to validate the accuracy and efficiency. However, the scope of this work will not focus on hardware-based real time deployment. The study will remain within the boundaries of deep learning-based tongue segmentation, emphasizing accuracy, efficiency and robustness as the primary objectives.

1.3.2 Classification

The primary objective of the classification component of this project is to develop deep learning models that are capable of classifying tongue images for diagnosis purposes in Traditional and Complementary Medicine. The study focuses on two related tasks: (1) binary classification of tongue coatings into stained and non-stained categories, (2) multi-class classification of pathological tongue coatings into clinically relevant subtypes such as white, yellow-greasy and grey-black. These classification tasks are critical in supporting accurate and objective tongue diagnosis so that the limitations of conventional visual inspection by TCM practitioners are addressed.

Firstly, separate convolutional neural network models will be constructed for each task. The binary classification model will distinguish food or drug-induced staining from pathological coatings while the multi-class model will categorize the non-stained coatings into distinct disease-indicative types. The models will be trained using a supervised learning approach with a curated dataset of annotated tongue images. Preprocessing techniques such as image segmentation and normalization will also be applied to ensure consistency in input quality.

The performance of the model will be evaluated using standard metrics such as accuracy, precision, recall, F1-Score and confusion matrices. In addition, explainable AI tools such as Grad-CAM will be integrated to visualize the regions of the tongue image that are influencing the model’s decision so that the practitioner can trust and interpret the model’s decision.

The project will be conducted in two phases. In the first phase, baseline models will be trained and evaluated to establish benchmarks for both classification tasks. In the second phase, the CNN variants will be tested and hyperparameter tuning will be performed to identify the most effective model configuration. The final models aim to be both accurate and lightweight which enable real-world deployment in clinical settings or mobile health applications.

1.4 Contributions

This study contributes to the advancement of intelligent diagnostic systems in TCM by introducing deep learning models tailored for tongue image classification. The study proposes two separate deep learning models for binary classification to differentiate stained from non-stained tongue coatings and also multi-class classification to categorize pathological coatings into clinically relevant classes. This separation of tasks ensures specialized learning and improved accuracy for both simpler and more granular diagnostic needs.

A significant contribution for this work is the development of lightweight and computationally efficient CNN architecture that is optimized for real-world applications. The solution is designed to operate effectively in resource-constrained environments. Besides that, one of the major challenges in medical image classification which is class imbalance is also addressed by applying data augmentation and applying weightage to the standard loss function based on the class in the imbalance dataset.

Furthermore, integration of explainable AI tools such as Grad-CAM is integrated into the model's pipeline. By visualizing the regions of tongue images that influence the model's decision, the system is able to provide greater transparency and interpretability to the TCM practitioners and healthcare professionals.

Finally, the study benchmarks the performance of the models against the state-of-the-art deep learning models and human expert evaluations. This comparative analysis not only validates the effectiveness of the models but also demonstrates the potential to outperform traditional manual diagnosis in certain cases. In overall, the research presents a practical and interpretable solution for enhancing the diagnostic process in TCM and contributes to the field of AI in healthcare.

1.5 Report Organization

The details of the reports are shown in the following chapters. In Chapter 2, a summary of existing research and knowledge on tongue image classification using deep learning approaches are conducted. Then, the design phase of the system is explained in detail in Chapter 3. Chapter 4 focuses on the system architecture and implementation. It explains the overall architecture of the proposed models for both binary and multi-class tasks which include architecture diagrams and technical frameworks. Chapter 5 shows the findings from the experiments conducted so that discussion and analysis can be carried out. Finally, Chapter 6 summarizes the research as a conclusion.

Chapter 2

Literature Review

2.1 Tongue Image Segmentation

According to [9], TongueNet was proposed by utilizing U-Net with morphological processing layer to segment the tongue images. The morphological processing layer is a post-processing step to refine U-Net's output using hole-filling, open/close operations, and morphological reconstruction. This addresses common raw predictions challenges such as noise, boundary irregularities, and internal holes. By leveraging this approach, TongueNet outperforms the other state-of-the-art segmentation methods such as Snake and Flood Fill by achieving 98.45% mean pixel accuracy and 93.11% mean Intersection-over-Union. However, the pixel prediction accuracy requires further refinement. This is because sample 11 demonstrated a lower recall value from TongueNet (97.94%) as compared to Region-based and Edge-based fusion (REF) approach (99.77%). This discrepancy indicates that TongueNet cannot consistently capture all pixels belonging to tongue regions.

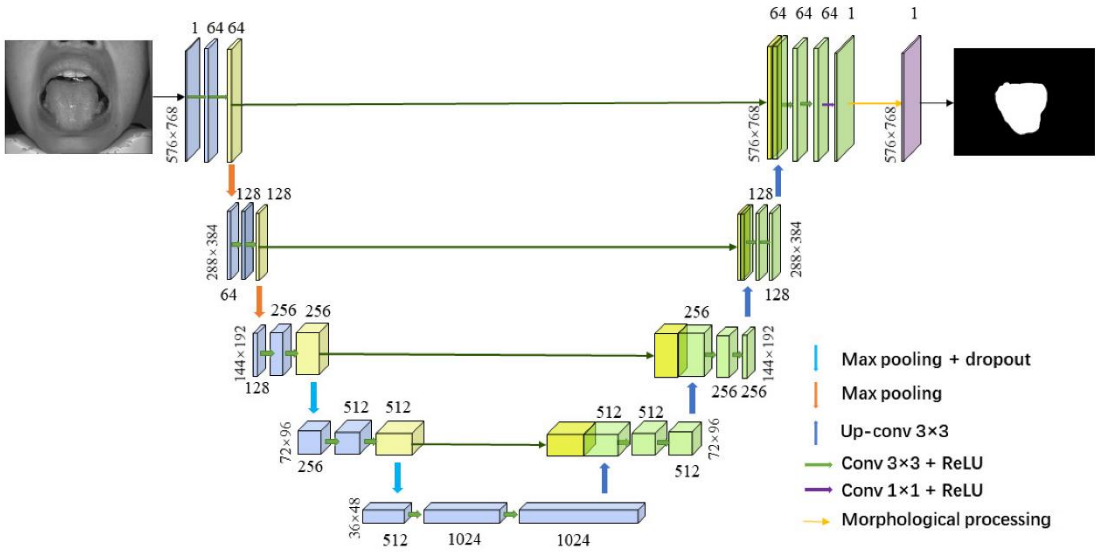


Figure 2.1 TongueNet Architecture [9]

Besides that, L. Yao et al. [10] introduced HPA-UNet model, which builds upon the U-Net architecture by incorporating a Hybrid Post-Processing Attention mechanism and an edge refinement strategy to improve segmentation performance. HPA-UNet enhances the standard U-Net by integrating Convolutional Block Attention Module (CBAM) that consists of both channel and spatial attention mechanisms. This design helps the model focus on important regions of the tongue while filtering out unnecessary background information. The spatial attention mechanism is applied in the downsampling path, while channel attention is used in the upsampling path, ensuring an optimal balance between feature representation and segmentation precision. On the post-processing side, the authors use a modified Sobel operator that extracts edge features from RGB channels instead of grayscale images. A lightweight segmentation model called Lite Reduced Atrous Spatial Pyramid Pooling (LR-ASPP) is then trained separately on these enhanced edge features. The results from the secondary network are then merged with the initial model predictions.

Results showed that HPA-UNet outperformed baseline models such as U-Net, U-Net++, ResUNet, and ResUNet++. With data augmentation and post-processing, HPA-UNet achieved a Mean IoU of 99.3% on the BioHit dataset and 98.4% on the HanYue-TongueSeg dataset.

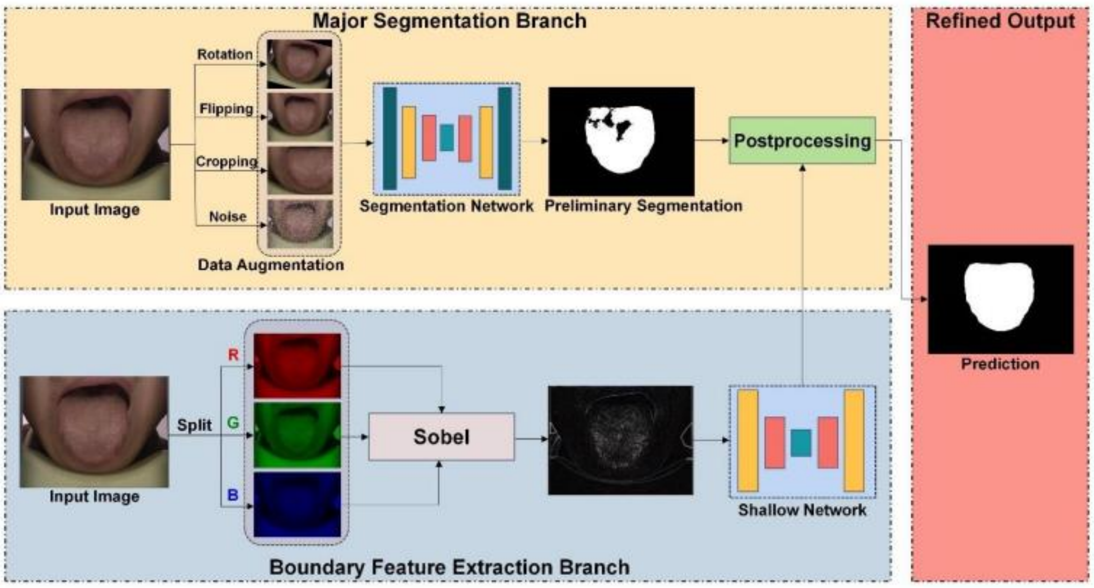


Figure 2.2 General Architecture of HPA-UNet [10]

Qu et al. [11] proposed tongue segmentation using SegNet and incorporate with an image quality evaluation method based on brightness statistics to determine whether an input image should be segmented. The model outperformed the traditional tongue image segmentation methods by achieving mean IoU scores of 95.89% and 90.72% respectively on TongueDataset1 and TongueDataset2. However, the segmentation performance declines when applied to images from open environments with uncontrolled lighting conditions as observed in TongueDataset2. This suggests that the model's robustness to extreme lighting variations needs improvement.

According to the research [12], the researchers propose an improved tongue image segmentation algorithm based on the DeeplabV3+ framework. The enhancements focus on three key areas: network structure optimization, a contour-based weighted cross-entropy (CBWCE) loss function, and a post-processing module. The network structure is refined by adjusting the atrous convolution scales in the ASPP module to better extract multi-scale information. Additionally, low-level features at different resolutions are incorporated to improve edge detection. To address segmentation errors, the CBWCE loss function assigns greater weight to pixels near tongue edges, ensuring the model prioritizes accurate boundary delineation. Lastly, a post-processing module leverages prior knowledge of tongue shape and connectivity to eliminate small, misclassified regions.

The experimental results demonstrate the effectiveness of these improvements. The proposed model achieves a mean intersection over union (MIOU) of 99.13%, surpassing baseline deep learning methods such as U-Net, SegNet, and PSPNet. The introduction of CBWCE and post-processing steps significantly reduces segmentation errors especially in complex cases that involves unclear tongue-lip boundaries or small background misclassifications.

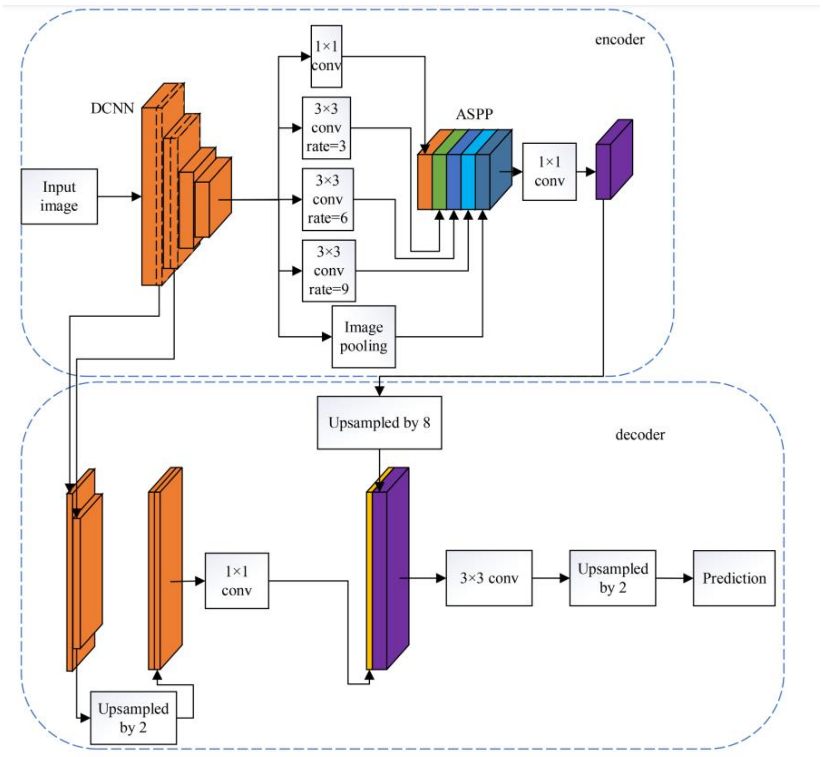


Figure 2.3 Enhanced DeeplabV3+ Structure [12]

Cao et al. [13] proposed TongueSAM, a universal tongue segmentation model that integrates Segment Anything Model (SAM) with a Prompt Generator based on object detection. This integration automates the generation of bounding box prompts so that the system able to deliver end-to-end segmentation without manual input. The study demonstrated that TongueSAM not only preserved SAM's generalization strengths but also optimized it for tongue-specific tasks by achieving robust performance even under challenging background. TongueSAM outperformed traditional deep learning methods and maintained superior segmentation accuracy under zero-shot conditions with mean Intersection over Union scores exceeding 95%.

Tang et al. [14] proposed a significant improvement to the DeepLabV3+ frameworkby integrating three key innovations. First, the backbone is replaced with MobileNet, which is a lightweight network that reduces parameter size and floating-point operation. Second, the authors enhance the ASPP module by embedding a Convolutional Block Attention Module (CBAM) into the branches so that better channel and spatial attention for richer semantic information extraction. Third, incorporate PointRend module to refine segmentation output. The experiment results demonstrate that the improved model not only outperforms classical architectures such as FCN, U-Net, PSPNet and standard DeepLab variants but also achieves substantial reductions of approximately 90% in parameter size.

2.2 Critical Remarks of Previous Works

Previous studies on tongue image segmentation have explored the deep learning-based approaches with its own strengths and limitations. Deep learning-based approaches have significantly improved segmentation performance by automatically learning hierarchical features from tongue images. Models such as U-Net are widely adopted due to their encoder-decoder structure, which enhances spatial localization and enables pixel-level segmentation. However, many of these models rely heavily on post-processing techniques, such as morphological processing layer and contour refinement algorithms to enhance segmentation accuracy. This dependency increases computational costs and training time, limiting the feasibility of real-time applications.

To address these limitations, this study proposes a hybrid Mobile U-Net model that reduces dependency on post-processing while improving segmentation accuracy. By leveraging U-Net's spatial localization strength and integrating MobileNet's lightweight efficient backbone, the proposed model enhances feature propagation and captures complex tongue patterns more effectively. The hybrid approach ensures end-to-end learning, optimizing both segmentation precision and computational efficiency.

2.3 Comparison of Previous Work (Segmentation)

Table 2.1 Comparison of Previous Work (Segmentation)

Author	Segmentation Approaches	Performance	Strength	Limitation
J. Zhou et al. [9]	U-Net with morphological processing layer	98.45% MPA and 93.11% mIoU	The model requires only 0.267 seconds per image to segment.	The model occasionally struggles with precise boundary delineation, leading to slight misclassification of edge pixels.
L. Yao et al. [10]	U-Net with Hybrid Post-Processing Attention mechanism and edge refinement strategy	99.3% mIoU on the BioHit dataset and 98.4% on the HanYue-TongueSeg dataset	The post-processing edge refinement module is independent, it can be applied to other segmentation networks beyond HPA-UNet.	The post-processing step relies on accurately labeled edge features.
P. Qu et al. [11]	SegNet with an image quality evaluation method based on brightness statistics	mIoU of 95.89% and 90.72% respectively on TongueDataset1 and TongueDataset2	Use of a large and diverse dataset ensures robustness, making the model more adaptable to real-world variations in image quality, illumination, and tongue morphology.	Performance declines when applied to images from open environments with uncontrolled lighting conditions.
X. Zhang et al. [12]	DeeplabV3+ with enhanced: network structure, optimization, CBWCE loss function, and a post-processing module	99.13% mIoU	Contour-Based Weighted Cross-Entropy (CBWCE) Loss Function places higher emphasis on tongue edges, thus reducing segmentation errors near the lips and teeth.	It cannot completely eliminate errors when tongue edges have significant distortions or irregularities.

Cao et al. [13]	TongueSAM	98.06% mIoU on BioHit while 97.85% on PaddlePaddle Dataset	Zero-shot capability that can achieve high accuracy even on unseen dataset without additional training.	Heavy computational resources required due to SAM is a large Transformer-based model that is pretrained on massive datasets.
Tang et al. [14]	DeepLabV3+ with MobileNet backbone, CBAM in ASPP module, PointRend	96.24 mIou on BioHit Dataset	Lightweight and efficient that have approximately 90% smaller parameter size compared to SOTA models.	Performance on the self-built dataset was weaker than on the public BioHit dataset, largely due to lower image resolution and cropping issues in clinical data acquisition.

2.4 Tongue Feature Recognition

Tiryaki et al. [3] introduced an objective and automated deep convolutional neural networks (DCNN) framework to distinguish between heathy tongues and four types of tongue lesions which are fissured tongue, coated tongue, geographic tongue and median rhomboid glossitis. Given the moderate size of the dataset and imbalance class nature, the researchers utilized transfer learning strategies where pre-trained models such as VGG19, ResNet50, ResNet101 and GoogleNet, are employed as the backbone for classification tasks. A unique feature of the study is the application of Fusion Based Majority Voting (FBMV) approach that was introduced. This method aggregates predictions from multiple DCNN to enhance overall classification robustness. In the binary classification task, ResNet101 achieved the highest individual accuracy of 93.53% which improved to 95.15% with FBMV approach. On the other hand, VGG19 led with an accuracy of 83.93% and was elevated to 88.76% using the FBMV method in the five-class classification task.

Zhong et al. [4] presents a novel application of deep learning to distinguish stained from non-stained tongue coatings. The researchers processed the tongue images by using GCYTD algorithm for tongue detection and DeepLabV3+ for segmentation to ensure clean tongue regions were analyzed. Then, ResNet50 architecture was employed for image classification due to its strong feature extraction capabilities and efficiency in training. The results demonstrated outstanding performance with an accuracy of 92%, 91% recall, and F1 score of 92%. The model also achieved high area under the curve (AUC) scores for both ROC (0.97) and PR (0.95) curves, indicating excellent classification capabilities. Most notably, the model outperformed three experienced TCM practitioners, who achieved an average diagnostic accuracy of only 69% when evaluating the same images.

Zhuang et al. [15] proposed a portable and human-computer interaction-based tongue diagnostic instrument in the study. The researchers used ResNet34, a residual neural network that was known for its depth and ability to avoid vanishing gradient, for classifying tongue features specifically teeth-printed tongues. The architecture outperforms the classical VGG16 model that is used as a baseline comparator by achieving the performance of 91.67% accuracy, 87.59% sensitivity and 95.39% specificity. The study also introduces a visual question answering component that merges computer vision with natural language processing. This multimodal approach allows the model to provide interpretable answers to user's health-related questions, therefore enhancing human-computer interaction and user experience.

Jiang et al. [16] leverage Faster Region-Based Convolutional Neural Network (Faster R-CNN) which is capable of simultaneously recognizing multiple tongue features within a single image in the research. The researchers utilized ResNet101 as the backbone network and trained the model on a large dataset of 8,676 expertly labeled tongue images that cover seven diagnostic categories. The proposed model achieved impressive performance with an average accuracy of 90.67%, precision of 99.28%, recall of 91.25%, and F1-score of 95.00%. Beyond the model development, the authors extended their work to real-world clinical application by analyzing 3,601 tongue images from individuals undergoing routine medical checkups. The analysis revealed gender and age-related differences in the prevalence of tongue features and established associations between specific tongue features and common metabolic disorders. For instance, fissured and tooth-marked tongues along with greasy coatings showed strong correlations with hypertension, dyslipidemia, overweight, and nonalcoholic fatty liver disease (NAFLD), which support traditional diagnostic claims in TCM with modern statistical validation.

The study by Wang et al. [17] addresses the need for an objective and automated system to evaluate greasy tongue coatings that were clinically significant in diagnosing various diseases such as gastroenteropathy and COVID-19. The authors developed and validated a deep learning model named GreasyCoatNet, that was built on ResNet architecture. The study evaluated three versions of GreasyCoatNet, which are GreasyCoatNet18, GreasyCoatNet34, GreasyCoatNet50 and identified that GreasyCoatNet34 as the most balanced and effective model. The model achieved 88.0% classification accuracy and 0.947 AUC in distinguishing among non-greasy, greasy and thick greasy coatings. The pretrained GreasyCoatNet was fine-tuned using a smaller set of tongue images from COVID-19 patients and healthy controls. The fine-tuned model outperformed randomly initialized networks in classifying COVID-related tongue features, achieving higher scores in accuracy, F1, and AUC.

2.5 Explainable Artificial Intelligence

Recent advancements in medical image classification using Convolutional Neural Networks have brough about high-performance diagnostic tools. However, the black box nature of deep learning models remains a significant barrier to clinical acceptance. Therefore, the literature on explainable AI (XAI) in medical imaging emphasizes the need for transparency in model decision-making.

The study by Cui et al. [18] employed techniques such as Grad-CAM, Grad-CAM++, Eigen-CAM, and Ablation-CAM to generate heatmaps that identify influential regions in input images. However, these methods are often qualitative and subject to subjective interpretation. The researchers address the gap by applying quantitative interpretability metrics which is the Heatmap Assisted Accuracy Score (HAAS) to evaluate the effectiveness of the heatmaps that were generated by CNN-based models' predictions. HAAS stands out as a robust and automated metric that evaluates the consistency of model predictions after adjusting images based on attribution maps and offering a model-centric alternative to annotation-dependent evaluations.

Wu et al. [19] integrates the explainable artificial intelligence with TCM tongue diagnosis to address a significant gap in the interpretability on existing deep learning-based diagnostic tools. The research compares and evaluates three types of diagnostic models which are convolutional neural networks, Transformer-based networks and feature-fusion approach that integrates colour, morphology and texture features of the tongue. Among the models, the feature-fusion model and ResNet-based CNN demonstrated superior classification performance on a dataset that categorized into four TCM syndromes. The authors employed ProtoPNet and Grad-CAM visualization techniques so that the identification of syndrome-specific tongue features and neural network attention areas is enhanced. The study found that ResNet yielded more clinically meaningful attention maps compared to VGG, which often misfocused on irrelevant image regions. Besides that, the study also revealed that neural networks can identify prototypical tongue images that align with traditional descriptions of TCM syndromes such as thin coating and pale colour for liver depression and spleen deficiency.

Musthafa et al. [20] integrate ResNet50 with Grad-CAM to both optimize detection performance and provide visual explanations for brain tumors prediction. The authors report a significant advancement over previous work by achieving testing accuracy of 98.52% and F1-score exceeding 98%. The model's performance surpasses that of several baseline studies, which generally report accuracies in the range of 91-97%. Importantly, the Grad-CAM visualizations enabled clinicians to confirm that the model was focusing on clinically relevant tumor regions.

The research by Umair et al. [21] studied on AI-assisted diagnostic methods for COVID-19 detection using medical imaging, specifically chest X-rays. The authors implemented a transfer learning approach with fine-tuning on four pretrained CNN models which are VGG16, ResNet50, MobileNet and DenseNet121 to classify chest X-ray images into COVID-19 positive and normal categories. Besides that, a significant methodological feature of the study is the application of the Grad-CAM visualization techniques which highlight the image regions of most influential to the model's decision-making. Among the tested models, DenseNet121 achieved the highest accuracy of 96.49%, followed closely by MobileNet (96.48%) and outperformed the popular COVID-Net benchmark (93.3%). The study further explores the impact of optimizers and hyperparameters such as batch size and learning rate. The findings show that RMSprop and a batch size of 32 is the optimal choice for the training hyperparameters.

2.6 Critical Remarks of Previous Work

Although previous studies have made notable contributions to automated tongue diagnosis using deep learning, several limitations remain when compared to the need of a more generalizable and clinically relevant solution. Tiryaki et al. [3] utilized fusion-based ensemble techniques to classify health versus diseased tongues, yet the reliance on majority voting will increase computational overhead. Besides that, Jiang et al. [16] developed a strong detection-based model that is capable of recognizing multiple tongue features with high accuracy, but the model's complexity and lack of explainability tools such as heatmaps or visual rationales limit its clinical transparency. Furthermore, only a few studies have prioritized model efficiency or suitability for deployment in settings with limited computational resources across these works. In comparison, our study introduces a lightweight model that requires low computational cost while maintaining strong classification performance. It also emphasizes model interpretability using explainable AI techniques such as Grad-CAM, therefore supporting transparent decision-making for TCM applications.

2.7 Comparison of Previous Work (Classification and Explainable AI)

Table 2.2 Comparison of Previous Work (Classification and Explainable AI)

Author	Network Architecture	XAI Techniques	Strength	Limitation
Tiryaki et al. [3]	VGG19, ResNet50, ResNet101, GoogleNet	-	Fusion Based Majority Voting that combines the strengths of multiple networks.	Small dataset size in each class, which may limit the model's generalizability.
Zhong et al. [4]	ResNet50	-	Include comparison between model and experienced TCM practitioners as a real-world benchmark.	Lack of model comparison, only ResNet50 was used in the study.
Zhuang et al. [15]	ResNet34	-	Combining deep learning with Visual Question Answering to allow users to interact and receive health suggestions from the system.	Limited tongue feature categories, features such as tongue fissures were not explored in detail.

Jiang et al. [16]	Faster R-CNN with ResNet101 backbone	-	Multi-label classification which enables multiple tongue features classification in one image.	Did not compare the results to other deep learning architecture such as YOLO.
Wang et al. [17]	GreasyCoatNet18, GreasyCoatNet34, GreasyCoatNet50	Grad-CAM	Effective use of transfer learning by fine-tuning pretrained GreasyCoatNet on a smaller COVID-19 dataset.	Imbalance distribution of classes in training data, only 85 for non-greasy class while 759 for greasy class.
Cui et al. [18]	ResNet18, ResNet50, VGG19, AlexNet	Grad-CAM, Grad-CAM++, Eigen-CAM, and Ablation-CAM	Integration of quantitative interpretability (HAAS) that quantifies the alignment between model attention and predictive accuracy.	Binary classification scope where the task was limited to distinguish stained from non-stained tongue coatings.
Wu et al. [19]	AlexNet, DenseNet, EfficientNet, EfficientNetV2, MobileNet, GoogleNet, RegNet, ResNet, VGG, ShuffleNet, Vision	ProtoPNet, Grad-CAM	Comparative model analysis that compares CNNs, Transformer-based and fusion model.	The overall accuracy and robustness may still be insufficient for clinical deployment.

	Transformer, Swin Transformer, Fusion Diagnosis			
Musthafa et al. [20]	EfficientNetB0, DenseNet201, Inception, Xception, MobileNet, ResNet50	Grad-CAM	The use of epoch-wise Grad-CAM visualizations offer deeper insights into the model's learning progression.	The high accuracy on the validation set (100%) suggests a possible risk of overfitting.
Umair et al. [21]	MobileNet, VGG16, DenseNet121, ResNet50	Grad-CAM	A systematic evaluation of learning rates, batch sizes and optimizers were conducted.	Did not include other lung diseases such as pneumonia which are often visually similar to COVID-19 in X-rays

Chapter 3
System Methodology/Approach

3.1 Design Specification

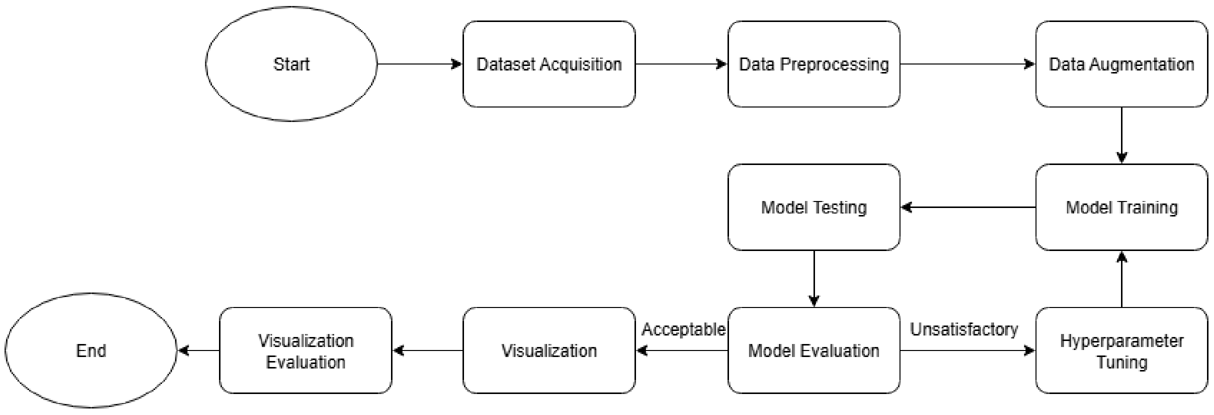


Figure 3.1 Model Development Flowchart

Figure 3.1 describes the model development flowchart to create and optimize the deep learning models. First, the dataset will be collected for image classification task. It is important to collect a good quality tongue image with high resolution, minimal noise and relevant content. This is because the quality of images will affect the performance of the models by having poor generalization and difficulty in learning meaningful features. Data preprocessing is carried out after dataset acquisition. The purpose of data preprocessing is to eliminate unwanted parts, normalize and convert the image into same size of arrays so that it can be used during the model training. After that, data augmentation techniques are applied to enhance the model generalization. The dataset diversity is increased by introducing the transformations such as rotation, flipping, scaling and shearing to the original dataset.

