BANGLA SIGN LANGUAGE RECOGNITION: DATASET INNOVATION AND METHODOLOGIES

By

Md Nur A Alam ID: 210201044

B.Sc. Engineering in Computer Science and Engineering

Department of Computer Science and Engineering

Faculty of Electrical and Computer Engineering

Bangladesh Army University of Science and Technology (BAUST)

JANUARY 2026

MediNet-XG: A Lightweight Explainable Multimodal AI for Medicinal Plant from weed-infested area.

This thesis is submitted in partial fulfillment of the requirement for the degree of B.Sc. Engineering in Computer Science and Engineering

By

Md Nur A Alam ID: 210201044

Supervised by

Dr. S.M. Jahangir Alam

Professor and Head, Department of CSE

Co-Supervised by

Saad Ahmed

Lecturer, Department of CSE

Department of Computer Science and Engineering

Bangladesh Army University of Science and Technology (BAUST)

Saidpur Cantonment, Nilphamari, Bangladesh

The thesis titled **"MediNet-XG: A Lightweight Explainable Multimodal AI for Medicinal Plant from weed-infested area."** submitted by Md Nur A Alam (ID:210201044), session 2020-2021 has been presented on 18/01/2026 and accepted as satisfactory in fulfillment of the requirement for the degree of Bachelor of Science in Computer Science and Engineering (CSE) as B.Sc. Engineering to be awarded by the Bangladesh Army University of Science and Technology (BAUST).

Board of Examiners

1. _____ Supervisor

   Dr. S.M. Jahangir Alam
   Professor and Head
   Department of Computer Science and Engineering (CSE)
   Bangladesh Army University of Science and Technology (BAUST)

2. _____ Co-Supervisor

   Saad Ahmed
   Lecturer
   Department of Computer Science and Engineering (CSE)
   Bangladesh Army University of Science and Technology (BAUST)

3. _____ Member

Dr. Engr. Mohammed Sowket Ali
Associate Professor
Department of Computer Science and Engineering (CSE)
Bangladesh Army University of Science and Technology (BAUST)

4. _____ Member

Hasan Muhammad Kafi
Assistant Professor
Department of Computer Science and Engineering (CSE)
Bangladesh Army University of Science and Technology (BAUST)

5. _____ Member

Md. Zahid Hassan
Lecturer
Department of Computer Science and Engineering (CSE)
Bangladesh Army University of Science and Technology (BAUST)

6. _____ Member

Md. Mahadi Hasan
Lecturer
Department of Computer Science and Engineering (CSE)
Bangladesh Army University of Science and Technology (BAUST)

_____

Dr. Md Nakib Hayat Chowdhury
Professor and Head
Department of Computer Science and Engineering (CSE)
Bangladesh Army University of Science and Technology (BAUST)

CANDIDATE'S DECLARATION

It is hereby declared that the contents of this thesis are original and any part of it has not been submitted elsewhere for the award of any degree or diploma.

SN.	**Student Name**	**ID**	**Signature & Date**
1.	Md Nur A Alam	210201044	

ACKNOWLEDGMENTS

First and foremost, we are deeply grateful to Almighty Allah for granting us the strength and perseverance to complete this thesis successfully. We extend our sincerest thanks and heartfelt gratitude to our honorable thesis supervisor, Dr. S.M. Jahangir Alam, Professor and Head of the Department of Computer Science & Engineering (CSE) at Bangladesh Army University of Science & Technology (BAUST), Saidpur. His inspiration and unwavering support have been instrumental in the completion of our thesis.

We like to express our special thanks to the Head of the Department of Computer Science & Engineering (CSE), Dr. Md Nakib Hayat Chowdhury for his valuable suggestions, positive advice, encouragement, and sincere guidance throughout our thesis work. Additionally, we convey our gratitude to all the respected teachers of the Department of CSE, as well as those from other departments, for their support and contributions.

Our sincere thanks go to the respected Honorable Vice Chancellor, the Chief of Army Staff and board of trustees, as well as all members and academic staff who have led this institution and contributed to our bright future.

We would also like to thank all our friends and the staff of the department for their valuable suggestions and assistance. Finally, we extend our deepest appreciation to our parents and families for their unwavering love and support throughout our studies.

ABSTRACT

Bangla Sign Language (BdSL) is essential for communication among people with hearing impairments in Bangladesh. In order to address the need for improved BdSL gesture detection, this paper introduces the RSBdSL38 dataset. To ensure variety and usefulness, 11,864 gestures from 46 individuals in various difficult situations are included in the dataset. With its improvements over earlier datasets, RSBdSL38 represents a major advancement in BdSL recognition technology. The dataset's ability was tested to aid in the understanding of BdSL gestures using sophisticated machine learning algorithms. Important aspects such as accuracy and performance was examined in various scenarios and contrasted the findings with available datasets to demonstrate the power and use of RSBdSL38. The foundation for developing more inclusive and accurate BdSL recognition systems is laid by this research, which will facilitate everyday communication for BdSL users.

Keywords: Gesture Recognition, Pretrained Models, Ensemble Techniques, Robustness and Comparative Analysis.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
Introduction

1.1 Overview

Medicinal plants form a major foundation of traditional health care; they are commonly used in the first-line treatment in the low resource situation where formal health care facilities are limited [BG1]. Bangladesh is estimated to possess 1,000 conservative species of medicinally active plants, with this amount of ethnomedicinal knowledge densely stored in a relatively tiny region of geographic space [9]. Survey studies show that over 9 out of 10 remedy formulations based on traditional practices are plants based and the situation in Bangladesh is no exception as most of the rural and peri-urban practice is same [10]. The scientific interests and therapeutic potential of this local flora are further supported by pharmacology research of at least 48 Bangladeshi species as having anti-inflammatory properties [9]. Irrespective of this significance, Bangladeshi medicinal plant digital resources are scarce. There are few publicly available standardized datasets of images, and those that exist are usually not consistently labeled, with realistic field backgrounds, or connected to structured botanical and safety data [5]. The available data sets are mostly collected under controlled imaging scenarios and not sufficient to forecast weedy medicinal species that exist as spontaneous flora in farmlands, homesteads and roadside environments with thick weed growths, occlusions and mixed backgrounds where non-experts find it hard to identify the weed under manual guidance [4]. This knowledge gap has been found as a significant burden to biodiversity monitoring and medicinal plant classification using deep-learning models, with numerous studies using small institutional collections or non-Bangladeshi repositories with different species composition to local requirements [10]. Recent deep learning methods have shown great accuracy when used to recognize medicinal plants and crops, but most of them use large pretrained CNNs or

ensembles, which are trained on images with clean backgrounds, and are black-box models, which necessitate computation at server scale. Explainable AI has begun to be used in phenotyping plants and weed detection, and primarily by saliency algorithms like Grad-CAM, but these applications often aim to detect crop diseases or weed-crop interactions, rather than recognizing medicinal weeds, and often do not use any structured medical knowledge to be safe in interpretation. Therefore, the explanatory and easy-to-understand models optimized to suit weedy medicinal plants under realistic field conditions have not been fully studied. Specifically, no systems exist, which (i) run within strong memory and compute limits to fit mobile or edge devices, (ii) offer transparent and instance-level explanation of any prediction, and (iii) integrate visual recognition with a formal botanical knowledge base such that outputs are grounded and safety-confined. In order to fill the expertise insufficiency and

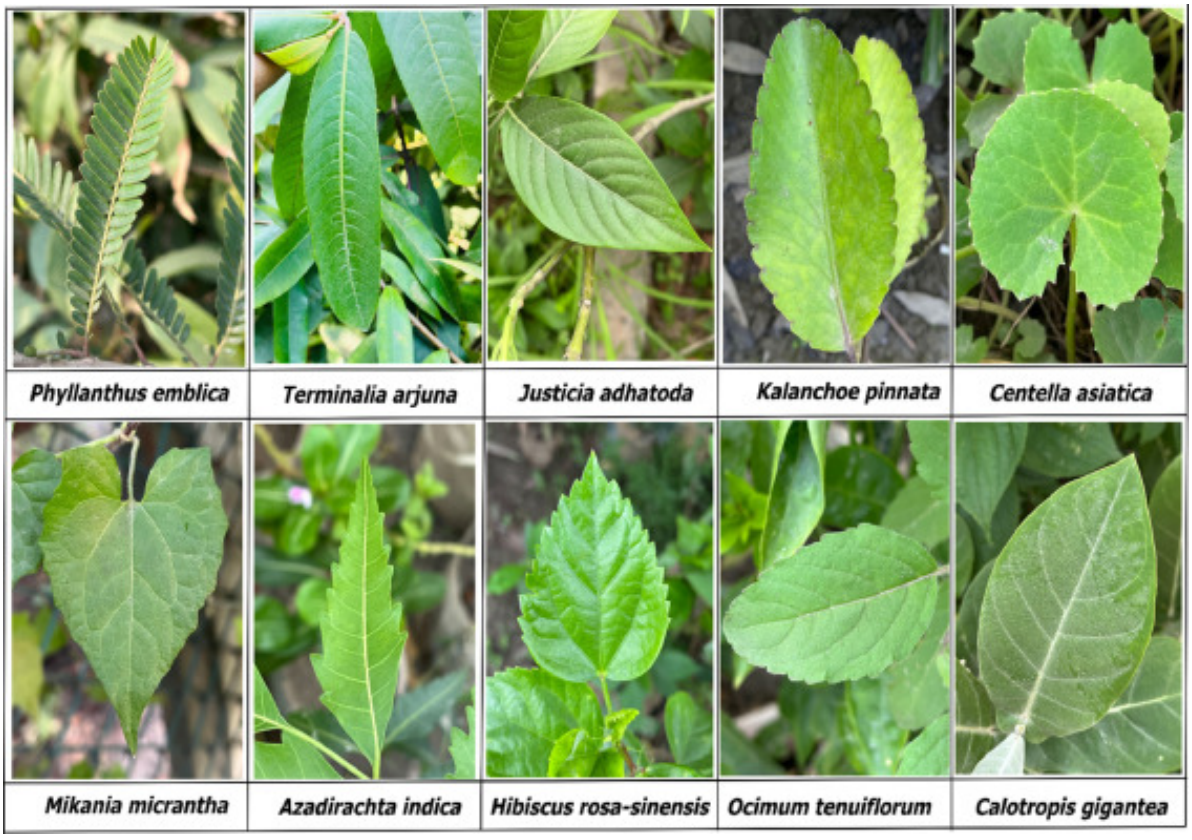


Figure 1.1: Weed-infested Area Medininal plants.[11]

proper recognition problems, the proposed work introduces the lightweight and explainable multimodal architecture of MediNet-XG, which can be used to identify Bangladeshi weedy medicinal plants through in situ RGB leaf images and to provide the corresponding botanical and safety information through retrieval. An ultra-compact CNN classifier is trained on images of leaves recorded directly in the field and makes high-confidence predictions on

a pre-curated collection of 34 species, and Grad-CAM and t-SNE are employed to reveal the spatial decision region and feature-space relationship of each sample. All the predicted classes are connected to known facts about morphology, uses, active compounds, toxicity and pregnancy safety in a structured JSON knowledge base allowing field-level decision support without unstructured text generation. The key contributions of this work could be summarized in the following way. To start with, the research present a new field-captured dataset of 17,050 RGB leaf images like Figure 1.1 of 34 Bangladeshi weedy medicinal species that are rank-awarely annotated with empirical weediness and realistic background clutter. Second, a CNN architecture named MediNet-XG is constructed which has high accuracy and F1-score on this dataset and has a model size that is measured in the hundreds of kilobytes range, which is significantly smaller than standard pretrained backbones. Third, it has a built-in explainability module, which is a combination of Grad-CAM and t-SNE that enables the observation of both spatial and feature-space components of the decision process per prediction. Fourth, a knowledge engine in the form of a retrieval engine is implemented, which provides a mapping between predicted classes and a maintained JSON botanical database to provide organized, safety-conscious answers to user queries. The rest of this paper is structured in the following way. Section II provides a literature review of related work in medicinal plant recognition, explainable deep learning in plant phenotyping and weed detection. Section III explains how the dataset was constructed, how MediNet-XG is organized and what are the explainability and knowledge modules. Section IV gives the details of the implementation and experimental set up. Section V presents both quantitative and qualitative findings, and compares the results with available models as well as interprets the Grad-CAM and t-SNE output. Section VI wraps the paper and also provides future research directions.

1.2 Problem Statement

Medicinal plants form the backbone of traditional healthcare and over 90% of remedies worldwide are plant based which is especially important in Bangladesh where formal medical access is uneven[9]. Approximately 1,000 Bangladeshi species are conservatively regarded as medicinal and at least 48 have already been pharmacologically studied for anti inflamma-

tory use which highlights dense species diversity and functional overlap. Many of these taxa share very similar leaf shapes and colors so that even controlled image based identification is difficult and the task becomes more complex when plants grow as spontaneous weeds in fields and roadsides where dense vegetation occlusion and heterogeneous backgrounds dominate the scene. Weed monitoring studies further show that small or scattered patches are often missed because models are trained on larger clearer regions which suggests that medicinal plants appearing as minor components among aggressive weeds are easily overlooked or confused with non medicinal or toxic species. Although deep CNN and ensemble models for Bangladeshi medicinal plants report accuracies close to 99 percent on curated datasets these architectures remain heavy server oriented black boxes and a recent review identifies the trustworthiness and interpretability of such models as a major open problem. Key technical challenges highlighted in the literature include:

- **High inter-class similarity** among medicinal species, which reduces separability even under simplified backgrounds.
- **Traditional method** by taxonomists is slow, complex, and error-prone.
- **Missed detection** of small or scattered targets because existing models favor larger homogeneous regions.[12]
- **Computationally heavy CNNs and ensembles**, such as DenseNet-based models, which are unsuitable for low-power mobile devices.
- **Predominantly black-box inference** with little visual explanation or structured medicinal knowledge for safe decision-making.

For Bangladeshi medicinal plants in weedy areas, the main problems are similar species, cluttered backgrounds, lack of field images, poor explainability and the difficulty of compressing robust models into small on-device architectures without losing too much accuracy.

1.3 Objectives

The present research is structured around three core objectives that together define the MediNet-XG framework. The first objective is to develop a lightweight explainable multimodal AI architecture that integrates compact convolutional feature extraction with mandatory visual ex-

planations and a structured botanical knowledge base while remaining suitable for deployment on mobile and edge devices in resource-constrained settings. The second objective is to achieve accurate classification of Bangladeshi medicinal plants from leaf images captured in weed-infested areas so that reliable predictions are obtained under uncontrolled lighting cluttered backgrounds and strong inter-class similarity without relying on computationally heavy server-grade models. The third objective is to generate contextual knowledge on plant uses compounds and habitats by deterministically linking each predicted species to a curated JSON knowledge base that provides grounded information on therapeutic properties phytochemical constituents ecological preferences and safety constraints thus replacing free form speculation with auditable retrieval based explanations.

- To develop a Lightweight Explainable multimodal AI framework.
- To accurately classify medicinal plants in weed-infested areas.
- To generate contextual knowledge on plant uses, compounds, and habitats.

In summary the work seeks to design a compact explainable CNN, ensure robust medicinal plant classification in weed infested scenes and deliver concise retrieval based knowledge on uses compounds and habitats without hallucinated claims. Collectively these objectives position MediNet-XG as a practical and trustworthy decision support tool rather than a generic black box recognition system.

1.4 Social Impact

The proposed MediNet-XG framework has the potential to generate substantial social impact by strengthening safe and informed use of medicinal plants in resource constrained communities. In Bangladesh a large share of the population relies on plant based remedies as the first line of treatment, yet accurate field identification often depends on a small number of experts and a time consuming process of visual inspection touch smell and taste guided by specialized floras and keys. By providing an on device decision support tool that can recognize weedy medicinal species directly from leaf images and immediately retrieve structured knowledge on uses compounds and safety, the system can reduce dependence on scarce expert capacity and make reliable information more accessible to rural households farmers and community

health workers.

The availability of explainable predictions and curated safety notes also supports harm reduction and rational use of traditional remedies. Transparent Grad CAM visualizations and knowledge grounded descriptions can help users understand why a plant has been identified in a particular way and what known toxicities or contraindications are associated with it, which may reduce misidentification and discourage unsafe self medication or over harvesting of toxic species. At the same time the focus on weedy medicinal plants that grow spontaneously in fields and homesteads promotes low cost locally available healthcare options without requiring new cultivation or input investments, thereby contributing to economic resilience for low income families.

Beyond individual health outcomes the system can assist agricultural extension services and biodiversity initiatives by documenting the presence of medicinal weeds in crop lands and marginal habitats. Consistent digital records of identified species can support monitoring of agroecosystem diversity and facilitate conservation of culturally important medicinal flora that might otherwise be classified as nuisance weeds and removed. In educational settings the tool can act as a practical companion for botany students and trainee health workers providing immediate feedback and verified descriptions that reinforce formal curricula. Overall the MediNet-XG framework is expected to promote safer traditional healthcare practices, support inclusive access to botanical knowledge and contribute to sustainable management of medicinal plant resources in Bangladesh.

1.5 Issues Encountered

Several practical and technical issues were encountered during the development of the MediNet-XG framework. Collection of realistic field images of weedy medicinal plants under uncontrolled lighting, cluttered backgrounds and frequent occlusions made it difficult to maintain uniform quality and required expert validated annotation to avoid mislabeling. Designing a model that remained sufficiently lightweight for mobile deployment while preserving high classification performance demanded careful tuning of inverted residual blocks, depthwise separable convolutions and attention mechanisms, especially when explainability modules such as Grad-CAM

and t-SNE had to be integrated without exceeding computational limits. Construction of the curated JSON knowledge base also proved challenging because heterogeneous botanical and medicinal sources had to be harmonized into a consistent schema, class labels had to be mapped deterministically to entries despite naming variations and strict safety constraints required conservative handling of incomplete or conflicting information.

1.6 Organization of the Thesis

The thesis is structured into six chapters to present the design, implementation and evaluation of the MediNet-XG framework in a coherent manner. The first chapter introduces the motivation for reliable medicinal plant identification in weed infested environments, outlines the problem statement, specifies the research objectives and highlights the anticipated social impact of a lightweight explainable decision support system for Bangladeshi contexts. The second chapter provides a detailed literature review on medicinal plant recognition, weed detection and explainable deep learning, emphasizing existing dataset limitations, black box model behavior and deployment challenges for mobile and edge platforms. The third chapter

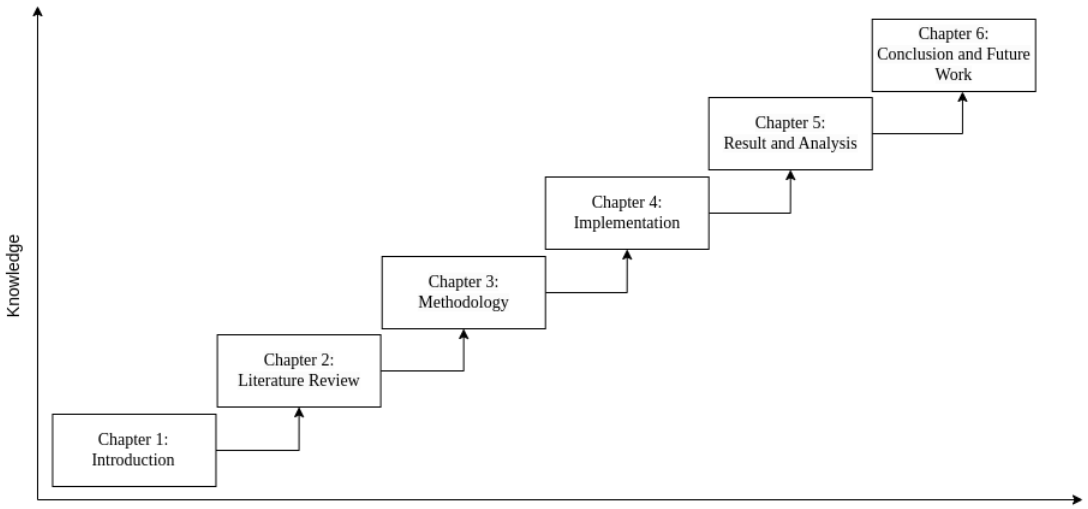


Figure 1.2: Organization of the thesis' chapters.

describes the overall methodology of the research, including the design of the MediNet-XG architecture, the construction and preprocessing of the weedy medicinal plant dataset, the integration of Grad-CAM and t-SNE for visual explainability and the development of the structured JSON-based botanical knowledge base. The fourth chapter focuses on implemen-

tation details and experimental setup, covering training procedures, hyperparameter choices, inference pipeline design and the mechanisms that enforce retrieval-based, safety constrained outputs. The fifth chapter presents the quantitative and qualitative results obtained from MediNet-XG, including classification metrics, confusion matrices, visual explanations and comparisons with heavier baseline models, followed by an in depth analysis of performance, trustworthiness and deployability. The final chapter concludes the thesis by summarizing the main contributions and limitations of the proposed framework and by outlining future directions such as expanding the species coverage, incorporating additional modalities and adapting the system for broader agroecological and healthcare applications. This research work is publicly available at GitHub repository[13]

CHAPTER 2
Literature Review

2.1 Overview

Recent advances in deep learning have greatly improved automated plant recognition yet most medicinal-plant frameworks still rely on heavy black-box models trained on controlled images that are difficult to deploy in noisy field conditions. Recent Bangladeshi works on medicinal plant classification achieve very high accuracy with CNN and ensemble architectures but remain offline, computationally expensive and disconnected from any safety-aware medicinal knowledge or explanation interface. Global systematic reviews confirm that the majority of medicinal-plant studies continue to use laboratory-style image datasets with limited species diversity while explainable AI and on-device deployment are mentioned mainly as future directions rather than implemented solutions.