During the model training phase, the model will learn to segment the training data by adjusting the parameters such as weights using optimization techniques iteratively and validate the performance with validation dataset. Then, the model is being tested on the testing dataset which is unseen during the model training phase. Performance metrics such as accuracy and F1-Score will be used to evaluate the performance of the model. Furthermore, hyperparameter tuning such as learning rate and batch size is performed to further refine the model's performance. The iterative cycle for model training, testing and evaluation is conducted until the model achieves optimal performance.

Finally, visual outputs such as heatmaps that highlight the important regions in the input image that contributed to the model's prediction will be generated. These generated heatmap will then be evaluated and compared using visualization benchmarks to assess the quality and effectiveness of the explanation provided.

3.2 System Requirements

3.2.1 Hardware

Table 3.1 Laptop Specifications

Description	**Specifications**
Processor	Intel Core i7-7700HQ
Operating System	System Windows 10
Graphic Card	NVIDIA GeForce GTX 1050 4GB GDDR5
Memory	12GB DDR4 RAM

3.2.2 Software

Table 3.2 Software Specifications

Description	**Software**
Programming Language	Pyhton 3.11.11
Library	TensorFlow 2.18.0, Keras 3.8.0
IDE	Google Colab with T4 GPU

Python is used as the main programming language in this project because it offers various built-in libraries such as TensorFlow or PyTorch library for developing deep learning models and image processing. Besides that, the syntax from Python is also easier to read and understand thus easing the development process.

TensorFlow is an open-source platform developed by Google for building machine learning and deep learning models while Keras is a high-level neural networks API that is written in Python and is capable to run on top of TensorFlow. It simplifies the model building process and therefore allows quick development and experimentation for deep learning models in this project.

Google Colab are the main integrated development environment (IDE) used in the development phase. Google Colab is a web-based IDE hosted by Google where Python code can be written on it in a Jupyter Notebook format. It does not require any setup and provide free access to GPU and TPU to train the deep learning model.

3.3 Dataset

3.3.1 Segmentation

BioHit Dataset

The dataset used in this research is BioHit Tongue Dataset, which was developed by the Harbin Institute of Technology [22]. It is a publicly available and widely utilized dataset for tongue segmentation. The dataset consists of 300 RGB tongue images with a resolution of 768x576 pixels and paired with manually annotated masks as the ground truth. The images were captured in a semi-enclosed setting under consistent lighting conditions. In this study, the dataset is being split into training, validation and testing set with the ratio of 7:2:1 respectively.

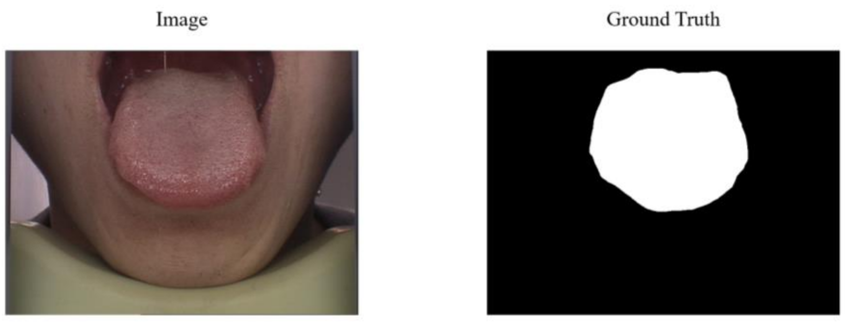


Figure 3.2 Sample Image and Its Ground Truth from BioHit Dataset

Roboflow Dataset

The second dataset was obtained from [23] which consists of 2,500 tongue images. There are around 500 images were captured under consistent lighting conditions while the remaining images were taken in open environments with unstable lighting. The images are 640x640 resolution and are provided in JPG format while the masks are in PNG format. The ground truth masks are labeled with binary labeling convention which is 0 for the background and 1 for the region of interest. The sample of the image and its preprocessed ground truth from the dataset is shown in Figure 3.3 and 3.4.

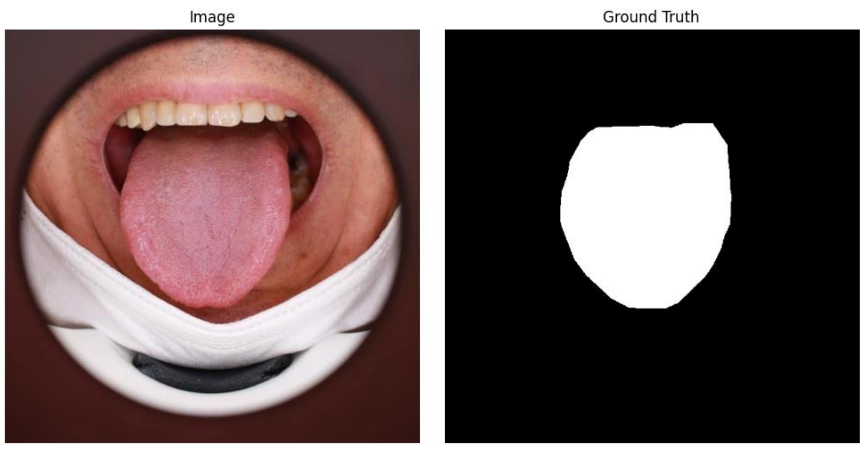


Figure 3.3 Sample of Consistent Lighting Tongue Image

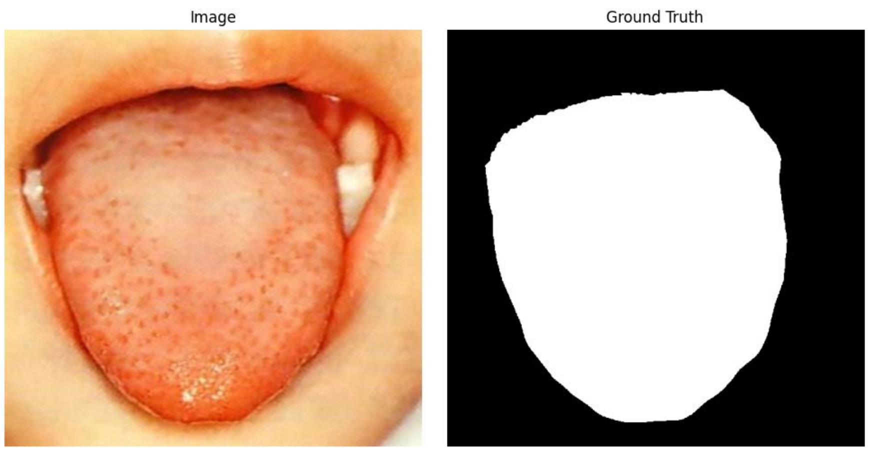


Figure 3.4 Sample of Open Environment Tongue Image

PaddlePaddle Dataset

The third dataset is a dataset that consists of 1,000 images selected from [24] with manual segmentation performed sing the Labelme tool by the author referenced in [13]. This collection includes a diverse array of tongue images captured various conditions including mobile devices and unconventional angles. The original images were resized to 400x400 resolution by the author to maintain uniform input size as the original images vary in resolution. The ground truth mask also appears as entirely black as the second dataset but however the pixel labeled [1,1,1] as tongue regions while [0,0,0] denotes the background. Figure 3.5 shows the sample of the tongue images from the third dataset.

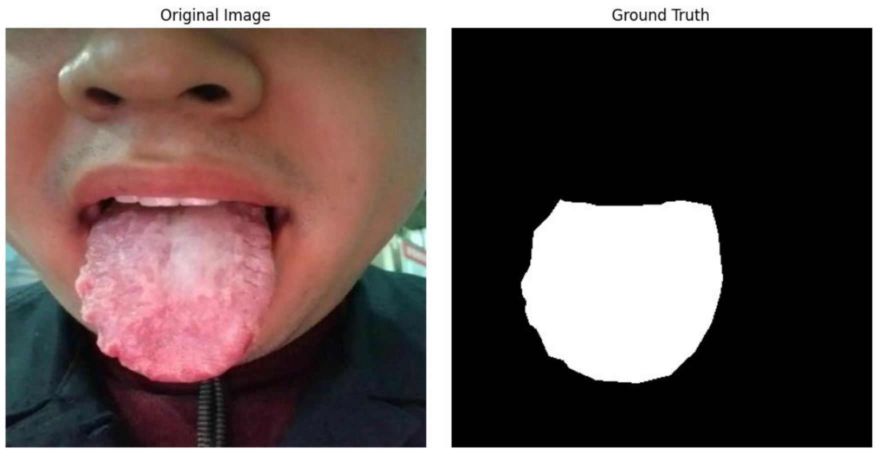


Figure 3.5 Sample of Tongue Image

3.3.2 Binary Classification

The dataset used for binary classification task is an open-source tongue dataset [25]. It consists of a total of 2,008 tongues images, which are divided into two balance categories:

Table 3.3 Distribution of Tongue Images in Binary Classification

Class	**Total**	**Proportion (%)**
Stained tongue coating	1,001	49.85
Non-stained tongue coating	1,007	50.15

The stained tongue coating images are collected from healthy student volunteers and the non-stained tongue coating images are collected from hospitalized patients that are diagnosed with conditions such as lung cancer, diabetes, and hypertension. The stained tongue images were captured at 5, 10, and 30 minutes after the participants consumed stained foods such as milk, mango and coffee in a fasting condition to control the confounding factors.

The dataset is split into training, validation and testing set with the ratio of 7:2:1 respectively during the model development. Furthermore, all images were uniformly resized to 384 x 384 pixels to ensure consistency throughout the study.

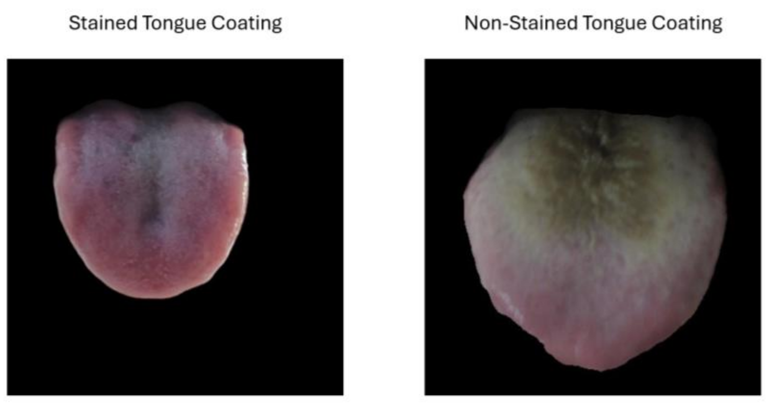


Figure 3.6 Sample of Tongue Image in Each Class (Binary)

3.3.3 Multi-Class Classification

The dataset used for multi-class classification is a proprietary dataset that was obtained via a commercial vendor at Taobao. It consists of a total of 1,501 images that are divided into 5 classes which are mirror-approximated, white-greasy, thin-white, yellow-greasy and grey-black tongue. The distribution of the tongue images for each class is shown at Table 3.4.

Table 3.4 Distribution of Tongue Images in Multi-Class Classification

Class	**Total**	**Proportion (%)**
Mirror-Approximated	122	8.13%
White-Greasy	698	46.50%
Thin-White	534	35.58%
Yellow-Greasy	91	6.06%
Grey-Black	56	3.73%

According to the study [26], the coating appearance of the tongue can be described as:

Table 3.5 Description of Tongue Coatings

Class	**Description**
Mirror-Approximated	No coating at tongue and appear as smooth, pink tongue surface.
White-Greasy	White in colour with thick, sticky and greasy coating.

Thin-White	A thin white layer of tongue, considered normal in healthy individuals.
Yellow-Greasy	Yellow-coloured coating with thick and greasy appearance.
Grey-Black	Dark grey to black tongue coating.

The dataset is split into training, validation and testing set with the ratio of 7:2:1 respectively during the model development. Furthermore, all images were uniformly resized to 384 x 384 pixels to ensure consistency throughout the study.

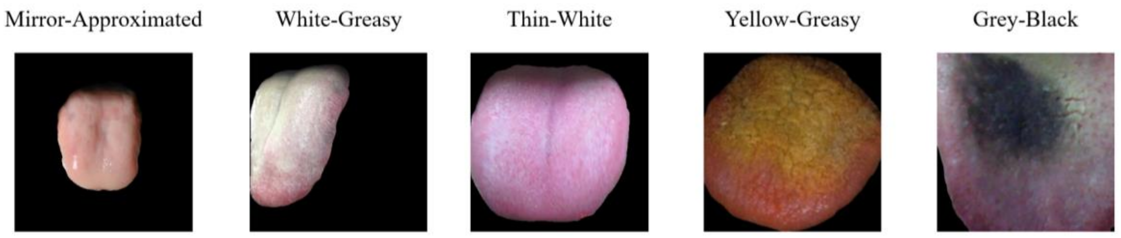


Figure 3.7 Sample of Tongue Image in Each Class (Multi-Class)

3.4 Model Architecture

3.4.1 LECA-EfficientNetV2

LECA-EfficientNetV2 is a lightweight deep learning model that was proposed in [27]. It is built upon the EfficientNetV2 architecture which is known for its optimized balance between performance and computational efficiency. The model incorporates a novel channel attention mechanism called Lightweight Efficient Channel Attention (LECA) module, that was designed to enhance feature extraction without increasing the computational burden. Unlike traditional attention modules like Squeeze-and-Excitation (SE) that use dimensionality reduction and can weaken feature learning, LECA avoids dimensionality reduction by leveraging a local cross-channel interaction strategy based on 1D convolutions. This allows it to preserve detailed information while adaptively weighting feature channels. The architecture of LECA-EfficientNetV2 closely follows the original structure of EfficientNetV2. The primary modification lies in the replacement of the SE modules with LECA module. Figure 3.8 shows the details of LECA module and Figure 3.9 shows the architecture of LECA-EfficientNetV2.

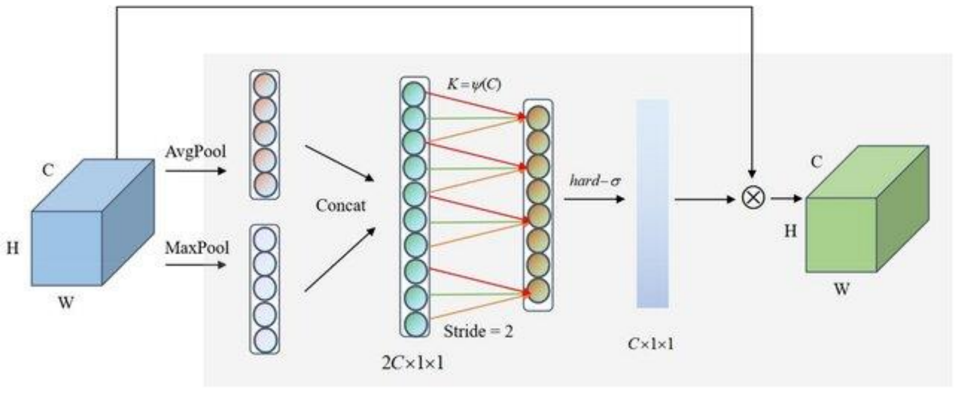


Figure 3.8 Details of LECA Module [27]

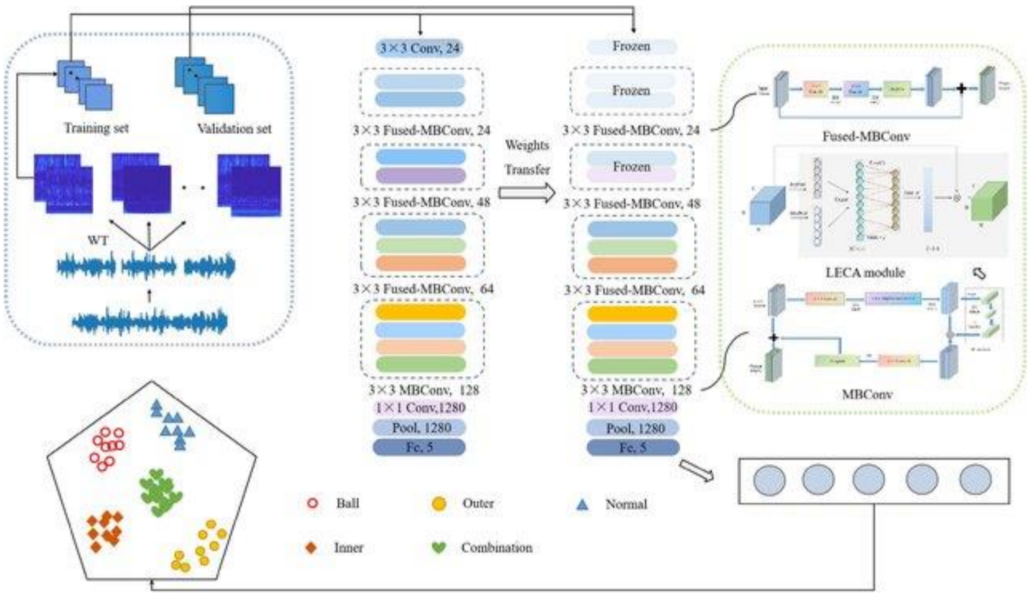


Figure 3.9 LECA-EfficientNetV2 Architecture [27]

3.4.2 Proposed Mobile U-Net

The model is a lightweight segmentation framework that combines the U-Net architecture [28] with MobileNetV2 [29] as its encoder backbone. The encoder utilizes a pretrained MobileNetV2 network to capture hierarchical image features while maintaining computational efficiency.

In the encoder, hierarchical features are extracted from different stages of MobileNetV2. These multi-scale features are then passed to the decoder through skip connections to preserve spatial details that are often lost during downsampling. The decoder progressively reconstructs the image resolution by psalming the encoded features and fusing them with the corresponding encoder features. This design enables the model to combine both high-level semantic information and fine-grained spatial details that are critical for accurate tongue segmentation.

In the decoder, the feature maps are progressively upsampled to reconstruct the spatial resolution of the input. An attention block is applied before concatenation at each skip connection to enhance the fusion of encoder and decoder features. The attention mechanism adaptively weights the encoder features to allow the network to emphasize the most relevant regions while suppressing irrelevant background information. This selective focus improves segmentation accuracy where boundaries may be ambiguous or corrupted by noise. Figure 3.10 shows the architecture of the Mobile U-Net with attention gate module.

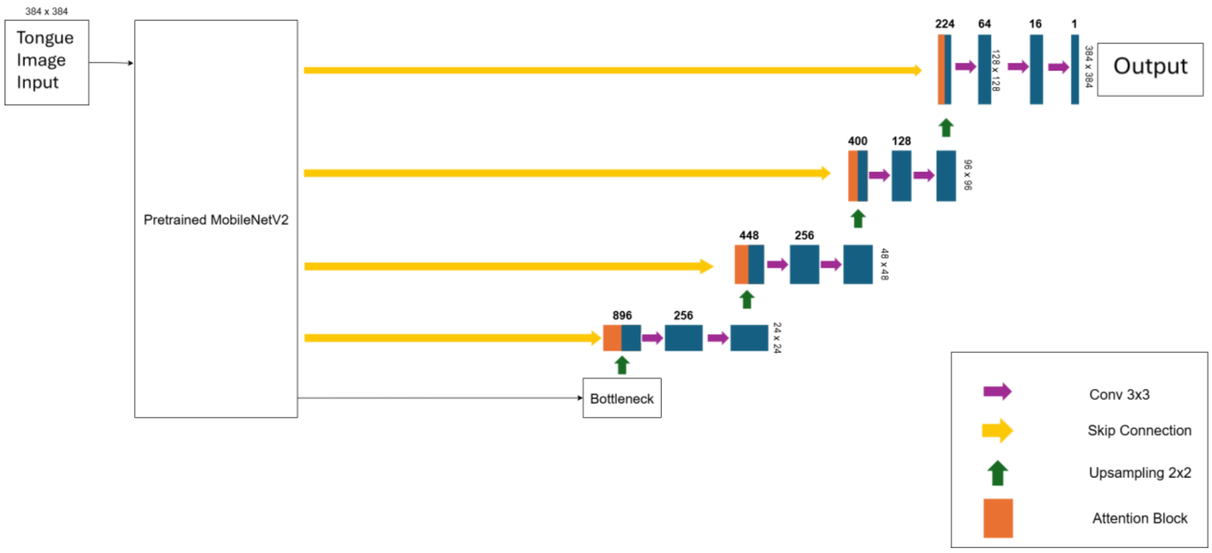


Figure 3.10 Mobile U-Net with Attention Gate Module

3.4.3 Proposed Mobile U-Net with Transformer

The second model enhances Mobile U-Net with Attention Gate model in section 3.4.1 by integrating a transformer module into the bottleneck stage. The original design relies primarily on convolutional operations, which are effective at capturing local spatial dependencies bt however these operations are limited in the ability to model long-range relationships. To address the limitation, a transformer bottleneck inspired by Vision Transformer (ViT) architectures [30] is introduced. This component enables the network to capture global contextual information across the entire feature map, which is beneficial for tongue segmentation tasks where local textures and global shape patterns must be considered simultaneously. Figure 3.11 shows the architecture of the model.

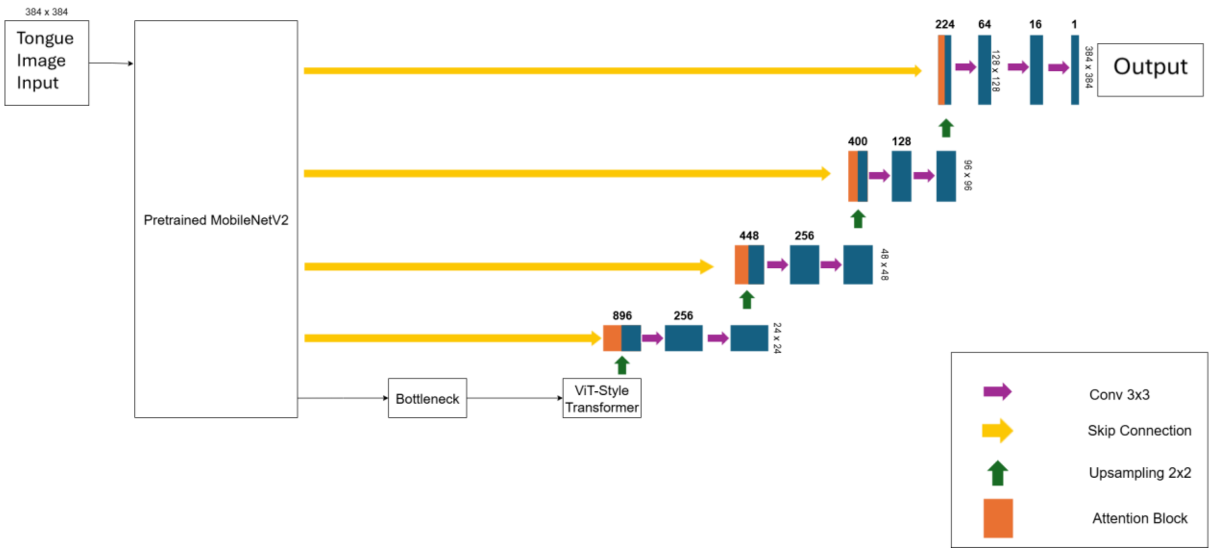


Figure 3.11 Mobile U-Net with Attention Gate and Transformer

3.4.4 Proposed Efficient-ResNet

The proposed model is a hybrid network that combines ResNet-style lightweight residual blocks with EfficientNetV2-inspired MBConv blocks for efficient feature extraction. To further improve channel-wise feature recalibration, the Lightweight Efficient Channel Attention (LECA) module in [27] is integrated within selected blocks. This hybrid design leverages the robust skip connections of ResNet and the parameter-efficient inverted bottleneck structure of EfficientNet so that the model is able to achieve a balance between accuracy and computational efficiency. Figure 3.12 shows the architecture along with the structure of the Reslite and MBConv block.

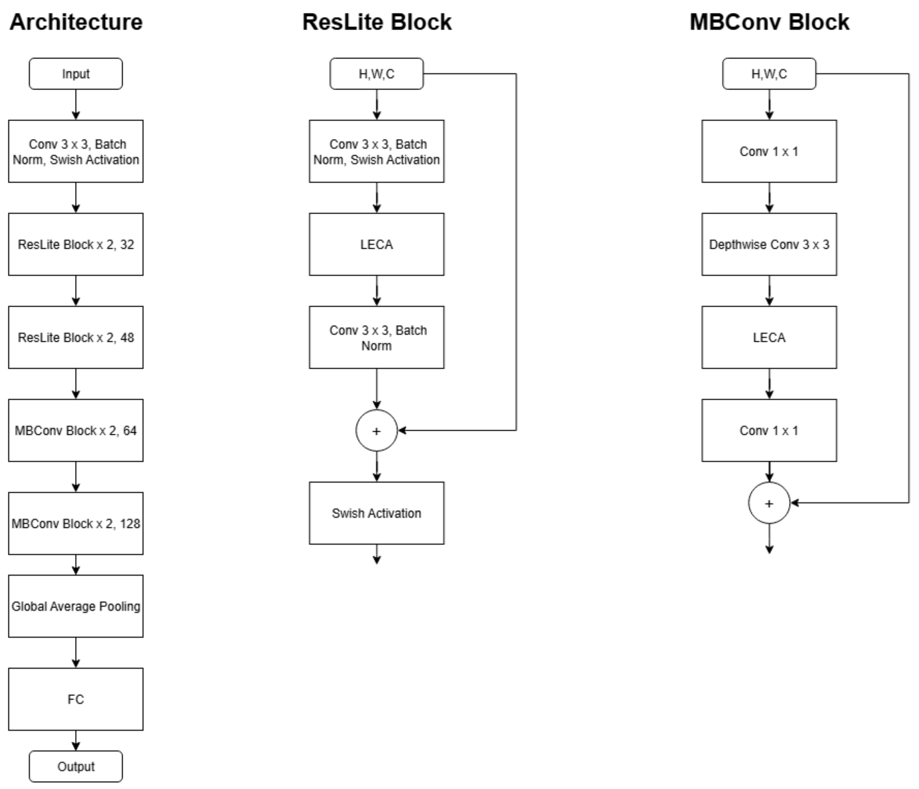


Figure 3.12 Architecture of Efficient-ResNet and Structure of Reslite and MBConv Block

3.5 Explainable Artificial Intelligence Techniques

3.5.1 Gradient-weighted Class Activation Mapping (Grad-CAM)

Grad-CAM is a popular visualization technique used to interpret the decision-making process of convolutional neural networks particularly in image classification tasks [31]. Grad-CAM works by computing the gradients of the output class with respect to the feature maps in the final convolutional layer. These gradients are used to produce a heatmap that highlights the spatial regions in the input image that contributes most significantly to the model’s prediction. By overlapping the heatmap onto the original image, Grad-CAM provides a visual explanation that helps researchers and practitioners understand which areas of the image the model is focusing on.

In the Grad-CAM heatmap, colours represent the importance or relevance of different regions of the input image. Red or yellow regions indicate high activation, which means the parts of the image are strongly influencing the model’s prediction for the chosen class. On the other hand, blue or dark areas represent low or no activation that suggests the regions had little to no contribution to the decision. By providing intuitive visual feedback, Grad-CAM enhances transparency and trust in deep learning models.

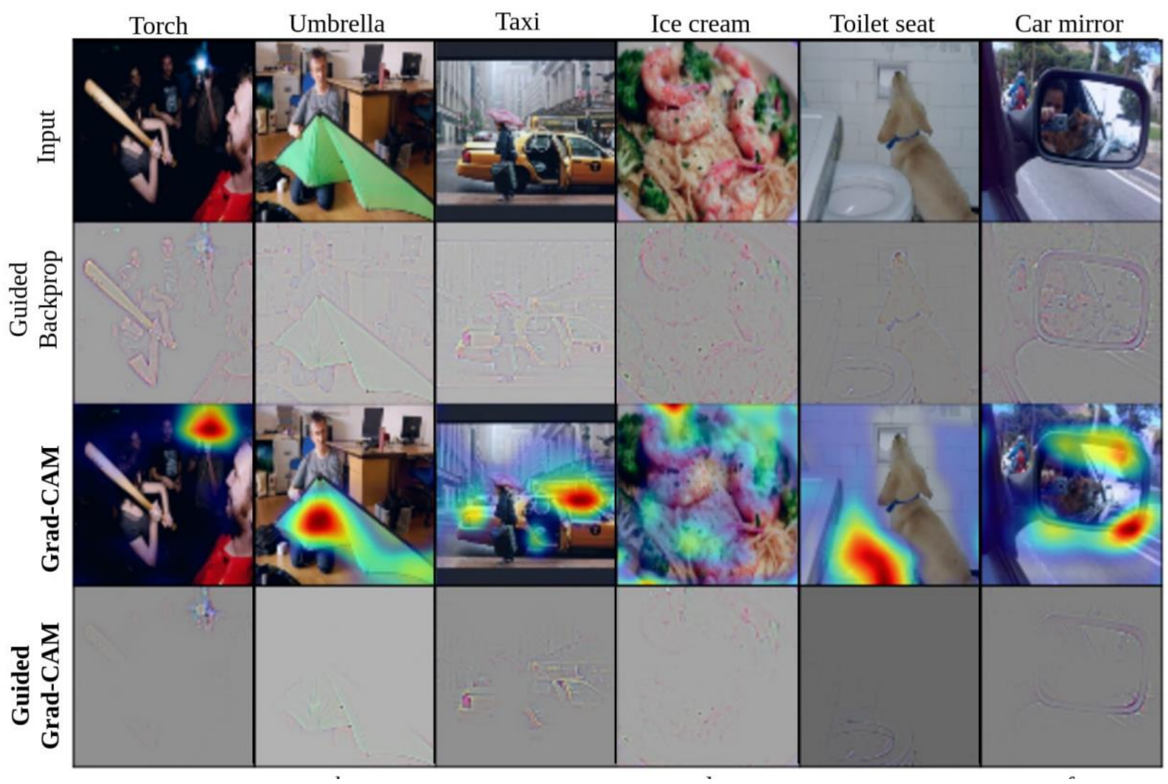


Figure 3.13 Visualization of Grad-CAM with Predicted Class [31]

3.6 Model Evaluation Techniques

3.6.1 Segmentation

The objective of the performance metrics is to evaluate the performance of the segmentation models by assessing the similarity between the predicted and ground truth segmentations. Several common evaluation metrics that are commonly used in medical image segmentation are referred from [32, 33] and are used in this study.

Mean Intersection over Union (MIoU)

The Mean Intersection over Union measures the average ratio of overlap between predicted and ground truth segmentations. It is commonly used to measure the accuracy of the model in image segmentation field.

$$MIoU = \frac{1}{k}\sum_{i=1}^{k}\frac{TP}{TP+FP+FN} \tag{3.1}$$

where:

k : Number of images segmented

TP : The overlapping area between the ground truth and predicted segmentation mask

FP : The area of predicted segmentation that extends beyond the ground truth

FN : The number of pixels in the ground truth that failed to predict

Mean Pixel Accuracy (MPA)

Pixel Accuracy represents the ratio of correctly predicted pixels to the total number of pixels. Mean Pixel Accuracy extends it by calculating the accuracy for average value across the segmentation.

$$MPA = \frac{1}{k}\sum_{i=1}^{k} \frac{TP + TN}{TP + TN + FP + FN} \tag{3.2}$$

where:

k : Number of images segmented

TP : The overlapping area between the ground truth and predicted segmentation mask

TN : The area that is correctly predicted that not belong to the region of interest

FP : The area of predicted segmentation that extends beyond the ground truth

FN : The number of pixels in the ground truth that failed to predict

Mean Dice Similarity Coefficient (MDSC)

The Dice Similarity Coefficient is defined as the harmonic mean between the sensitivity and precision. The difference between MDSC and MIoU is that MIoU imposes a stronger penalty on under-segmentation and over segmentation towards the images.

$$MDSC = \frac{1}{k}\sum_{i=1}^{k} \frac{2TP}{2TP + FP + FN} \tag{3.3}$$

where:

k : Number of images segmented

TP : The overlapping area between the ground truth and predicted segmentation mask

FP : The area of predicted segmentation that extends beyond the ground truth

FN : The number of pixels in the ground truth that failed to predict

3.6.2 Classification

The objective of performance metrics is to objectively evaluate how well a model can correctly identify and categorize images into their respective classes. These metrics provide insight into different aspects of the model's behavior. The performance metrics that are commonly used in medical image classification and benchmark for evaluating visual explanations are referred from [34, 35, 36] and are used in this study.

Accuracy

Accuracy is a fundamental performance metric that measures the overall correctness of the model by calculating the proportion of correctly classified images out of the total number of predictions. In tongue classification tasks, accuracy indicates how frequently the model correctly identifies the tongue type regardless of the specific class.

$$Accuracy = \frac{TP + TN}{TP + FP + TN + FN} \tag{3.4}$$

where:

True Positive (TP): The model correctly predicts the image that belongs to a specific class and in actual the image also belongs to the class. For example, the model predicts an image is “yellow-greasy” and the actual label is also “yellow-greasy”.

True Negative (TN): The model correctly predicts the image that does not belong to a specific class and it indeed does not belong to the class. For example, the model predicts an image is not “yellow-greasy” and the actual label is also not “yellow-greasy”.

False Positive (FP): The model incorrectly predicts that an image belongs to a class when it actually does not. For example, the model predicts an image is “yellow-greasy” but the actual label is not “yellow-greasy”.

False Negative (FN): The model incorrectly predicts the image does not belong to a specific class when it actually does. For example, the model predicts an image is not “yellow-greasy” but the actual label is “yellow-greasy”.

Precision

Precision measures the proportion of true positive predictions among all instances that the model predicted as a specific tongue type. In the context of tongue classification, high precision indicates that when the model classifies an image as a particular tongue category, it is likely to be correct.

$$Precision = \frac{TP}{TP + FP} \tag{3.5}$$

Recall

Recall is also known as sensitivity or the true positive rate. It refers to the proportion of actual positive cases that the model correctly identifies. In tongue classification, recall assess the model's ability to detect all images that truly belong to a given tongue type. A high recall value means that the model is effective in identifying most instances of that tongue category so that the risk of false negative is minimized. A high recall is important in medical diagnostics as a false negative will lead to untreated conditions.

$$Recall = \frac{TP}{TP + FN} \tag{3.6}$$

F1-Score

F1-Score is the harmonic means of precision and recall that provides a balance trade-off between false positive and false negative. In tongue classification, the F1-Score is particularly valuable when dealing with imbalanced datasets where certain tongue types may have fewer samples than others.

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3.7}$$

Confusion Matrix

Confusion matrix is a tabular representation of the classification results that summarizes the number of true positives, true negatives, false positives and false negatives for each class. This helps in identifying specific weaknesses in the model and guides further refinement in model training or data preparation.

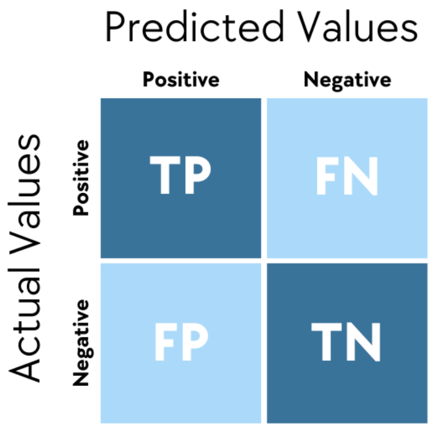


Figure 3.14 Confusion Matrix [31]

Pointing Game

Pointing Game is a metric used to evaluate the accuracy of visual explanations by assessing whether the most activated point in a Grad-CAM heatmap falls within the ground truth region of interest. If the pixel with the highest activation value in the Grad-CAM lies inside the ground truth mask, then the prediction is counted as "hit" otherwise "miss". The final score is calculated as the ratio of hits to the total number of evaluated images. This method provides a simple yet effective way to measure whether the model is focusing on the most relevant part of the image even if it does not highlight the entire region.

$$Pointing\ Game = \frac{\sum_{i=1}^{N} 1[MaxLoc(E_i) \in A_i]}{N} \qquad (3.6)$$

where:

N: Number of samples.

MaxLoc(E_i): The location of the highest activation in the Grad-CAM saliency map for the i-th image.

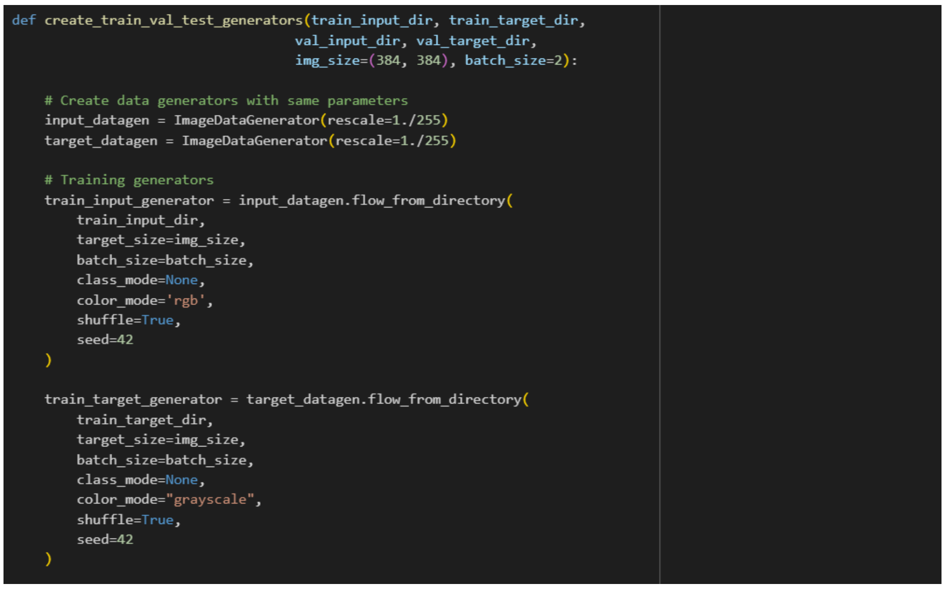
A_i: Ground truth mask for the i-th image.

Chapter 4

System Implementation

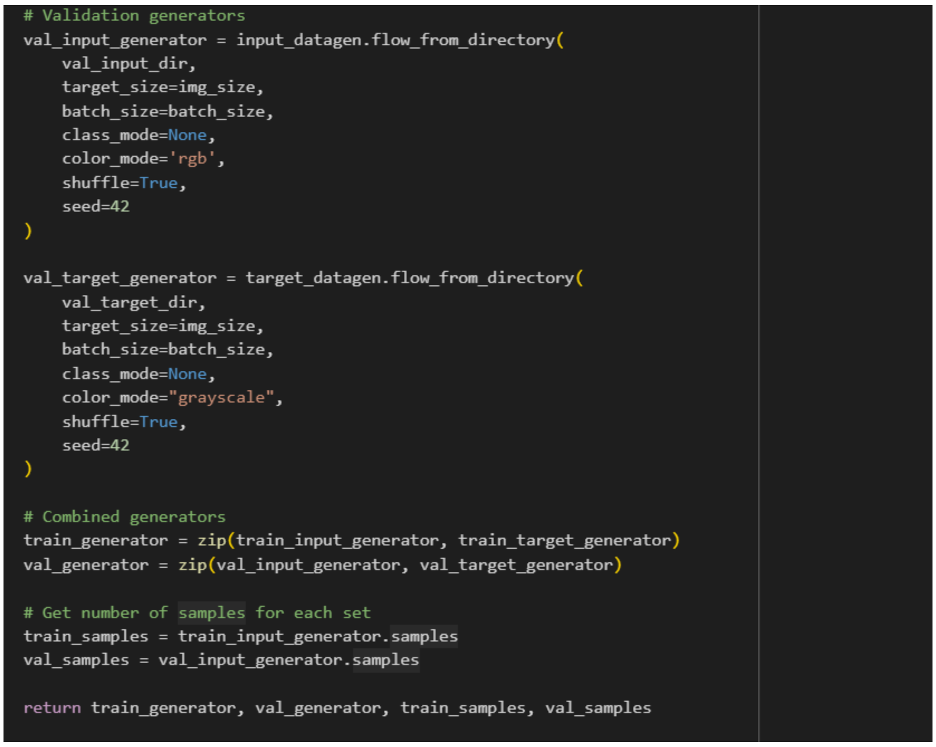
4.1 Segmentation

4.1.1 Data Preprocessing



```python
def create_train_val_test_generators(train_input_dir, train_target_dir,
                                     val_input_dir, val_target_dir,
                                     img_size=(384, 384), batch_size=2):

    # Create data generators with same parameters
    input_datagen = ImageDataGenerator(rescale=1./255)
    target_datagen = ImageDataGenerator(rescale=1./255)

    # Training generators
    train_input_generator = input_datagen.flow_from_directory(
        train_input_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode=None,
        color_mode='rgb',
        shuffle=True,
        seed=42
    )

    train_target_generator = target_datagen.flow_from_directory(
        train_target_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode=None,
        color_mode="grayscale",
        shuffle=True,
        seed=42
    )
```

Figure 4.1 Data Pipeline for Segmentation

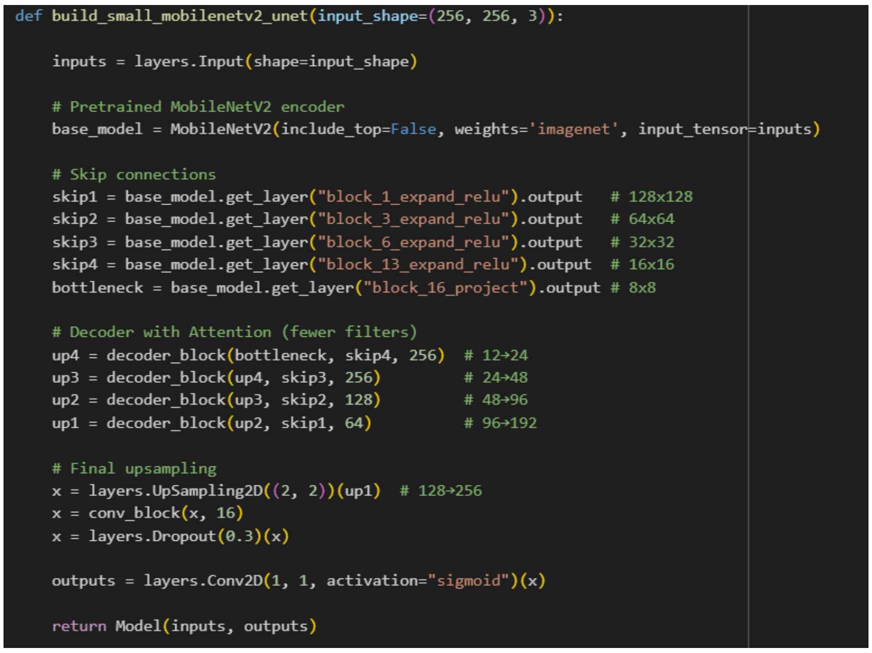


```python
    # Validation generators
    val_input_generator = input_datagen.flow_from_directory(
        val_input_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode=None,
        color_mode='rgb',
        shuffle=True,
        seed=42
    )

    val_target_generator = target_datagen.flow_from_directory(
        val_target_dir,
        target_size=img_size,
        batch_size=batch_size,
        class_mode=None,
        color_mode="grayscale",
        shuffle=True,
        seed=42
    )

    # Combined generators
    train_generator = zip(train_input_generator, train_target_generator)
    val_generator = zip(val_input_generator, val_target_generator)

    # Get number of samples for each set
    train_samples = train_input_generator.samples
    val_samples = val_input_generator.samples

    return train_generator, val_generator, train_samples, val_samples
```

Figure 4.2 Data Pipeline for Segmentation

Figure 4.1 and Figure 4.2 shows the function of the data pipeline. This function prepares training and validation data generators for image segmentation tasks using Keras' ImageDataGenerator. Both input images (RGB) and target masks (grayscale) are normalized to a [0,1] range. To maintain alignment between inputs and targets, the same random seed is applied during shuffling. The generators are combined using zip so that each batch yields paired (input, target) samples. Additionally, the function returns the total number of samples in the training and validation sets, which are required to define training steps during model fitting.

4.1.2 Modelling



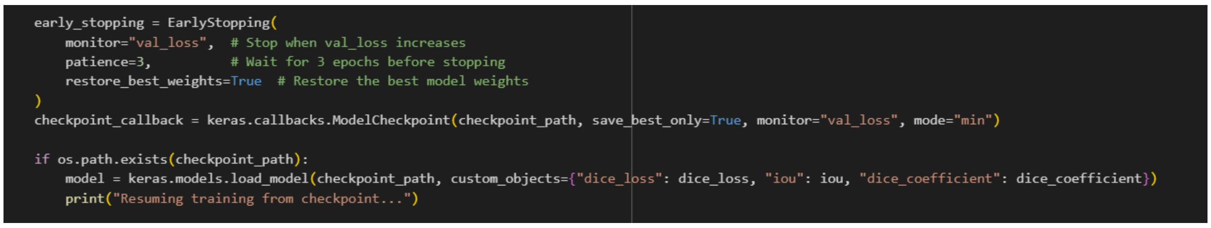
```
def build_small_mobilenetv2_unet(input_shape=(256, 256, 3)):

    inputs = layers.Input(shape=input_shape)

    # Pretrained MobileNetV2 encoder
    base_model = MobileNetV2(include_top=False, weights='imagenet', input_tensor=inputs)

    # Skip connections
    skip1 = base_model.get_layer("block_1_expand_relu").output   # 128x128
    skip2 = base_model.get_layer("block_3_expand_relu").output   # 64x64
    skip3 = base_model.get_layer("block_6_expand_relu").output   # 32x32
    skip4 = base_model.get_layer("block_13_expand_relu").output  # 16x16
    bottleneck = base_model.get_layer("block_16_project").output # 8x8

    # Decoder with Attention (fewer filters)
    up4 = decoder_block(bottleneck, skip4, 256)  # 12→24
    up3 = decoder_block(up4, skip3, 256)         # 24→48
    up2 = decoder_block(up3, skip2, 128)         # 48→96
    up1 = decoder_block(up2, skip1, 64)          # 96→192

    # Final upsampling
    x = layers.UpSampling2D((2, 2))(up1)  # 128→256
    x = conv_block(x, 16)
    x = layers.Dropout(0.3)(x)

    outputs = layers.Conv2D(1, 1, activation="sigmoid")(x)

    return Model(inputs, outputs)
```

Figure 4.3 Mobile U-Net Modelling

Figure 4.3 shows the function to model Mobile U-Net. The function uses pretrained MobileNetV2 as its encoder. It is initialized with pretrained ImageNet weights and the intermediate feature maps from selected layers are extracted as skip connection. The "block_16_project", which is the deepest feature representation serves as the bottleneck of the model so that it helps the network to learn abstract features from the tongue image.

The decoder reconstructs the segmentation map through a series of upsampling operations that are combined with skip connections. After the final upsampling, the output is passed through a convolutional block, a dropout layer for regularization and a final convolution layer with a sigmoid activation to generate the segmentation mask.

4.1.3 Early Stopping and Model Checkpoint Callbacks

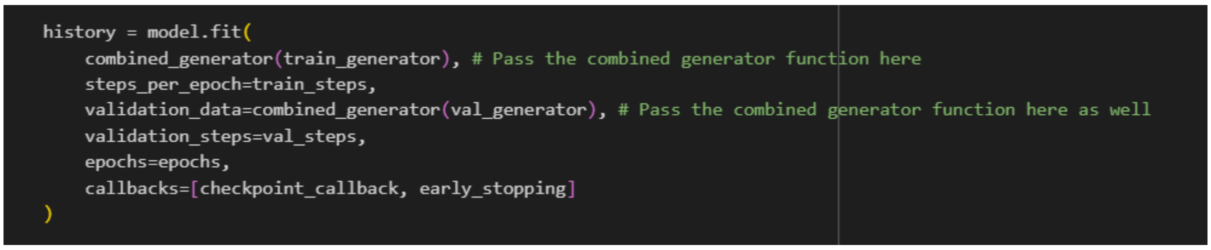


```
early_stopping = EarlyStopping(
    monitor="val_loss",  # Stop when val_loss increases
    patience=3,          # Wait for 3 epochs before stopping
    restore_best_weights=True  # Restore the best model weights
)
checkpoint_callback = keras.callbacks.ModelCheckpoint(checkpoint_path, save_best_only=True, monitor="val_loss", mode="min")

if os.path.exists(checkpoint_path):
    model = keras.models.load_model(checkpoint_path, custom_objects={"dice_loss": dice_loss, "iou": iou, "dice_coefficient": dice_coefficient})
    print("Resuming training from checkpoint...")
```

Figure 4.4 Early Stopping and Model Checkpoint Callbacks

In Figure 4.4, the code configures a training strategy that includes early stopping and checkpointing across all segmentation models to improve model training efficiency and reliability. The early stopping callback monitors the validation loss and stops the training if it does not improve for three consecutive epochs. Besides that, the Model Checkpoint callback saves the model whenever a new minimum validation loss is reached. Before the training starts, the script will check whether a saved model already exists. If found, the model will be loaded so the training can resume from the last best state rather than starting the training from scratch again.

4.1.4 Model Training



```
history = model.fit(
    combined_generator(train_generator), # Pass the combined generator function here
    steps_per_epoch=train_steps,
    validation_data=combined_generator(val_generator), # Pass the combined generator function here as well
    validation_steps=val_steps,
    epochs=epochs,
    callbacks=[checkpoint_callback, early_stopping]
)
```

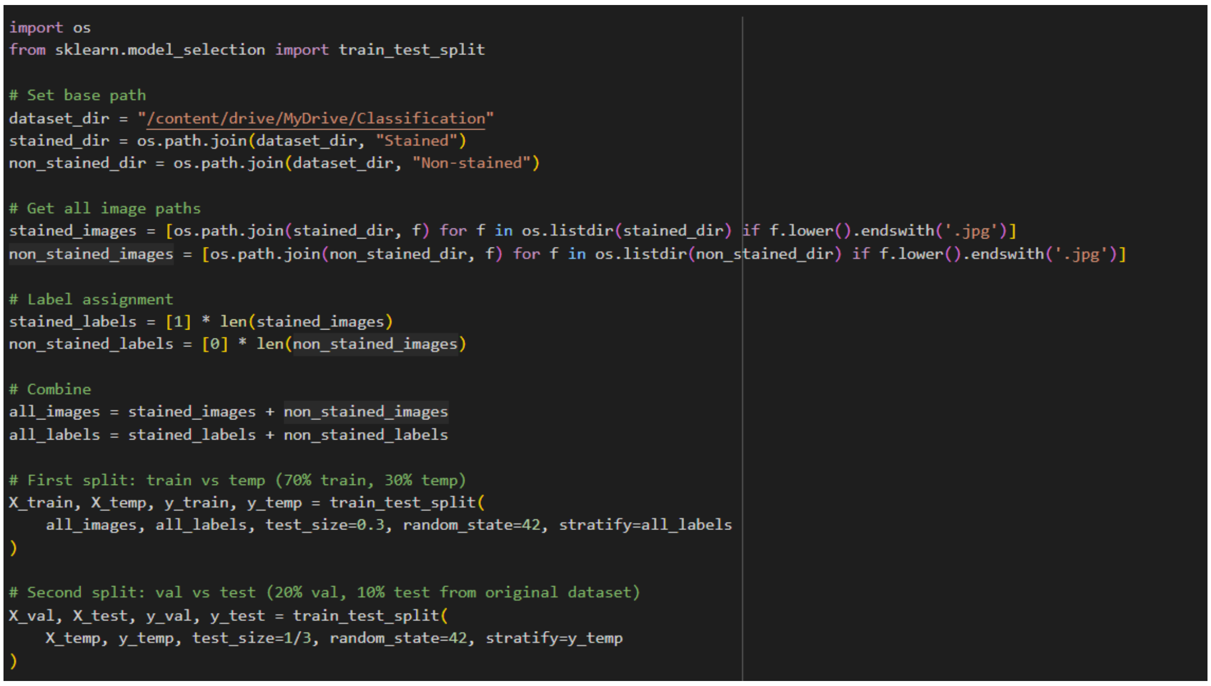
Figure 4.5 Model Training

Figure 4.5 shows the method to train the model. The code uses fit method with combined generators for both training and validation data. The batch size is set to 8 while the training is set to run for 10 epochs maximum. Two callbacks are included during the training process which are Model Checkpoint and Early Stopping that were described in 4.1.3. This setup ensures efficient training by preventing overfitting and preserving the best model weights.

4.2 Classification

4.2.1 Data Preprocessing

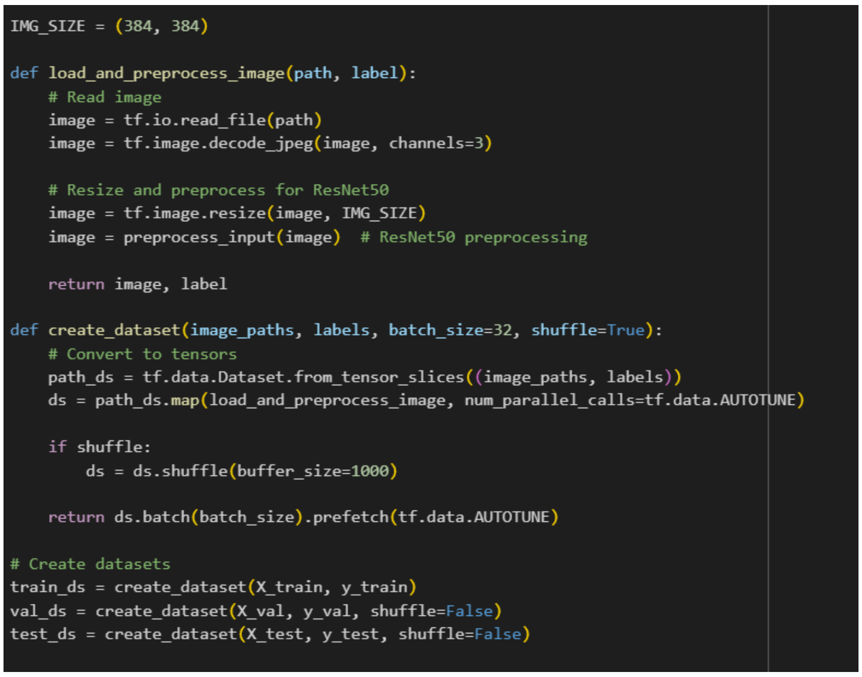
Binary Classification Preprocessing



```python
import os
from sklearn.model_selection import train_test_split

# Set base path
dataset_dir = "/content/drive/MyDrive/Classification"
stained_dir = os.path.join(dataset_dir, "Stained")
non_stained_dir = os.path.join(dataset_dir, "Non-stained")

# Get all image paths
stained_images = [os.path.join(stained_dir, f) for f in os.listdir(stained_dir) if f.lower().endswith('.jpg')]
non_stained_images = [os.path.join(non_stained_dir, f) for f in os.listdir(non_stained_dir) if f.lower().endswith('.jpg')]

# Label assignment
stained_labels = [1] * len(stained_images)
non_stained_labels = [0] * len(non_stained_images)

# Combine
all_images = stained_images + non_stained_images
all_labels = stained_labels + non_stained_labels

# First split: train vs temp (70% train, 30% temp)
X_train, X_temp, y_train, y_temp = train_test_split(
    all_images, all_labels, test_size=0.3, random_state=42, stratify=all_labels
)

# Second split: val vs test (20% val, 10% test from original dataset)
X_val, X_test, y_val, y_test = train_test_split(
    X_temp, y_temp, test_size=1/3, random_state=42, stratify=y_temp
)
```

Figure 4.6 Binary Classification Dataset Splitting

To prepare the dataset for model training and evaluation, a structured data splitting approach was implemented as shown in Figure 4.6. The dataset was initially stored in separate directories within the root folder. Then, each image was retrieved using the “os” module from Python and every image was labelled according to its respective category where stained images were assigned label 1 and non-stained images were label 0. The complete dataset is split with 7:2:1 ratio for training, validation and testing dataset with a stratified sampling strategy to maintain the original class distribution across all subsets.