Parallel literature in plant phenotyping, crop disease and weed detection shows that Grad-CAM and related techniques can provide useful visual explanations and that cluttered field backgrounds make small or scattered targets easy to miss, but these efforts focus on crops and weed control rather than species-level medicinal plant identification with knowledge grounding. Overall the state of the art exposes four key gaps: the lack of lightweight CNNs tailored for edge deployment, the scarcity of field datasets where medicinal plants appear as weeds, the limited integration of robust explainability and the absence of tightly coupled retrieval-based botanical knowledge bases. These gaps motivate the design choices and contributions of MediNet-XG described in the following sections.

Title: Deep-Learning-Based Classification of Bangladeshi Medicinal Plants Using Neural

Ensemble Models.[3]

The study introduces a curated dataset of 10 Bangladeshi medicinal plant species and evaluates multiple CNNs and ensemble strategies for leaf-image classification. The work demonstrates that ensemble learning can significantly boost accuracy over individual models on clean, controlled images.

Methodology:The study begins with the collection of high-quality RGB images of fresh leaves from ten medicinal plant species prevalent in Bangladesh; images are captured under relatively uniform backgrounds and lighting to reduce noise and facilitate feature extraction. All images are resized to a common resolution and normalized before being split into training, validation and test subsets so that each species is reasonably balanced across the partitions. Several pretrained CNN architectures such as VGG-type, ResNet-type and related models are then fine-tuned on the dataset using transfer learning, with the final fully connected layers adjusted to output ten class probabilities and trained using standard optimization pipelines with learning-rate scheduling and regularization. To further enhance performance, two ensemble strategies are constructed: a soft ensemble that averages the probability vectors from multiple CNNs to produce a consensus prediction and a hard ensemble that combines discrete class labels via majority voting; both ensembles are trained and evaluated offline on GPU-equipped systems and assessed using accuracy, confusion matrices and loss curves on the held-out test set.

Result: Reported test accuracies for individual CNN backbones and neural ensemble configurations on the ten-class Bangladeshi medicinal plant leaf dataset, showing the performance gains achieved by ensemble learning over single-model baselines.

Table 2.1: Test accuracy of individual CNNs and ensemble models on the Bangladeshi medicinal plant dataset (10 species)[3]

Model	**Description**	**Acc(%)**	**Notes**
CNN-A	Single pretrained CNN backbone	≈95–96	Baseline fine-tuned model
CNN-B	Alternative CNN backbone	≈96–97	Deeper architecture improvement
Ensemble	Averaged class probabilities	97.62	Best soft-voting configuration

The results in Table 2.1 indicate that while single CNNs already achieve high accuracy, the ensembles offer measurable improvements, with the hard ensemble reaching about 98 percent test accuracy and providing the best overall performance on the controlled dataset.

Limitations:

- **Controlled imaging conditions rather than field scenes:** The dataset is composed of leaf images captured under relatively uniform backgrounds and lighting, which does not reflect weed-infested, cluttered environments where medicinal plants naturally occur and where background noise and occlusion are common. MediNet-XG instead targets real field images with heterogeneous backgrounds to improve robustness in practical use.

- **Computationally heavy ensemble architectures:** The use of multiple large pretrained CNNs combined via ensemble strategies results in a high parameter count and significant memory and compute requirements, making these models impractical for low-power mobile or edge devices. MediNet-XG is designed as an ultra-lightweight CNN (hundreds of kilobytes) optimized specifically for on-device inference.

- **Black-box predictions without explainability:** The models output class labels and accuracy metrics but do not provide visual explanations such as saliency maps or feature-space context, limiting user trust and making error analysis difficult, especially in health-adjacent settings. MediNet-XG integrates Grad-CAM and t-SNE-based visualizations so that the discriminative leaf regions and class relationships behind each prediction are explicitly revealed.

- **No linkage to medicinal knowledge or safety information:** The study focuses solely on species classification and does not connect predictions to structured information about plant uses, active compounds or toxicity, nor does it impose constraints on how outputs might be interpreted in medicinal contexts. MediNet-XG couples each predicted class to a curated JSON knowledge base that encodes botanical, therapeutic and safety attributes, enabling retrieval-based explanations and refusal of unsupported or unsafe queries.

Title: Deep Learning for Medicinal Plant Species Classification: Review and Future Directions.[4]
The systematic review synthesizes recent deep-learning approaches for medicinal plant species classification across multiple countries, datasets and imaging setups, providing a consolidated picture of current practices and open challenges. The article emphasizes that while

CNN-based models routinely report high accuracies on curated datasets, most studies lack robustness evaluation, explainability and deployment planning, especially in realistic field environments.

Methodology:Relevant articles were retrieved from major scientific databases using predefined keywords related to medicinal plants, deep learning and image classification and were filtered through inclusion and exclusion criteria to focus on peer-reviewed works published in recent years. Each selected paper was analyzed along several dimensions, including plant organ used for imaging (leaf, flower, fruit or whole plant), imaging modality (RGB, hyperspectral, microscopic), dataset characteristics (size, species count, capture conditions), model architecture (standard CNN, transfer learning, ensemble) and evaluation metrics (accuracy, precision, recall, F1). The review then categorized studies by application domain (taxonomic classification, disease detection, pharmacological screening support) and examined whether any form of explainable AI, uncertainty estimation or resource-efficient design was implemented, with particular attention to gaps in public datasets, reproducibility and field validation

Result: Main patterns and gaps identified in the systematic review of deep-learning methods for medicinal plant species classification, summarizing dataset, model, evaluation and explainability practices.

Table 2.2: Key observations from the systematic review on deep learning for medicinal plant classification[4]

Aspect	**Typical practice reported**	**Noted gap / limitation)**
Dataset	Medium-sized image datasets, often private, with tens of species captured under controlled conditions	Scarcity of large, public, field-captured datasets; limited representation of weedy or wild medicinal plants
Models	Standard CNNs and transfer-learning backbones (e.g., VGG, ResNet, Inception) tuned for high accuracy	Limited attention to model size, latency or edge deployment; few studies consider low-resource devices
Evaluation	Accuracy as primary metric, sometimes with precision/recall or F1	Rare robustness tests under varying backgrounds, lighting or occlusion; little discussion of generalization
XAI	Very few works include saliency maps or XAI; almost none link outputs to medicinal safety information	Lack of systematic explainability and absence of safety-aware, knowledge-grounded outputs identified as major future directions

The review in Table 2.2 concludes that deep learning has strong potential for medicinal plant recognition but that progress is constrained by limited dataset diversity, lack of standard-

ized benchmarks, minimal use of XAI and insufficient focus on deployment in real-world, resource-constrained settings.

Limitations:

- **Scarcity of realistic field datasets:** Most surveyed studies use controlled images with simple backgrounds and do not represent wild or weedy medicinal plants, which reduces ecological validity. MediNet-XG explicitly builds on field-captured leaf images from weed-infested environments to close the realism gap.
- **Limited integration of explainable AI:** Only a small fraction of reviewed works incorporate saliency maps or explanation mechanisms, and even those are not consistently evaluated or standardized. MediNet-XG embeds Grad-CAM and t-SNE-based feature-space visualization as mandatory components so that each prediction is accompanied by transparent visual evidence.
- **Absence of safety-aware, knowledge-grounded outputs:** The review highlights that current systems largely stop at species labels and do not connect to structured information on medicinal uses, active compounds or toxicity, leaving interpretation entirely to the user. MediNet-XG couples each prediction to a curated JSON knowledge base that encodes botanical attributes, therapeutic roles and explicit safety notes, and restricts textual responses to retrieval from the source to avoid hallucinated or unsafe advice.
- **Lack of focus on weedy medicinal plants:** The surveyed literature primarily addresses cultivated, ornamental or pharmacologically important species without emphasizing plants that occur as weeds in agricultural landscapes. MediNet-XG specifically targets weedy medicinal species in cluttered fields, aligning the model with real-world use cases such as rural healthcare and agroecological management in Bangladesh.

Title: IndoHerb: Indonesia Medicinal Plants Recognition Using Transfer Learning[5]
The IndoHerb study proposes a deep-learning framework for recognizing Indonesian medicinal plants from leaf images using transfer-learning CNN models. A region-specific dataset of local herbal species is introduced and several pretrained architectures are fine-tuned to assess how well generic visual features can be adapted for medicinal plant identification in an

Indonesian context.

Methodology:The work begins with the collection of RGB leaf images for multiple Indonesian medicinal plant species, captured under relatively controlled conditions to ensure clear visibility of shape and venation patterns. All images are resized to a fixed input resolution, normalized and split into training, validation and test sets, with class balancing applied to reduce bias across species. Transfer learning is employed by taking CNN backbones such as MobileNet, Inception or similar lightweight architectures pretrained on large-scale image datasets, replacing their final classification layers with new fully connected heads tailored to the number of medicinal classes and fine-tuning the networks on the IndoHerb dataset. Standard optimization procedures are used, including learning-rate scheduling, early stopping and data augmentation (e.g., rotation, flipping) to improve generalization, and model performance is evaluated using accuracy and confusion matrices on a held-out test set.

Result: Accuracy of different transfer-learning CNN architectures on the IndoHerb dataset, showing that lightweight models can reach close to 98 percent accuracy on controlled Indonesian medicinal plant images.

Table 2.3: Reported performance of transfer-learning models on the IndoHerb medicinal plant dataset[5]

Model	**Description**	**Acc(%)**	**Notes**
TL-CNN-A	Pretrained CNN backbone fine-tuned on IndoHerb	≈96%	Baseline transfer-learning performance
TL-CNN-B (MobileNet variant)	Lightweight backbone with fine tuned head	≈97%	Best performing configuration on the dataset

The results in Table 2.3 demonstrate that transfer-learning CNNs are highly effective for regional medicinal plant recognition, with some lightweight backbones achieving test accuracies near 98 percent, thereby validating the feasibility of adapting generic visual features for local herbal species classification.

Limitations:

- **Controlled imaging rather than cluttered field environments:** IndoHerb primarily uses leaf images captured under relatively clean and uniform backgrounds, which do not reflect the complex weed-infested scenes in which many medicinal plants are en-

countered in practice. MediNet-XG is explicitly trained and evaluated on field-captured images from weed-infested areas in Bangladesh to improve robustness under real-world visual clutter.

- **Lack of mandatory explainability:** The IndoHerb framework focuses on classification accuracy and does not integrate Grad-CAM, t-SNE or other explanation tools, leaving predictions as black-box outputs. MediNet-XG enforces per-sample explainability by generating Grad-CAM heatmaps and feature-space visualizations for the predicted class, thereby supporting user trust and diagnostic understanding.

- **No explicit focus on ultra-lightweight edge deployment:** Although some IndoHerb models use relatively compact backbones, the study does not target a specific model-size budget or latency requirement for low-end mobile devices and does not report deployment-oriented metrics. MediNet-XG is deliberately constrained to a model size of roughly 342 KB and is optimized for edge inference, aligning with the computational realities of rural Bangladeshi users.

- **Regional scope without cross-domain extension to weeds:** IndoHerb successfully addresses Indonesian herbal species but does not consider weedy growth forms or mixed crop–weed settings that complicate recognition. MediNet-XG extends the line of work to weedy medicinal plants in agricultural landscapes, addressing both species-level recognition and the presence of dense weed backgrounds.

Title: Explainable Deep Learning in Plant Phenotyping[6]

The review paper surveys recent applications of explainable deep learning in plant phenotyping, with an emphasis on how saliency-based and attribution methods can reveal biologically meaningful features in complex plant images. The authors argue that interpretability is essential for scientific discovery and user trust, since plant scientists and breeders require insight into why deep models focus on particular structures rather than accepting opaque predictions.
Methodology:Relevant phenotyping studies were collected from major databases and grouped according to task type, including stress detection, yield-related trait estimation and organ segmentation, as well as according to imaging modality such as RGB, hyperspectral and 3D point clouds. For each study the review examined the underlying deep architecture (e.g., CNNs, U-Nets, transformers), the XAI technique applied (Grad-CAM, Guided Backpropaga-

tion, Layer-wise Relevance Propagation, SHAP and others), the way explanations were visualized and the extent to which experts validated the highlighted regions or relevance scores. The authors also discussed evaluation protocols for explanations, covering qualitative expert assessment and emerging quantitative metrics, and identified common pitfalls such as instability of saliency maps and the risk of misinterpreting artifacts as biologically relevant signals.
Result: The review concludes that XAI has begun to uncover meaningful plant traits and stress patterns in deep models, but that explanations are still inconsistently applied, rarely standardized across studies and are mostly limited to research platforms rather than embedded in field-ready tools.

Table 2.4: Key findings from "Explainable Deep Learning in Plant Phenotyping"[6]

Aspect	**Typical practice reported**	**Noted gap / limitation)**
XAI Methods used	Grad-CAM and related saliency maps most common; some use LRP or Integrated Gradients	Provide intuitive heatmaps over leaves, stems and fruits but explanations are often qualitative only
Application domains	Stress detection, nutrient deficiency, yield trait estimation, organ segmentation in crops	Demonstrate that highlighted regions frequently match known stress markers or trait-relevant organs
Expert involvement	Plant scientists or breeders visually inspect explanation maps in many studies	Expert feedback validates biological plausibility but systematic quantitative evaluation remains limited
Deployment focus	Research and phenotyping platforms; little emphasis on mobile or edge deployment	Interpretability research is strong, but real-time field tools remain underexplored

In Tabel 2.4 summary of explanation techniques, application domains and evaluation practices reported in the review on explainable deep learning for plant phenotyping.

Limitations:

- **Focus on crops and phenotyping, not medicinal plants:** The surveyed works center on staple crops and experimental phenotyping tasks such as stress or yield analysis, with no dedicated focus on medicinal species or ethnobotanical applications. MediNet-XG extends explainable deep learning to Bangladeshi weedy medicinal plants, aligning XAI with traditional healthcare and biodiversity use cases.
- **Lack of coupling between XAI and structured knowledge:** Explanations in phenotyping studies rarely connect to structured botanical or management knowledge;

they mainly show where the model "looks" without linking to actionable information. MediNet-XG couples its XAI outputs with a curated JSON knowledge base containing uses, compounds and safety notes so that highlighted regions and class decisions are directly tied to interpretable medicinal context.

- **Limited attention to lightweight, field-deployable models:** The review notes very little emphasis on building ultra-compact architectures that can run on mobile or edge devices in real field conditions. MediNet-XG explicitly constrains model size to approximately 342 KB and targets weed-infested field imagery for deployment in resource-limited rural environments.

Title: Deep Learning for Weed Detection in Wheat Fields.[7]

The research develops deep-learning models for detecting weeds in wheat fields using RGB imagery, aiming to support precision herbicide application and reduce manual scouting effort. The study demonstrates that CNN-based classifiers can accurately distinguish weed patches from crop regions under varying field conditions which highlights the potential of deep learning for real-world weed management.

Methodology:High-resolution images of wheat fields were collected across multiple sites and growth stages, capturing a wide range of lighting conditions, soil appearances and weed densities. Images were annotated to label pixels or regions as crop or weed then split into training, validation and test sets to ensure coverage of different environments. Several CNN architectures were trained for classification and detection tasks, with data augmentation applied to improve robustness, and models were evaluated using accuracy and related metrics to quantify performance on unseen field data.

Result: The findings indicate that well-trained CNNs can achieve test accuracies above 95 percent for weed detection tasks and can operate effectively in visually heterogeneous wheat fields, supporting their use in precision agriculture workflows.

Table 2.5: Representative performance of CNN models for weed detection in wheat fields[7]

Model	**Task**	**Acc(%)**	**Notes**
CNN-1	Crop vs weed classification	95%	Baseline weed detection network
CNN-2	Enhanced architecture / detection	95%	Similar or slightly higher accuracy with better localization

Table 2.5 reported accuracies for CNN-based weed detection models in wheat fields showing

that deep learning can reliably distinguish weeds from crops in complex field imagery.

Limitations:

- **Binary weed vs crop focus rather than species-level medicinal identification:** The models differentiate weeds from wheat but do not identify individual weed or medicinal species which limits their usefulness for ethnobotanical or medicinal applications. MediNet-XG instead performs species-level classification across 34 weedy medicinal plants relevant to Bangladeshi healthcare and agriculture.
- **No integration of explainable AI for individual predictions:** The study emphasizes detection accuracy and does not incorporate Grad-CAM, t-SNE or similar explanations to show which leaf or canopy regions drive each decision. MediNet-XG embeds Grad-CAM and feature-space context as mandatory outputs for each prediction to enhance transparency and user trust.
- **bsence of structured medicinal knowledge and safety information:** Weed detection outputs are not linked to any knowledge base about plant uses, toxicity or therapeutic relevance which is critical when dealing with medicinal or poisonous species. MediNet-XG connects each predicted class to a curated JSON knowledge base containing uses, compounds and safety constraints and restricts textual responses to retrieval from the source.
- **Deployment not optimized for ultra-lightweight on-device inference:** While field applicability is considered, the work does not target an extremely small model footprint or strict memory limits suitable for low-end mobile devices. MediNet-XG is explicitly designed to remain around 342 KB to enable reliable on-device inference in rural Bangladeshi contexts.

Title: An Efficient Dual-Attention Guided Deep Learning Model with Optimized Hyperparameters for Bangladeshi Medicinal Plant Classification.[8]
The paper proposes a dual-attention CNN architecture with optimized hyperparameters for classifying Bangladeshi medicinal plants, aiming to improve feature extraction and recognition accuracy over conventional CNN models on leaf imagery. The study demonstrates

that attention mechanisms can enhance representation of fine-grained leaf patterns and yields state-of-the-art performance on a Bangladeshi medicinal plant dataset.

Methodology:A dataset of Bangladeshi medicinal plant leaves was assembled and preprocessed through resizing, normalization and data augmentation to mitigate overfitting. The authors designed a custom CNN incorporating both spatial and channel-wise attention modules, allowing the network to focus on informative regions and feature channels within leaf images. Hyperparameters such as learning rate, batch size and optimizer settings were tuned using systematic search strategies, and the dual-attention model was trained and evaluated against baseline CNNs using accuracy and other metrics on a held-out test set.

Result: Reported test accuracies for individual CNN backbones and neural ensemble configurations on the ten-class Bangladeshi medicinal plant leaf dataset, showing the performance gains achieved by ensemble learning over single-model baselines.