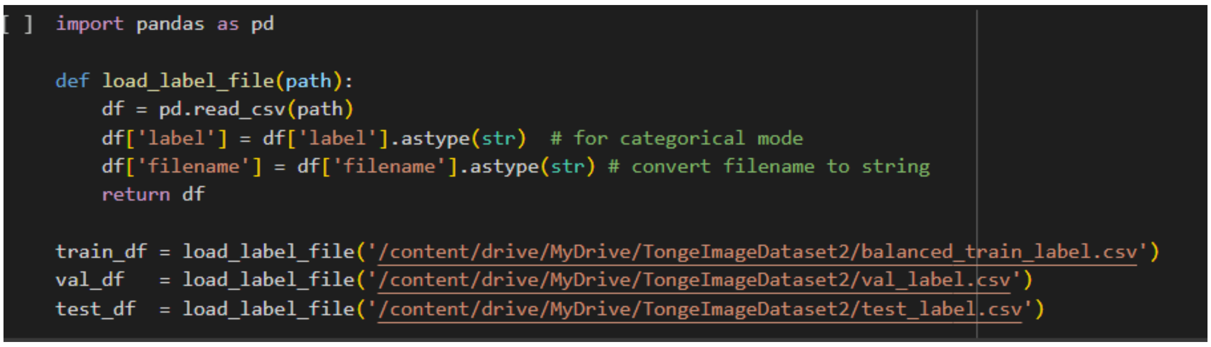


```python
IMG_SIZE = (384, 384)

def load_and_preprocess_image(path, label):
    # Read image
    image = tf.io.read_file(path)
    image = tf.image.decode_jpeg(image, channels=3)

    # Resize and preprocess for ResNet50
    image = tf.image.resize(image, IMG_SIZE)
    image = preprocess_input(image)  # ResNet50 preprocessing

    return image, label

def create_dataset(image_paths, labels, batch_size=32, shuffle=True):
    # Convert to tensors
    path_ds = tf.data.Dataset.from_tensor_slices((image_paths, labels))
    ds = path_ds.map(load_and_preprocess_image, num_parallel_calls=tf.data.AUTOTUNE)

    if shuffle:
        ds = ds.shuffle(buffer_size=1000)

    return ds.batch(batch_size).prefetch(tf.data.AUTOTUNE)

# Create datasets
train_ds = create_dataset(X_train, y_train)
val_ds = create_dataset(X_val, y_val, shuffle=False)
test_ds = create_dataset(X_test, y_test, shuffle=False)
```

Figure 4.7 Image Preprocessing

After the dataset was split into training, validation and testing subsets, the create_dataset function converts the image paths and the corresponding labels into TensorFlow tensors. Each image was then resized to a fixed dimension of 384 x 384 pixels to ensure consistency input. ResNet50 preprocessing steps was applied to normalize the image according to the requirements of the pre-trained model. The dataset is then batched with a batch size of 32 and shuffling was applied to the training set to enhance generalization and avoid learning order-specific patterns during the training process.

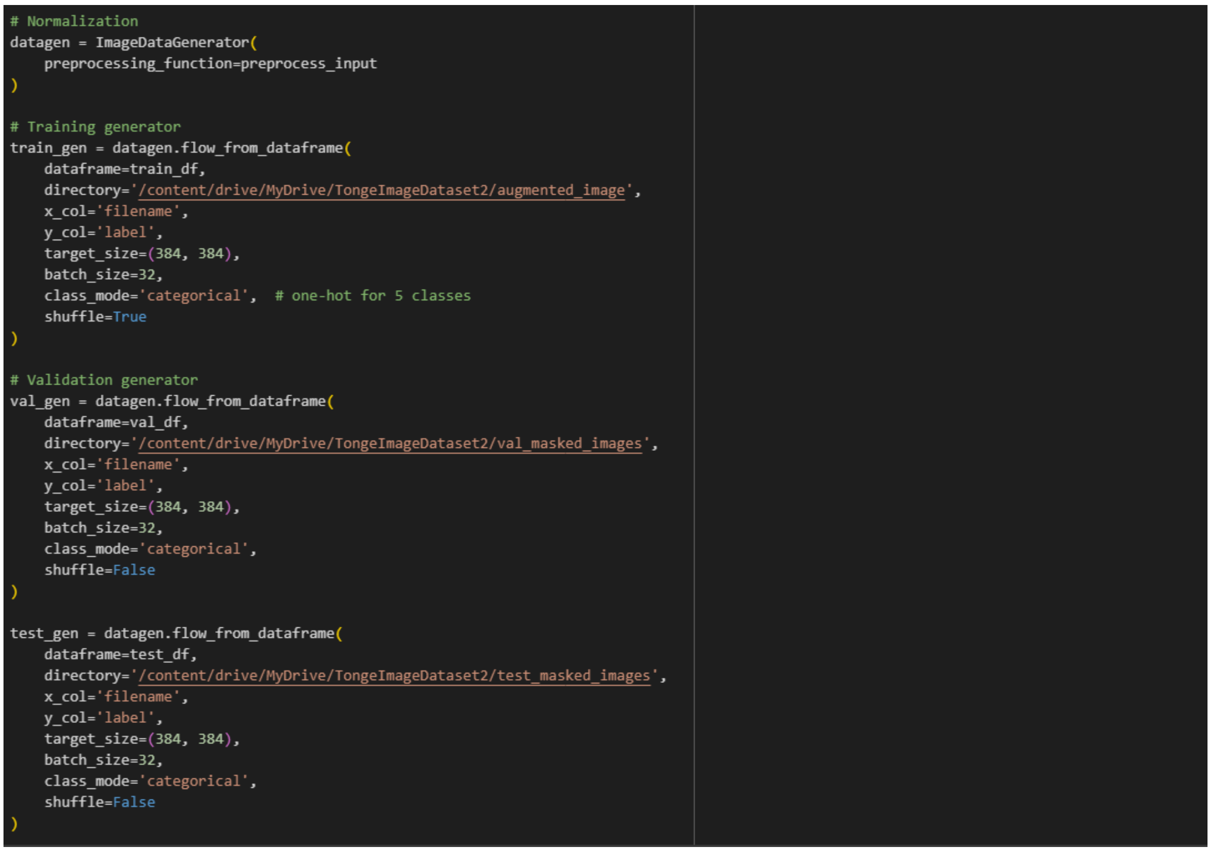
Multi-Class Classification Preprocessing



```python
[ ]  import pandas as pd

     def load_label_file(path):
         df = pd.read_csv(path)
         df['label'] = df['label'].astype(str)  # for categorical mode
         df['filename'] = df['filename'].astype(str) # convert filename to string
         return df

     train_df = load_label_file('/content/drive/MyDrive/TongeImageDataset2/balanced_train_label.csv')
     val_df   = load_label_file('/content/drive/MyDrive/TongeImageDataset2/val_label.csv')
     test_df  = load_label_file('/content/drive/MyDrive/TongeImageDataset2/test_label.csv')
```

Figure 4.8 Reading CSV Files

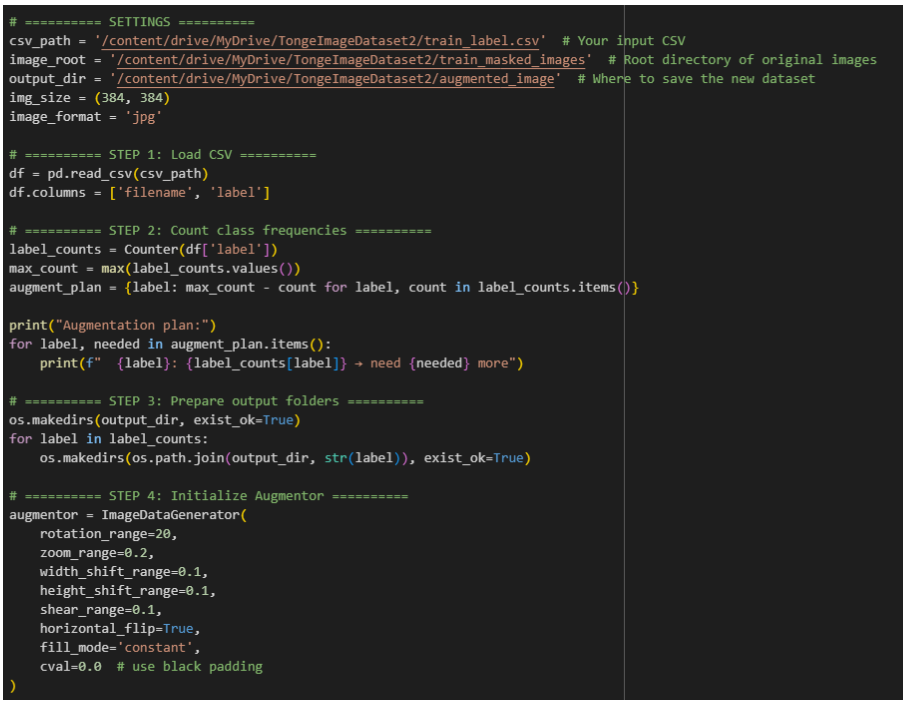
Due to the different methods of data storage, the image metadata for the multi-class classification task was organized and stored in CSV files. Each CSV file contained the filenames of the images along with their corresponding class labels. Both columns were explicitly cast to string format to ensure compatibility with TensorFlow's data pipeline.



```
# Normalization
datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input
)

# Training generator
train_gen = datagen.flow_from_dataframe(
    dataframe=train_df,
    directory='/content/drive/MyDrive/TongeImageDataset2/augmented_image',
    x_col='filename',
    y_col='label',
    target_size=(384, 384),
    batch_size=32,
    class_mode='categorical',  # one-hot for 5 classes
    shuffle=True
)

# Validation generator
val_gen = datagen.flow_from_dataframe(
    dataframe=val_df,
    directory='/content/drive/MyDrive/TongeImageDataset2/val_masked_images',
    x_col='filename',
    y_col='label',
    target_size=(384, 384),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)

test_gen = datagen.flow_from_dataframe(
    dataframe=test_df,
    directory='/content/drive/MyDrive/TongeImageDataset2/test_masked_images',
    x_col='filename',
    y_col='label',
    target_size=(384, 384),
    batch_size=32,
    class_mode='categorical',
    shuffle=False
)
```

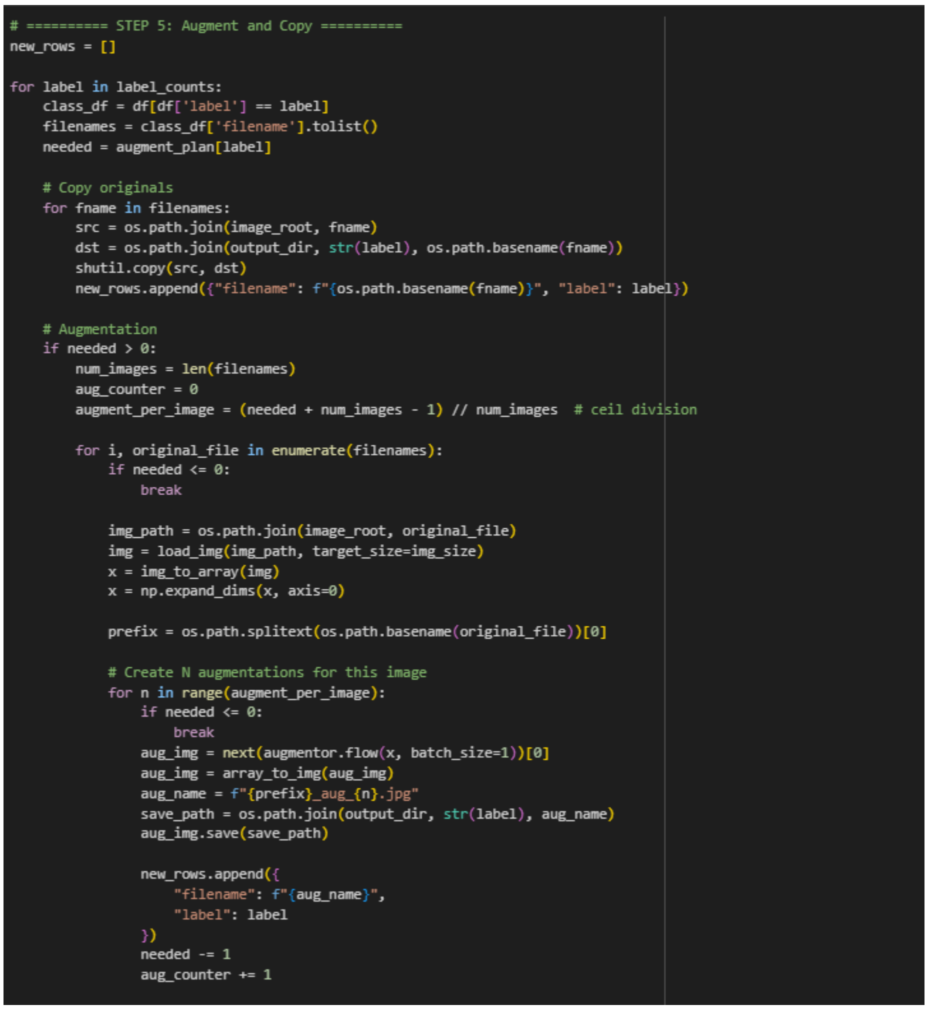
Figure 4.9 Data Generator

Image processing was managed using the ImageDataGenerator class from Keras that facilitated the normalization and efficient loading of image data. The flow_from_dataframe method was used to generate batches of images directly from the labeled dataframes and their corresponding directories. Besides that, each image was also resized to a fixed resolution of 384 x 384 pixels to ensure consistent input size. Finally, the class labels were one-hot encoded using the categorical mode for the five-class classification task.



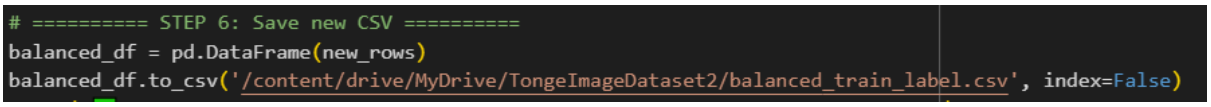
```python
# ========== SETTINGS ==========
csv_path = '/content/drive/MyDrive/TongeImageDataset2/train_label.csv'  # Your input CSV
image_root = '/content/drive/MyDrive/TongeImageDataset2/train_masked_images'  # Root directory of original images
output_dir = '/content/drive/MyDrive/TongeImageDataset2/augmented_image'  # Where to save the new dataset
img_size = (384, 384)
image_format = 'jpg'

# ========== STEP 1: Load CSV ==========
df = pd.read_csv(csv_path)
df.columns = ['filename', 'label']

# ========== STEP 2: Count class frequencies ==========
label_counts = Counter(df['label'])
max_count = max(label_counts.values())
augment_plan = {label: max_count - count for label, count in label_counts.items()}

print("Augmentation plan:")
for label, needed in augment_plan.items():
    print(f"  {label}: {label_counts[label]} → need {needed} more")

# ========== STEP 3: Prepare output folders ==========
os.makedirs(output_dir, exist_ok=True)
for label in label_counts:
    os.makedirs(os.path.join(output_dir, str(label)), exist_ok=True)

# ========== STEP 4: Initialize Augmentor ==========
augmentor = ImageDataGenerator(
    rotation_range=20,
    zoom_range=0.2,
    width_shift_range=0.1,
    height_shift_range=0.1,
    shear_range=0.1,
    horizontal_flip=True,
    fill_mode='constant',
    cval=0.0  # use black padding
)
```

Figure 4.10 Data Augmentation



```python
# ========== STEP 5: Augment and Copy ==========
new_rows = []

for label in label_counts:
    class_df = df[df['label'] == label]
    filenames = class_df['filename'].tolist()
    needed = augment_plan[label]

    # Copy originals
    for fname in filenames:
        src = os.path.join(image_root, fname)
        dst = os.path.join(output_dir, str(label), os.path.basename(fname))
        shutil.copy(src, dst)
        new_rows.append({"filename": f"{os.path.basename(fname)}", "label": label})

    # Augmentation
    if needed > 0:
        num_images = len(filenames)
        aug_counter = 0
        augment_per_image = (needed + num_images - 1) // num_images  # ceil division

        for i, original_file in enumerate(filenames):
            if needed <= 0:
                break

            img_path = os.path.join(image_root, original_file)
            img = load_img(img_path, target_size=img_size)
            x = img_to_array(img)
            x = np.expand_dims(x, axis=0)

            prefix = os.path.splitext(os.path.basename(original_file))[0]

            # Create N augmentations for this image
            for n in range(augment_per_image):
                if needed <= 0:
                    break
                aug_img = next(augmentor.flow(x, batch_size=1))[0]
                aug_img = array_to_img(aug_img)
                aug_name = f"{prefix}_aug_{n}.jpg"
                save_path = os.path.join(output_dir, str(label), aug_name)
                aug_img.save(save_path)

                new_rows.append({
                    "filename": f"{aug_name}",
                    "label": label
                })
                needed -= 1
                aug_counter += 1
```

Figure 4.11 Data Augmentation

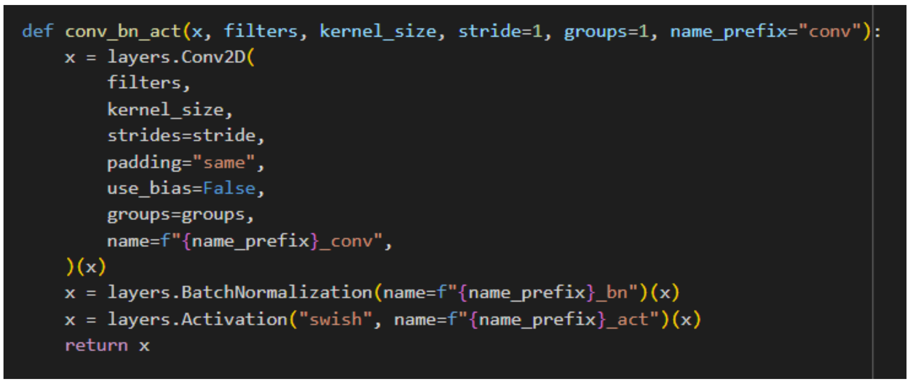


```
# ========== STEP 6: Save new CSV ==========
balanced_df = pd.DataFrame(new_rows)
balanced_df.to_csv('/content/drive/MyDrive/TongeImageDataset2/balanced_train_label.csv', index=False)
```

Figure 4.12 Saving CSV File

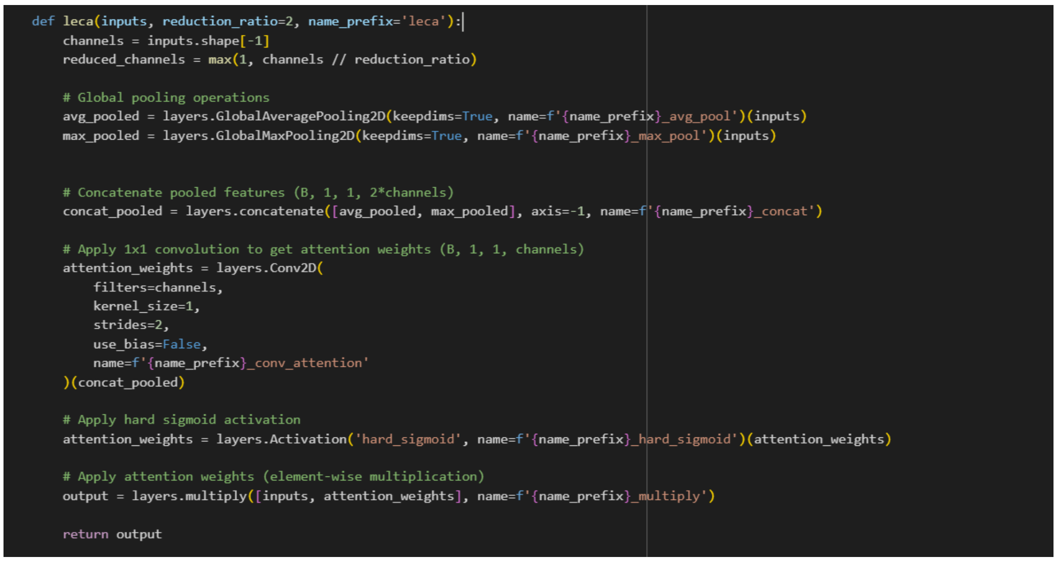
Figure 4.10 to Figure 4.12 shows the process of data augmentation. First, the class distribution was analyzed to determine the number of additional images required for each class to achieve a balance dataset. The image augmentation techniques were configured with a combination of geometric transformations such as rotation, zoom, width, height shifts, shear transformations and horizontal flipping. The fill_mode parameter was set constantly with a black padding value to preserve image consistency after transformations. Finally, the augmented images were saved into respective class-specific subdirectories and a new CSV file containing the filenames of the augmented images along with their class labels was generated.

4.2.2 Modelling



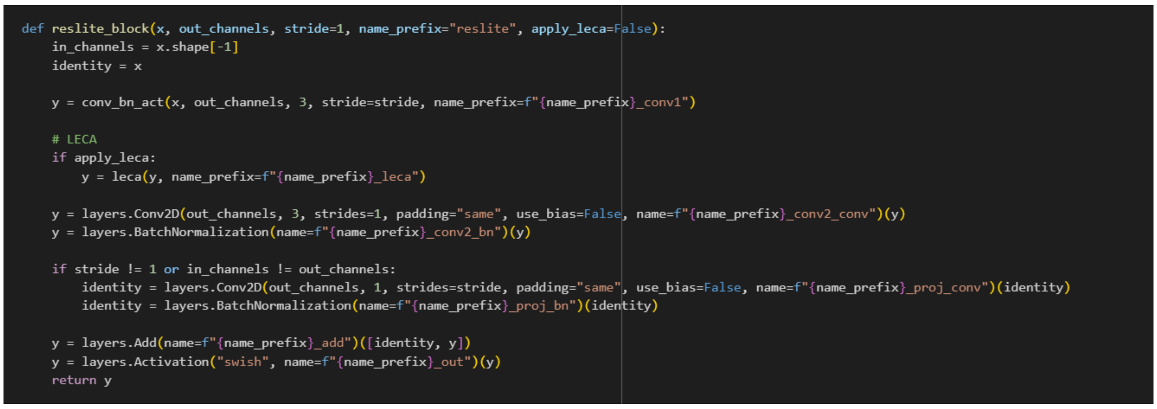
```
def conv_bn_act(x, filters, kernel_size, stride=1, groups=1, name_prefix="conv"):
    x = layers.Conv2D(
        filters,
        kernel_size,
        strides=stride,
        padding="same",
        use_bias=False,
        groups=groups,
        name=f"{name_prefix}_conv",
    )(x)
    x = layers.BatchNormalization(name=f"{name_prefix}_bn")(x)
    x = layers.Activation("swish", name=f"{name_prefix}_act")(x)
    return x
```

Figure 4.13 Convolution, Batch Normalization and Activation Layer



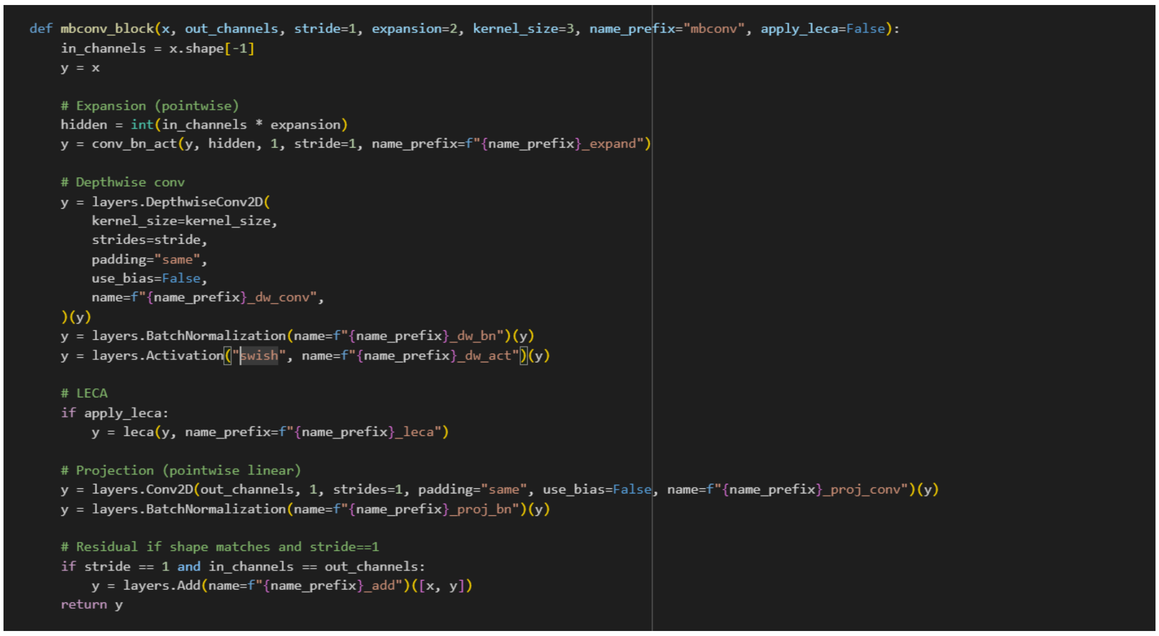
```
def leca(inputs, reduction_ratio=2, name_prefix='leca'):
    channels = inputs.shape[-1]
    reduced_channels = max(1, channels // reduction_ratio)

    # Global pooling operations
    avg_pooled = layers.GlobalAveragePooling2D(keepdims=True, name=f'{name_prefix}_avg_pool')(inputs)
    max_pooled = layers.GlobalMaxPooling2D(keepdims=True, name=f'{name_prefix}_max_pool')(inputs)


    # Concatenate pooled features (B, 1, 1, 2*channels)
    concat_pooled = layers.concatenate([avg_pooled, max_pooled], axis=-1, name=f'{name_prefix}_concat')

    # Apply 1x1 convolution to get attention weights (B, 1, 1, channels)
    attention_weights = layers.Conv2D(
        filters=channels,
        kernel_size=1,
        strides=2,
        use_bias=False,
        name=f'{name_prefix}_conv_attention'
    )(concat_pooled)

    # Apply hard sigmoid activation
    attention_weights = layers.Activation('hard_sigmoid', name=f'{name_prefix}_hard_sigmoid')(attention_weights)

    # Apply attention weights (element-wise multiplication)
    output = layers.multiply([inputs, attention_weights], name=f'{name_prefix}_multiply')

    return output
```

Figure 4.14 LECA Module



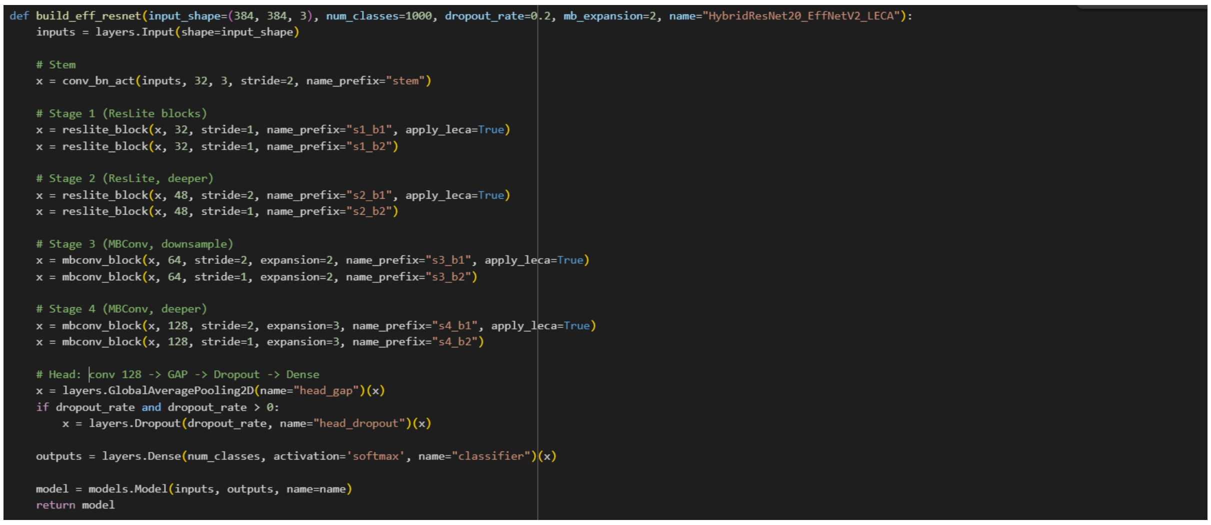
```python
def reslite_block(x, out_channels, stride=1, name_prefix="reslite", apply_leca=False):
    in_channels = x.shape[-1]
    identity = x

    y = conv_bn_act(x, out_channels, 3, stride=stride, name_prefix=f"{name_prefix}_conv1")

    # LECA
    if apply_leca:
        y = leca(y, name_prefix=f"{name_prefix}_leca")

    y = layers.Conv2D(out_channels, 3, strides=1, padding="same", use_bias=False, name=f"{name_prefix}_conv2_conv")(y)
    y = layers.BatchNormalization(name=f"{name_prefix}_conv2_bn")(y)

    if stride != 1 or in_channels != out_channels:
        identity = layers.Conv2D(out_channels, 1, strides=stride, padding="same", use_bias=False, name=f"{name_prefix}_proj_conv")(identity)
        identity = layers.BatchNormalization(name=f"{name_prefix}_proj_bn")(identity)

    y = layers.Add(name=f"{name_prefix}_add")([identity, y])
    y = layers.Activation("swish", name=f"{name_prefix}_out")(y)
    return y
```

Figure 4.15 ResLite Block



```python
def mbconv_block(x, out_channels, stride=1, expansion=2, kernel_size=3, name_prefix="mbconv", apply_leca=False):
    in_channels = x.shape[-1]
    y = x

    # Expansion (pointwise)
    hidden = int(in_channels * expansion)
    y = conv_bn_act(y, hidden, 1, stride=1, name_prefix=f"{name_prefix}_expand")

    # Depthwise conv
    y = layers.DepthwiseConv2D(
        kernel_size=kernel_size,
        strides=stride,
        padding="same",
        use_bias=False,
        name=f"{name_prefix}_dw_conv",
    )(y)
    y = layers.BatchNormalization(name=f"{name_prefix}_dw_bn")(y)
    y = layers.Activation("swish", name=f"{name_prefix}_dw_act")(y)

    # LECA
    if apply_leca:
        y = leca(y, name_prefix=f"{name_prefix}_leca")

    # Projection (pointwise linear)
    y = layers.Conv2D(out_channels, 1, strides=1, padding="same", use_bias=False, name=f"{name_prefix}_proj_conv")(y)
    y = layers.BatchNormalization(name=f"{name_prefix}_proj_bn")(y)

    # Residual if shape matches and stride==1
    if stride == 1 and in_channels == out_channels:
        y = layers.Add(name=f"{name_prefix}_add")([x, y])
    return y
```

Figure 4.16 MBConv Block



```python
def build_eff_resnet(input_shape=(384, 384, 3), num_classes=1000, dropout_rate=0.2, mb_expansion=2, name="HybridResNet20_EffNetV2_LECA"):
    inputs = layers.Input(shape=input_shape)

    # Stem
    x = conv_bn_act(inputs, 32, 3, stride=2, name_prefix="stem")

    # Stage 1 (ResLite blocks)
    x = reslite_block(x, 32, stride=1, name_prefix="s1_b1", apply_leca=True)
    x = reslite_block(x, 32, stride=1, name_prefix="s1_b2")

    # Stage 2 (ResLite, deeper)
    x = reslite_block(x, 48, stride=2, name_prefix="s2_b1", apply_leca=True)
    x = reslite_block(x, 48, stride=1, name_prefix="s2_b2")

    # Stage 3 (MBConv, downsample)
    x = mbconv_block(x, 64, stride=2, expansion=2, name_prefix="s3_b1", apply_leca=True)
    x = mbconv_block(x, 64, stride=1, expansion=2, name_prefix="s3_b2")

    # Stage 4 (MBConv, deeper)
    x = mbconv_block(x, 128, stride=2, expansion=3, name_prefix="s4_b1", apply_leca=True)
    x = mbconv_block(x, 128, stride=1, expansion=3, name_prefix="s4_b2")

    # Head: conv 128 -> GAP -> Dropout -> Dense
    x = layers.GlobalAveragePooling2D(name="head_gap")(x)
    if dropout_rate and dropout_rate > 0:
        x = layers.Dropout(dropout_rate, name="head_dropout")(x)

    outputs = layers.Dense(num_classes, activation='softmax', name="classifier")(x)

    model = models.Model(inputs, outputs, name=name)
    return model
```

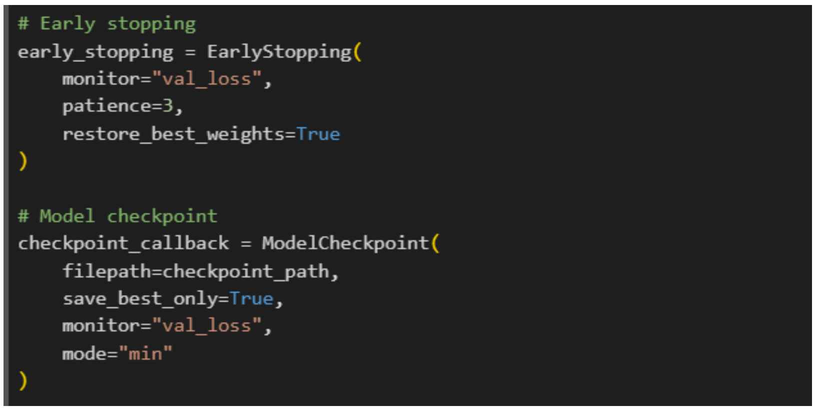
Figure 4.17 Efficient-ResNet Model Creation

Figure 4.13 to Figure 4.17 shows the creation of Efficient-ResNet Model. The model combines ResNet-style residual connections, EfficientNetV2-inspired MBConv blocks and a Lightweight Efficient Channel Attention mechanism. The core building units are modularized for flexibility. The conv_bn_act utility applies a convolutional layer followed by batch normalization and a Swish activation, serving as the basic operation for feature extraction. The LECA module performs global average pooling and global max pooling in parallel to capture complementary global context, concatenates the pooled descriptors, and applies a pointwise convolution with hard-sigmoid activation to generate channel-wise attention weights. These weights are used to re-scale the feature maps, thereby enhancing informative channels while suppressing redundant ones.

The ResLite block is a lightweight residual unit composed of two consecutive convolutional operations with batch normalization and non-linear activation which then connected through an identity mapping. When spatial resolution or channel dimension changes, a projection shortcut with a 1×1 convolution is used to align dimensions. LECA can optionally be integrated within the block to improve channel recalibration. The MBConv block employs an expansion phase via a pointwise convolution, followed by depthwise convolution to capture spatial information with reduced complexity, and a projection phase that restores channel dimensionality. Residual connections are maintained when input and output dimensions match and stride equal one.

The complete model begins with a stem convolution for low-level feature extraction and progresses through sequential stages of ResLite and MBConv blocks with increasing channel depth and downsampling at each stage. The feature extraction backbone is terminated with a global average pooling layer, followed by dropout for regularization, and a dense softmax classifier for multi-class prediction. This architectural design strategically combines the efficiency of MBConv operations, the stability of residual learning, and the representational enhancement of lightweight channel attention, resulting in a computationally efficient yet expressive deep learning model suitable for resource-constrained environments.

4.2.3 Early Stopping and Model Checkpoint Callbacks

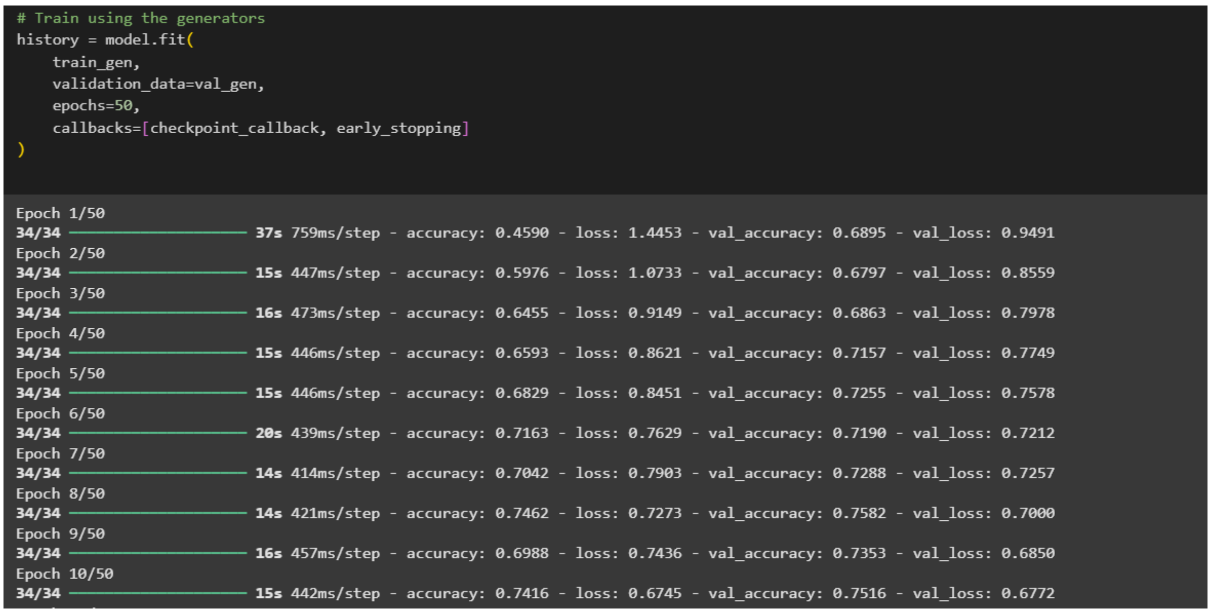


```
# Early stopping
early_stopping = EarlyStopping(
    monitor="val_loss",
    patience=3,
    restore_best_weights=True
)

# Model checkpoint
checkpoint_callback = ModelCheckpoint(
    filepath=checkpoint_path,
    save_best_only=True,
    monitor="val_loss",
    mode="min"
)
```

Figure 4.18 Early Stopping and Model Checkpoint Callbacks

To optimize training efficiency and prevent overfitting, two callback strategies were developed as shown in Figure 4.18. Early stopping was configured to monitor the validation loss during training and terminate the training process when there is no improvement was observed for three consecutive epochs. Additionally, the final model will retain the weights from the epoch with the lowest validation loss. Model checkpoint was implemented to save the best-performing model throughout the training based on the minimum validation loss observed. It stored the model weights in the predefined checkpoint path and updated the saved file only when a new performance improvement was detected.

4.2.4 Model Training



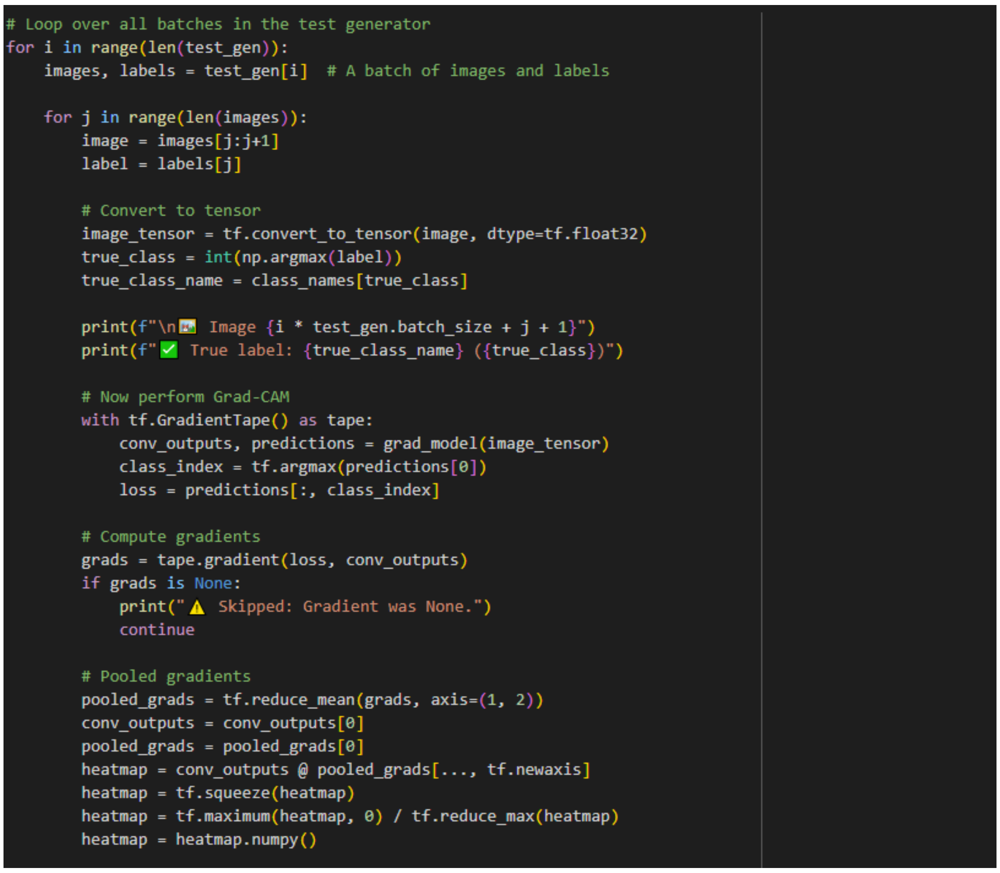
```
# Train using the generators
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=50,
    callbacks=[checkpoint_callback, early_stopping]
)
```

```
Epoch 1/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 37s 759ms/step - accuracy: 0.4590 - loss: 1.4453 - val_accuracy: 0.6895 - val_loss: 0.9491
Epoch 2/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 15s 447ms/step - accuracy: 0.5976 - loss: 1.0733 - val_accuracy: 0.6797 - val_loss: 0.8559
Epoch 3/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 16s 473ms/step - accuracy: 0.6455 - loss: 0.9149 - val_accuracy: 0.6863 - val_loss: 0.7978
Epoch 4/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 15s 446ms/step - accuracy: 0.6593 - loss: 0.8621 - val_accuracy: 0.7157 - val_loss: 0.7749
Epoch 5/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 15s 446ms/step - accuracy: 0.6829 - loss: 0.8451 - val_accuracy: 0.7255 - val_loss: 0.7578
Epoch 6/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 20s 439ms/step - accuracy: 0.7163 - loss: 0.7629 - val_accuracy: 0.7190 - val_loss: 0.7212
Epoch 7/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 14s 414ms/step - accuracy: 0.7042 - loss: 0.7903 - val_accuracy: 0.7288 - val_loss: 0.7257
Epoch 8/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 14s 421ms/step - accuracy: 0.7462 - loss: 0.7273 - val_accuracy: 0.7582 - val_loss: 0.7000
Epoch 9/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 16s 457ms/step - accuracy: 0.6988 - loss: 0.7436 - val_accuracy: 0.7353 - val_loss: 0.6850
Epoch 10/50
34/34 ━━━━━━━━━━━━━━━━━━━━ 15s 442ms/step - accuracy: 0.7416 - loss: 0.6745 - val_accuracy: 0.7516 - val_loss: 0.6772
```

Figure 4.19 Model Training

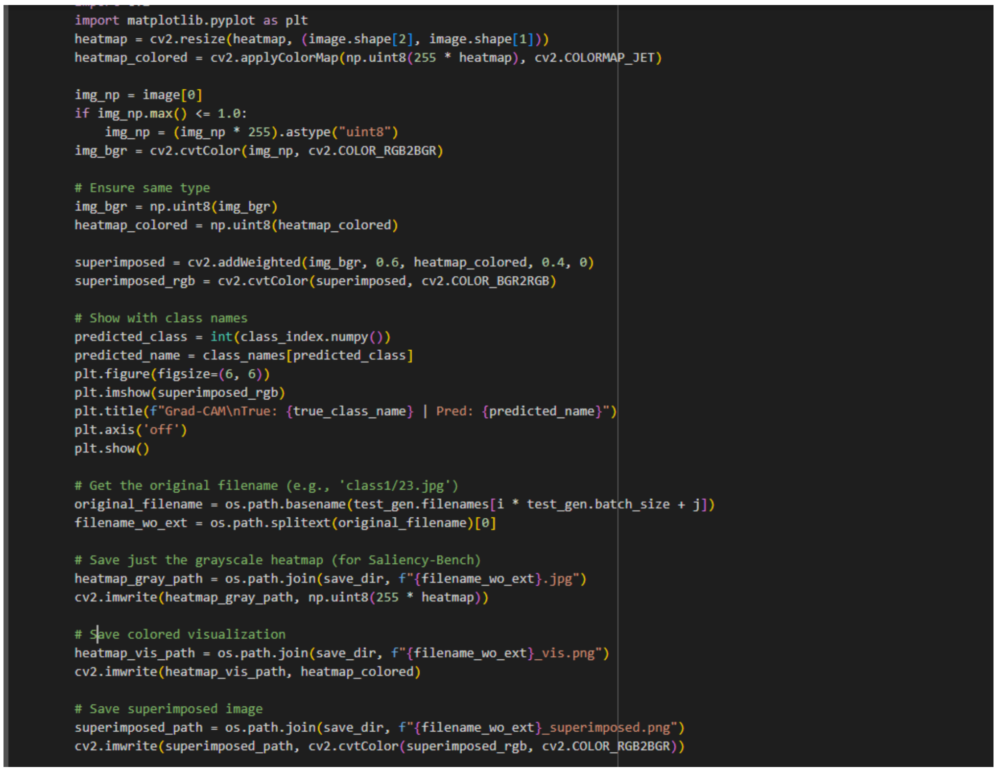
The model training process was conducted using the fit method with the training dataset serving as the input and validation dataset used for performance monitoring. The training was scheduled for a maximum of 50 epochs. However, the actual number of training iterations could be reduced due to the early stopping mechanism. Both the early stopping and model checkpoint callbacks were integrated into the training loop to ensure that the model retain its optimal weights and avoid overfitting.

4.2.5 Visualization



```
# Loop over all batches in the test generator
for i in range(len(test_gen)):
    images, labels = test_gen[i]  # A batch of images and labels

    for j in range(len(images)):
        image = images[j:j+1]
        label = labels[j]

        # Convert to tensor
        image_tensor = tf.convert_to_tensor(image, dtype=tf.float32)
        true_class = int(np.argmax(label))
        true_class_name = class_names[true_class]

        print(f"\n🖼️ Image {i * test_gen.batch_size + j + 1}")
        print(f"✅ True label: {true_class_name} ({true_class})")

        # Now perform Grad-CAM
        with tf.GradientTape() as tape:
            conv_outputs, predictions = grad_model(image_tensor)
            class_index = tf.argmax(predictions[0])
            loss = predictions[:, class_index]

        # Compute gradients
        grads = tape.gradient(loss, conv_outputs)
        if grads is None:
            print("⚠️ Skipped: Gradient was None.")
            continue

        # Pooled gradients
        pooled_grads = tf.reduce_mean(grads, axis=(1, 2))
        conv_outputs = conv_outputs[0]
        pooled_grads = pooled_grads[0]
        heatmap = conv_outputs @ pooled_grads[..., tf.newaxis]
        heatmap = tf.squeeze(heatmap)
        heatmap = tf.maximum(heatmap, 0) / tf.reduce_max(heatmap)
        heatmap = heatmap.numpy()
```

Figure 4.20 Grad-CAM Visualization



```python
import matplotlib.pyplot as plt
heatmap = cv2.resize(heatmap, (image.shape[2], image.shape[1]))
heatmap_colored = cv2.applyColorMap(np.uint8(255 * heatmap), cv2.COLORMAP_JET)

img_np = image[0]
if img_np.max() <= 1.0:
    img_np = (img_np * 255).astype("uint8")
img_bgr = cv2.cvtColor(img_np, cv2.COLOR_RGB2BGR)

# Ensure same type
img_bgr = np.uint8(img_bgr)
heatmap_colored = np.uint8(heatmap_colored)

superimposed = cv2.addWeighted(img_bgr, 0.6, heatmap_colored, 0.4, 0)
superimposed_rgb = cv2.cvtColor(superimposed, cv2.COLOR_BGR2RGB)

# Show with class names
predicted_class = int(class_index.numpy())
predicted_name = class_names[predicted_class]
plt.figure(figsize=(6, 6))
plt.imshow(superimposed_rgb)
plt.title(f"Grad-CAM\nTrue: {true_class_name} | Pred: {predicted_name}")
plt.axis('off')
plt.show()

# Get the original filename (e.g., 'class1/23.jpg')
original_filename = os.path.basename(test_gen.filenames[i * test_gen.batch_size + j])
filename_wo_ext = os.path.splitext(original_filename)[0]

# Save just the grayscale heatmap (for Saliency-Bench)
heatmap_gray_path = os.path.join(save_dir, f"{filename_wo_ext}.jpg")
cv2.imwrite(heatmap_gray_path, np.uint8(255 * heatmap))

# Save colored visualization
heatmap_vis_path = os.path.join(save_dir, f"{filename_wo_ext}_vis.png")
cv2.imwrite(heatmap_vis_path, heatmap_colored)

# Save superimposed image
superimposed_path = os.path.join(save_dir, f"{filename_wo_ext}_superimposed.png")
cv2.imwrite(superimposed_path, cv2.cvtColor(superimposed_rgb, cv2.COLOR_RGB2BGR))
```

Figure 4.21 Display and Save Generated Visualization

Figure 4.20 and Figure 4.21 shows the Grad-CAM visualization process. For each image in the test dataset, a forward pass was performed to obtain predictions and the gradient of the predicted class score with respect to the convolutional feature maps was computed. The gradients were global average pooled to obtain channel importance weights, which were then combined with the feature maps to generate a class localization heatmap. The heatmap was then normalized and colour-mapped using OpenCV's COLORMAP_JET. A weighted overlay of the heatmap on the original image was created to highlight salient regions that influencing the model's decision. Finally, the grayscale heatmap, colour heatmap and the superimposed visualization were saved for each image for further analysis.

Chapter 5
System Evaluation and Discussion

5.1 Segmentation Experimental Results

5.1.1 Quantitative Evaluation

In this study, four baseline deep learning models which are U-Net, SegNet, PSPNet and DeepLabv3+ were evaluated for tongue segmentation based on the MIoU, MPA, MDSC as discussed in Chapter 3 along with the number of parameters. BioHit and Roboflow datasets were collected under consistent lighting environments while PaddlePaddle dataset was captured using a mobile phone with inconsistent lighting conditions. This setup allows a realistic assessment of each model's robustness and generalizability in practical scenarios.

Table 5.1 Performance Results of the Models (MIoU)

Model	BioHit	Roboflow	PaddlePaddle	Average
U-Net [27]	0.9468	0.9631	0.8838	0.9312
SegNet [37]	0.9601	0.9578	0.9177	0.9452
PSPNet [38]	0.9674	0.9702	0.9573	0.9650
DeepLabv3+ [39]	0.9710	**0.9712**	0.9589	**0.9670**
Mobile U-Net - Small	0.9614	0.9432	0.9321	0.9456
Mobile U-Net - Large	**0.9740**	0.9617	**0.9644**	0.9667
Mobile U-Net - Transformer	0.9522	0.9513	0.9390	0.9475

Table 5,1 presents the mean Intersection over Union performance of seven segmentation models evaluated on the three benchmark datasets. On the BioHit dataset, Mobile U-Net Large achieved the highest mIoU (0.9740), slightly outperforming DeepLabv3+ (0.9710) and PSPNet (0.9674). Traditional models such as SegNet (0.9601) and U-Net (0.9468) demonstrated competitive but relatively lower scores, indicating that while classical encoder–decoder designs remain effective while enhancements in deeper or more optimized architectures contribute to incremental improvements.

In the Roboflow dataset, DeepLabv3+ obtained the best score (0.9712), closely followed by PSPNet (0.9702) and Mobile U-Net Large (0.9617). Despite its relative simplicity, U-Net still demonstrated robust performance (0.9631) and outperforming the Mobile U-Net Small variant (0.9432), which showed a decline likely due to reduced model capacity. The results suggest that while lightweight models are attractive for efficiency, they may not capture sufficient feature richness when applied to more complex data.

For the PaddlePaddle dataset, Mobile U-Net Large once again delivered the best performance (0.9644), followed by DeepLabv3+ (0.9589) and PSPNet (0.9573). Notably, U-Net dropped significantly to 0.8838, indicating sensitivity to variations in lighting and image quality. The Mobile U-Net Transformer variant (0.9390) performed better than the small configuration but did not surpass the large version, suggesting that simply adding transformer layers does not fully mitigate the challenges posed by inconsistent imaging environments.

The performance gap between Mobile U-Net Small and Mobile U-Net Transformer illustrates the specific impact of transformer integration. Both models share the same convolutional filter configurations with the only difference being the inclusion of a ViT-style transformer block at the bottleneck. On the controlled BioHit and Roboflow datasets, the transformer variant achieved slightly higher mIoU values than the small configuration (0.9522 vs. 0.9614 on BioHit, 0.9513 vs. 0.9432 on Roboflow), suggesting that the attention mechanism improved global feature modeling under consistent imaging conditions. However, on the more challenging PaddlePaddle dataset, the transformer model (0.9390) did not surpass the large variant (0.9644) and only modestly outperformed the small version (0.9321). This indicates that while transformer integration can enhance contextual feature extraction, it is not sufficient on its own to fully address domain shifts caused by inconsistent lighting or lower image quality. Instead, increasing network capacity, as in Mobile U-Net Large, appears more effective for achieving robust performance across varied imaging conditions.

In summary, Mobile U-Net Large consistently ranked among the top performers, demonstrating strong adaptability even under challenging conditions. DeepLabv3+ and PSPNet also showed stable and competitive performance across datasets, confirming their reliability in different settings. In contrast, U-Net and Mobile U-Net Small were more vulnerable to performance degradation especially in the PaddlePaddle dataset, underscoring their limited generalization capacity. These findings indicate that models with higher representational power or more sophisticated feature extraction strategies are better equipped to handle the variability introduced by uncontrolled imaging conditions, whereas lighter architectures may sacrifice accuracy for efficiency.

Table 5.2 Performance Results of the Models (MPA)

Model	**BioHit**	**Roboflow**	**PaddlePaddle**	**Average**
U-Net [27]	0.9788	0.9909	0.9542	0.9746
SegNet [37]	0.9820	0.9901	0.9630	0.9784
PSPNet [38]	0.9848	0.9921	**0.9828**	0.9866
DeepLabv3+ [39]	**0.9860**	**0.9927**	0.9826	**0.9871**
Mobile U-Net - Small	0.9789	0.9866	0.9693	0.9783
Mobile U-Net - Large	0.9847	0.9902	0.9823	0.9857
Mobile U-Net - Transformer	0.9790	0.9889	0.9744	0.9808

Table 5.2 shows the mean pixel accuracy of the seven segmentation models across the datasets. On the BioHit dataset, DeepLabv3+ achieved the highest mPA (0.9860), followed closely by PSPNet (0.9848) and Mobile U-Net Large (0.9847). SegNet (0.9820) and U-Net (0.9788) performed slightly lower but still demonstrated strong reliability in controlled imaging conditions. Both Mobile U-Net Small (0.9789) and Transformer (0.9790) attained similar performance to the baseline U-Net, indicating that lightweight or transformer-enhanced variants were less advantageous when the dataset posed fewer challenges.

In the Roboflow dataset, DeepLabv3+ again achieved the top performance (0.9927), with PSPNet (0.9921) and U-Net (0.9909) following closely. Mobile U-Net Large (0.9902) remained competitive, while Mobile U-Net Small dropped slightly to 0.9866. Interestingly, SegNet (0.9901) was also among the strongest performers, highlighting that classical architectures remain effective under consistent lighting conditions.

For the PaddlePaddle dataset, which was captured under inconsistent lighting conditions, the differences between models became more apparent. PSPNet (0.9828) obtained the highest mPA, with DeepLabv3+ (0.9826) and Mobile U-Net Large (0.9823) performing almost equally well. SegNet and Mobile U-Net Transformer achieved moderate scores (0.9630 and 0.9744, respectively), while U-Net recorded the lowest value (0.9542). These results suggest that larger and more advanced architectures are more resilient to variability in imaging conditions, whereas lightweight or classical designs exhibit greater sensitivity to noise and inconsistencies.

When averaged across datasets, DeepLabv3+ (0.9871) and PSPNet (0.9866) delivered the best overall mPA, followed by Mobile U-Net Large (0.9857). The transformer-enhanced Mobile U-Net (0.9808) achieved modest improvements over the small variant (0.9783) but did not approach the performance of the larger configuration, confirming that increased model capacity plays a more critical role than transformer integration alone in achieving robust segmentation performance.

In summary, the mPA results demonstrate that all models performed strongly under controlled conditions, with modern architectures such as DeepLabv3+ and PSPNet consistently outperforming U-Net and SegNet, especially in the more challenging PaddlePaddle dataset. While Mobile U-Net variants strike a balance between efficiency and accuracy, only the large configuration consistently matched the performance of state-of-the-art models, underscoring the importance of model capacity for handling complex real-world data variability.