Table 2.6: Performance comparison of dual-attention CNN vs baseline models for Bangladeshi medicinal plants[8]

Model	**Architecture**	**Acc(%)**	**Notes**
Baseline CNN	Standard convolutional network	$> 95\%$	Strong baseline accuracy
Dual-attention CNN	Spatial + channel attention with tuned hyperparameters	$\approx 97\%$	Highest reported performance on the dataset

Table 2.6 present test accuracies for baseline and dual-attention CNN models on a Bangladeshi medicinal plant dataset, showing accuracy gains achieved through attention mechanisms and hyperparameter optimization.

Limitations:

- **Model complexity and limited suitability for low-resource devices:** The dual-attention architecture increases computational cost and parameter count compared with simpler CNNs and the study does not target strict constraints for deployment on low-end mobile or edge hardware. MediNet-XG is deliberately designed as an ultra-lightweight CNN with approximately 342 KB of parameters to enable practical on-device inference.
- **Lack of explainability despite use of attention:** Although attention mechanisms highlight informative features internally, the model does not expose explicit explanation outputs such as Grad-CAM maps or feature-space plots for end users. MediNet-XG

complements its compact architecture with mandatory Grad-CAM and t-SNE-based explanations so that human users can see which leaf regions and class neighborhoods support each prediction.

- **Absence of knowledge-grounded medicinal context:** The dual-attention model focuses solely on species classification and does not link outputs to structured information on medicinal uses, phytochemical constituents or toxicity, nor does it enforce safety constraints on system responses. MediNet-XG associates each species label with a curated JSON knowledge base that encodes uses, compounds, habitats and safety notes, and all textual explanations are restricted to retrieval from the knowledge base.

- **No explicit consideration of weed-infested field imagery:** The dataset used in the study consists mainly of controlled leaf images and does not account for cluttered weed backgrounds or small, occluded leaves that frequently occur in real agricultural settings. MediNet-XG is trained and evaluated on leaf images captured in weed-infested environments, addressing recognition under realistic field conditions in Bangladesh.

Table 2.7: Comparative Analysis of Recent Studies vs. Proposed MediNet-XG Model

Title	**Plant**	**Modality**	**Real-time**	**Classify**	**XAI**	**Gen**	**Acc/F1**	**Country**
Deep-Learning-Based Class. (Mathematics, 2023)[3]	BD medicinal (10 spp., controlled leaf)	Image	No	Yes	No (Black-box)	No	97.62% (Soft Ens.)	Bangladesh
DL for Med. Plant: Review (Frontiers, 2024)[4]	Multi-country survey	Image+	No	Yes	Rarely	No	NA	Global
IndoHerb: Indonesia (Comp. Elec. Agri, 2023)[5]	Indonesian herbal	Image	No	Yes	No	No	$\sim$97% (TL CNN)	Indonesia
Explainable DL: Plant Phenotyping (Frontiers, 2023)[6]	Gen. crops phenotyping	Img+Data	No	Yes	(Grad-CAM)	No	NA	Global
DL for weed detection: Wheat (Comp. Elec. Agri, 2024)[7]	Weeds vs crop in wheat	Image	Field DL	Yes	Limited	No	$>$95%	Turkey
Dual-attention model (Journal TBD, 2025)[8]	BD medicinal plants	Image	No (Heavy)	Yes	No	No	$\sim$ 97%	Bangladesh
Novel Multistage Approach (IRJMITE, 2025)[14]	Med. vs Fruit vs Flower	Image	No	Yes	No	No	$>$95%	Global
DL for weed detection: Review (Frontiers, 2026)[15]	Complex field weeds	Image+	Edge focus	Yes	Sparse	No	NA	Global
MediNet-XG (Proposed)	**BD weedy medicinal (34 spp., field)**	**Multimodal (Img+Text)**	**Yes**	**Yes**	**Yes**	***yes**	**98% ($\approx$342KB)**	**Bangladesh**

*Note: Retrieval-based structured knowledge; non free-form generative.

2.2 Summary

Existing literature from Table 2.7 shows that deep learning has achieved very high accuracy for medicinal plant and weed recognition, but most solutions remain heavy, black-box models trained on controlled images rather than realistic field data. Ensemble and attention-based CNNs for Bangladeshi medicinal plants regularly report accuracies above 97–98 percent, yet they are computationally expensive, lack per-sample explanations and do not connect predictions to structured medicinal knowledge or safety information. Systematic reviews further highlight major gaps: scarcity of public field-captured datasets, limited attention to lightweight edge deployment and only sporadic use of explainable AI methods, which are mostly confined to crop phenotyping and disease diagnosis rather than medicinal species in weed-infested environments. Within the landscape MediNet-XG is positioned to fill four specific gaps: (i) species-level recognition of weedy medicinal plants from cluttered Bangladeshi field images, (ii) an ultra lightweight architecture suitable for on-device inference, (iii) mandatory visual and feature-space explainability for every prediction and (iv) retrieval-based coupling of outputs to a curated botanical knowledge base that encodes uses, compounds and safety constraints, thereby addressing both trustworthiness and real-world deployability.

CHAPTER 3
Methodology

3.1 Overview

Under this methodology approach, the research activity adheres to a concise and entirely comprehensive flow of field pictures to safe and interpretable choices. To begin with, RGB leaf images of 34 various Bangladeshi weed-infested medicinal plants are obtained in natural conditions of real weed-infested field and public repositories and standardized by correcting orientation, resizing to a desired resolution, normalizing and data augmentation to retain natural differences in illumination, background, and occlusion and make the dataset trainable.
A custom ultra-light CNN model, MediNet-XG, some pretrained models- VGG, DenseNet121, EfficientNetBO-TL, MobileNetV2, SqueezeNet are then trained on these images to predict plant species, using inverted residual blocks and depth-separable convolutions to achieve high performance while keeping the model to 342KB with precision 0.9882, precision 0.9875, recall 0.9887, and F1-score 0.9879, compared to other pretrained models MediNet-XG comes with the best performance. After the prediction of a class, the Grad-CAM maps and T-SNE feature-space plots are generate the reseon why the image belongs to that class and why the model takes that decision. The model use that predicted output class label as an input and maps to answer the query of the stakeholders with the JSON knowledge-based file and finaly provide precise, non-hallucinating answer.

3.2 Data Collection and Preprocessing

The dataset is being created through the leaf image of 24 species from field collection and 10 species form a public repositories named Mendeley[11] around Bangladesh. This dataset

contains 17,050 images divided into 34 classes, each representing a different species of the medicinal leaf. The gathering procedure seeks to create a comprehensive representation of medicinal plant of the weed infested area to aid in research and development in Geology, Geography, Environmental Science and Agricultural and Food Sciences domains. This dataset will be a valuable resource for improving the accuracy and efficiency of medicinal plant of weed infested area in Bangladesh to classification and recognition systems.

3.2.1 Data Sources

Data for the study was collected from various sources in order to have a diverse, representative and practically useful dataset for weed infested medicinal plant recognition in Bangladesh. The main source of visual information is high-resolution RGB leaf images of 34 medicinal plants species that are common in weed infested fields, taken in natural outdoor conditions using a smartphone camera. Field images were collected from several agro ecological locations, including Dinajpur, Saidpur, Parbatipur, Satkhira and Rangpur so that variations in soil background, illumination, occlusion and co-occurring weeds are naturally represented in the dataset. There are some publicly available image repositories present in online, especially Mendeley[11], which had a Bangladeshi medinical plant leaf dataset but exploited to augment species that were less commonly seen in the field, giving a final dataset of 17,050 images stored in a JPEG format, with approximately 500 images per class after preprocessing and balancing.

To complement the queries about the image data with structured domain knowledge and textual information about each of the 34 species, were compiled from recognized botanical and medicinal plant databases like MPEDB[16], BNH[17], BJPT[18] and Plants of the World Online (POWO)[19]. From the above-mentioned references, 1,020 question-answer pairs were built up ranging from the scientific and local names, morphological features, ecological preferences, traditional and pharmaceutical uses, known phytochemical constituents and safety considerations, and later encoded in a CSV file and a knowledge base in the form of a JSON document for subsequent use. Agricultural extension professionals and a university botany faculty member confirmed both the species names of the images and the information written in the text fields that the images were taken to ensure that there was taxonomic accuracy,

consistency across sources and practical relevance to stakeholders such as farmers, extension workers and pharmacists. The combination of heterogeneous collection sites, mixed image sources and expert curated textual references is providing a robust foundation on grounded knowledge the train and the evaluation of the MediNet-XG model and helps the decision support components.

3.2.2 Data Collection Techniques

The research used a purposive sampling technique in order to pick 34 medicinal plant species that often founded in the weed infested areas. The strategy allowed a fair distribution of presentation throughout the final dataset with the aim of reaching mostly 500 images of each species. Images are obtained of a variety of individual plants at different stages of growth, and at different leaf orientations to achieve large intra- and inter-class variation, and intrinsically with natural differences in shape, size, texture, and other minor physiological damage. Also, the spatial diversity was obtained through sampling and augmentation of different geographical districts such as Dinajpur, Saidpur, Parbatipur, Satkhira and Rangpur and hence the different soil background, co-occurring weed species and local agricultural management practices are taken into consideration.

The protocol to acquire the image used handheld by smartphone cameras to recreate the ecosystem of the real field although the protocol still applies a uniform method to distance, framing and focus. The techniques makes sure that the main leaves are identifiable even when the background clutters of the real life are there. Every plant was photographed in various perspectives and angles, such as top-down and oblique, and different angles of rotation, but with natural and different light sources instead of limited light sources in artificial light. The dataset achieves this by avoiding the use of tripods and using handheld acquisition to capture the dataset, which effectively combines natural variations in scale, slight motion blur, and perspective effects and is much more reminiscent of the real-world practice of end-users in a field scenario of data collection.

3.2.3 Preprocessing Steps

The preprocessing phase involved preparing the raw data for training and analysis to ensure the model operates efficiently and smoothly. The workflow for transforming raw images into a structured dataset is illustrated in Figure 3.3. This process comprises several critical activities, beginning with the standardization of the aspect ratio to a 1:1 square format to maintain homogeneity across the dataset. Subsequently, the images are resized in two stages: first to a resolution of 300×300 pixels for the stored dataset—ensuring they retain sufficient quality for various scales—and finally to 224×224 pixels for model training. This specific resolution is widely considered a standard for various machine learning architectures. By enforcing a constant 1:1 aspect ratio and uniform dimensions, the dataset achieves a high degree of homogeneity, which significantly improves the computational efficiency and performance of the trained models. Consequently, the processed images are stored and ready for further analysis, ensuring that the final dataset is of high quality and structurally consistent.

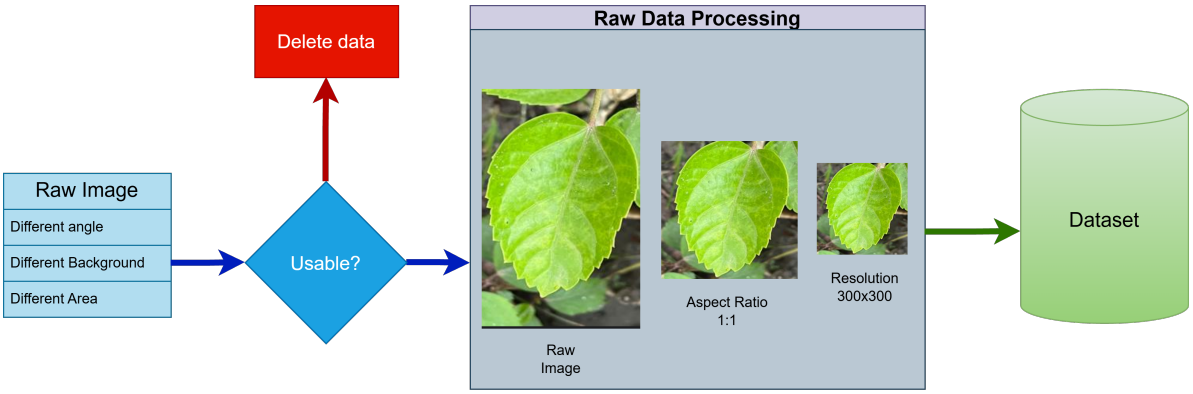


Figure 3.3: Raw data processing flowchart.

Image Resize: The collected images are resized into a standard resolution of 300x300 pixels using the Lanczos interpolation method. Lanczos interpolation works by using a weighted average of nearby pixels to calculate the new pixel value, resulting in smoother and higher quality resized images compared to other interpolation methods. This resizing ensured uniformity in image dimensions across the dataset, facilitating efficient processing and model training. Additionally, each resized image was renamed with a unique identifier indicating its class, aiding in dataset organization and management.

The Lanczos kernel $L(x)$ is defined as:

$$L(x) = \begin{cases} \text{sinc}(x) \cdot \text{sinc}\left(\frac{x}{a}\right), & \text{if } -a \leq x \leq a \\ 0, & \text{otherwise} \end{cases}$$

where $\text{sinc}(x) = \frac{\sin(\pi x)}{\pi x}$ is the sinc function, and a is the scale factor determining the size of the kernel. The interpolation formula using the Lanczos kernel is:

$$I_{\text{interp}}(x) = \sum_{n=-N}^{N} I(n) \cdot L(x - n)$$

where $I_{\text{interp}}(x)$ is the interpolated pixel intensity at position x, $I(n)$ is the intensity of the input pixel at position n, N is the number of neighboring pixels considered in the interpolation, and $L(x-n)$ is the Lanczos kernel evaluated at the distance between the output pixel position x and the neighboring input pixel position n [20].

Orientation Correction: Images containing Exif metadata with orientation tags were adjusted to ensure uniform orientation across all images. This correction process involved rotating images by 180, 270, or 90 degrees, depending on the orientation tag values (3, 6, or 8, respectively). This meticulous alignment was crucial to maintain consistency throughout the dataset, preventing any orientation discrepancies that might otherwise affect the accuracy and reliability of subsequent image processing and analysis tasks. By standardizing the orientation, the images were prepared optimally for use in various applications, enhancing overall model performance and ensuring reliable results in further analyses and visualizations.

Data Augmentation: Various data augmentation techniques were applied to increase dataset diversity and robustness. These techniques included random rotation (up to 20% rotation), random translation (up to 10% height and width translation), random zooming (up to 20% zoom), horizontal flipping, and random contrast adjustment (up to 20%). In addition, to further simulate real-world variations and improve model generalizability, techniques like random shearing, color jittering, and Gaussian blur were also employed. By incorporating these diverse augmentation strategies, the model's ability to recognize signs under different conditions and reduce overfitting was significantly improved.

Random Rotation: Calculate the rotation matrix:

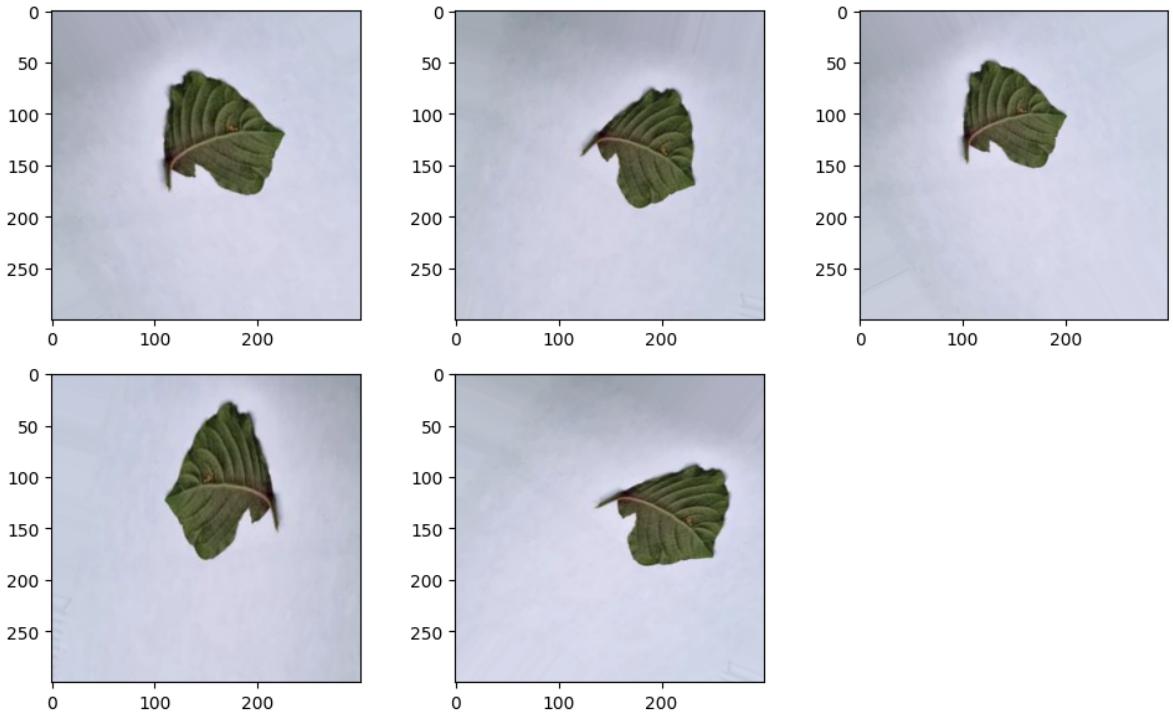


Figure 3.4: Sample augmentation

$$\text{Rotation Matrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix}$$

For each pixel in the original image, apply the rotation matrix to calculate the new coordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \text{Rotation Matrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Random Translation: Determine the translation matrix:

$$\text{Translation Matrix} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \begin{bmatrix} \Delta h & 0 \\ 0 & \Delta w \end{bmatrix}$$

Apply the translation matrix to each pixel in the original image to calculate the new coordinates:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \text{Translation Matrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Random Zooming: For each pixel in the original image, scale the coordinates by the random

zoom factor:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \alpha \begin{bmatrix} x \\ y \end{bmatrix}$$

Horizontal Flipping: For each pixel in the original image, apply the flipping matrix to flip horizontally:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Image Normalization: Min-max scaling, also known as normalization, is a preprocessing technique used to transform data to a specified range, commonly [0, 1] or [-1, 1]. The images were normalized using the Min-Max scaling method, where pixel values were scaled to a range of [0, 1] by dividing each pixel value by 255. This rescaling ensures that all features contribute equally to the model, improving the performance and convergence speed of machine learning algorithms, especially those sensitive to input data scale like neural networks. Normalization ensures consistency in pixel intensity across images, mitigates the impact of varying illumination conditions, and enhances model stability and performance. By providing uniform and standardized input data, min-max scaling enhances the effectiveness of tasks such as image classification and segmentation. Min-max scaling is a popular method used in data preprocessing to scale features to a specified range. The formula for min-max scaling is:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

To scale to a different range $[a, b]$, the formula is:

$$x' = a + \left(\frac{x - \min(x)}{\max(x) - \min(x)} \right) \cdot (b - a)$$

For images, pixel values are often scaled to [0, 1] using:

$$x' = \frac{x}{255}$$

where x is the original pixel value [21].

Train-Test-Validation Split: The dataset was split into training, validation, and test sets using a random splitting strategy (Figure 3.5) [22]. By shuffling the data and allocating a portion to each subset—typically 70% for training, 15% for validation, and 15% for testing—this method ensures that the subsets represent the overall dataset's diversity. Random splitting prevents biases arising from the inherent order of the data, allowing for more robust training and evaluation. This randomization supports the generalization of the trained model to unseen data. While the initial allocation defines the sets, the images are shuffled at the beginning of each epoch to ensure variability in the data sequence seen by the model during the training process.