Table 5.3 Performance Results of the Models (MDSC)

Model	BioHit	Roboflow	PaddlePaddle	Average
U-Net [27]	0.9763	0.9783	0.9356	0.9634
SegNet [37]	0.9808	0.9760	0.9527	0.9698
PSPNet [38]	0.9865	0.9825	0.9793	0.9828
DeepLabv3+ [39]	**0.9886**	**0.9842**	0.9806	**0.9845**
Mobile U-Net - Small	0.9759	0.9649	0.9603	0.9670
Mobile U-Net - Large	0.9862	0.9760	**0.9807**	0.9810
Mobile U-Net - Transformer	0.9767	0.9724	0.9642	0.9711

Table 5.3 summarizes the mean Dice Similarity Coefficient of the evaluated segmentation model across the BioHit, Roboflow and PaddlePaddle datasets. On the BioHit dataset, DeepLabv3+ achieved the best performance (0.9886), marginally outperforming PSPNet (0.9865) and Mobile U-Net Large (0.9862). SegNet (0.9808) and U-Net (0.9763) produced competitive but lower scores, while Mobile U-Net Small (0.9759) and Transformer (0.9767) trailed close to the baseline U-Net, indicating that lightweight variants and bottleneck-level transformer integration provided limited benefits under consistent imaging conditions.

For the Roboflow dataset, DeepLabv3+ again achieved the highest mDSC (0.9842), followed by PSPNet (0.9825). U-Net (0.9783) remained reliable, though Mobile U-Net Small dropped to 0.9649, the lowest score on this dataset. Interestingly, both Mobile U-Net Large (0.9760) and SegNet (0.9760) produced identical results, reinforcing the stability of encoder–decoder structures under controlled conditions but also showing that deeper feature extraction in PSPNet and DeepLabv3+ leads to consistently higher overlap accuracy.

In the PaddlePaddle dataset, which posed greater challenges due to inconsistent lighting and capture conditions, Mobile U-Net Large and DeepLabv3+ performed nearly equally well (0.9807 and 0.9806, respectively), with PSPNet close behind (0.9793). These three models significantly outperformed U-Net (0.9356), highlighting the vulnerability of simpler architectures to domain shifts. The transformer variant (0.9642) slightly improved over the small model (0.9603), but neither approached the accuracy of the larger or state-of-the-art architecture.

In summary, the results reinforce the trends observed in mIoU and mPA. DeepLabv3+ and PSPNet consistently achieved the strongest performance, followed closely by Mobile U-Net Large, particularly in challenging datasets such as PaddlePaddle. By contrast, U-Net and Mobile U-Net Small showed more significant performance degradation under uncontrolled imaging conditions. These findings highlight that advanced feature extraction strategies and larger model capacity are crucial for achieving robust and reliable segmentation performance across diverse real-world environments.

Table 5.4 Number of Parameters of the Models

Model	Number of Parameters (10^6)
U-Net [27]	31.04
SegNet [37]	29.44
PSPNet [38]	24.82
DeepLabv3+ [39]	**17.86**
Mobile U-Net - Small	2.54
Mobile U-Net - Large	4.38
Mobile U-Net - Transformer	3.80

An important aspect of this analysis is the computational efficiency of each model. The number of trainable parameters provides insights into the complexity and potential computational cost of each architecture. In Table 5.4, the results shows that the Mobile U-Net variants were by far the most parameter-efficient. The small configuration used only 2.54M parameters, nearly an order of magnitude smaller than DeepLabv3+ and more than ten times smaller than U-Net. The large configuration required 4.38M parameters, striking a balance between compactness and representational capacity, while the transformer-enhanced variant used 3.80M parameters, falling between the small and large designs. Despite their drastically reduced size, the Mobile U-Net models delivered competitive segmentation accuracy which closely matched or exceeded state-of-the-art architectures in several datasets especially the large version of Mobile U-Net.

5.1.2 Performance Comparison with Previous Researchers

Table 5.5 Segmentation Performance Comparison with Previous Researchers

Model	**Dataset**	**mIoU**	**Number of Parameters (10^6)**	**Source**
U-Net & Attention & Edge Refinement	BioHit	0.9930	7.90	L. Yao et al. [10]
TongueSAM	BioHit / PaddlePaddle	0. 9862/ 0.9785	641.09	S. Cao, Q. Wu, and L. Ma [13]
DeepLabV3+(Mobilenet) & CBAM & PointRend	BioHit	0.9624	5.87	Y. Tang et al. [14]
Mobile U-Net - Small	BioHit / PaddlePaddle	0.9614 / 0.9321	2.54	This study
Mobile U-Net - Large	BioHit / PaddlePaddle	0.9740 / 0.9644	4.38	This study
Mobile U-Net - Transformer	BioHit / PaddlePaddle	0.9522 / 0.9390	3.80	This study

Table 5.5 presents a comparison of segmentation performance in mIoU and model complexity between the proposed Mobile U-Net variants and prior research. The results highlight the trade-offs between segmentation accuracy and model efficiency, as well as the contributions of this study in advancing lightweight tongue segmentation models.

The highest mIoU was reported by U-Net with Attention and Edge Refinement on the BioHit dataset (0.9930) as presented by Yao et al. [10]. However, the number of parameters was not reported, and the use of multiple refinements suggests a considerably higher model complexity than standard U-Net. Similarly, TongueSAM achieved very high accuracy (0.9862 on BioHit and 0.9785 on PaddlePaddle) but required 641.09M parameters, making it computationally prohibitive for practical clinical or mobile deployment. Other prior work, such as DeepLabv3+(MobileNet) combined with CBAM and PointRend, demonstrated strong performance on BioHit (0.9624 mIoU) with a parameter count of 5.87M [14], striking a reasonable balance between accuracy and efficiency.

In comparison, the Mobile U-Net variants developed in this study achieved competitive accuracy while requiring significantly fewer parameters. Specifically, Mobile U-Net Large obtained 0.9740 mIoU on BioHit and 0.9644 on PaddlePaddle with only 4.38M parameters, closely approaching TongueSAM's accuracy on PaddlePaddle while being more than 146 times smaller in parameter count. The Mobile U-Net Small and Transformer-enhanced variants also produced respectable results (0.9614/0.9321 and 0.9522/0.9390, respectively), further demonstrating that efficient architectures can deliver strong performance under controlled and uncontrolled imaging conditions.

These findings indicate that while state-of-the-art models such as TongueSAM or attention-refined U-Nets achieve slightly higher accuracy, the Mobile U-Net family offers a far superior balance between performance and computational cost. This efficiency makes the proposed models particularly suitable for deployment in real-world clinical applications where hardware resources may be limited, such as mobile health platforms and point-of-care diagnostic systems.

5.1.3 Qualitative Evaluation

To further assess the performance of the segmentation models, a qualitative comparison was conducted using selected test images inspired by the challenge as mentioned in [9] and the results are shown in Figure 5.1, 5.2 and 5.3. The selected images represent various challenging tongue conditions and are split across 3 datasets, including:

(a) Tongue in irregular poses
(b) Tongue that are not fully protruded
(c) Tongue with imprints from teeth along the edges
(d) Tongue with visible gaps in the mouth
(e) Tongue exhibiting abnormal colour
(f) Tongue where teeth are visible

(g) Tongues closely surrounded by the lips

(h) Tongue with abnormal texture

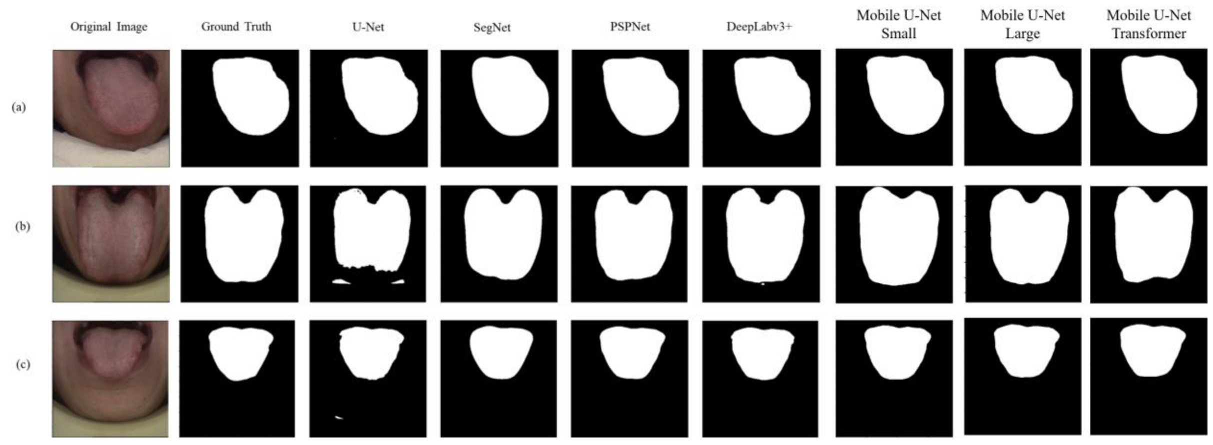


Figure 5.1 Comparison of Segmentation Results in BioHit dataset

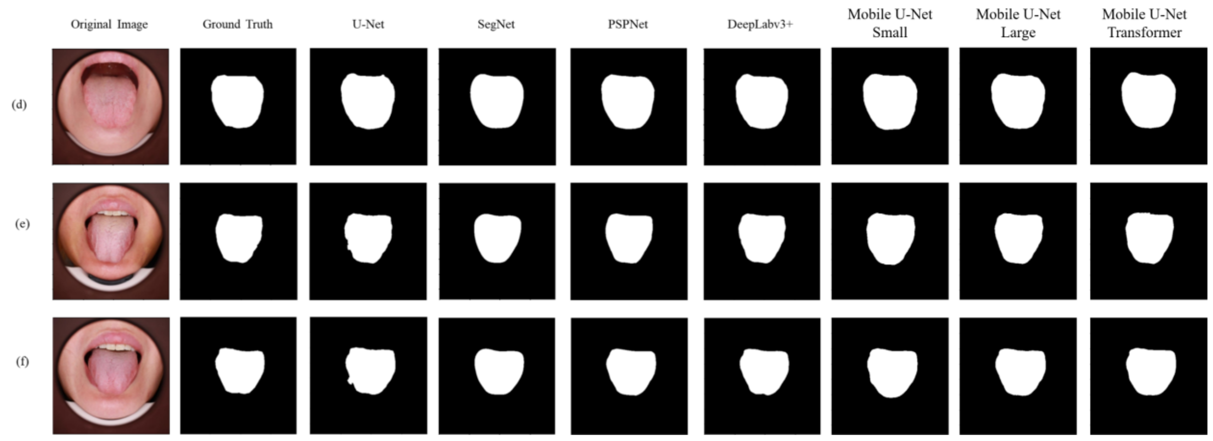


Figure 5.2 Comparison of Segmentation Results in Roboflow dataset

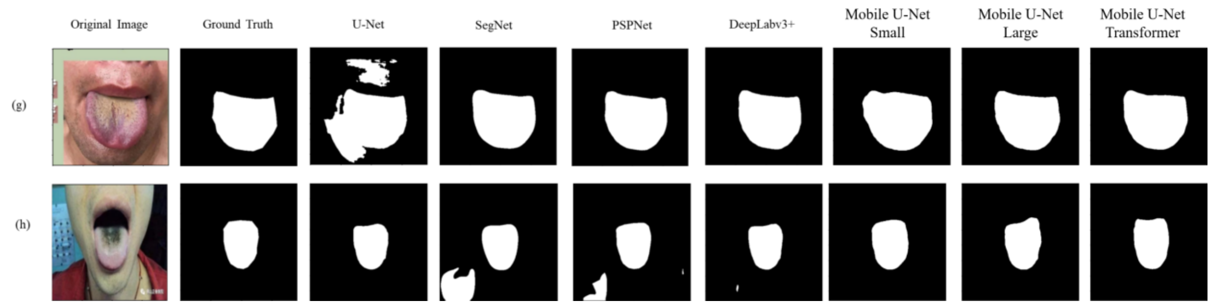


Figure 5.3 Comparison of Segmentation Results in PaddlePaddle dataset

For the BioHit and Roboflow datasets which were captured under stable lighting and standardized acquisition, most models achieve accurate tongue delineation. Architectures such as PSPNet and DeepLabv3+ consistently preserve smooth contours and avoid over-segmentation into non-tongue regions. The Mobile U-Net (Large) shows comparable performance with boundaries that are visually indistinguishable from the ground truth in many cases despite using far fewer parameters. In contrast, U-Net occasionally produces irregular borders or incomplete segmentations especially around the tongue tip and lateral edges, therefore suggesting weaker robustness to subtle variations in texture.

In PaddlePaddle dataset, more complex challenges are presented as the lighting conditions are uncontrolled. This includes lips closely encircling the tongue and the presence of abnormal tongue textures such as black coatings. U-Net generally performs more prone to false positives in other scenarios but however performs comparatively well by effective segmenting abnormal tongue texture image. Its ability to preserve texture information allows it to highlight coated region better than other models. SegNet and PSPNet struggle in delineating the tongue boundary due to the red shirt of the patient which has a similar colour to the tongue. The Mobile U-Net with Transformer shows improved attention to global tongue shape but sometimes sacrifices fine-grained edge sharpness compared to the large variant.

Across all datasets, the qualitative findings align well with the quantitative results reported earlier. DeepLabv3+ provides the most consistent segmentation quality, while Mobile U-Net (Large) achieves a favorable balance between accuracy and efficiency. Notably, Mobile U-Net (Small) demonstrates its potential for lightweight deployment as it remains visually competitive although its boundaries appear slightly less smooth.

5.2 Binary Classification Experimental Results

5.2.1 Quantitative Evaluation

Table 5.6 Binary Classification Experiment Results

Model	**Number of Parameter (10^6)**	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
ResNet18	11.18	97.01	0.9701	0.9701	0.9701
ResNet20	**0.28**	89.05	0.8946	0.8903	0.8902
ResNet50	23.85	**99.00**	**0.9903**	**0.9900**	**0.9900**
VGG19	20.09	96.52	0.9652	0.9652	0.9652
AlexNet	16.86	95.52	0.9557	0.9551	0.9552
EfficientNetV2-B0	6.08	96.52	0.9652	0.9651	0.9652
EfficientNetV2-S	20.50	97.51	0.9751	0.9751	0.9751

LECA-EfficientNetV2-B0	1.72	98.51	0.9856	0.9850	0.9851
LECA-EfficientNetV2-S	5.97	**99.00**	0.9900	**0.9900**	**0.9900**
Efficient-ResNet	0.31	98.51	0.9856	0.9850	0.9851

Table 5.6 shows the experiment results of the binary classification task. Among the conventional backbones, ResNet50 achieved the highest performance by reaching an accuracy of 99.00% with precision, recall, and F1-score values of 0.99. This confirms its strong feature extraction capability but comes at the cost of a relatively high parameter count (23.85M). ResNet18 and VGG19 also produced strong results (97.01% and 96.52% accuracy, respectively), though their efficiency-to-accuracy trade-off is less favourable compared to newer architectures. AlexNet despite being historically significant, showed lower accuracy (95.52%) and demonstrates limitations in handling complex tongue image features due to its shallower design.

The EfficientNet family displayed strong balance between accuracy and model size. EfficientNetV2-B0 achieved 96.52% accuracy with only 6.08M parameters, while EfficientNetV2-S reached 97.51% with 20.50M parameters, which the results are comparable with the deeper ResNet variants. The introduction of LECA (Lightweight Efficient Channel Attention) further enhanced performance. LECA-EfficientNetV2-B0 improved to 98.51% accuracy with just 1.72M parameters, while LECA-EfficientNetV2-S matched ResNet50's 99.00% accuracy but with only uses 25% of the parameters compared to ResNet50 (5.97M vs. 23.85M). These results demonstrate the effectiveness of channel attention in improving discriminative power without substantially increasing complexity.

The Efficient-ResNet also proved highly competitive, achieving 98.51% accuracy with only 0.31M parameters, making it the most parameter-efficient model in this experiment. This suggests that combining residual learning with efficient scaling strategies provides a promising direction for lightweight tongue image classification, especially for mobile or resource-constrained clinical applications.

In overall, the results highlight three key findings. First, deeper conventional CNNs such as ResNet50 remain strong performers but are computationally expensive. Second, EfficientNet-based models offer superior efficiency-accuracy trade-offs, particularly when enhanced with

LECA. Third, hybrid lightweight models (Efficient-ResNet, LECA-EfficientNetV2-B0) achieve near state-of-the-art accuracy with minimal parameter counts, making them well-suited for deployment in real-time or embedded diagnostic systems.

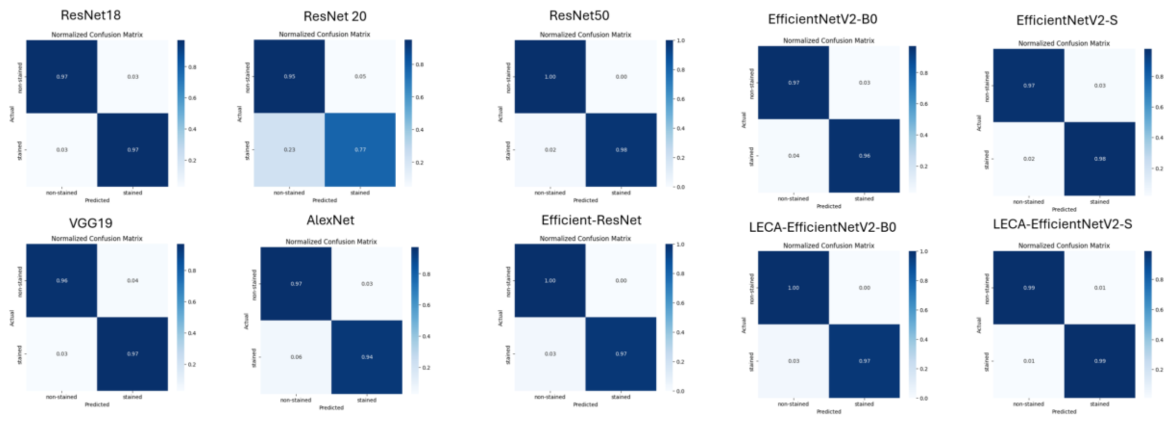


Figure 5.4 Normalized Confusion Matrix for Each Model

Figure 5.4 shows the normalized confusion matrix for each model. The normalized confusion matrix provides additional insight into the classification behaviour of each model in terms of false positive and false negative tendencies.

Among classical networks, VGG19 maintained solid performance, with 96% and 97% correct predictions for non-stained and stained tongues respectively. On the other hand, AlexNet showed weaker performance particularly for stained tongues, where the accuracy dropped to 94%, with 6% misclassified. This reflects the limitations of early CNN designs in capturing subtle tongue coating features compared to deeper or more efficient networks.

ResNet18 demonstrated a balanced outcome, with both non-stained and stained classes achieving a high recognition rate of 97%, and only 3% misclassification for each class. This indicates that the network was able to generalize well without strong bias toward either category. In contrast, ResNet20 exhibited clear limitations, particularly in classifying stained samples. While non-stained tongues were correctly identified 95% of the time, the accuracy for stained tongues dropped to 77%, with 23% being misclassified as non-stained. This imbalance suggests that ResNet20 struggled to capture the distinctive features of stained tongues, potentially due to its relatively shallow depth compared to other models. In addition, ResNet50 outperformed its smaller counterparts, achieving nearly perfect recognition with 100% accuracy for non-stained tongues and 98% for stained tongues.

The EfficientNet family, including the standard EfficientNetV2 models, the LECA-enhanced variants and the hybrid Efficient-ResNet, consistently demonstrated strong performance in the tongue classification task. The baseline EfficientNetV2-B0 achieved 97% accuracy for non-stained and 96% for stained tongues which offer balanced and reliable classification. Building on this, EfficientNetV2-S improved stained tongue recognition to 98% while maintaining 97% accuracy for non-stained samples. These findings emphasize the strength of the EfficientNet scaling strategy which achieves efficient yet powerful feature extraction.

The LECA-enhanced EfficientNetV2 models further advanced performance by incorporating local enhancement modules. LECA-EfficientNetV2-B0 matched the strongest ResNet and hybrid networks, achieving 100% accuracy for non-stained tongues and 97% for stained tongues. Most notably, LECA-EfficientNetV2-S achieved the highest accuracy overall, with 99% recognition for both classes and only 1% misclassification. This demonstrates that local enhancement strategies significantly strengthen the network's capacity to capture subtle coating variations, providing superior robustness and generalization.

The hybrid Efficient-ResNet also produced highly competitive results, achieving perfect recognition of non-stained tongues (100%) and 97% accuracy for stained samples. Its performance was comparable to ResNet50, suggesting that the integration of EfficientNet scaling with ResNet residual connections enhances both accuracy and generalization.

In overall, grouping the models highlights clear trends. Classical architectures like VGG19 provided stable but slightly lower results compared to modern designs. In contrast, deeper and more advanced models particularly ResNet50 and the EfficientNet family, consistently achieved high accuracy with minimal misclassification. The LECA-EfficientNetV2-S stood out as the best overall performer, demonstrating near-perfect classification across both classes. These findings underscore the importance of network depth, architectural refinements, and feature enhancement modules in achieving reliable tongue diagnosis, with modern EfficientNet-based designs offering the greatest potential for clinical applications.

5.2.2 Performance Comparison with Previous Researchers

Table 5.7 Performance Comparison with Previous Researchers in Binary Classification

Model	**Number of Parameter (10^6)**	**Accuracy (%)**	**Recall (%)**	**F1-Score (%)**
ResNet50 [4]	25.64	92.00	91.00	92.00
ResNet18 [12]	11.70	95.50	91.20	94.90
ResNet50 [12]	25.64	95.00	92.30	94.40
VGG19 [12]	144.00	92.50	94.50	92.00
AlexNet [12]	62.38	95.50	93.40	95.00
Efficient-ResNet	**0.31**	**98.51**	**98.56**	**98.50**

The results in Table 5.7 demonstrate that the proposed Efficient-ResNet model significantly outperforms previous state-of-the-art architectures in binary classification tasks. While traditional deep learning models such as ResNet50 [4], ResNet18 [12], and VGG19 [12] achieved strong performance with accuracies ranging from 92.00% to 95.50%, their parameter sizes remain relatively large, ranging between 11.70M and 144.00M. Similarly, AlexNet [12] achieved an accuracy of 95.50% with 62.38M parameters. In contrast, the Efficient-ResNet achieves the highest accuracy of 98.51%, recall of 98.56%, and F1-score of 98.50% with only 0.31M parameters. This highlights a substantial improvement in both predictive performance and computational efficiency. The reduction in model complexity without compromising accuracy demonstrates the advantage of integrating efficiency-oriented architectural design, making Efficient-ResNet highly suitable for resource-constrained environments and real-time clinical applications.

5.2.3 Visualization and Benchmark

To investigate the specific visual features emphasized by each CNN model during the tongue classification task, a total of 4 tongue images (2 stained and 2 non-stained coating samples) were randomly selected. The Grad-CAM visualizations for these samples reveal notable differences in the attention distribution across models. Figure 5.5 shows the stained tongue coating samples generated by each CNN model.

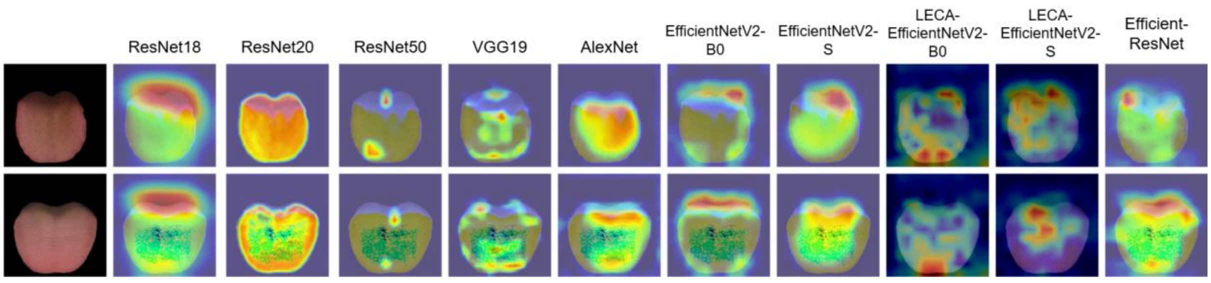


Figure 5.5 Stained Tongue Coatings

ResNet18 produces broad and diffuse activations that span most of the tongue surface, often extending into non-relevant areas. While this indicates sensitivity to overall colour and texture variations, the lack of precise localization reduces its interpretability. ResNet20, despite its lightweight design, shows even weaker and noisier activations, with scattered attention maps that fail to consistently highlight stained regions. This suggests that while the smaller architecture reduces computational cost, it compromises the model's ability to capture discriminative pathological features. ResNet50 exhibits more concentrated activations within central tongue regions, therefore effectively reducing background interference compared to ResNet18, although it occasionally neglects diagnostically relevant peripheral coating areas.

VGG19 demonstrates sharply localized hotspots, suggesting strong attention to specific coating regions. However, this narrow focus may fail to capture the broader spatial patterns necessary for accurate classification. AlexNet displays widespread activations with prominent hotspots in the central and upper tongue regions, capturing general shape and texture differences but with less selective targeting as it often includes large non-critical areas.

The EfficientNetV2-B0 model achieves a balanced activation pattern by attending to multiple coating areas without excessive background interference, indicating that it effectively integrates local detail with broader context, though occasional edge activations persist. EfficientNetV2-S improves upon this by producing more precise and smaller hotspots while still preserving adequate contextual coverage, offering a strong compromise between localization accuracy and completeness.

The LECA-modified models exhibit higher selectivity. LECA-EfficientNetV2-B0 concentrates attention on small and well-defined patches, potentially enhancing fine-grained texture analysis but risking the omission of diffuse coating regions. LECA-EfficientNetV2-S maintains this selectivity while distributing activations more coherently across the tongue, resulting in better coverage than its B0 counterpart. This combination of sharpness and coherence demonstrates the interpretability advantage of LECA integration, particularly when analyzing clinically subtle coating patterns.

The Efficient-ResNet hybrid shows moderate activations that capture the coated regions more reliably than classical CNNs, but its attention tends to be broader and less sharply defined compared to EfficientNetV2, indicating a trade-off between efficiency and localization precision.

Overall, these findings demonstrate that the LECA-enhanced models not only achieve parameter efficiency but also offer superior interpretability in the context of tongue coating recognition. Their ability to specifically highlight pathological coatings rather than broadly activating across the tongue surface, thus aligns with clinical expectations and strengthens their potential applicability in real-world diagnostic settings.

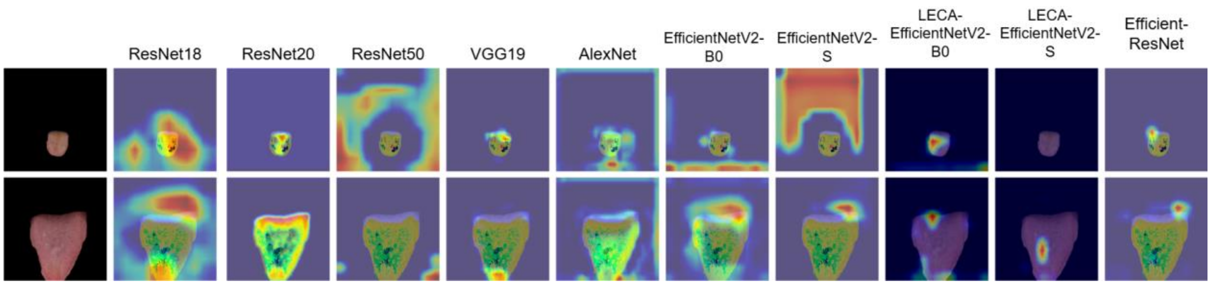


Figure 5.6 Non-Stained Tongue Coatings

For non-stained tongues, the Grad-CAM visualizations show clear differences in how each model allocates attention when no pathological coating is present. ResNet18 again produces broad activations and often extending beyond the tongue surface into irrelevant background areas. This overgeneralization suggests that the model relies on overall shape and edge cues rather than subtle texture absence which reducing its interpretability in negative cases. Besides that, ResNet20 generates activations that cover almost the entire tongue surface, indicating a tendency to capture global shape and textural patterns rather than selectively attending to diagnostically important coating areas. ResNet50 also exhibits considerable attention outside the tongue region, suggesting suboptimal localization and possible reliance on peripheral artifacts instead of true diagnostic areas. VGG19 shows more confined activation within the tongue boundary, but the hotspots are narrowly concentrated in small regions, potentially missing distributed non-stained patterns.