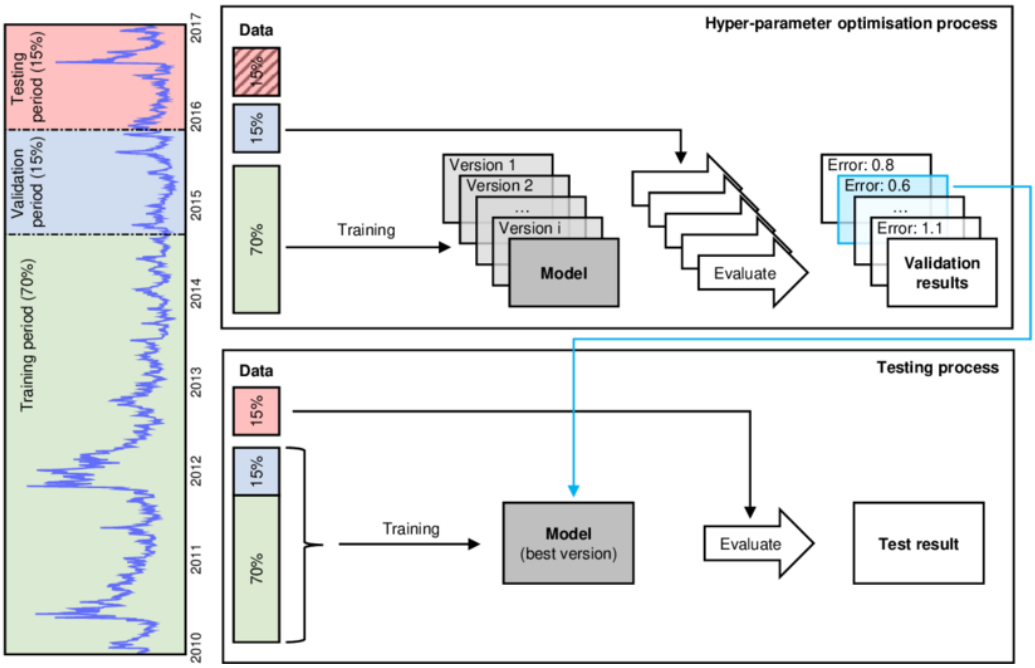


Figure 3.5: Random data splitting method [22].

3.2.4 Annotation Process

Annotation in this experiment consisted of labeling every image and the corresponding knowledge with organized identifiers in a systematic manner. This ensured the dataset could be easily organized, accessed, and processed for supervised learning. Beyond dividing the 34 species of medicinal plants into classes, the annotation encoded the frequency at which each plant was found in naturally weedy or unmanaged habitats to provide domain-specific

context to each label. For the visual modality, a rank-aware naming scheme of the form `rank_serial_plantname` was used. Here, the *rank* (1–7) represents empirical availability in weedy fields, the *serial* (1–34) denotes the fixed index of the species in the database, and the *plantname* is the standardized local/common name in lowercase without spaces (e.g., `1_1_thankuni`, `4_18_joba`, or `7_34_betal` as shown in Figure 3.6). These rank assignments were based on repeated field observations within weedy Bangladeshi habitats and remained constant across all images of a particular species. This convention allowed folder hierarchies to be automatically parsable, making data loading, class balancing, and performance evaluation straightforward. Annotation also involved a manual verification process. All labeled images were cross-referenced with the master species list and field notes to ensure that observable morphology—including leaf shape, venation, and margin—correctly matched the assigned class and its rank of availability.

3.2.5 Dataset Description

The dataset, encompassing 17,050 images across 34 classes (Figure 3.7), serves as a foundational resource for advancing computer vision research, particularly in the realm of MediNet-XG. With an average of approximately 500 images per class, this collection provides a robust foundation for training and testing machine learning algorithms effectively. The predominant use of the `.jpg` format ensures seamless integration and compatibility across diverse platforms and frameworks. Figure 3.7 displays the class distribution; the y-axis lists the 34 medicinal plant classes, while the x-axis indicates the total count of images within each class. Each bar's width signifies the number of images in that particular class. It is evident that the dataset is relatively balanced, with the number of images per class ranging between approximately 500 and 530. However, slight variations exist, with two classes having slightly more images than others. The balanced nature of this dataset suggests that it is suitable for training machine learning models, reducing the likelihood of bias towards any particular class. Class 7_31_aloe_vera and 4_19_sojina stands out with the two highest image count, totaling 530 and 520 images, accounting for about 3.1% and 3.05% of the dataset. Conversely, rest of the classes contain 500 images, representing approximately 2.94% of the dataset. This balanced distribution across classes supports comprehensive exploration across a spectrum of visual recognition tasks, ranging from image classification to object detection and pattern recogni-

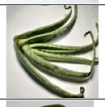
Sl	Image	Annotation	Sl	Image	Annotation	Sl	Image	Annotation
1		1_1_thankuni	13		4_13_tulsi	25		6_25_amloki
2		1_2_pathorkuchi	14		4_14_ramtulshi	26		6_26_haritoki
3		1_3_apang	15		4_15_basok	27		6_27_bohera
4		1_4_kalomegh	16		4_16_neem	28		6_28_oshwagandha
5		2_5_kaluchitra	17		4_17_pudina	29		6_29_devils_backbone
6		2_6_ayapan	18		4_18_joba	30		7_30_lemongrass
7		2_7_akondo	19		4_19_sojina	31		7_31_aloe_vera
8		2_8_bamonhati	20		5_20_gandha	32		7_32_nayontara
9		3_9_kalodhutra	21		5_21_shotomuli	33		7_33_longevity_spinach (gainura)
10		3_10_tarulata	22		6_22_sarpagandha	34		7_34_betal
11		4_11_punarnava	23		6_23_anantamul			
12		4_12_agnishikha	24		6_24_chapalish			

Figure 3.6: Visualization of the 34 medicinal plant classes with their rank-aware identifiers [1].

tion. Moreover, the dataset is enriched by data collected from 5 different districts and a public repository, ensuring a balanced geographic medicinal plant collection representation that enhances the dataset's inclusivity and applicability to real-world scenarios. This diversity not only mitigates potential biases but also broadens the dataset's utility in addressing varied perspectives and contexts. By providing such a diverse and meticulously structured dataset, researchers are empowered to push the boundaries of computer vision capabilities. This dataset not only fosters technical advancements but also promotes ethical considerations, contributing to a more equitable development of artificial intelligence. The provided JSON file serves as a comprehensive knowledge base for 34 medicinal plant species each of containing 30 different, structured to facilitate grounded and safety constrained responses to user queries. For each entry, it provides formal botanical data, including scientific names, families, and specific

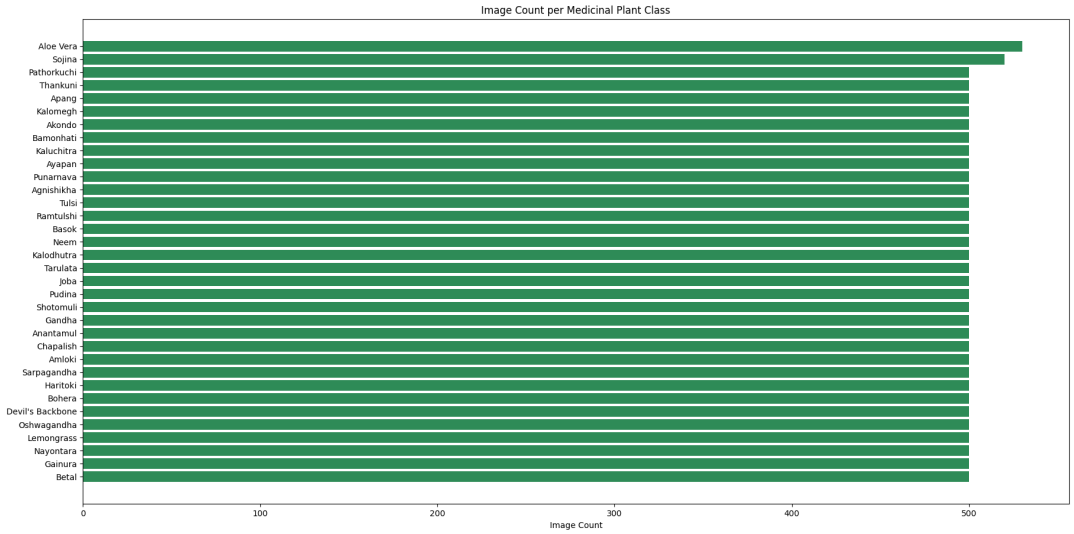


Figure 3.7: Class distribution in the dataset.

leaf morphology. Beyond taxonomy, the dataset details pharmacological profiles such as primary active compounds and therapeutic properties, while emphasizing safety through specific toxicity levels, contraindications, and pregnancy safety guidelines. It also includes practical administration data, covering standard dosages and both internal and external preparation methods, such as decoctions or poultices. This structured approach ensures that when a plant is predicted, the system can return authoritative botanical and pharmacological information while imposing critical safety constraints on the textual output.

3.3 Pre-trained Models for Comparative Analysis

3.3.1 EfficientNetB0

EfficientNetB0 is chosen as a powerful benchmark to leverage high-precision image recognition within limited computational resources. The model is the lightest of the EfficientNet family and is based on Mobile Inverted Bottleneck (MBConv) blocks, depthwise separable convolutions, and squeeze-and-excitation (SE) channel attention. These features enable it to reach competitive accuracy with approximately 5 million parameters and a 224×224 input size (Figure 3.10). Another important characteristic of EfficientNetB0 is the concept of compound scaling, where the depth and width of the network are scaled together with the input resolution rather than individually. This approach offers a principled trade-off between accu-

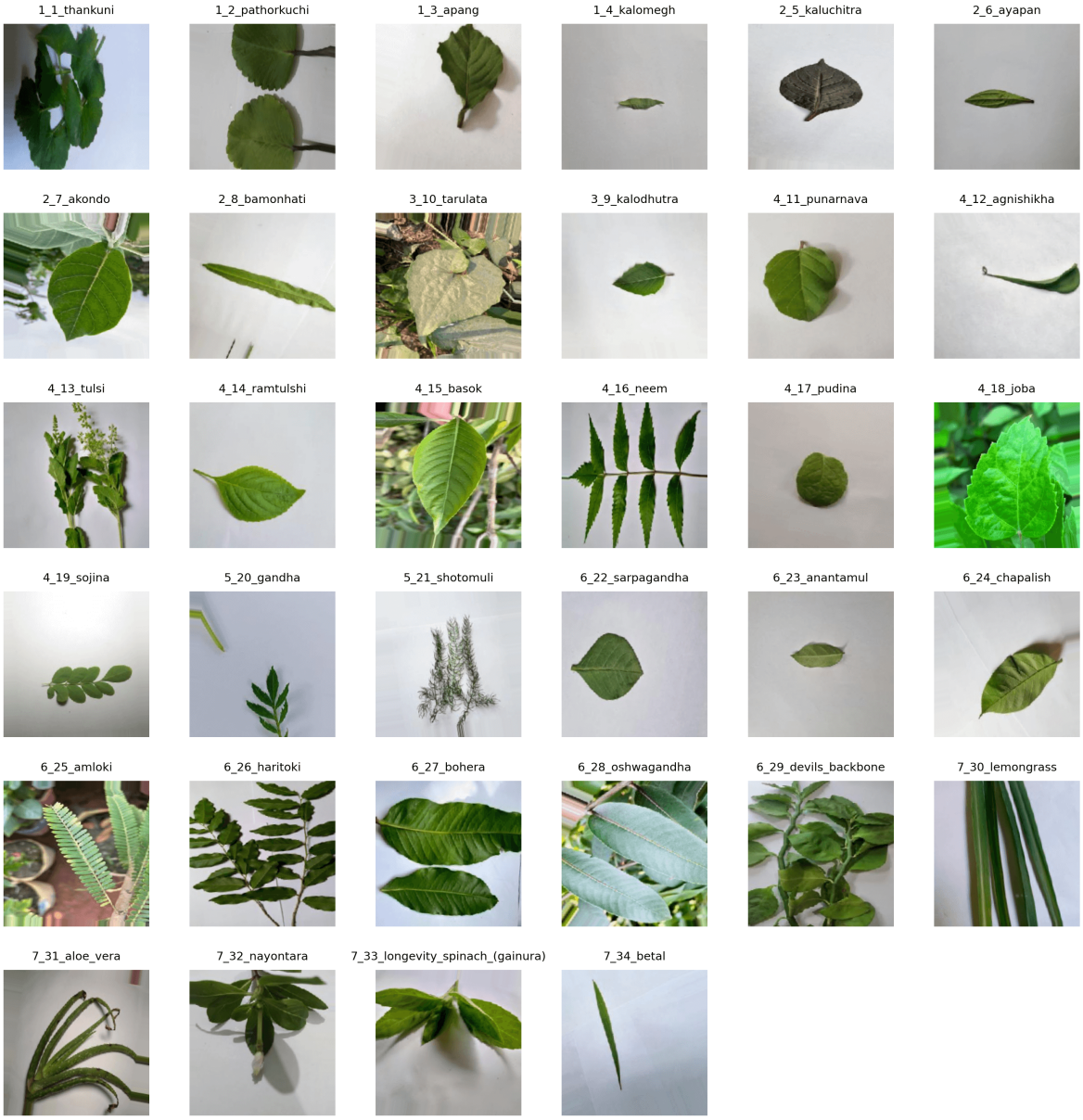


Figure 3.8: Dataset annotation.

racy and efficiency, making the B0 variant an excellent choice as a transfer-learning backbone for small and medium-sized datasets.[23].

3.3.2 MobileNetV2

ResNet-50 uses shortcut connections, convolutional layers, and batch normalization to reduce vanishing gradients in deep networks. In contrast, MobileNetV2 was chosen for its lightweight and efficient design, which is critical in resource-constrained situations such as mobile and embedded devices (Figure 3.11). It has inverted residuals and linear bottlenecks,

```
"1_1_thankuni": {
    "Scientific Name": "Centella asiatica",
    "Botanical Family": "Apiaceae",
    "Genus & Species": "Centella / asiatica",
    "Local/Vernacular Names": "Bengali: Thankuni; English: Indian Pennywort, Gotu Kola; Hindi: Brahmi",
    "Leaf Morphology": "Orbicular-reniform, 3-5cm across, crenate or sub-entire margins, glabrous, radiat
    "Plant Habit": "Herb",
    "Life Span": "Perennial",
    "Geographical Origin": "Indian Subcontinent",
    "Native Habitat": "Tropical wetlands, marshy areas, tropical forests",
    "Global Distribution": "India, Bangladesh, Sri Lanka, Southeast Asia, Africa",
    "Preferred Soil Type": "Loamy, well-drained, moist soil",
    "Climate Requirements": "Tropical, 20-30\u00b0C, high humidity, 1500-2500mm rainfall annually",
    "Primary Active Compounds": "Asiaticoside, madecassoside, asiatic acid, madecassic acid",
    "Secondary Metabolites": "Flavonoids, phenolic acids, triterpenes, alkaloids",
    "Essential Oils": "Trace amounts of volatile oils",
    "Nutritional Profile": "Vitamin C, iron, potassium, calcium, magnesium, amino acids",
    "Therapeutic Properties": "Cognitive enhancement, wound healing, anti-inflammatory, antioxidant, anxi
    "Traditional Uses": "Ayurvedic brain tonic, memory enhancement, skin conditions, fever",
    "Modern Clinical Uses": "Treatment of cognitive disorders, varicose veins, cellulite, anxiety managem
    "Medicinal Part Used": "Whole aerial part, leaves preferred",
    "Standard Dosage": "3-6g dried herb daily, or 500-1500mg extract daily",
    "Internal Preparation Method": "Decoction, infusion, powder capsules, standardized extract",
    "External Application Method": "Poultice, herbal paste, oil infusion, topical cream",
    "Culinary Uses": "Edible leaf in salads, herbal teas, smoothies, traditional curries",
    "Extraction Process": "Aqueous extraction, alcoholic extraction (60-70% ethanol), CO2 extraction",
    "Toxicity Levels": "Very low toxicity; LD50 >5000 mg/kg in animal models",
    "Contraindications": "Pregnancy (large doses), lactation (caution), estrogen-sensitive tumors",
```


Figure 3.9: Knowledge based JSON file structure.

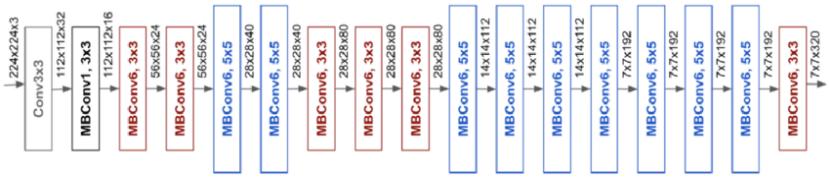


Figure 3.10: The EfficientNetB0 architecture [23].

and it uses lightweight depthwise separable convolutions to reduce parameters and computing costs while keeping excellent accuracy [24].

3.3.3 DenseNet121

DenseNet121 is selected for its unique dense connectivity pattern, where each layer is directly connected to every other layer in a feed-forward manner. This architecture promotes extensive feature reuse and facilitates robust gradient flow throughout the network, enabling efficient learning of complex representations while minimizing the number of parameters. The compact design of DenseNet121 (Figure 3.12), coupled with its demonstrated high accuracy, renders it particularly suitable for environments constrained by computational resources yet demanding exceptional performance. Furthermore, the efficient layer connections en-

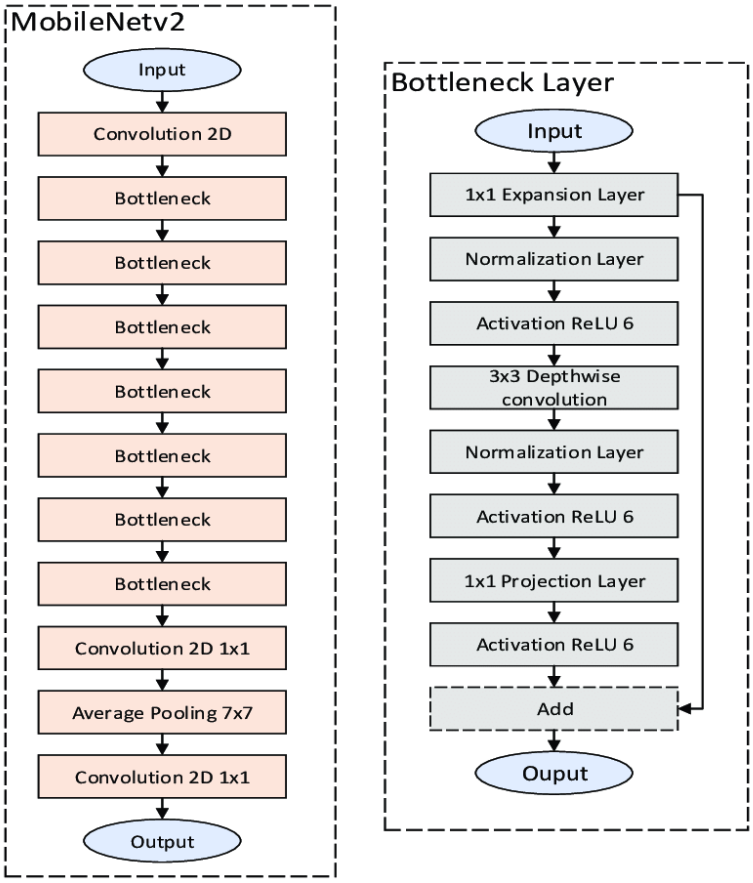


Figure 3.11: The MobileNetV2 architecture [24].

hance information propagation and mitigate issues like vanishing gradients, ensuring stable and effective training even with deep networks. This makes DenseNet121 an ideal choice for applications requiring high performance with limited computational power, providing a balance of speed, accuracy, and resource efficiency [25].

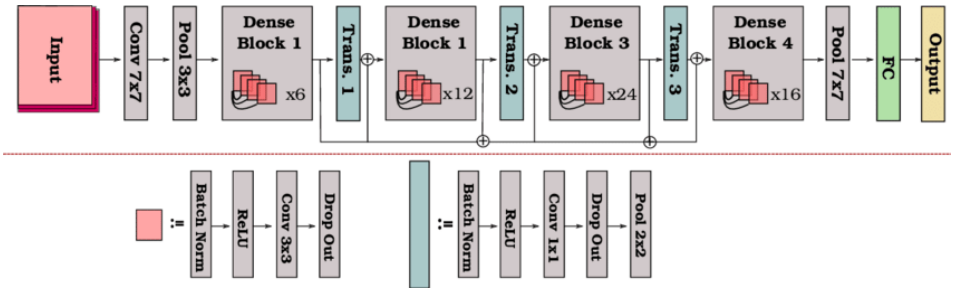


Figure 3.12: The DenseNet121 architecture [25].