AlexNet generates sparse activations with inconsistent focus, occasionally highlighting irrelevant peripheral zones. EfficientNetV2-B0 also demonstrates poor localization with its most intense activations extending outside the tongue surface and along image borders, thus reducing its interpretability. Besides that, EfficientNetV2-S performs similarly poorly in this context with strong activations largely outside the tongue region.

The LECA-modified EfficientNetV2 models achieve the most precise localization. LECA-EfficientNetV2-B0 produces small, well-defined activations primarily within the tongue boundary, and LECA-EfficientNetV2-S maintains coherent attention across the tongue body with minimal background influence.

On the other hand, Efficient-ResNet consistently demonstrates activations that extend beyond the tongue into background areas. This suggests reduced robustness, as the model may depend on external image artefacts rather than medically relevant tongue features, limiting its interpretability in non-stained cases as well.

Table 5.8 Visualization Benchmark for Binary Classification

Model	**Pointing Game**
ResNet18	0.4579
ResNet20	**0.9801**
ResNet50	0.5670
VGG19	0.8756
AlexNet	0.5771
EfficientNetV2-B0	0.6318
EfficientNetV2-S	0.5821
LECA-EfficientNetV2-B0	0.7662
LECA-EfficientNetV2-S	0.7622
Efficient-ResNet	0.6135

The pointing game results provide a quantitative measure of the alignment between Grad-CAM activations and clinically relevant tongue regions. Interestingly, ResNet20 achieves the highest score (0.9801), far surpassing both shallow and deeper ResNet variants. This suggests that although the qualitative Grad-CAM inspection revealed diffuse activation across the entire tongue, the model consistently covers the ground-truth annotated region, leading to a strong pointing game score. However, this highlights a limitation of the metric: high coverage does not necessarily imply precise or selective localization but rather that the relevant region is encompassed within the broad activations.

VGG19 also performs strongly (0.8756), which aligns with its sharply localized Grad-CAM hotspots that tend to fall within stained or diagnostically relevant regions. AlexNet (0.5771) and ResNet50 (0.5670) show more modest scores, reflecting their tendency to generate broader or less focused activations that do not consistently overlap with annotated areas.

EfficientNetV2 models achieve moderate scores, with EfficientNetV2-B0 (0.6318) outperforming EfficientNetV2-S (0.5821) despite the latter showing better quantitative classification performance. This indicates that while EfficientNetV2-S is more accurate for classification, its attention maps are less well aligned with annotated regions, pointing to a potential trade-off between predictive accuracy and interpretability.

The LECA-modified models improve alignment, with LECA-EfficientNetV2-B0 and LECA-EfficientNetV2-S achieving 0.7662 and 0.7622, respectively. The improved scores correspond with qualitative observations that these models exhibit more selective and coherent activations across relevant tongue coating regions, demonstrating that lightweight channel attention enhances interpretability without sacrificing performance.

Efficient-ResNet achieves a moderate pointing game score (0.6135). While this indicates some ability to capture relevant regions, its Grad-CAM results frequently highlight background areas, which reduces interpretability despite the high classification accuracy observed earlier.

Overall, these results reveal that pointing game scores can complement but also contrast with qualitative interpretations. Models like ResNet20 achieve very high pointing game accuracy due to their extensive coverage of the tongue region, even though their interpretability is reduced by diffuse activations. In contrast, models such as VGG19 and LECA variants strike a better balance between precise localization and consistent region coverage, offering stronger evidence of clinically meaningful attention behavior.

5.2.4 Discussion

The binary classification experiments reveal important insights into the trade-offs between accuracy, efficiency, and interpretability across the tested architectures. ResNet50 and LECA-EfficientNetV2-S achieve the highest classification accuracy (99.00%), with consistently strong precision, recall, and F1-scores. These results demonstrate that deeper networks and models enhanced with lightweight channel attention can capture highly discriminative features for tongue coating classification. In comparison, ResNet18, AlexNet, and VGG19 deliver slightly lower accuracies (95–97%), reflecting the limitations of shallower or older architectures in modeling complex texture variations. ResNet20, despite its small parameter size, performs notably worse (89.05%), highlighting that excessive downsizing compromises feature extraction capacity and generalization.

Model efficiency also plays a significant role in practical applicability. Efficient-ResNet achieves high performance (98.51% accuracy) with only 0.31M parameters, while LECA-EfficientNetV2-B0 offers 98.51% accuracy at 1.72M parameters. These results indicate that lightweight models can approach the accuracy of larger architectures while remaining computationally efficient, making them well suited for mobile or resource-constrained diagnostic applications. By contrast, conventional models such as ResNet50 (23.85M parameters) or VGG19 (20.09M parameters) incur high computational costs, limiting their deployment in real-world point-of-care settings.

The visualization analysis through Grad-CAM highlights differences in model interpretability. ResNet18 and ResNet20 exhibit diffuse activations that spread across the entire tongue surface, capturing global shape and texture but failing to consistently isolate diagnostically relevant regions. ResNet50 and EfficientNetV2 variants provide more localized focus on coating regions, improving interpretability while maintaining classification accuracy. The LECA-modified EfficientNetV2 models further enhance attention selectivity, generating coherent and clinically meaningful activations across the tongue surface. However, Efficient-ResNet, despite its excellent efficiency, shows attention leakage into background regions, which raises concerns about robustness and reliance on artefactual cues.

The pointing game results (Table 5.8) complement these findings by quantitatively evaluating localization. ResNet20 achieves the highest score (0.9801), indicating that its broad coverage consistently overlaps with annotated regions, even though its qualitative interpretability remains limited. VGG19 also performs strongly (0.8756), reflecting its sharply localized hotspots within relevant regions. LECA-modified models outperform their baseline counterparts, confirming that lightweight attention improves the alignment of activations with clinically significant areas. Meanwhile, Efficient-ResNet and EfficientNetV2 variants achieve moderate scores (0.58–0.63), reflecting their tendency to balance classification accuracy with only partial localization fidelity.

Overall, the binary classification experiments demonstrate that while deeper or attention-enhanced models achieve the best performance, lightweight architectures can offer competitive accuracy with far fewer parameters, making them suitable for mobile diagnostic tools. At the same time, the visualization analysis underscores that interpretability does not always correlate directly with predictive performance, as models like ResNet20 achieve excellent pointing game accuracy despite poor classification results. This highlights the need to consider both quantitative and qualitative metrics when selecting models for clinically oriented applications.

5.3 Multi-Class Classification Experimental Results

5.3.1 Quantitative Evaluation

Table 5.9 Multi-Class Classification Experiment Results

Model	**Number of Parameter (10^6)**	**Accuracy (%)**	**Precision**	**Recall**	**F1-Score**
ResNet18	11.19	57.50%	0.7241	0.6253	0.5855
Resnet20	0.32	60.83%	0.6500	0.5155	0.5627
ResNet50	23.85	77.50%	0.8741	0.6431	0.7126
VGG19	20.09	76.67%	0.8587	0.6904	0.7528
AlexNet	16.86	65.83%	0.6196	0.6310	0.6147
EfficientNetV2-B0	6.08	83.33%	0.8317	0.7868	0.7990
EfficientNetV2-B3	13.13	**85.00%**	**0.8889**	0.7806	0.8127
EfficientNetV2-S	20.50	80.83%	0.8406	**0.7957**	**0.8157**
LECA-EfficientNetV2-B0	1.72	75.83%	0.7748	0.6622	0.7048
LECA-EfficientNetV2-B3	3.51	75.83%	0.7678	0.7264	0.7441

LECA-EfficientNetV2-S	5.97	79.17%	0.8035	0.7555	0.7738
Efficient-ResNet - Small	**0.30**	73.33%	0.7800	0.6841	0.7219
Efficient-ResNet - Medium	0.83	76.67%	0.7433	0.7775	0.7522
Efficient-ResNet - Large	1.16	79.17%	0.7594	0.7736	0.7658

The multi-class classification experiment reveals a clear performance gradient across different architectures, reflecting the trade-offs between model complexity, parameter count, and generalization ability. Among the classical convolutional backbones, ResNet18 and ResNet20 achieved relatively modest performance, with accuracies of 57.50% and 60.83% respectively. Despite ResNet20 being significantly smaller in size (0.32M parameters), its recall dropped to 0.5155, suggesting that while the lightweight design reduces computation, it also struggles to capture the diverse feature variations across multiple tongue categories. ResNet50 improved substantially, achieving 77.50% accuracy and a balanced F1-score of 0.7126 but its 23.85M parameters make it computationally expensive. Similarly, VGG19 reached 76.67% accuracy with a stronger F1-score of 0.7528 but its heavy parameter load (20.09M) limits efficiency. AlexNet despite having a moderate parameter size (16.86M), achieved only 65.83% accuracy, which reflects the limitations of early CNN architectures in capturing complex tongue features.

By contrast, the EfficientNetV2 family consistently outperformed the traditional baselines. EfficientNetV2-B0 and B3 achieved accuracies of 83.33% and 85.00%, with high F1-scores of 0.7990 and 0.8127 respectively, while requiring far fewer parameters than VGG19 or ResNet50. EfficientNetV2-S also demonstrated competitive performance (80.83%, F1-score 0.8157), showing that compound scaling effectively balances depth, width, and resolution for multi-class classification.

The LECA-modified EfficientNet variants displayed competitive but slightly reduced performance compared to their vanilla counterparts. LECA-EfficientNetV2-B0, B3, and S achieved accuracies between 75–79%, with F1-scores ranging from 0.7048 to 0.7738. Although their performance was lower than the standard EfficeintNetV2, their significantly reduced parameter sizes (1.72M–5.97M) highlight the benefit of lightweight adaptation, especially for deployment scenarios requiring efficiency.

The Efficient-ResNet models further confirmed the effectiveness of lightweight architectures. The small variant (0.30M parameters) achieved 73.33% accuracy with an F1-score of 0.7219, while the medium and large versions scaled performance to 76.67% and 79.17%, respectively. Notably, even the Efficient-ResNet-Large (1.16M parameters) outperformed ResNet18, ResNet20, and AlexNet by a large margin, despite being substantially smaller. This underscores the advantage of hybrid design principles that integrate EfficientNet's scaling with ResNet's residual learning.

Overall, the results suggest that EfficientNetV2-B3 provides the best trade-off between accuracy and reliability, achieving the highest classification performance (85.00%, F1 = 0.8127). However, the Efficient-ResNet demonstrates strong potential as compact alternatives which delivers competitive performance with dramatically fewer parameters. This indicates that lightweight models may be more suitable for real-world clinical applications where computational resources are limited, while larger EfficientNetV2 variants remain advantageous in research settings prioritizing accuracy.

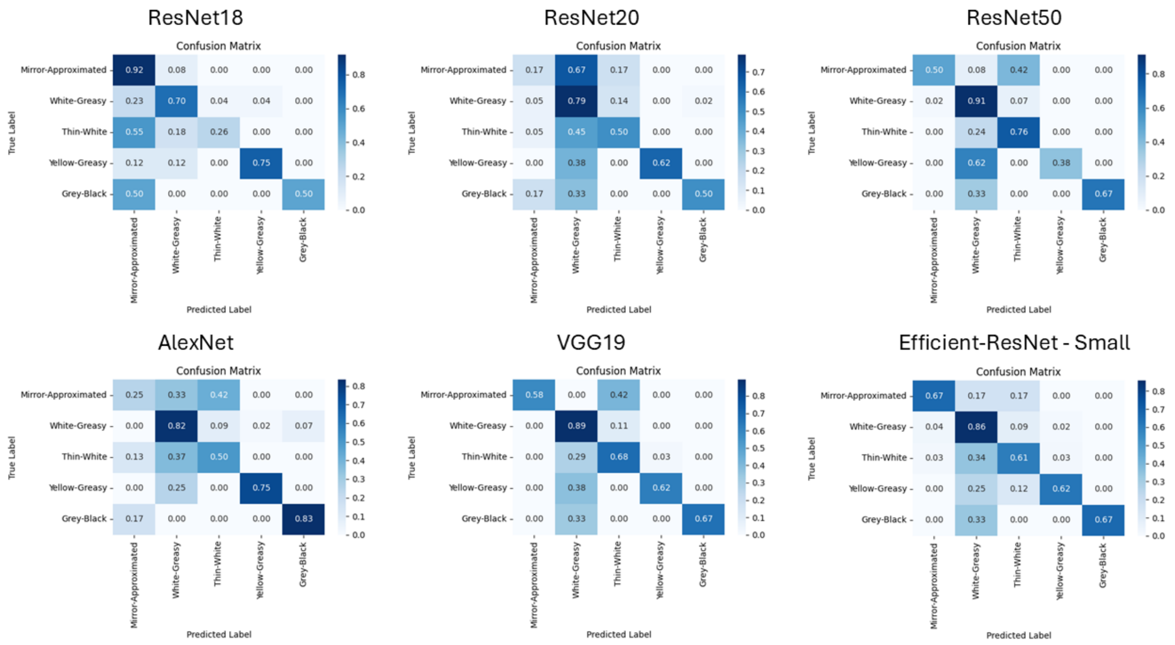


Figure 5.7 Normalized Confusion Matrix for Each Model

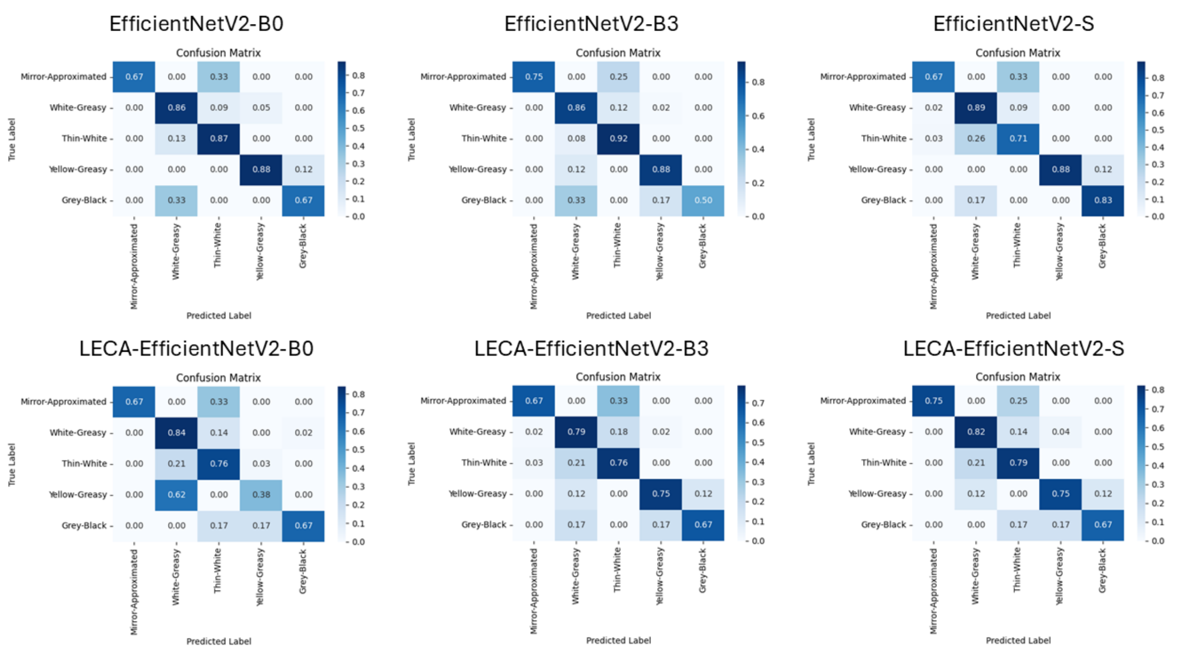


Figure 5.8 Normalized Confusion Matrix for Each Model

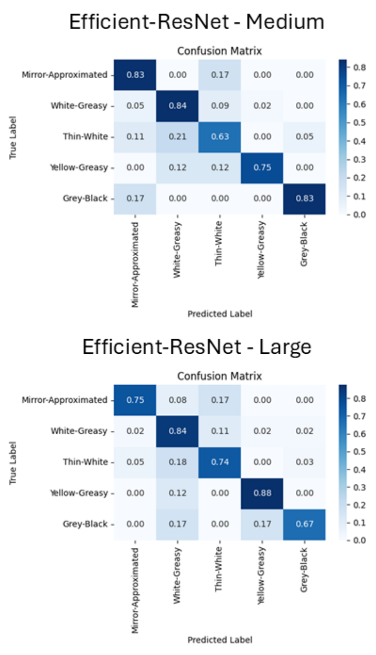


Figure 5.9 Normalized Confusion Matrix for Each Model

Figure 5.7 to Figure 5.9 show the normalized confusion matrix for each model. The traditional convolutional neural networks, comprising the ResNet family, VGG19, and AlexNet, demonstrated varied capabilities in handling the tongue coating classification task. Among the ResNet variants, ResNet18 achieved strong recognition in Mirror-Approximated (0.92) and Yellow-Greasy (0.75) but exhibited frequent misclassification in Thin-White and Grey-Black, often confusing them with Mirror-Approximated. ResNet20 maintained moderate performance in White-Greasy (0.79) and Yellow-Greasy (0.62), though substantial confusion remained between Mirror-Approximated and Thin-White. The deeper ResNet50 improved stability across classes, performing particularly well in White-Greasy (0.91) and Thin-White (0.76), while still showing some overlap between visually similar categories.

In contrast, AlexNet, despite being a shallower network, delivered strong recognition of Grey-Black (0.83) and White-Greasy (0.82). However, its performance in Mirror-Approximated was limited with predictions heavily distributed across other categories. VGG19 performed reliably in White-Greasy (0.89) and Thin-White (0.68) but struggled with Mirror-Approximated (0.58), where frequent confusion with Thin-White suggested challenges in separating reflective from light-colored coating textures. Collectively, these traditional CNNs highlighted the trade-off between network depth and classification consistency, with deeper models like ResNet50 and VGG19 offering more stable performance, while AlexNet showed notable limitations in feature discrimination.

The EfficientNetV2 models consistently outperformed the traditional CNNs, showing higher accuracy across most coating categories. EfficientNetV2-B0 delivered strong recognition in Thin-White (0.87) and Yellow-Greasy (0.88), while EfficientNetV2-B3 further improved Thin-White classification (0.92), reflecting the advantages of compound scaling. EfficientNetV2-S maintained balanced accuracy across all classes, excelling in Grey-Black (0.83) compared to earlier networks. The LECA variants introduced slight shifts in performance. While LECA-EfficientNetV2-B0 maintained high performance in Thin-White (0.76), its Yellow-Greasy recognition dropped due to misclassifications. Conversely, LECA-EfficientNetV2-S provided more balanced results, achieving higher performance in Mirror-Approximated (0.75) and strong consistency across other categories. These results confirm the strength of EfficientNet architectures in capturing fine-grained coating variations, particularly for Thin-White and Yellow-Greasy.

In the Efficient-ResNet family, Efficient-ResNet-Small achieved robust performance in White-Greasy (0.86) and Grey-Black (0.67), though some confusion persisted between Thin-White and White-Greasy. Efficient-ResNet-Medium improved recognition for Mirror-Approximated (0.83) and Grey-Black (0.83), showing balanced performance across categories but slightly weaker accuracy in Thin-White (0.63). The Efficient-ResNet-Large produced the most consistent results within this group, excelling in Yellow-Greasy (0.88) and White-Greasy (0.84), while maintaining good recognition across the other categories. Overall, the Efficient-ResNet family provided a reliable balance between efficiency and accuracy, outperforming traditional CNNs in stability while showing slightly less fine-grained sensitivity compared to EfficientNetV2 models.

5.3.2 Visualization and Benchmark

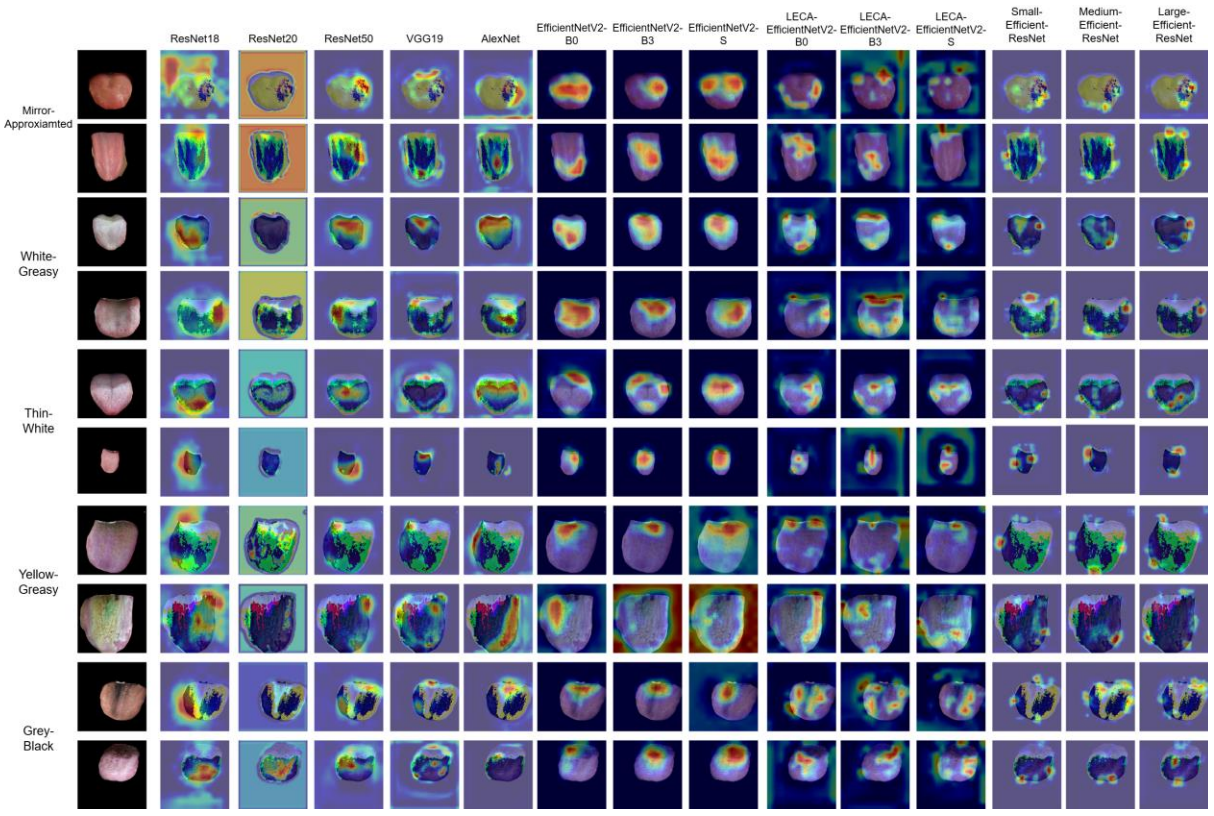


Figure 5.10 Grad-CAM Heatmaps for Multi-Class Tongue Coatings

From the full-class visualization as shown in Figure 5.10, it is evident that the models exhibit substantial variation in their focus regions. ResNet18 and ResNet20 frequently highlight peripheral or irrelevant areas, particularly in Thin-White and Grey-Black samples which reflecting unstable localization. In contrast, ResNet50 demonstrates stronger focus with heatmaps concentrated on the central coating textures, suggesting improved discriminative ability. On the other hand, AlexNet produces more diffuse activations which indicates weaker capability in isolating class-specific cues.

The VGG19 model demonstrates more structured attention compared to the ResNet family. Its activations are especially well aligned with coating patterns in White-Greasy and Yellow-

Greasy categories, where heatmaps highlight the defining surface textures. However, VGG19 often extends its focus beyond the gloss features in Mirror-Approximated samples , this shows that VGG19 had limited robustness when handling reflective surfaces. The EfficientNetV2 family (B0, B3, and S) performs consistently well, with compact and centralized activations across all classes. These models excel at capturing fine discriminative details such as the subtle surface patterns of Thin-White and Yellow-Greasy coatings.

In contrast, the LECA-EfficientNetV2 variants display more distributed attention patterns, with heatmaps often covering broader regions of the tongue surface. This suggests an emphasis on contextual information, which improves coverage but sometimes reduces pinpoint localization. The effect is especially visible in Mirror-Approximated samples, where activations spread across the entire surface instead of isolating gloss regions. The Efficient-ResNet family (Small, Medium, and Large) shows intermediate behavior, balancing between the scattered focus of ResNets and the sharp localization of EfficientNets.

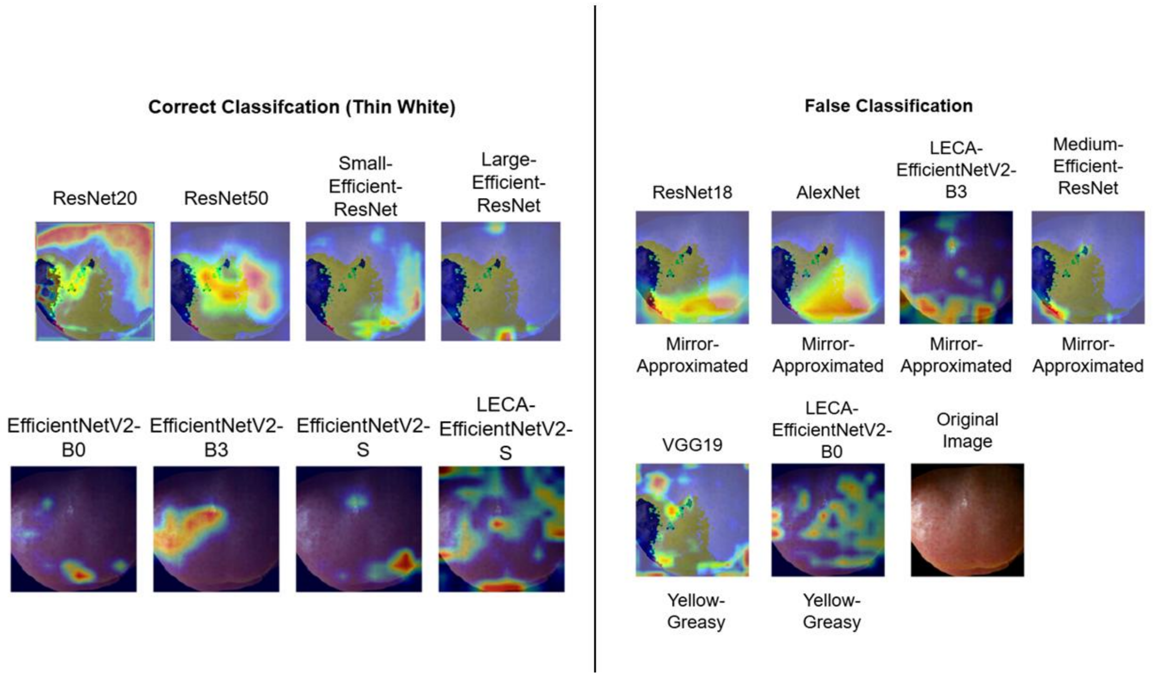


Figure 5.11 Grad-CAM Visualization by Prediction

Figure 5.11 shows Grad-CAM visualization by categorising the prediction of Thin-White tongue so that the comparison between correct and misclassified samples provides additional understanding of model behaviour. In correctly classified Thin-White samples, models such as ResNet50, EfficientNetV2-B0, focus strongly on the lighter central surface of the tongue. Their sharp and localized activations indicate reliable detection of Thin-White's defining coating features. Conversely, misclassified cases reveal systematic weaknesses. ResNet18, AlexNet, and LECA-EfficientNetV2-B3 frequently confuse Thin-White with Mirror-Approximated, as their attention shifts toward glossy edges rather than coating density. Similarly, VGG19 and LECA-EfficientNetV2-B0 sometimes misinterpret Thin-White as Yellow-Greasy, with activations extending into peripheral color variations instead of focusing on central whiteness. Medium-Efficient-ResNet also displays scattered activations, which contributes to frequent confusion with Mirror-Approximated.

Table 5.10 Visualization Benchmark for Multi-Class Classification

Model	**Pointing Game**
ResNet18	0.8273
ResNet20	0.8000
ResNet50	0.9455
VGG19	0.9545
AlexNet	0.8818
EfficientNetV2-B0	0.9545
EfficientNetV2-B3	0.9545
EfficientNetV2-S	**0.9727**
LECA-EfficientNetV2-B0	0.8364
LECA-EfficientNetV2-B3	0.9000
LECA-EfficientNetV2-S	0.9000
Efficient-ResNet - Small	0.8091
Efficient-ResNet - Medium	0.8273
Efficient-ResNet - Large	0.7818

Table 5.10 shows the visualization benchmark for multi-class classification. Among the ResNet family, ResNet50 achieved the highest pointing game score (0.9455), reflecting improved localization compared to ResNet18 (0.8273) and ResNet20 (0.8000). This suggests that deeper residual connections enhance the capacity to capture discriminative features relevant to tongue coatings. In contrast, shallower variants tended to spread activations across broader, less specific regions, limiting interpretability.