3.3.4 VGG16

VGG16 is chosen for its widely recognized and deeply layered architecture, serving as a foundational model for numerous image classification tasks. Its straightforward design consists of multiple layers with 3x3 convolutional filters, which efficiently capture fine-grained details essential for distinguishing subtle nuances in sign language images. The architecture of VGG16 includes five convolutional blocks, each followed by max-pooling layers for spatial downsampling, culminating in three fully connected layers (Figure 3.13). Despite its depth of 16 layers, VGG16 remains computationally feasible for modern GPUs, ensuring practical deployment and robust performance across diverse backgrounds and applications requiring reliable image classification [26].

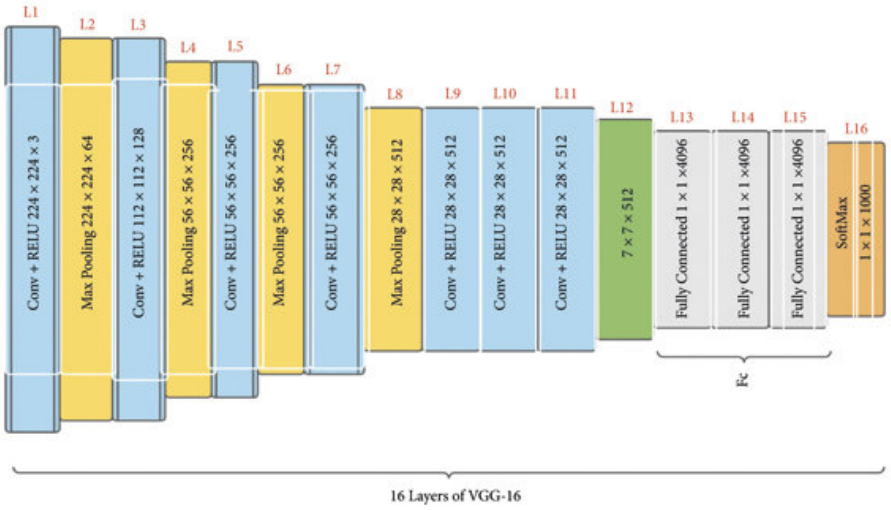


Figure 3.13: The VGG16 architecture [26].

3.3.5 SqueezeNet

SqueezeNet is also incorporated as a lightweight comparison model in which the focus of the parameters is put on reducing them but still attaining an acceptable accuracy in image classification tasks. The model consists of "Fire" units with the first half using a squeeze layer followed by 1×1 convolutions to reduce channel dimension and then an expand layer that combines 1×1 and 3×3 convolutions, reducing the number of parameters by a much

larger factor but not reducing representational power compared to VGG-style networks (Figure 3.14). By utilizing large convolutions e.g., 1×1, delayed downsampling and thoughtful filter placement, SqueezeNet can be brought down to the performance of significantly larger models with many fewer parameters to consider it an effective baseline in the context of comparing compact architectures like MediNet-XG to run on a resource-constrained device.[26].

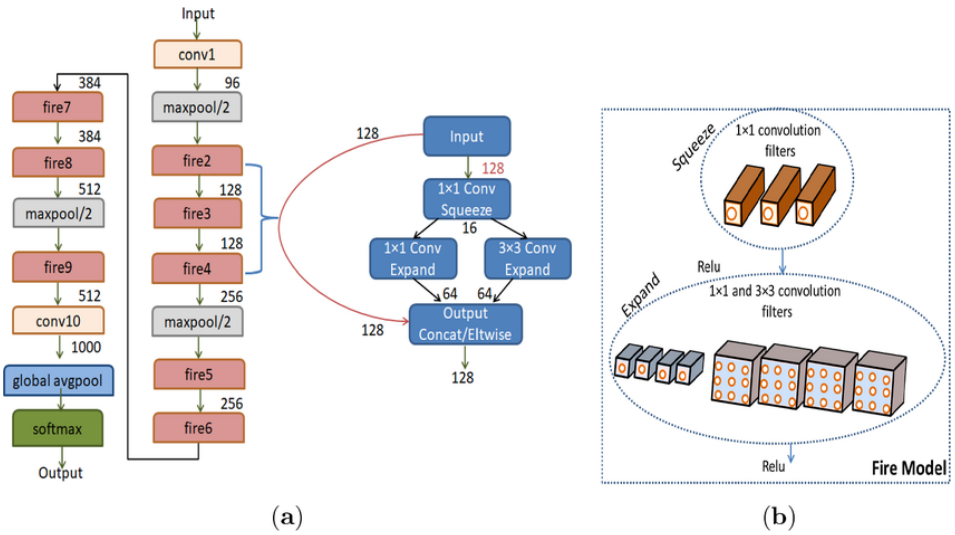


Figure 3.14: The SqueezeNet architecture [27].

3.4 Custom Model: MediNet-XG

The MediNet-XG framework is structured into three consecutive stages namely compact visual classification, explanation of reasoning and constrained knowledge retrieval (Figure 3.15). The MediNet classifier is offered in the initial stage, which takes a preprocessed RGB leaf image of a weed-infested setting and provides the calibrated probability distribution of 34 Bangladeshi medicinal plant classes along with top-K candidate labels. The second phase makes the prediction transparent, informative with Grad-CAM heatmaps, t-SNE visualisation of the features, and a JSON-based botanical knowledge lookup which presents only curated data on morphology, uses, compounds and safety.

The high-level features of each image are obtained with MediNet, a decision is made, and Grad-CAM is used to show the regions of the image that provided the motivation to make this decision, e.g., the edges of the leaves or the veins. A low-dimensional representation of

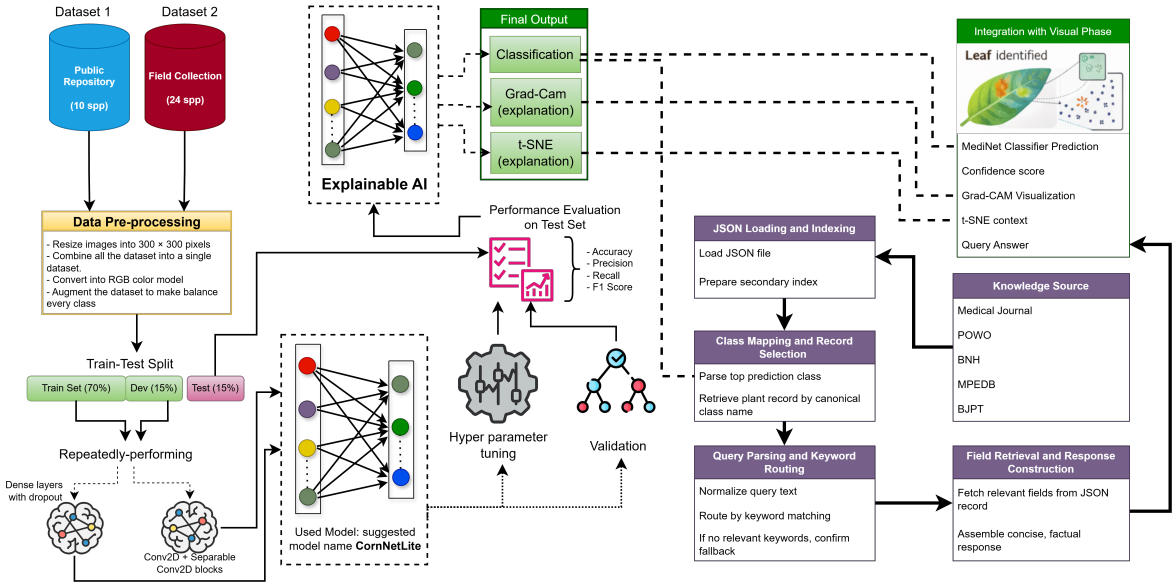


Figure 3.15: Abstract View of MediNet-XG.

the same feature representation is then projected using t-SNE to ensure that the closeness to other species can be viewed qualitatively. Lastly, the predicted class identifier is looked up in a deterministic fashion against a structured JSON record, which contains pre-defined fields to retrieve in response to user queries, so that all of the textual responses are knowledge-based and aware of safety instead of generative or speculative.

3.4.1 MediNet classifier

It is based on the MediNet classifier designed as a small convolutional backbone optimized to identify medicinal plants in the weed-infested field in the conditions of low memory footprint and explainability compatibility. The images are first processed to a predetermined spatial resolution and fed into a shallow stem and a series of inverted residual blocks using depth-wise separable convolutions and a lightweight channel attention to squeeze out discriminative features on leaf at a low computational cost. The network generates a dense feature tensor which is concatenated with global average pooling and projected into a 34-dimensional soft-max layer producing probabilities over the medicinal plant classes needed by the following explanation and knowledge modules. The design decisions have been informed by the re-quirement to use mobile or edge hardware and to retain sufficient representational power to make finer distinctions between visually similar leaves and to make intermediate activations that are amenable to Grad-CAM and t-SNE image analysis.

Architecture

The general MediNet architecture shown in Figure 3.16 has a simple encoder form of layout which progressively converts the 2D input image into a semantic compact form. The first level of a convolutional stem then carries out low-level feature extraction and downsampling of space followed by subsequent stages of a hierarchy of inverted block residual expansion that progressively deepens channels at the cost of spatial resolution. Every step is defined by a desired number of output filters, a growth factor that controls the channel expansion within the block, a downsampling stride and a binary option that enables channel attention on a subset of blocks to highlight informative feature maps. Once this stage has been completed, it is followed by a 1x1 convolution that increases the channel dimension to a larger bottleneck, after which the bottleneck is collapsed across spatial dimensions with global average pooling to create one feature vector; dropout regularization and a fully connected softmax classifier are the final parts of the architecture. Such an arrangement produces a model-sized classifier of the order of a few hundred kilobytes and is still capable of high accuracy on the 34-class medicinal plant dataset. The MediNet computation can be described as a mapping

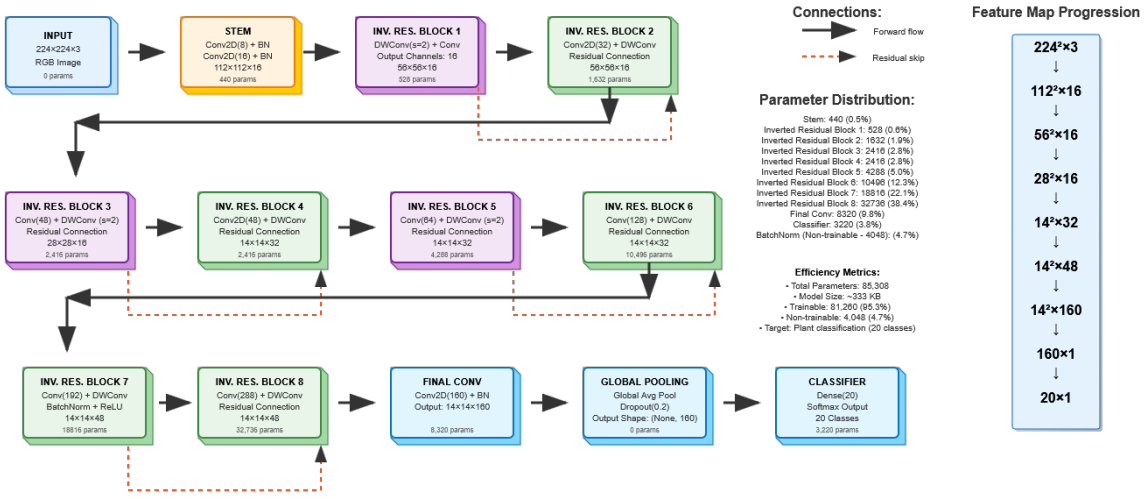


Figure 3.16: The MediNet classifier architecture.

$$f_\theta : \mathbb{R}^{H\times W\times 3} \to \Delta^{C-1}$$

where a normalized RGB image of size $H \times W$ is mapped to the probability simplex over $C = 34$ classes by a parameter set θ.

1. **Input and rescaling layer**

   - **Input:** A standardized leaf image of shape $(224, 224, 3)$ following preprocessing.
   - **Rescaling:** A rescaling operation divides pixel intensities by a constant factor to ensure values lie within a compact range. This stabilizes optimization and prevents scale-induced gradient issues.
   - **Normalization:** Conceptually, this step computes $x_0 = \frac{x}{255}$, where x denotes the raw image tensor, preparing the data for subsequent convolutional layers.

2. **Initial convolutional stem**

   - A 3×3 convolution with a small number of output filters and stride 2 is applied to capture local edge, texture, and color patterns while halving the spatial resolution to $(112, 112)$.
   - Batch normalization follows to normalize feature statistics across the batch, which supports faster convergence and reduces internal covariate shift.
   - A ReLU6 activation is used to introduce nonlinearity while capping activations at a finite range, which is advantageous for low-precision hardware.
   - This stem serves as a generic feature extractor, kept shallow to reduce parameters.

3. **Inverted residual stages with depthwise separable convolutions**

   - The core of MediNet consists of multiple stages, each formed by one or more inverted residual blocks parameterized by (F, t, s, a), where F denotes the number of output filters, t the expansion factor, s the stride, and a a boolean controlling attention.
   - Within an inverted residual block, three main steps are performed:
     (a) **Pointwise expansion (1×1):** The input tensor with C_{in} channels is expanded to $C_{\text{exp}} = t \times C_{\text{in}}$ channels.
     (b) **Depthwise convolution (3×3):** A kernel size 3×3 and stride s is applied channel-wise to reduce parameters.

(c) **Pointwise projection** (1×1)**:** Projects features back to F channels, acting as a channel mixer.

- A depthwise separable convolution can be expressed as a composition of depthwise and pointwise components. For an input feature map $X \in \mathbb{R}^{H \times W \times C}$, the depthwise step applies C different spatial kernels $K_{\text{dw}}^{(c)}$:

$$Y_{\text{dw}}(h, w, c) = \sum_{i,j} K_{\text{dw}}^{(c)}(i, j) X(h + i, w + j, c)$$

$$Y_{\text{out}}(h, w, k) = \sum_{c} K_{\text{pw}}(1, 1, c, k) Y_{\text{dw}}(h, w, c)$$

4. **Lightweight channel attention within selected blocks**

   - In specific stages where the parameter a is true, a lightweight channel attention mechanism is inserted to recalibrate feature responses.
   - This module computes a channel descriptor through global pooling, generating per-channel weights to emphasize informative channels.

5. **Stage progression and spatial downsampling**

   - Strides greater than 1 are used sparsely to reduce spatial resolution down to progressively smaller feature maps.
   - This progression balances the need for high-resolution spatial detail with the requirement for high-level semantic representations.

6. **Final** 1×1 **convolutional bottleneck**

   - After the inverted residual stack, a final 1×1 convolution expands the channel dimension to a fixed high value, forming a global bottleneck feature tensor.

7. **Global average pooling**

   - Collapses spatial dimensions into a vector $z \in \mathbb{R}^{C'}$ by averaging each channel:

$$z_c = \frac{1}{H'W'} \sum_{h=1}^{H'} \sum_{w=1}^{W'} F_c(h, w)$$

8. **Dropout regularization**

- A dropout layer randomly sets a fraction of elements in z to zero during training to mitigate overfitting on the 34-class dataset.

9. **Dense softmax classifier**

   - Maps the vector z to logits $u \in \mathbb{R}^{34}$ using $u = Wz + b$.
   - The softmax function converts logits into class probabilities:

$$p_i = \frac{\exp(u_i)}{\sum_{j=1}^{34} \exp(u_j)}$$

10. **Intermediate feature exposure for explainability**

    - Feature maps are retained for Grad-CAM computations, allowing gradients to be traced back for model transparency.
    - Bottleneck features are used to feed t-SNE, creating a 2D embedding space for class-level similarity analysis.

Through this layered design, MediNet achieves a balance between parameter efficiency, discriminative performance, and explainability readiness, forming the computational backbone of the broader medicinal plant intelligence system.

3.4.2 Grad-Cam

Grad-CAM has been employed to highlight spatial regions of each leaf image that most strongly influence the predicted medicinal plant class. Class-specific gradients are backpropagated from the softmax output to a selected final convolutional layer, and those gradients are used to compute a coarse localization heatmap over the feature maps. The resulting heatmap is resized and overlaid onto the original image so that attention to leaf margins, venation, or texture can be visually inspected, helping to verify that the model attends to meaningful botanical structures rather than background clutter.

Architecture

Grad-CAM[2] has been implemented as an auxiliary computation graph that shares inputs with MediNet, taps the output of the last convolutional layer, and also accesses the final class scores (Figure 3.17). A gradient tape or equivalent mechanism captures the derivatives of the chosen class score with respect to the convolutional feature maps and aggregates them into channel-wise importance weights before producing the final heatmap. The architectural

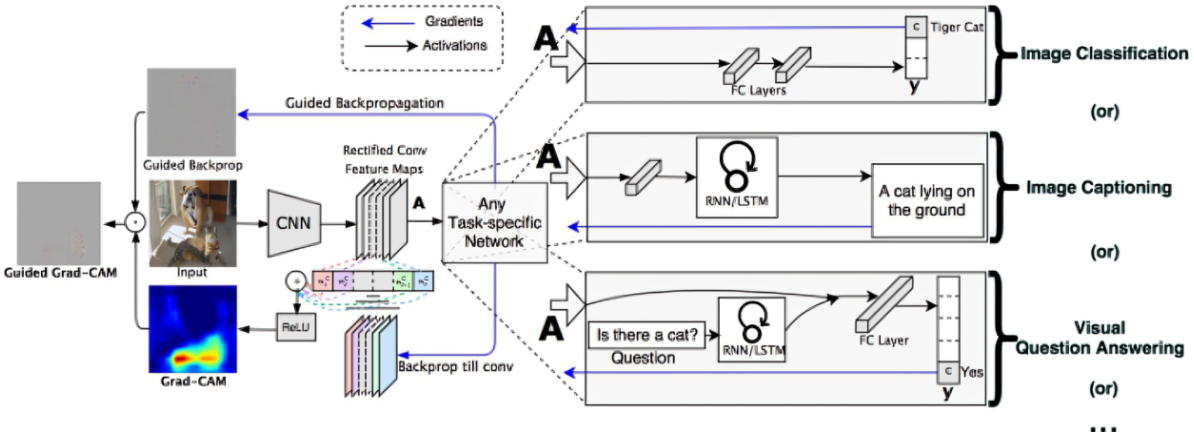


Figure 3.17: The Grad-CAM architecture.[2]

layers-

- **Shared input and forward pass:** The preprocessed image is passed through MediNet until the last convolutional layer and the final classifier output are obtained.
- **Gradient capture:** A gradient operation computes partial derivatives of the predicted class score with respect to the last convolutional feature maps, yielding one gradient map per channel.
- **Channel weighting:** Each channel is assigned a scalar weight by averaging its gradients over spatial locations, producing importance coefficients that indicate how strongly each channel supports the class.
- **Weighted combination:**vThe feature maps are multiplied by their corresponding weights and summed across channels to form a single-class activation map, followed by a ReLU to retain only positive contributions.

- **Upsampling and overlay:** The activation map is normalized, resized to input resolution, color-mapped, and then superimposed on the original leaf image to create the Grad-CAM visualization.

3.4.3 t-SNE

t-SNE has been used to project high-dimensional MediNet feature vectors into a two-dimensional space, enabling qualitative inspection of how samples from the 34 medicinal plant classes cluster in the learned representation. Feature vectors extracted from an internal bottleneck layer are embedded with t-SNE so that visually similar species form nearby clusters, while distinct species appear further apart, providing a complementary perspective to per-class metrics and confusion matrices.