VGG19 and EfficientNetV2 models demonstrated the strongest interpretability, with scores exceeding 0.95 in most cases. In particular, EfficientNetV2-S achieved the highest score (0.9727), indicating consistently precise localization of diagnostically relevant regions. This highlights the benefit of compound scaling in EfficientNet, which balances depth, width, and resolution to improve both performance and explainability.

The LECA-enhanced EfficientNetV2 variants produced competitive results, though their scores (0.8364–0.9000) were slightly lower than the EfficientNetV2 variants. This outcome suggests that while LECA modifications reduce parameter count and preserve predictive accuracy, they may trade off some localization precision in the visualization benchmark.

The Efficient-ResNet family showed mixed results with pointing game scores ranging between 0.7818 and 0.8273. Although efficient in parameter usage, these models did not achieve the same interpretability level as deeper conventional CNNs or EfficientNetV2 variants. Their moderate scores reflect a balance between efficiency and localization but indicate room for improvement in attention alignment with clinically relevant coating areas.

In summary, the visualization metrics highlight a trade-off between precision and coverage. Models such as EfficientNetV2-B0 and EfficientNetV2-S excel in pinpoint accuracy, effectively localizing key coating regions, but may underrepresent peripheral yet clinically relevant features. Conversely, the LECA variants provide broader coverage at the expense of fine-grained precision. ResNet50 and VGG19 achieve a balanced performance, maintaining both strong localization and comprehensive region coverage. These findings suggest that model selection should be guided by the intended clinical application, particularly whether exact localization of a focal feature or more expansive coverage of the coating region is prioritized.

5.3.3 Multi-Class Classification with Segmentation Preprocessing

Table 5.11 Experiment Results of Multi-Class Classification with Segmentation

Segment	Classification	Accuracy	Precision	Recall	F1-Score
Mobile U-Net - Small	ResNet50	72.50	**0.8760**	0.5999	0.6802
	EfficientNetV2-B0	**79.17**	0.7983	**0.7571**	**0.7750**
	LECA-EfficientNetV2-B0	70.83	0.7093	0.6173	0.6449

	LECA-EfficientNetV2-S	76.67	0.7757	0.6905	0.7234
	Efficient-ResNet - Small	61.67	0.7177	0.5271	0.5880
	Efficient-ResNet - Medium	67.50	0.6913	0.6706	0.6771
	Efficient-ResNet - Large	60.00	0.5220	0.5315	0.5223
Mobile U-Net with Transformer	ResNet50	73.33	**0.8856**	0.5821	0.6582
	EfficientNetV2-B0	**78.33**	0.7757	**0.7683**	**0.7703**
	LECA-EfficientNetV2-B0	73.33	0.7872	0.6675	0.6934
	LECA-EfficientNetV2-S	75.83	0.7602	0.7019	0.7234
	Efficient-ResNet - Small	66.67	0.6989	0.6146	0.6496
	Efficient-ResNet - Medium	70.00	0.6481	0.7211	0.6714
	Efficient-ResNet - Large	64.17	0.6337	0.6164	0.6052

The results in Table 5.11 show distinct trends when combining segmentation models with different classifiers. Under the Mobile U-Net – Small configuration, EfficientNetV2-B0 achieved the highest overall accuracy (79.17%) and the most balanced performance across all metrics, with an F1-score of 0.7750. Although ResNet50 delivers the highest precision (0.8760), the model suffered from low recall (0.5999) and resulting in a lower F1-score of 0.6802. The LECA variants under this segmentation backbone showed moderate results;

LECA-EfficientNetV2-S performed better (accuracy 76.67%, F1-score 0.7234) than LECA-EfficientNetV2-B0 (accuracy 70.83%, F1-score 0.6449). In contrast, the Efficient-ResNet family demonstrated varying effectiveness. The medium variant achieved a fair balance with an F1-score of 0.6771, outperforming both the small and large variants, which showed lower accuracies (61.67% and 60.00%) and weaker F1-scores (0.5880 and 0.5223).

When using Mobile U-Net with Transformer, performance patterns were largely consistent. EfficientNetV2-B0 again produced strong and balanced results, with 78.33% accuracy and an F1-score of 0.7703, slightly lower than the U-Net–Small case but still stable. ResNet50 once more achieved the highest precision (0.8856) but was limited by low recall (0.5821), giving an F1-score of 0.6582. LECA-EfficientNetV2 models maintained moderate performance; LECA-EfficientNetV2-S achieved 75.83% accuracy and an F1-score of 0.7234, matching its performance under U-Net–Small, while LECA-EfficientNetV2-B0 scored slightly lower at 73.33% accuracy and 0.6934 F1. The Efficient-ResNet family again displayed similar trends, where the medium variant outperformed the small and large versions. The medium variant achieved 70.00% accuracy with an F1-score of 0.6714, compared to the small (66.67% accuracy, 0.6496 F1) and large variants (64.17% accuracy, 0.6052 F1).

In overall, the analysis indicates that EfficientNetV2-B0 consistently delivered the best balance of accuracy, precision, recall, and F1-score across both segmentation backbones. ResNet50 showed recall limitations although excelling in precision and the Efficient-ResNet family demonstrated more reliable performance in the medium variant compared to its small and large counterparts.

5.3.4 Discussion

The multi-class classification experiments highlight important trade-offs between model accuracy, efficiency, and interpretability. EfficientNetV2-B3 achieved the highest accuracy (85.00%) with strong precision and F1-score, which demonstrates its effectiveness in handling fine-grained tongue coating categories. EfficientNetV2-B0 and EfficientNetV2-S also delivered robust results (83.33% and 80.83% accuracy), confirming the scalability of this family of models. Traditional CNNs such as ResNet50 (77.50%) and VGG19 (76.67%) performed competitively but required substantially more parameters, while AlexNet and shallow ResNets (ResNet18 and ResNet20) underperformed, reflecting their limited ability to capture complex coating variations.

Model efficiency was also an important factor. The proposed Efficient-ResNet family offered competitive accuracies (73–79%) with extremely low parameter counts (0.30–1.16M), significantly reducing computational overhead compared to larger models. Notably, Efficient-ResNet-Large achieved 79.17% accuracy with only 1.16M parameters, outperforming LECA-EfficientNetV2-B0 (75.83% at 1.72M parameters). This demonstrates that Efficient-ResNet can deliver superior accuracy at a smaller computational footprint, making it a strong candidate for mobile or resource-constrained applications. However, compared to EfficientNetV2-B3, these lightweight variants showed a modest drop in performance, suggesting that efficiency gains still come at the cost of representational depth.

The visualization analysis provided additional insights into interpretability. VGG19 and EfficientNetV2-S achieved the highest pointing game scores (0.9545 and 0.9727), reflecting their ability to attend to diagnostically relevant coating regions. In contrast, Efficient-ResNet models obtained lower scores between 0.7818 to 0.8273, suggesting weaker spatial focus despite their strong efficiency profile. These findings indicate that while Efficient-ResNet is competitive in predictive accuracy, further refinement of its attention mechanisms may be required to improve clinical interpretability.

In addition, segmentation-assisted experiments revealed mixed outcomes. When paired with Mobile U-Net or Mobile U-Net with Transformer, EfficientNetV2-B0 consistently maintained high accuracy (79.17% and 78.33%), while Efficient-ResNet models experienced notable performance degradation particularly in the large variant. This suggests that EfficientNetV2 is more robust to pre-processed segmentation inputs, whereas Efficient-ResNet may be more sensitive to variations in input feature quality.

Overall, the multi-class classification results demonstrate that EfficientNetV2 models remain the strongest choice for accurate and interpretable tongue coating categorization. At the same time, Efficient-ResNet provides a lightweight alternative with favorable efficiency–accuracy trade-offs, and importantly, shows superiority over LECA-based models at similar parameter sizes. These findings highlight the potential of Efficient-ResNet as a practical model for deployment in real-world diagnostic workflows.

5.4 Project Challenges

During the implementation of the deep learning-based tongue classification model, several challenges and limitations arise due to the constraints of using the free-tier Google Colab environment. The primary issues were out-of-memory (OOM) errors and limited computing resources. The available hardware consisted of 12.7 GB of system memory and 15.0 GB of GPU memory, which posed a significant limitation when training memory-intensive deep architectures such as the VGG19 model. The architecture requires substantial memory when increasing the batch size to accelerate convergence and improve generalization. To avoid OOM errors, the batch size had to be restricted to 16-32, which not only prolonged training time but also introduced potential instability in model performance due to smaller gradient estimates.

In addition, the free-tier Google Colab service imposes session timeouts and daily GPU usage limits. The T4 GPU allocation was limited to approximately 4 hours per day. Although a checkpointing mechanism was implemented in this study to allow resumption of interrupted sessions, the restricted runtime still limit the ability to perform extensive hyperparameter tuning experiments, thereby constraining the exploration of optimal model configurations.

Finally, a technical limitation was encountered when applying the Heatmap Assisted Accuracy Score (HAAS) for model's visualization evaluation. According to [12], the tongue coating images should be normalized to the range of [-1,1] before computing HAAS values for the Class Activation Map (CAM) algorithms on the test dataset. However, because of the generated heatmaps often contained substantial blue regions, the normalization process caused these areas to appear greyish when overlaid onto the original images. This colour distortion reduced the visibility of important coating features such as subtle colour and texture variations, which in turn misled the model's feature recognition and introduced a bias toward coating categories with less distinct boundaries such as thin-white and white-greasy. Figure 5.12 shows the example of Heatmap-Assisted Image for model prediction.

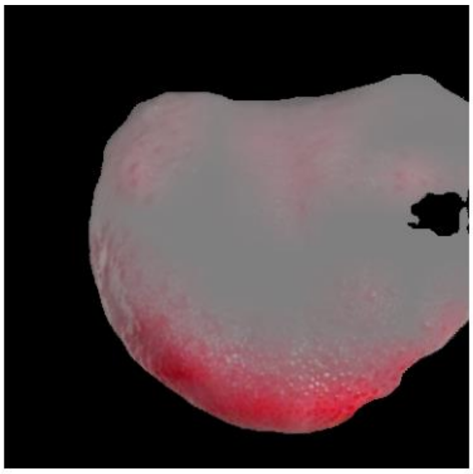


Figure 5.12 Example of Heatmap-Assisted Image

5.5 Objective Evaluation

The main objective of this study was to develop and evaluate deep learning-based framework for tongue image analysis to assist Traditional and Complementary Medicine diagnosis and improve diagnostic accuracy. This was accomplished through the achievement of four sub-objectives as evaluated below.

First, a Mobile U-Net architecture was developed to perform tongue image segmentation. By combining U-Net's spatial localization strength with MobileNet's lightweight and efficient feature extraction, the model achieved fast and accurate segmentation of tongue regions. This step ensured that subsequent classification was performed on well-isolated tongue areas, reducing the influence of irrelevant background features.

Second, development of binary and multi-class classification models for tongue coatings. Deep learning models were successfully developed for both binary classification (distinguishing stained from non-stained tongue coatings) and multi-class classification (categorizing pathological tongue coatings into specific types). The implemented architectures included ResNet18, ResNet50, VGG19, AlexNet, EfficientNetV2 variants, LECA-based EfficientNetV2 variants as well as Efficient-ResNet family. Results showed that EfficientNetV2-B3 and EfficientNetV2-S achieved the strongest overall accuracy in multi-class classification, while Efficient-ResNet demonstrated competitive recognition with far fewer parameters.

Third, lightweight architectures were designed to balance diagnostic accuracy with computational efficiency. Unlike the traditional architectures, Efficient-ResNet achieved strong classification accuracy while requiring less than 1M parameters in binary classification task, thus making it significantly smaller than ResNet50 (23.85M) and VGG19 (20.09M). The medium Efficient-ResNet variant in particular demonstrated a favorable trade-off, maintaining robust accuracy across multiple coating types while remaining computationally efficient. These results show that Efficient-ResNet provides a scalable, lightweight solution well-suited for clinical applications where resource efficiency is critical.

Finally, incorporating explainable AI for interpretability and trust. Explainable AI methods were integrated into the workflow specifically Grad-CAM for visual saliency mapping. These methods provided insight into the regions of the tongue images most influential in model decision making. Grad-CAM visualizations revealed that Efficient-ResNet produced coherent activation patterns that often emphasized broader tongue regions, although occasionally extending to background areas. While Efficient-ResNet did not achieve the highest Pointing Game score compared to larger models such as EfficientNetV2-S or VGG19, it demonstrated competitive localization ability considering its compact size. These results suggest that Efficient-ResNet can provide interpretable outputs while maintaining efficiency, making it more suitable for resource-constrained clinical settings.

In summary, all four sub-objectives were successfully achieved. This study delivered segmentation and classification models for tongue coatings, introducing the Efficient-ResNet family as a lightweight yet accurate architecture and demonstrated the importance of explainable AI for clinical transparency. By combining accuracy, efficiency, and interpretability, the proposed framework provides a practical and trustworthy tool to support diagnostic decision-making in Traditional and Complementary Medicine.

Chapter 6
Conclusion and Recommendations

6.1 Conclusion

Tongue diagnosis plays a crucial role in Traditional and Complementary Medicine, offering valuable insights into a patient's health status through visual inspection of tongue characteristics. However, the process remains subjective and highly dependent on practitioner experience, thus leading to inconsistencies and potential diagnostic inaccuracies. This subjectivity is further compound the challenges of limited annotated tongue datasets which hinder the development of robust and generalizable automated diagnostic systems. Motivated by the need to standardize and enhance the reliability of tongue diagnosis, this study aims to develop efficient and interpretable deep learning models capable of accurately classifying tongue coating conditions.

To address these challenges, deep learning-based classification models were developed for both binary and multi-class tongue coating classification tasks. In the binary task, the models distinguished between stained and non-stained tongue coatings while in the multi-class task, the models categorized pathological coatings into specific types. To overcome the limitations of small datasets and computational constraints, lightweight architectures such as Efficient-ResNet were introduced and achieved a balance between high classification accuracy and reduced computational complexity. Furthermore, explainable AI techniques including Grad-CAM was incorporated to provide visual interpretability, thereby enhancing transparency and the potential for integration into clinical decision support systems.

Experimental results demonstrated that the proposed Efficient-ResNet architecture achieved superior performance compared to baseline models by offering a better balance between accuracy and efficiency. While LECA-EfficientNetV2-S showed strong results, Efficient-ResNet consistently delivered robust outcomes across multiple evaluation metrics, making it a more reliable candidate for practical deployment. These findings contribute to the development of reliable, interpretable and efficient AI tools for TCM tongue diagnosis that pave the way for standardize objective and accessible diagnostic practices in both clinical and telemedicine settings.

6.2 Future Work

One of the key directions for future work is the expansion of the dataset to improve model robustness and generalization. The current study was conducted using a limited number of annotated tongue images captured under relatively controlled conditions. To address this, future studies could focus on collecting a larger and more diverse dataset by including images from individuals of different ages, ethnicities and health conditions. Collaborating with UTAR Hospital would be a valuable approach to achieve this as it would provide access to a broader patient base and allow data collection under varying clinical settings. This increased diversity would help the models adapt better to real-world scenarios where environmental factors and patient variability are inevitable.

Besides that, another promising direction is the integration of multi-modal data to enhance diagnostic performance. While this research focused solely on tongue image analysis, incorporating additional patient information such as pulse diagnosis, medical history and laboratory test results could allow for a more holistic approach to TCM diagnosis. Multi-modal deep learning architectures could be explored to combine visual and non-visual features, thus potentially leading to more accurate and clinically relevant predictions.

Furthermore, the development of real-time mobile or web applications could bring the models closer to clinical practice and public use. By leveraging the lightweight architecture developed in this study, the models could be deployed on resource-constrained devices such as smartphones or embedded systems. This would enable practitioners and patients to perform quick and standardized tongue assessments remotely. Collaborating with UTAR Hospital during the application development stage would also allow for user acceptance testing and feedback collection in a real clinical environment, therefore ensuring the solution meets practical needs.

REFERENCES

[1] J. Li *et al.*, “Automatic Classification Framework of Tongue Feature Based on Convolutional Neural Networks,” *Micromachines*, vol. 13, no. 4, pp. 501–501, Mar. 2022, doi: https://doi.org/10.3390/mi13040501.

[2] Q. Zhang, J. Zhou, and B. Zhang, “Computational Traditional Chinese Medicine Diagnosis: A Literature Survey,” *Computers in Biology and Medicine*, vol. 133, p. 104358, Mar. 2021, doi: https://doi.org/10.1016/j.compbiomed.2021.104358.

[3] Burcu Tiryaki, Kubra Torenek-Agirman, Ozkan Miloglu, B. Korkmaz, İbrahim Yucel Ozbek, and Emin Argun Oral, “Artificial intelligence in tongue diagnosis: classification of tongue lesions and normal tongue images using deep convolutional neural network,” *BMC Medical Imaging*, vol. 24, no. 1, Mar. 2024, doi: https://doi.org/10.1186/s12880-024-01234-3.

[4] L. Zhong, G. Xin, Q. Peng, J. Cui, L. Zhu, and H. Liang, “Deep learning-based recognition of stained tongue coating images,” *Digital Chinese Medicine*, vol. 7, no. 2, pp. 129–136, Nov. 2024, doi: https://doi.org/10.1016/j.dcmed.2024.09.004.

[5] D. Tan *et al.*, “Tongue-LiteSAM: A Lightweight Model for Tongue Image Segmentation with Zero-Shot,” *IEEE Access*, vol. 13, pp. 11689–11703, Jan. 2025, doi: https://doi.org/10.1109/access.2025.3528658.

[6] S.-J. Lee *et al.*, “Enhancing deep learning classification performance of tongue lesions in imbalanced data: mosaic-based soft labeling with curriculum learning,” *BMC oral health*, vol. 24, no. 1, Feb. 2024, doi: https://doi.org/10.1186/s12903-024-03898-3.

[7] J. Ni, Z. Yan, and J. Jiang, “TongueCaps: An Improved Capsule Network Model for Multi-Classification of Tongue Color,” *Diagnostics*, vol. 12, no. 3, p. 653, Mar. 2022, doi: https://doi.org/10.3390/diagnostics12030653.

[8] V. Buhrmester, D. Münch, and M. Arens, “Analysis of Explainers of Black Box Deep Neural Networks for Computer Vision: A Survey,” *Machine Learning and Knowledge Extraction*, vol. 3, no. 4, pp. 966–989, Dec. 2021, doi: https://doi.org/10.3390/make3040048.

[9] J. Zhou, Q. Zhang, B. Zhang, and X. Chen, "TongueNet: A Precise and Fast Tongue Segmentation System Using U-Net with a Morphological Processing Layer," *Applied sciences*, vol. 9, no. 15, pp. 3128–3128, Aug. 2019, doi: https://doi.org/10.3390/app9153128.

[10] L. Yao *et al.*, "HPA-UNet: A Hybrid Post-Processing Attention U-Net for Tongue Segmentation," *IEEE Journal of Biomedical and Health Informatics*, pp. 1–12, Jan. 2024, doi: https://doi.org/10.1109/jbhi.2024.3446623.

[11] P. Qu, H. Zhang, L. Zhuo, J. Zhang, and G. Chen, "Automatic Tongue Image Segmentation for Traditional Chinese Medicine Using Deep Neural Network," *Lecture notes in computer science*, vol. 10361, pp. 247–259, Jan. 2017, doi: https://doi.org/10.1007/978-3-319-63309-1_23.

[12] X. Zhang, H. Bian, Y. Cai, K. Zhang, and H. Li, "An improved tongue image segmentation algorithm based on Deeplabv3+ framework," *IET Image Processing*, vol. 16, no. 5, pp. 1473–1485, Jan. 2022, doi: https://doi.org/10.1049/ipr2.12425.

[13] S. Cao, Q. Wu, and L. Ma, "TongueSAM: An Universal Tongue Segmentation Model Based on SAM with Zero-Shot," Dec. 2023. Available: https://arxiv.org/pdf/2308.06444

[14] Y. Tang *et al.*, "An improved lightweight tongue image semantic segmentation model based on DeepLabV3+," *Biomedical Signal Processing and Control*, vol. 109, pp. 107911–107911, May 2025, doi: https://doi.org/10.1016/j.bspc.2025.107911.

[15] Q. Zhuang, S. Gan, and L. Zhang, "Human-computer interaction based health diagnostics using ResNet34 for tongue image classification," *Computer Methods and Programs in Biomedicine*, vol. 226, p. 107096, Nov. 2022, doi: https://doi.org/10.1016/j.cmpb.2022.107096.

[16] T. Jiang *et al.*, "Deep Learning Multi-label Tongue Image Analysis and Its Application in a Population Undergoing Routine Medical Checkup," *Evidence-based complementary and alternative medicine*, vol. 2022, pp. 1–12, Sep. 2022, doi: https://doi.org/10.1155/2022/3384209.

[17] X. Wang *et al.*, "Constructing tongue coating recognition model using deep transfer learning to assist syndrome diagnosis and its potential in noninvasive ethnopharmacological

evaluation," *Journal of Ethnopharmacology*, vol. 285, p. 114905, Mar. 2022, doi: https://doi.org/10.1016/j.jep.2021.114905.

[18] N. Cui, Y. Wu, G. Xin, J. Wu, L. Zhong, and H. Liang, "Application of Quantitative Interpretability to Evaluate CNN-Based Models for Medical Image Classification," *IEEE Access*, vol. 13, pp. 89386–89398, 2025, doi: https://doi.org/10.1109/access.2025.3567775.

[19] Q. Wu, S. Cao, L. Cao, and Q. Ruan, "Visualization Analysis of Tongue Syndrome Diagnosis by Interpretability Neural Networks," *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, vol. 8, pp. 3946–3953, Dec. 2023, doi: https://doi.org/10.1109/bibm58861.2023.10385515.

[20] Mohamed Musthafa M, M. T. R, Vinoth Kumar V, and Suresh Guluwadi, "Enhancing brain tumor detection in MRI images through explainable AI using Grad-CAM with Resnet 50," *BMC medical imaging*, vol. 24, no. 1, May 2024, doi: https://doi.org/10.1186/s12880-024-01292-7.

[21] M. Umair *et al.*, "Detection of COVID-19 Using Transfer Learning and Grad-CAM Visualization on Indigenously Collected X-ray Dataset," *Sensors*, vol. 21, no. 17, p. 5813, Aug. 2021, doi: https://doi.org/10.3390/s21175813.

[22] BioHit, "GitHub - BioHit/TongeImageDataset: This tongue image data set contains 300 tongue images. All the images in the data set were acquired by our image acquisition device and the image size was 576*768. Manual segmentations were used as the ground truth.," *GitHub*, 2014. https://github.com/BioHit/TongeImageDataset (accessed Sep. 02, 2024).

[23] M. HA, "Tongue Segmentation Dataset," *Roboflow*, 2023. https://universe.roboflow.com/minh-ha-tixet/tongue-segmentation-deq4x (accessed Apr. 07, 2025).

[24] Clear2073, "舌象数据集-neo_数据集-飞桨 AI Studio 星河社区," *Baidu.com*, 2025. https://aistudio.baidu.com/datasetdetail/196398 (accessed Apr. 07, 2025).

[25] Zhong Li Qin *et al.*, "A dataset of stained tongue fur images of TCM," *Scidb.cn*, Jul. 2023, doi: https://doi.org/10.57760/sciencedb.j00001.00822.

[26] D. Zhang, W. Pang, K. Wang, F. Yang, and J. Zhang, "Tongue Coating Grading Identification Using Deep Learning for Hyperspectral Imaging Data," *IEEE Access*, vol. 11, pp. 93151–93159, 2023, doi: https://doi.org/10.1109/access.2023.3308602.

[27] X. Cheng, S. Dou, Y. Du, and Z. Wang, "Gearbox fault diagnosis method based on lightweight channel attention mechanism and transfer learning," *Scientific Reports*, vol. 14, no. 1, Jan. 2024, doi: https://doi.org/10.1038/s41598-023-50826-6.

[28] O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional networks for biomedical image segmentation," *arXiv (Cornell University)*, May 2015, doi: https://doi.org/10.48550/arxiv.1505.04597.

[29] M. Sandler, A. W. Howard, M. Zhu, Andrey Zhmoginov, and L.-C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," *arXiv (Cornell University)*, Jan. 2018, doi: https://doi.org/10.48550/arxiv.1801.04381.

[30] A. Dosovitskiy *et al.*, "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale," *arXiv.org*, Jun. 03, 2021. https://www.arxiv.org/abs/2010.11929v2

[31] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization," *International Journal of Computer Vision*, vol. 128, no. 2, pp. 336–359, Feb. 2020, doi: https://doi.org/10.1007/s11263-019-01228-7.

[32] D. Müller, I. Soto-Rey, and F. Kramer, "Towards a guideline for evaluation metrics in medical image segmentation," *BMC Research Notes*, vol. 15, no. 1, Jun. 2022, doi: https://doi.org/10.1186/s13104-022-06096-y.

[33] L. I. U. Wei, C. H. E. N. Jinming, L. I. U. Bo, H. U. Wei, W. U. Xingjin, and Z. H. O. U. Hui, "Tongue image segmentation and tongue color classification based on deep learning," *Digital Chinese Medicine*, vol. 5, no. 3, pp. 253–263, Sep. 2022, doi: https://doi.org/10.1016/j.dcmed.2022.10.002.

[34] S. A. Hicks *et al.*, "On evaluation metrics for medical applications of artificial intelligence," *Scientific Reports*, vol. 12, no. 1, p. 5979, Apr. 2022, doi: https://doi.org/10.1038/s41598-022-09954-8.

[35] P. Soni, “Confusion Matrix, Precision, and Recall - Train in Data’s Blog,” *Train in Data’s Blog*, Sep. 05, 2024. https://www.blog.trainindata.com/confusion-matrix-precision-and-recall/

[36] Y. Zhang *et al.*, “Saliency-Bench: A Comprehensive Benchmark for Evaluating Visual Explanations,” *arXiv.org*, 2023. https://arxiv.org/abs/2310.08537 (accessed Aug. 08, 2025).

[37] V. Badrinarayanan, A. Kendall, and R. Cipolla, “SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.

[38] L. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, “Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation,” *arXiv (Cornell University)*, Feb. 2018, doi: https://doi.org/10.48550/arxiv.1802.02611.

[39] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, “Pyramid Scene Parsing Network,” *arXiv.org*, Apr. 27, 2017. https://arxiv.org/abs/1612.01105

POSTER

Applying Deep Learning Techniques to Segmentize and Classify Tongue Regions for Traditional and Complementary Medicine Diagnosis

Yeap Chun Hong FICT/CS Supervisor: Ts. Dr. Lee Wai Kong

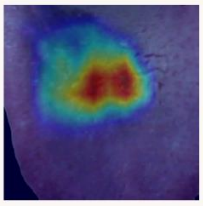
Introduction

Tongue diagnosis is a key component of Traditional Chinese Medicine (TCM), but manual visual inspection is subjective and inconsistent. This study develops deep learning-based tongue image analysis to improve accuracy and reliability. The research integrates three components:

Segmentation

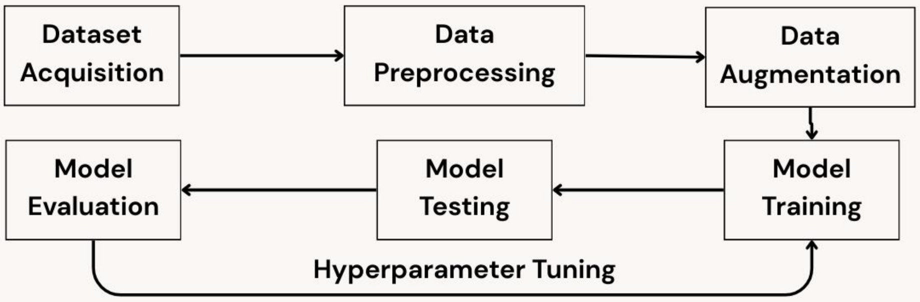
Black-Grey

Classification



Explainable AI

Methodology



Results

Segmentation: Mobile U-Net achieved **97.40% accuracy** with **40% smaller size**.

Binary Classification: Efficient-ResNet achieved **98.51% accuracy** with just **0.31M parameters.**

Multi-Class Classification: EfficientNetV2 variants performed best results (**85.00% accracy**), while Efficient-ResNet delivered competitive results with **10x fewer parameters (79.17% in 1.16M parameters)** .

Discussion

Accuracy vs. Efficiency: Larger models are accurate but computationally expensive. Lightweight models deliver near state-of-the-art accuracy at a fraction of the size, making them ideal for mobile and real-time applications .

Conclusion

Mobile U-Net and Efficient-ResNet deliver lightweight and efficient performance, achieving high diagnostic accuracy while Grad-CAM ensures interpretability. Together, these contributions provide a practical and trustworthy solution for tongue diagnosis in clinical and telemedicine settings.