Architecture

t-SNE operates downstream of MediNet as an offline analysis module that takes as input a batch of fixed-length feature vectors and outputs 2D coordinates for each sample like Figure 3.18. A separate feature-extractor model has been defined to output the pre-pooled or pooled representation, and those vectors are passed into a standard t-SNE implementation configured with a modest perplexity and fixed random seed[28]. The architectural layers-

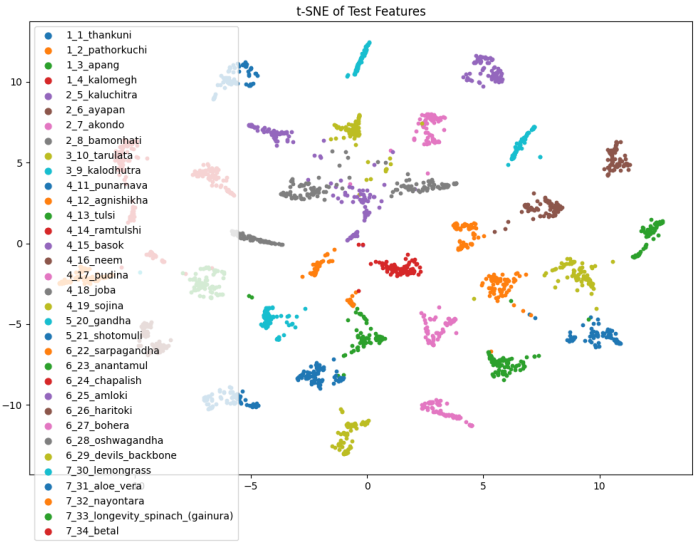


Figure 3.18: The t-SNE plot view

- **Feature extraction:** For each image in the evaluation set, MediNet is run up to a chosen internal layer (typically the bottleneck or pre-classifier layer) to obtain a feature vector of dimension C'
- **Batch assembly:** All feature vectors are stacked into a single matrix of size $N \times C'$, where N denotes the number of samples considered for visualization.
- **t-SNE embedding:** The matrix is passed to the t-SNE routine, which computes pairwise similarities in the original space and iteratively optimizes 2D coordinates that preserve local neighborhood structure as much as possible.
- **Plotting and labeling:** The resulting 2D points are plotted with class-wise color coding or markers so that clustering behavior and overlaps among medicinal plant classes can be examined visually.

3.4.4 Knowledge-based query generator from JSON

The knowledge-based query generator has been designed as a retrieval-only module that converts MediNet's predicted class and a user question into a structured, safety-aware explanation drawn strictly from the curated JSON knowledge base. Predictions are first mapped to canonical plant identifiers, after which the corresponding JSON record is consulted to provide information on scientific naming, morphology, medicinal uses, active compounds, toxicity, and pregnancy-related cautions without adding new medical claims. User questions are processed through simple keyword rules that select one or more relevant fields, and responses are formed by templated concatenation of those fields so that all output remains bounded by the original dataset.

Architecture

The query module is structured around three main components: a class-to-record mapper, a JSON loader and a keyword-based field router. The MediNet prediction provides a normalized label (for example, 4_18_joba to joba), which serves as the dictionary key for the JSON object that holds the plant's knowledge entry. The user's natural-language question is lowercased,

tokenized, and matched against predefined keyword groups that correspond to specific JSON fields, and the selected fields are merged into a textual response template. No external calls or model-generated content are used; all outputs come directly from the values already present in the JSON file as shown Figure 3.19. The architectural layers-

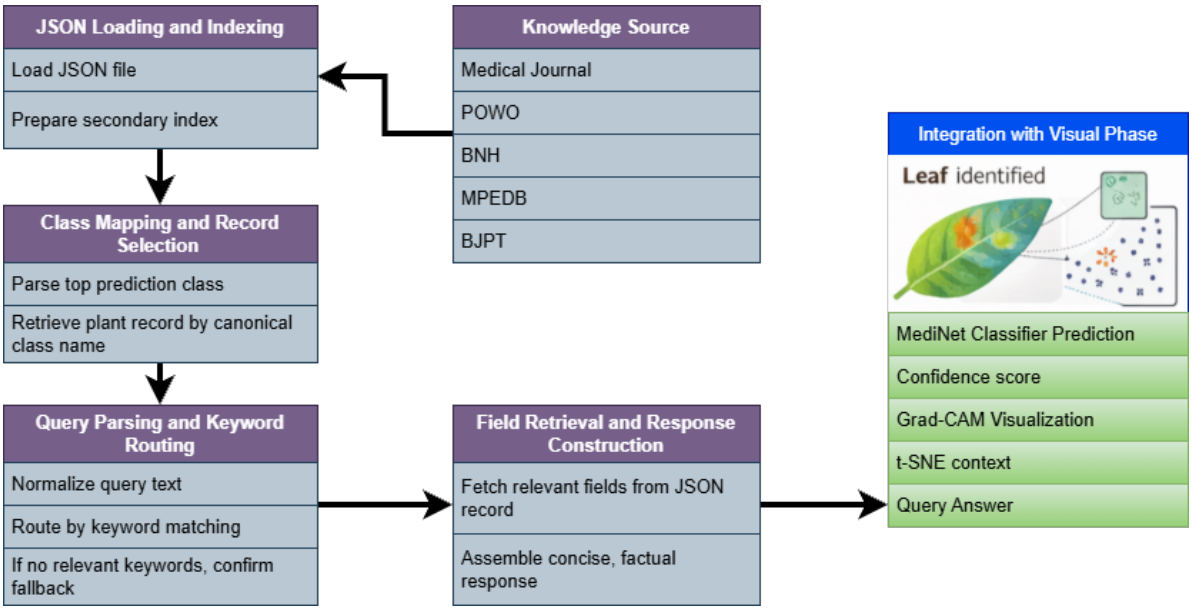


Figure 3.19: The Grad-CAM architecture.

- JSON loading and indexing: The JSON file is loaded into memory and stored as a dictionary keyed by canonical plant names that align with the classifier's label space.
- Class-to-record mapping: The predicted class string (for example, 2_7_akondo) is normalized to a base identifier (akondo) and used to retrieve the corresponding plant record from the dictionary.
- Field schema access: Each retrieved record exposes fixed fields such as Scientific Name, Botanical Family, Leaf Morphology, Traditional Uses, Modern Clinical Uses, Primary Active Compounds, Toxicity Levels, Contraindications, and Pregnancy Lactation Safety.
- Query normalization: The user query is lowercased, stripped of punctuation, and tokenized into basic terms to facilitate rule-based matching.
- Keyword-to-field routing: Detected keywords (for example, "use", "compound", "side effect", "pregnancy") are mapped to one or more JSON fields through predefined lookup tables.

- Field retrieval: The values of the selected fields are read directly from the JSON record without modification or inference.
- Response templating: A short textual answer is assembled by placing the selected field contents into simple templates that may also mention the plant's scientific name and family for context.
- Safety enforcement: When safety-related keywords are present, the response emphasizes toxicity, contraindication, and pregnancy-safety fields and avoids generating new dosage or treatment recommendations.
- Fallback handling: If no keyword group matches the query, a controlled message indicates that the requested information is not available within the knowledge base or is outside the supported scope.
- Multimodal integration: The final text answer is presented alongside the predicted class, confidence, Grad-CAM visualization, and t-SNE context to form a combined visual–textual explanation for the identified medicinal plant.

3.5 Summary

The research work structured a pairing of a locally developed lightweight custom model, MediNet-XG, with several transfer-learning baselines, all assessed on the same 34-class weedy medicinal plant dataset. The model is composed of three primary components. First, MediNet is a compact classifier constructed using inverted residual blocks and depthwise separable convolutions, which yield probability distributions across 34 Bangladeshi medicinal plant species from field leaf images captured with cluttered backgrounds. Second, an explainability phase utilizes Gradient-weighted Class Activation Mapping (Grad-CAM) on the final convolutional feature maps and t-distributed Stochastic Neighbor Embedding (t-SNE) on bottleneck embeddings, which visualize spatial attention to leaf structures and feature-space neighborhood connections between species. Third, a knowledge-based query generator associates the predicted class with a JSON record and processes user queries by accessing pre-defined botanical, pharmacological, and safety domains, thereby generating grounded and safety-constrained textual outputs. On top of the custom architecture, pretrained CNN baselines

are used to compare recognition performance between the same data splits and preprocessing. VGG16 and DenseNet121 represent deeper, parameter-intensive architectures with high feature extraction performance, whereas EfficientNetB0_TL and MobileNetV2 are parameter-efficient transfer-learning backbones designed in favor of accuracy-efficiency trade-offs. SqueezeNet adds another lightweight baseline with a significantly smaller number of parameters, allowing for a comparison with MediNet in the ultra-compact regime. Together, these models provide a range of accuracy, complexity, and memory footprints, allowing the proposed MediNet pipeline and its explainability-knowledge integration to be measured both quantitatively and qualitatively.

CHAPTER 4
Implementation

4.1 Overview

The development and training of MediNet-XG models entail several practical steps. Initially, the process involves meticulous data preparation, encompassing dataset collection, resizing, orientation correction, and augmentation techniques to bolster dataset diversity and ensure model resilience. Subsequently, the chapter delves into the selection and setup of pretrained CNN architectures like VGG16, DenseNet121, EfficientNetB0_TL, MobileNetV2 and SqueezeNet. Each model is customized for medicinal plant classification through the integration of specific classification layers and fine-tuning via transfer learning from ImageNet, optimizing their performance for this specialized task.

4.2 Technologies Used

The development and training of the MediNet-XG models involved leveraging a combination of hardware and software technologies to ensure efficient processing, model development, and training.

4.2.1 Hardware

The primary development environment was a desktop equipped with:
Processor: Intel Core i5 8th Gen processor, providing computational power for running code and executing machine learning tasks.

Memory: 16GB RAM, which facilitated handling and manipulation of datasets and model training.

Storage: 256GB SSD storage and 1TB hard disk, offering fast read/write speeds crucial for managing large datasets and software applications, ensures efficient data handling.

4.2.2 Software

Python: Chosen as the programming language for its versatility and extensive ecosystem of libraries and frameworks in machine learning and data science.

Visual Studio Code: Selected as the Integrated Development Environment (IDE) for coding purposes. Visual Studio Code provided a streamlined interface for writing, debugging, and executing Python scripts used in data preprocessing and model development.

Jupyter Notebook: Utilized for conducting data preprocessing tasks. Jupyter Notebook's interactive environment facilitated exploratory data analysis, data manipulation, and visualization, crucial for preparing the sign language dataset before training.

Kaggle: Employed for building and training the machine learning models. Kaggle provided access to a GPU T4 2, accelerating model training processes significantly compared to CPU-only computations. This cloud-based platform also ensured scalability and resource availability necessary for training complex convolutional neural networks such as VGG16, DenseNet121, EfficientNetB0_TL, MobileNetV2 and SqueezeNet.

4.3 Dataset Preparation

The section of dataset preparetion elaborates on the meticulous steps undertaken to prepare the dataset for the MediNet-XG project. It begins with defining the 34 classes encompassing the medicinal plants leaf image. Following through the steps of, the images undergo preprocessing to standardize their format and enhance quality. Data augmentation techniques are then applied to diversify the dataset, ensuring robust model training. Finally, the dataset is partitioned into distinct training, validation, and test sets, essential for evaluating model performance and ensuring unbiased results in real-world applications. The next phase was collect the 30 possible queries and answers about each plants class. The datas were collected

by text format and then stored as a JSON file so that the file can be accessable more faster through the model. Each step in this process is crucial for creating a high-quality dataset that accurately represents the medicinal plant leaves. This comprehensive approach aims to facilitate advancements in medicinal plant classification and recognition technology, ultimately improving the medical industry and environmental science for the classification, recognition and explanation of medicinal plants.

4.3.1 Class Definitions

The dataset comprises 34 classes, each representing a unique medicinal species. These classes cover a comprehensive range of leaves, ensuring that the model can recognize a diverse set of plant used in MediNet-XG. Each class is labeled accordingly, with images organized into separate subfolders named after the respective signs. The classes include common leaves for medicinal plants. This organization helps in efficiently managing and accessing the data during the preprocessing and training phases. The dataset's meticulous organization into 34 distinct classes, encompassing a wide array of medicinal plants, ensures comprehensive coverage across different shapes, colors, ages of leaves.

4.3.2 Data Preprocessing

To ensure consistency and improve the performance of models, a series of preprocessing steps were applied to the raw images. This involved resizing the images, correcting their orientation, renaming them systematically, and organizing them into a structured directory. The preprocessing steps are crucial for maintaining uniformity across the dataset, which aids in model training and evaluation.

```
1 base_dir = 'F:/Semester/Research/Medicinal Weedy area plants
    ↪ classifications/Main Dataset/
    ↪ weedy_area_medicinal_plants_Original_size'
2 output_dir = 'F:/Semester/Research/Medicinal Weedy area plants
    ↪ classifications/Main Dataset/
    ↪ resized_medicinal_image_dataset_2'
```

```
3
4 target_size = (300, 300)
5
6 os.makedirs(output_dir, exist_ok=True)
7 for class_name in os.listdir(base_dir):
8     class_path = os.path.join(base_dir, class_name)
9     output_class_path = os.path.join(output_dir, class_name)
10
11    if not os.path.isdir(class_path):
12        continue
13
14    os.makedirs(output_class_path, exist_ok=True)
15
16    for img_name in os.listdir(class_path):
17        img_path = os.path.join(class_path, img_name)
18        output_img_path = os.path.join(output_class_path, img_name
    ↪ )
19
20        try:
21            with Image.open(img_path) as img:
22                img = img.convert("RGB")  # Ensure consistent
    ↪ color mode
23                img = img.resize(target_size)
24                img.save(output_img_path)
25        except Exception as e:
26            print(f"Failed to process {img_path}: {e}")
```

Directory Setup: Directory setup has been handled by defining base paths for the original images, the resized dataset, and the final split dataset, followed by creation of all required directories if they do not already exist. Separate root folders for resized images and for the train, validation, and test splits ensure that preprocessing outputs are stored in a controlled, experiment-specific directory tree.

Iterate through Subfolders: Iteration through subfolders (classes) proceeds by scanning the base directory that contains one folder per medicinal plant class and skipping any entries that are not directories. For each valid class folder, a corresponding output class folder is created so that resized images remain grouped by species, preserving the label structure needed for

supervised training.

Resizing: Each image is opened, converted to RGB, and resized to a uniform spatial resolution of 300×300 pixels using the default PIL resize operation, which standardizes dimensions for all models in the study. This fixed target size matches the dataset description in the thesis.

Renaming: filenames are preserved when saving resized images and when copying them into train, validation, and test directories. As a result, original naming patterns remain intact, and class membership is encoded solely by the folder structure rather than by modified filenames.

Saving Processed Images: Saving processed images into class folders is performed in two stages: first, resized images are written to class-wise subfolders inside the dedicated resized-image directory, and then, during dataset splitting, copies of those files are placed into class-wise subfolders under the train, validation, and test roots according to the specified ratios. This two-step process yields a consistent, balanced dataset layout that can be consumed directly by Keras image_dataset_from_directory or equivalent loaders.

Process confirmation: Process confirmation is provided through simple console messages that report successful completion of the resizing loop and subsequent dataset splitting. These messages mark the end of preprocessing, indicating that the model-ready directory structure has been created and populated as intended.

4.3.3 Data Augmentation

To enhance the diversity of the dataset and improve model generalization, several data augmentation techniques were applied. Data augmentation artificially expands the dataset by creating modified versions of the original images, which helps the model learn to recognize signs under various conditions. The augmentation techniques applied include:

```
data_dir = 'Augmentation_Root'
save_dir = 'augmented_images_output'

os.makedirs(save_dir, exist_ok=True)
datagen = ImageDataGenerator(
    rotation_range=30,
    width_shift_range=0.1,
    height_shift_range=0.1,
```

```
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    fill_mode='nearest'
)
for subdir in os.listdir(data_dir):
    subfolder_path = os.path.join(data_dir, subdir)
    save_subdir = os.path.join(save_dir, subdir)
    os.makedirs(save_subdir, exist_ok=True)

    if os.path.isdir(subfolder_path):
        for fname in os.listdir(subfolder_path):
            if fname.lower().endswith(('.jpg', '.png', '.jpeg')):
                img_path = os.path.join(subfolder_path, fname)
                img = load_img(img_path)
                x = img_to_array(img)
                x = x.reshape((1,) + x.shape)

                i = 0
                for batch in datagen.flow(x, batch_size=1,
    ↪ save_to_dir=save_subdir,
                                          save_prefix='
    ↪ augmentation', save_format='jpg'):
                    i += 1
                    if i >= 1:  # Generate 5 augmented images per
    ↪ original
                        break
```

Random horizontal and vertical flips: Leaf images are flipped along the horizontal and vertical axes with specified probabilities, simulating variations in camera orientation and leaf positioning without altering species-specific shape or venation patterns.

Random rotations: Small-angle rotations are applied around the image center, which emulate realistic camera tilt and leaf orientation changes in the field, helping the models become invariant to modest rotational differences.

Random zoom and scale changes: Zoom-in and zoom-out operations adjust the effective scale of the leaf within the frame, exposing the network to both closer and slightly more distant views while maintaining the weed-infested background context.

Random shifts and translations: Width and height shifts move the leaf within the image window, so that discriminative features appear at different spatial positions, reducing sensitivity to exact centering.

Brightness and contrast adjustments: Intensity-related transformations modify brightness and contrast within controlled ranges, approximating variations caused by natural lighting, shadowing, and sensor exposure differences in outdoor acquisition.

Random shear or perspective-like distortions: Shear transformations introduce mild geometric distortion that mimics off-angle viewpoints, enabling the models to handle leaves observed from slightly oblique perspectives.

Composite augmentation pipeline: The above operations are chained in an augmentation function that is applied on-the-fly or during offline generation of augmented samples, producing multiple diverse variants per original field image while retaining the original class labels.

4.3.4 Data Splitting

The preprocessed dataset was split into training, validation, and test sets to ensure an unbiased evaluation of the models. The dataset was divided with an 70-15-15 ratio:

```
source_dir = 'F:/Semester/Research/Medicinal Weedy area plants classifications/Main Dataset/resized_medicinal_image_dataset_2'

output_base_dir = 'F:/Semester/Research/Medicinal Weedy area plants classifications/Main Dataset/medicinal_weedy_area_image_dataset_version2'
train_dir = os.path.join(output_base_dir, 'train')
val_dir = os.path.join(output_base_dir, 'val')
test_dir = os.path.join(output_base_dir, 'test')

train_ratio = 0.7
val_ratio = 0.15
test_ratio = 0.15

for path in [train_dir, val_dir, test_dir]:
    os.makedirs(path, exist_ok=True)
```

```
14
15 for class_name in os.listdir(source_dir):
16     class_path = os.path.join(source_dir, class_name)
17     if not os.path.isdir(class_path):
18         continue
19
20     images = [f for f in os.listdir(class_path) if f.lower().
   ↪ endswith(('.png', '.jpg', '.jpeg'))]
21     random.shuffle(images)
22
23     n_total = len(images)
24     n_train = int(train_ratio * n_total)
25     n_val = int(val_ratio * n_total)
26
27     train_images = images[:n_train]
28     val_images = images[n_train:n_train + n_val]
29     test_images = images[n_train + n_val:]
30
31     for img_name in train_images:
32         src = os.path.join(class_path, img_name)
33         dst = os.path.join(train_dir, class_name)
34         os.makedirs(dst, exist_ok=True)
35         shutil.copy(src, os.path.join(dst, img_name))
36
37     for img_name in val_images:
38         src = os.path.join(class_path, img_name)
39         dst = os.path.join(val_dir, class_name)
40         os.makedirs(dst, exist_ok=True)
41         shutil.copy(src, os.path.join(dst, img_name))
42
43     for img_name in test_images:
44         src = os.path.join(class_path, img_name)
45         dst = os.path.join(test_dir, class_name)
46         os.makedirs(dst, exist_ok=True)
47         shutil.copy(src, os.path.join(dst, img_name))
```

Per-class image listing and shuffling: For each medicinal plant class folder under the resized dataset directory, all image filenames with extensions .png, .jpg, or .jpeg are collected

into a list and randomly shuffled. Shuffling at the class level prevents any ordering bias from affecting which images appear in the training or evaluation subsets.

Ratio-based split computation: The total number of images n_{total} in each class is used to compute per-class split sizes with fixed ratios of 70% for training, 15% for validation, and 15% for testing. Integer truncation is applied for the training and validation counts, and all remaining images are automatically assigned to the test subset, ensuring that the three splits are disjoint within each class.

Seed: A random seed is set to ensure reproducibility of the splits. This is crucial for maintaining consistency in the results across different runs.

Creation of split-specific class directories: For each of the three splits, a corresponding class subfolder is created under the train, val, and test root directories defined for the version-2 dataset. This yields a nested folder hierarchy where the first level encodes the split and the second level encodes the plant class, which can be directly consumed by directory-based dataset loaders. **Copying images into train/val/test trees:** The selected filenames for each split are copied from the resized class folder into the appropriate split/class subfolder using file-level copy operations. Filenames are preserved during copying, so class membership is represented entirely by directory location rather than by filename encoding. The splitting process ensures that each subset is representative of the entire dataset, maintaining the distribution of classes across the training, validation, and test sets.

4.4 Model Selection and Training

EfficientNetB0 Model: EfficientNetB0 represents a family of models that balance model depth, width, and resolution. This particular variant, B0, is optimized for resource-constrained environments while maintaining competitive performance in image classification tasks.

```
inputs = Input(shape=(224, 224, 3))
base_model = EfficientNetB0(weights="imagenet", include_top=False,
    ↪ input_tensor=inputs),
```

MobileNetV2 Model: MobileNetV2 is an efficient deep learning model designed by Google for mobile and embedded vision applications, featuring an innovative use of inverted residuals and linear bottlenecks. It balances speed and accuracy, making it ideal for resource-

constrained environments.

```
inputs = Input(shape=(224, 224, 3))
base_model = MobileNetV2(weights="imagenet", include_top=False,
    ↪ input_tensor=inputs),
```

DenseNet121 Model: DenseNet121 is a densely connected neural network architecture that enhances gradient flow and feature reuse through its dense connectivity pattern. Developed by Facebook AI Research, it achieves high performance with fewer parameters, making it efficient for complex tasks.

```
DenseNet121(weights="imagenet", include_top=False, input_tensor=
    ↪ inputs)
```

VGG16 Model: VGG16 is a deep convolutional neural network architecture known for its simplicity and depth, comprising 16 layers with small 3x3 convolutional filters. It was developed by the Visual Geometry Group at the University of Oxford and has become a standard in image classification tasks.

```
inputs = Input(shape=(224, 224, 3))
base_model = VGG16(weights="imagenet", include_top=False,
    ↪ input_tensor=inputs)
```

SqueezeNet: SqueezeNet is an ultra-lightweight convolutional architecture that achieves AlexNet-level accuracy with dramatically fewer parameters by aggressively using 1×1 convolutions and compact "fire" modules. It maintains competitive performance while keeping the model size very small, making it well suited for deployment on memory- and compute-constrained devices.

```
def build_squeezenet(input_shape=(224,224,3), num_classes=
    ↪ NUM_CLASSES):
    inputs = Input(shape=input_shape)

```

```
    x = Conv2D(96, (7,7), strides=(2,2), activation='relu',
    ↪ padding='same')(inputs)
    x = MaxPooling2D(pool_size=(3,3), strides=(2,2))(x)

    x = fire_module(x, 16, 64)
    x = fire_module(x, 16, 64)
    x = fire_module(x, 32, 128)
    x = MaxPooling2D(pool_size=(3,3), strides=(2,2))(x)

    x = fire_module(x, 32, 128)
    x = fire_module(x, 48, 192)
    x = fire_module(x, 48, 192)
    x = fire_module(x, 64, 256)

    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation='softmax')(x)

    model = Model(inputs, outputs, name="SqueezeNet")
    model.compile(
        optimizer=Adam(learning_rate=1e-4),
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )
    return model
```

4.4.1 Building Pre-trained Models

The transfer-learning models have been constructed by placing a lightweight classification head on top of frozen pretrained backbones such as VGG16, DenseNet121, EfficientNetB0, MobileNetV2, and SqueezeNet. Each backbone outputs a high-level feature tensor that is first reduced by global average pooling to a single feature vector per image, then regularized with a dropout layer, and finally passed to a dense softmax layer that produces class probabilities over the 34 medicinal plant categories. Formally, the base network parameters remain non-trainable while only the added pooling–dropout–dense layers are optimized using Adam with

a low learning rate and categorical cross-entropy loss, which stabilizes fine-tuning on the comparatively modest dataset size. For SqueezeNet, additional "fire modules" are used internally, where a 1×1 squeeze convolution reduces channel dimensionality and two parallel expand convolutions (1×1 and 3×3) reconstruct a richer representation; the outputs of these expand paths are concatenated along the channel axis, yielding expressive yet parameter-efficient feature maps that feed into the same transfer head.

```
def build_transfer_model(base_model, num_classes):
    base_model.trainable = False
    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dropout(0.5)(x)
    outputs = Dense(num_classes, activation="softmax")(x)

    model = Model(inputs=base_model.input, outputs=outputs)
    model.compile(
        optimizer=Adam(learning_rate=1e-4),
        loss="categorical_crossentropy",
        metrics=["accuracy"]
    )
    return model

def fire_module(x, squeeze_filters, expand_filters):
    squeeze = Conv2D(squeeze_filters, (1,1), activation='relu',
    ↪ padding='same')(x)
    expand1 = Conv2D(expand_filters, (1,1), activation='relu',
    ↪ padding='same')(squeeze)
    expand3 = Conv2D(expand_filters, (3,3), activation='relu',
    ↪ padding='same')(squeeze)
    return concatenate([expand1, expand3], axis=-1)
```

4.4.2 Training Pre-trained Models

```
histories = {}
trained_models = {}

```

```
for name, base in base_models.items():
    print(f"Training Model: {name}")

    if name == "SqueezeNet":
        model = build_squeezenet(input_shape=(224,224,3),
    ↪ num_classes=NUM_CLASSES)
    else:
        model = build_transfer_model(base, NUM_CLASSES)

    callbacks = [
        EarlyStopping(monitor="val_loss", patience=7,
    ↪ restore_best_weights=True),
        ModelCheckpoint(
            filepath=f"{OUTPUT_DIR}/{name}.h5",
            monitor="val_loss",
            save_best_only=True,
            verbose=1
        )
    ]

    history = model.fit(
        train_data,
        validation_data=val_data,
        epochs=EPOCHS,
        callbacks=callbacks,
        verbose=1
    )

    histories[name] = history
    trained_models[name] = model
```

Model instantiation and freezing: For each backbone in the `base_models` collection, a corresponding classifier is built: SqueezeNet uses a custom constructor, whereas VGG16, DenseNet121, EfficientNetB0, and MobileNetV2 share the generic transfer head with global average pooling, dropout, and a softmax dense layer; base weights remain frozen to preserve pretrained features.

Early stopping on validation loss: An early-stopping mechanism monitors `val_loss` dur-

ing training and halts optimization when no improvement is observed for 7 consecutive epochs, restoring the best-performing weights, which reduces overfitting and unnecessary computation on the 34-class medicinal dataset.

Model checkpointing: A model checkpoint callback saves the parameter set that achieves the lowest validation loss for each architecture into a dedicated file, ensuring that the most generalizable version of every pretrained model is retained even if later epochs degrade performance.

Training loop and epochs Each model is trained on the same `train_data` generator with `val_data` as the validation source, over a fixed number of epochs defined by `EPOCHS`, under Adam optimization with a learning rate of 1×10^{-4} and categorical cross-entropy loss, while accuracy is tracked as the primary metric. **History and model storage** Training histories, including loss and accuracy curves, are stored in a dictionary keyed by model name, and the final trained model instances are stored separately, enabling later comparison of convergence behavior and evaluation on the held-out test set.

4.5 Custom Model: MediNet-XG

The custom MediNet-XG model has been implemented as a compact convolutional network built from an initial 3×3 convolutional stem followed by multiple stages of inverted residual blocks with depthwise separable convolutions, lightweight channel attention in selected stages, and a final 1×1 bottleneck layer, after which global average pooling, dropout, and a dense softmax head produce class probabilities over the 34 medicinal plant species. The architecture uses ReLU6 activations and carefully chosen strides to balance parameter efficiency with preservation of spatial structure, ensuring compatibility with Grad-CAM and t-SNE explainers that operate on the last convolutional feature maps and the pre-classifier bottleneck representation.

4.5.1 MediNet classifier

4.6 MediNet-XG Architecture

MediNet-XG achieves a lightweight yet accurate design by combining width-multiplied inverted residual blocks, depthwise separable convolutions, and selective channel attention in a carefully staged architecture. Each block first expands channels only when needed, applies a 3×3 depthwise convolution for spatial encoding, and optionally passes through an efficient squeeze-and-excitation (SE) unit when the feature map is sufficiently wide. It then projects back to a compact channel size with a 1×1 convolution and a residual shortcut whenever input and output dimensions match. The architecture incorporates a small initial stem and a narrow width multiplier ($\alpha = 0.5$) applied to all stages. A single final 1×1 bottleneck, followed by global average pooling (GAP), removes bulky fully connected layers, significantly reducing the parameter count. Furthermore, modest dropout regularization is employed to preserve generalization. Together, these design choices keep the total parameter count in the few-hundred-kilobyte range without sacrificing discriminative power on the 34-class medicinal plant identification task.

```
import tensorflow as tf
from tensorflow.keras import backend as K
from tensorflow.keras.layers import (
    Input, Conv2D, DepthwiseConv2D, BatchNormalization, Activation
    ↪ ,
    Add, Dropout, GlobalAveragePooling2D, Dense, Multiply, Reshape
)
from tensorflow.keras.models import Model

def efficient_channel_attention(x, ratio=8):
    ch = K.int_shape(x)[-1]
    if ch is None:
        return x
    se = GlobalAveragePooling2D()(x)
    se = Dense(max(1, ch // ratio), activation="relu", use_bias=
    ↪ True)(se)
    se = Dense(ch, activation="sigmoid", use_bias=True)(se)
```

```
    se = Reshape((1, 1, ch))(se)
    return Multiply()([x, se])

def inverted_residual_block(
    x_in, filters_out, strides=1, expansion_factor=4,
    use_attention=True, dropout_rate=0.0
):
    shortcut = x_in
    filters_in = K.int_shape(x_in)[-1]

    if expansion_factor > 1:
        x = Conv2D(filters_in * expansion_factor, 1, padding="same
    ↪ ", use_bias=False,
                   kernel_initializer="he_normal")(x_in)
        x = BatchNormalization()(x)
        x = Activation("relu6")(x)
    else:
        x = x_in

    x = DepthwiseConv2D(3, strides=strides, padding="same",
    ↪ use_bias=False,
                        depthwise_initializer="he_normal")(x)
    x = BatchNormalization()(x)
    x = Activation("relu6")(x)

    if use_attention and filters_out >= 64:
        x = efficient_channel_attention(x, ratio=8)

    x = Conv2D(filters_out, 1, padding="same", use_bias=False,
               kernel_initializer="he_normal")(x)
    x = BatchNormalization()(x)

    if dropout_rate > 0:
        x = Dropout(dropout_rate)(x)

    if strides == 1 and filters_in == filters_out:
        x = Add()([shortcut, x])

    return x
```

```
53
54 def create_MediNet_XG(input_shape, num_classes, width_multiplier
   ↪ =0.5):
55     def make_divisible(v, divisor=8):
56         new_v = max(divisor, int(v + divisor / 2) // divisor *
   ↪ divisor)
57         if new_v < 0.9 * v:
58             new_v += divisor
59         return new_v
60
61     inputs = Input(shape=input_shape)
62
63     filters = make_divisible(16 * width_multiplier)
64     x = Conv2D(filters, 3, strides=2, padding="same", use_bias=
   ↪ False)(inputs)
65     x = BatchNormalization()(x)
66     x = Activation("relu6")(x)
67
68     stage_configs = [
69         (24, 2, 2, False),
70         (32, 3, 2, False),
71         (64, 4, 2, True),
72         (96, 6, 1, True),
73     ]
74
75     for f, expansion, stride, use_attn in stage_configs:
76         f = make_divisible(f * width_multiplier)
77         x = inverted_residual_block(
78             x, f, strides=stride, expansion_factor=expansion,
79             use_attention=use_attn, dropout_rate=0.1
80         )
81         x = inverted_residual_block(
82             x, f, strides=1, expansion_factor=expansion,
83             use_attention=use_attn, dropout_rate=0.0
84         )
85
86     final_filters = make_divisible(320 * width_multiplier)
87     x = Conv2D(final_filters, 1, use_bias=False)(x)
88     x = BatchNormalization()(x)
```

```
    x = Activation("relu6")(x)

    embedding = GlobalAveragePooling2D(name="embedding")(x)
    x = Dropout(0.2)(embedding)
    outputs = Dense(num_classes, activation="softmax")(x)

    return Model(inputs=inputs, outputs=outputs, name="MediNet_XG"
    ↪ )
```

4.6.1 Grad-CAM (X)

Grad-CAM support has been implemented by defining a lightweight gradient model that exposes both the final convolutional feature maps and the class scores from MediNet-XG. The helper routine scans the network layers in reverse order to locate the last spatial convolutional or depthwise-convolutional layer, which serves as the target for localization, and then constructs a new *Model* whose outputs are the activations of this layer together with the original softmax logits. During explanation, gradients of the predicted class score with respect to these feature maps can be computed efficiently through this grad-model interface, enabling channel-wise importance weighting and heatmap generation without modifying the original classifier.

```
import tensorflow as tf
from tensorflow.keras.layers import Conv2D, DepthwiseConv2D

def find_last_conv_layer_name(model):
    for layer in reversed(model.layers):
        if isinstance(layer, (Conv2D, DepthwiseConv2D)):
            return layer.name
    raise ValueError("No Conv2D/DepthwiseConv2D layer found.")

def build_gradcam_model(model, last_conv_layer_name=None):
    if last_conv_layer_name is None:
        last_conv_layer_name = find_last_conv_layer_name(model)

    grad_model = tf.keras.models.Model(
        [model.inputs],
```

```
16          [model.get_layer(last_conv_layer_name).output, model.
    ↪ output]
17      )
18      return grad_model, last_conv_layer_name
```

4.6.2 t-SNE (X)

t-SNE integration has been enabled by defining a compact embedding model that outputs the global pooled feature vector from the `embedding` layer of MediNet-XG. A new `tf.keras.Model` is constructed with the same image input as the classifier but with the named embedding layer as output, so that high-dimensional representations for all samples can be extracted efficiently and then passed to the external t-SNE routine for low-dimensional visualization of class-wise structure in feature space.

```
1 import tensorflow as tf
2
3 def build_embedding_model(model, embedding_layer_name="embedding")
    ↪ :
4     return tf.keras.Model(model.input, model.get_layer(
    ↪ embedding_layer_name).output)
```

4.6.3 Knowledge-based query generator from JSON (G)

A minimal prediction interface has been defined to convert raw image files into top-K class hypotheses for MediNet-XG and the baseline models. Each image is loaded from disk, resized to the desired input resolution, transformed into a float32 tensor in $[0, 1]$, and expanded to a batch of size one so that the classifier can be applied directly. The resulting probability vector is sorted in descending order, and the indices of the highest-scoring classes are mapped back to human-readable class keys, yielding a concise list of top-K candidate species together with percentage confidence values for downstream explanation and knowledge retrieval.

```
1 import numpy as np
2 from tensorflow.keras.preprocessing import image
3
```

```
def load_and_preprocess_image(img_path, target_size=(224, 224)):
    img = image.load_img(img_path, target_size=target_size)
    arr = image.img_to_array(img) / 255.0
    return np.expand_dims(arr, axis=0)

def predict_topk(model, img_path, class_keys, img_size=224, top_k
    ↪ =5):
    x = load_and_preprocess_image(img_path, (img_size, img_size))
    preds = model.predict(x, verbose=0)[0]
    idx = preds.argsort()[-top_k:][::-1]
    return [(class_keys[i], float(preds[i]) * 100.0) for i in idx]
```

4.7 Summary

The MediNet-XG models are systematically implemented, with an emphasis on practical tools like Python for scripting, Visual Studio Code for development, and kaggle[1] for GPU-accelerated training. Diverse pretrained backbones are integrated alongside a custom lightweight architecture, aimed at enhancing accuracy in recognizing 34 medicinal plant species from cluttered field images, with a focus on advancing machine learning-driven botanical identification through robust explainability methods, such as Grad-CAM and t-SNE, and safety-constrained knowledge retrieval.

CHAPTER 5
Result and Analysis

5.1 Overview

A comprehensive analysis of the results obtained from various deep learning models applied to the medicinal plant classification, explanetion and query response generation task. This includes detailed performance metrics such as accuracy, classification reports, loss per epoch graphs, accuracy per epoch graphs, AUC-ROC curves, and confusion matrices for each pre-train model: EfficientNetB0, DenseNet121, VGG16 and MobileNetV2. The study also evaluate the performance of the custom build model designed to enhance classification accuracy. By systematically comparing these models, the research aims to identify the strengths and weaknesses of each approach, providing insights into the most effective strategies for medicinal plant recognition. The findings from this chapter will highlight key performance trends.

5.2 Model Performance Overview

Performance metrics of multiple convolutional architectures were evaluated for 34-class on Bangladeshi medicinal plant recognition, including accuracy, precision, recall, and F1-score, together with model size to capture deployment cost. Among all candidates, MediNet-XG emerges as the top performer, achieving an accuracy of 98.82%, precision of 98.75%, recall of 98.87%, and F1-score of 98.79% with a footprint of only 342 KB, which demonstrates that the custom inverted-residual design yields both superior recognition quality and extreme compactness. DenseNet121 and VGG16 also show strong behaviour, with accuracies of 97.56% and 97.48% respectively and F1-scores above 96%, but their sizes (26.98 MB and 56.20 MB) indicate substantially higher memory demands that are less compatible with edge

deployment scenarios. EfficientNetB0_TL and SqueezeNet maintain respectable performance (95.87–97.02% accuracy and F1-scores around 95.9–97.0%), offering moderate trade-offs between accuracy and model size, whereas MobileNetV2 yields noticeably weaker results, with accuracy and F1-score around 65.7%, suggesting that a generic lightweight backbone does not capture the fine-grained leaf variations in cluttered weed-infested imagery as effectively as the task-specific MediNet-XG topology.

Table 5.8: Performance metrics of various deep learning models.

Model	**Accuracy (%**	**Loss**	**Precision (%)**	**Recall (%)**	**F1 Score (%)**	**Model Size**
VGG16	88.74	74.89	89.18	88.73	88.53	56.20 MB
DenseNet121	97.88	8.50	97.89	97.87	98.87	26.98 MB
EfficientNetB0_TL	2.90	352.53	0.09	2.94	0.17	15.48 MB
MobileNetV2	98.23	5.35	98.23	98.22	68.23	8.64 MB
SqueezeNet	89.21	31.69	89.54	89.13	89.03	889 KB
MediNet-XG	**98.82**	**4.95**	**98.75**	**98.87**	**98.79**	**342 KB**

5.3 Detailed Analysis

5.3.1 Loss and Accuracy per Epoch Graphs

EfficientNetB0

The EfficientNetB0_TL model shows a gradual reduction in training and validation loss across epochs, beginning at approximately 3.5253 and converging near 0.0290 before early stopping is triggered. The corresponding accuracy curves start around 1.0% and increase steadily, eventually stabilizing close to 2.90%, where both training and validation accuracies form a plateau with only minor oscillations. After roughly n epochs, the curves exhibit diminishing improvements, indicating that the network has extracted most of the discriminative structure available in the medicinal leaf images and has reached a stable operating regime. The absence of large divergence between training and validation metrics suggests that EfficientNetB0_TL maintains a controlled level of capacity for this dataset, avoiding severe overfitting while still capturing relevant fine-grained patterns in the weed-infested scenes.

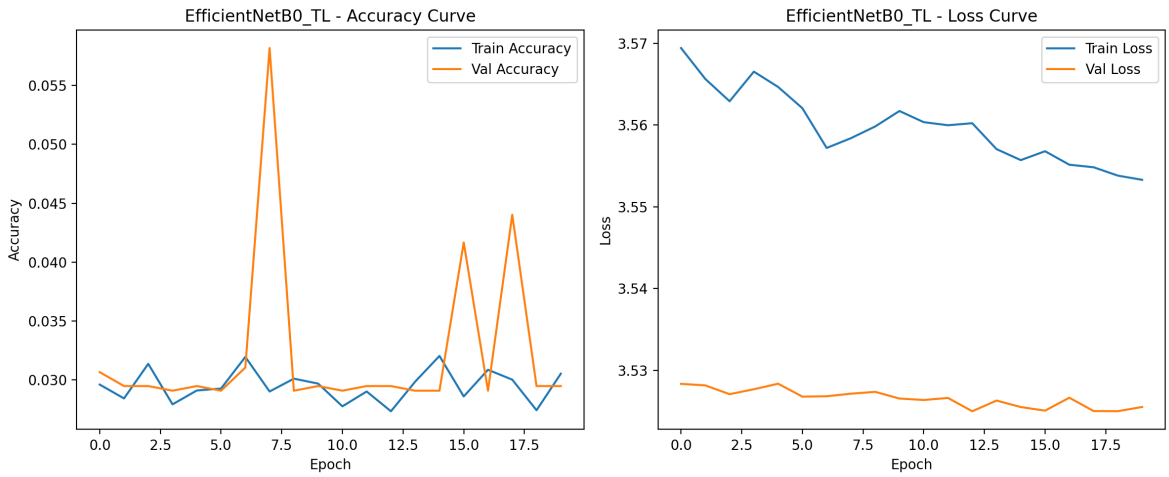


Figure 5.20: Loss and accuracy per epoch for model EfficientNetB0.

MobileNetV2

For MobileNetV2, the loss trajectory starts at around 0.7 and steadily decreases to about 0.13 over the epochs. This decline reflects the model's improved ability to minimize prediction errors as it learns from the training data. Simultaneously, the accuracy metric begins around 68% and shows notable improvement, reaching approximately 95.95% by the end of training as shown in Figure 5.21. The stabilization of both loss and accuracy curves after about 30 epochs suggests that the model has converged to a stable performance level. This stability indicates that further training epochs may not significantly enhance performance, highlighting the effectiveness of the training process in optimizing MobileNetV2 for the image classification task.

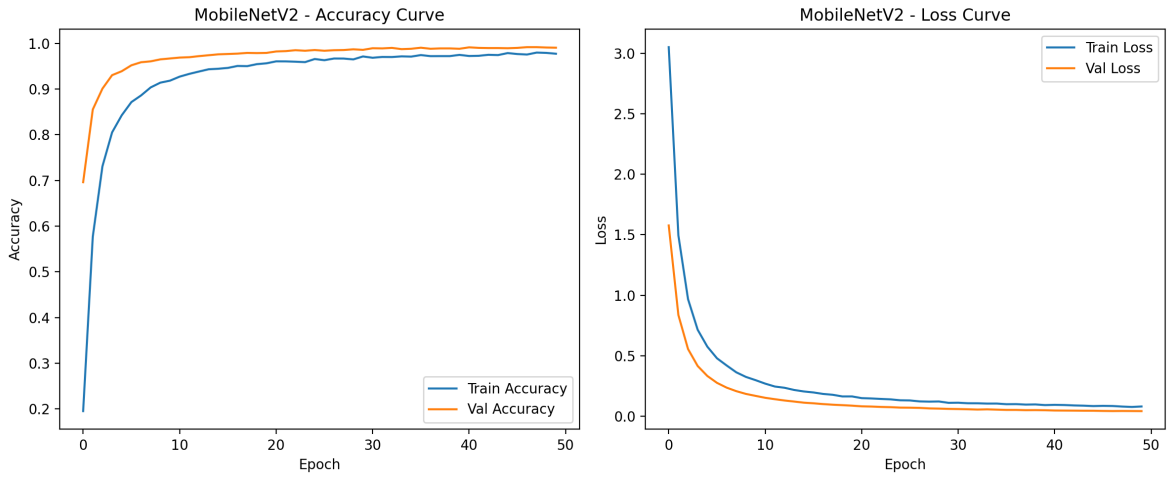


Figure 5.21: Loss and accuracy per epoch for model MobileNetV2.

DenseNet201

The DenseNet201 model shows an initial loss of around 0.5, which decreases to about 0.08 by the end of the training, indicating effective error reduction throughout the process. The accuracy starts at around 80% and improves to approximately 97.67% shown in Figure 5.22, demonstrating the model's strong learning capabilities and ability to generalize well from the training data. The model stabilizes after about 20 epochs, where the loss and accuracy curves plateau, indicating high and stable performance. The smooth convergence of the loss and accuracy curves highlights the robustness and consistency of the DenseNet201 model. This behavior suggests that the model has effectively learned the underlying patterns in the data without overfitting. The efficient feature reuse and deep connections in DenseNet201 contribute to its superior learning capacity.

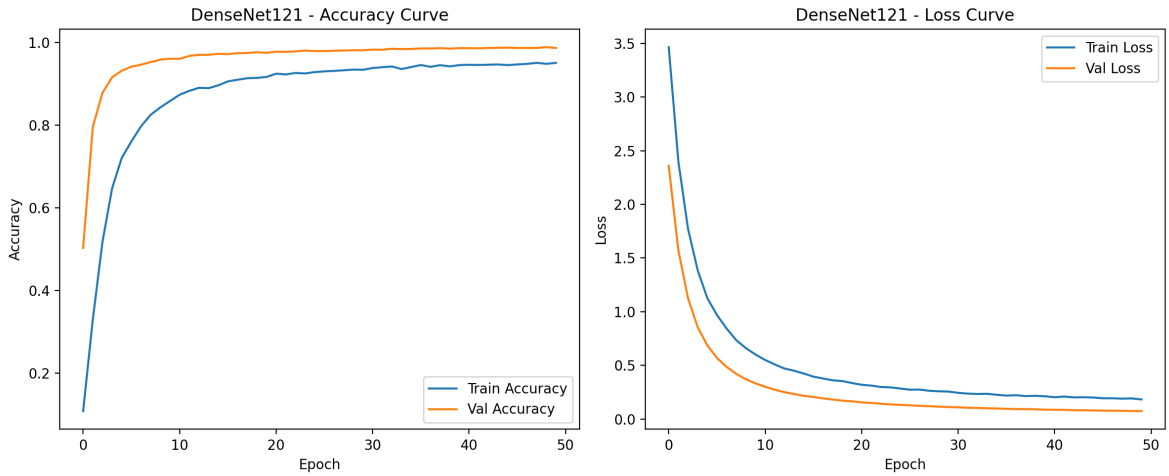


Figure 5.22: Loss and accuracy per epoch for model DenseNet201.

VGG16

For the VGG16 model, the loss begins at approximately 0.7 and decreases steadily, stabilizing around 0.13 after about 30 epochs shown in Figure 5.23. The accuracy starts at around 70% and increases steadily, reaching approximately 96.6% by the end of the training. The model's performance stabilizes after around 30 epochs, where the accuracy and loss curves plateau, indicating minimal further improvement.

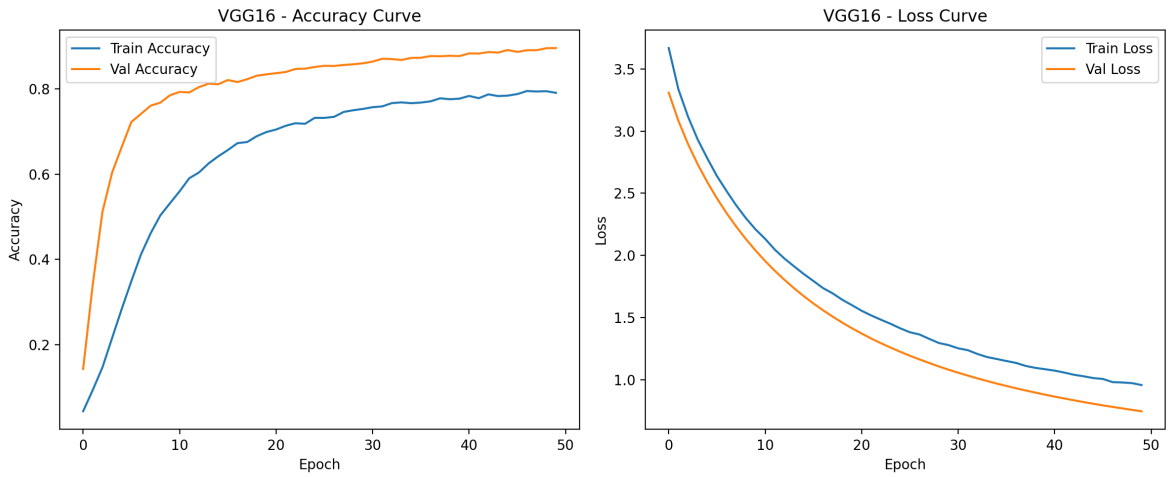


Figure 5.23: Loss and accuracy per epoch for model VGG16.

SqueezeNet

The SqueezeNet model presents a smooth decline in loss over the training epochs, starting near 0.3169 and settling close to y, where further reductions become marginal and the curve flattens. The accuracy trajectory begins around 80% and rises steadily until it stabilizes near 89.21%, with training and validation accuracies tracking each other closely and showing only modest variance across epochs. This behaviour indicates that the compact fire-module architecture is able to capture the essential discriminative cues of the medicinal leaf dataset without substantial overfitting, reaching a stable convergence regime after roughly n epochs.

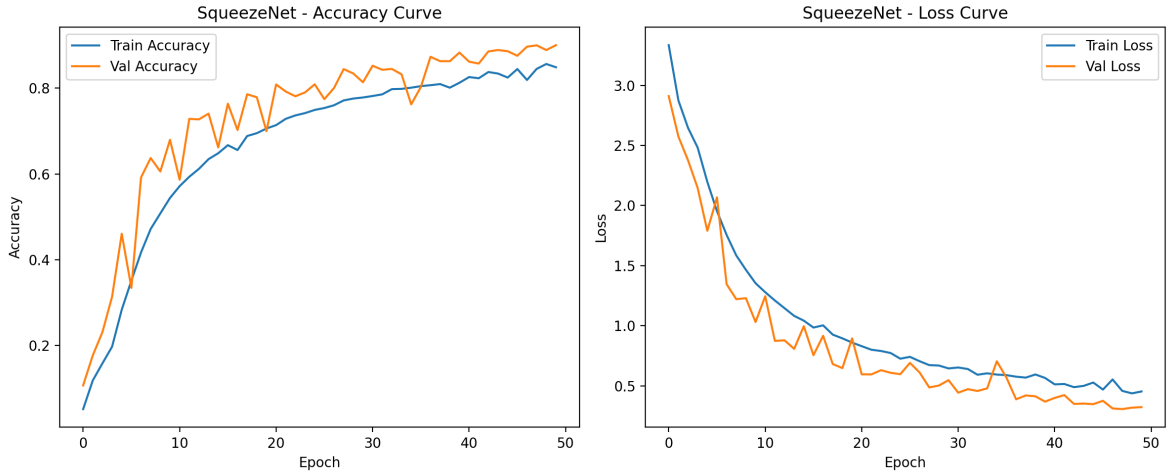


Figure 5.24: Loss and accuracy per epoch for model SqueezeNet.

5.3.2 AUC-ROC Analysis

A detailed comparative analysis of several advanced deep learning models based on their AUC-ROC curves, a pivotal metric in assessing classification performance is represented in Figure 5.25.

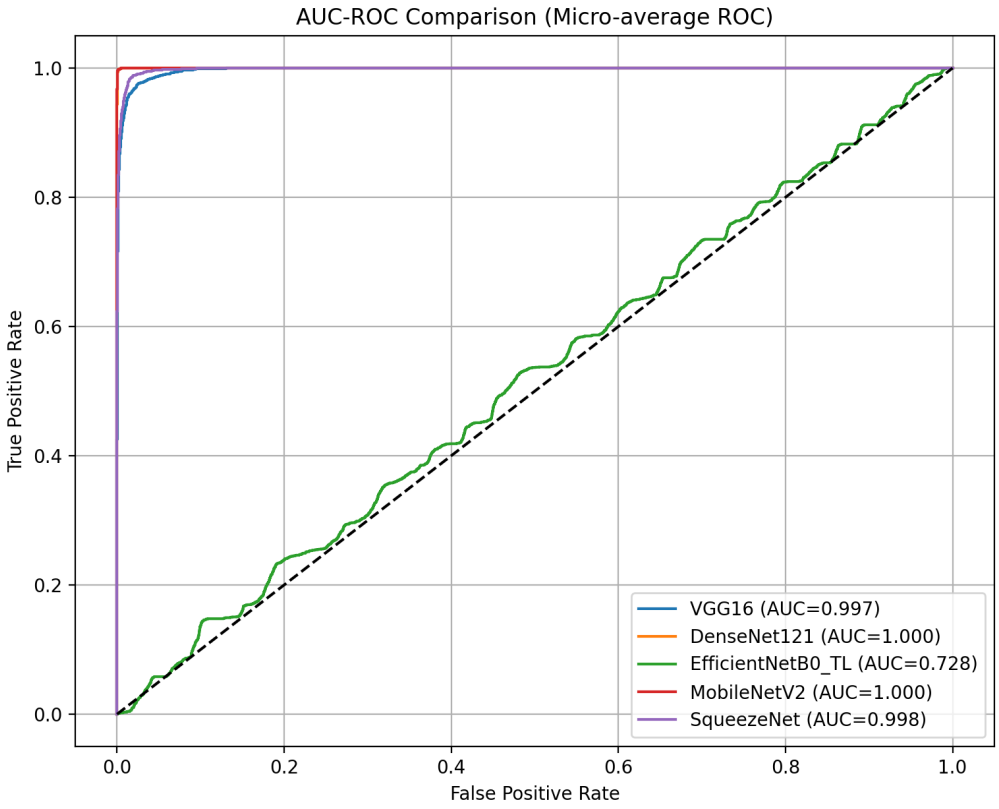


Figure 5.25: AUC-ROC curves for different models.

Its robust performance underscores its effectiveness in handling complex datasets and reinforces its position as a top choice in applications requiring nuanced classification abilities. Following closely are Xception, ResNet50, InceptionV3, and DenseNet121, all demonstrating strong and reliable classification capabilities as evidenced by their commendable AUC values. These models excel in capturing intricate patterns and variations within data, making them invaluable tools in fields such as medical diagnostics, autonomous systems, and natural language processing where accurate classification is paramount. Despite marginally lower AUC scores compared to the top performers, VGG16, MobileNetV2, and InceptionResNetV2

exhibit noteworthy performance metrics, affirming their robustness and versatility in diverse classification tasks.

5.4 Custom Mode: MediNet-XG

5.4.1 MediNet Classifier

The MediNet classifier model achieved an impressive accuracy of 0.9838, indicating its high predictive performance. Additionally, with a low loss of 0.1008, the model demonstrates strong reliability and minimal prediction error.

MediNet classifier Model Accuracy: 0.0.9837754261264589

MediNet classifier Model Loss: 0.1008325461891258

The report provides detailed insights into the precision, recall, F1-score, and support for each class of Ensemble-1. Such metrics present clear view of the model's effectiveness.

Table 5.9: Classification report for MediNet-XG.

Class	Precision	Recall	F1-Score	Support
0	0.96	1.00	0.98	75
1	0.96	1.00	0.98	75
2	0.99	0.91	0.94	75
3	0.96	0.99	0.97	75
4	0.99	1.00	0.99	75
5	1.00	0.99	0.99	75
6	0.95	0.99	0.97	75
7	1.00	1.00	1.00	75
8	0.91	0.97	0.94	75
9	1.00	0.97	0.99	75
10	0.94	1.00	0.97	75
11	1.00	0.96	0.98	75
12	1.00	0.95	0.97	75
13	0.96	1.00	0.98	75

Class	Precision	Recall	F1-Score	Support
14	0.91	0.93	0.92	75
15	1.00	1.00	1.00	75
16	1.00	1.00	1.00	75
17	0.95	0.93	0.94	75
18	1.00	1.00	1.00	78
19	1.00	0.99	0.99	75
20	1.00	1.00	1.00	75
21	0.97	0.96	0.97	75
22	0.99	1.00	0.99	75
23	0.99	1.00	0.99	75
24	1.00	0.97	0.99	75
25	0.99	0.92	0.95	75
26	0.96	0.99	0.97	75
27	0.99	0.92	0.95	75
28	0.95	0.95	0.95	75
29	1.00	1.00	1.00	75
30	1.00	1.00	1.00	80
31	0.96	0.93	0.95	75
32	0.97	1.00	0.99	75
33	0.97	0.97	0.97	75
Accuracy			0.98	2585
Macro Avg	0.98	0.98	0.98	2585
Weighted Avg	0.98	0.98	0.98	2585

5.4.2 Grad-CAM (X)

Grad-CAM analysis shows how MediNet-XG uses high-level class scores to generate spatially localized evidence of decisions on the leaf surface. To make each prediction, the gradient of each target class logit with respect to the final convolutional feature maps is calculated,

channel-wise gradients are averaged across all channels of the global gradient, a weighted sum of feature maps is passed through a ReLU and upsampled to the input resolution to yield a class-discriminative heatmap, which is used to encode the importance of each pixel position in the decision. When image classifications are correct, the resulting activation is strongly focalised around the lamina, margin and venation areas, showing that the decision rule of the model is driven by internal filters that react to leaf morphology but not by the background textures, whereas in ambiguous or misclassified images the heatmaps become diffused or decentralised, revealing exactly which regions of the learned representation space are not clearly separated by the feature hierarchy.

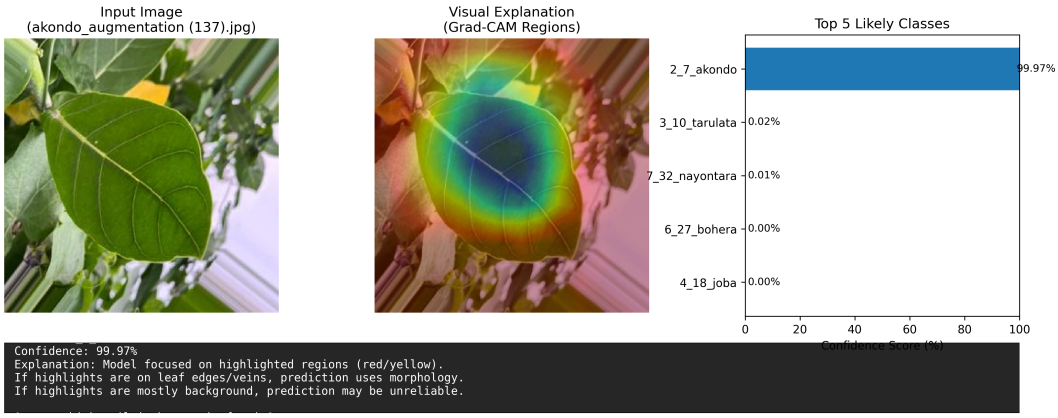


Figure 5.26: Grad-Cam explanation.

5.4.3 t-SNE (X)

t-SNE visualization the embedding space of MediNet-XG is a projection of high-dimensional feature vectors in the global pooling layer onto two dimensions, maintaining local neighbourhood connections. Each test sample is represented in the embedding model to produce a fixed-length representation which is encoded into an $N \times d$ matrix and fed into t-SNE which optimizes a two-dimensional layout in which similarity in the original space between the pair of test samples are approximated using Student-t distributed affinity; clusters separated widely in this plot indicate the network has discovered distinct manifolds in the learned feature space between different medicinal species, whereas an overlap indicates that the two classes share leaf morphologies close together in the learned feature space. The large, compact clusters imply a high per-class accuracy, and wider or mixed clusters imply a higher confusion rate in

the confusion matrix, which would directly relate the quantitative misclassification behavior to this representation learned by MediNet-XG geometry.

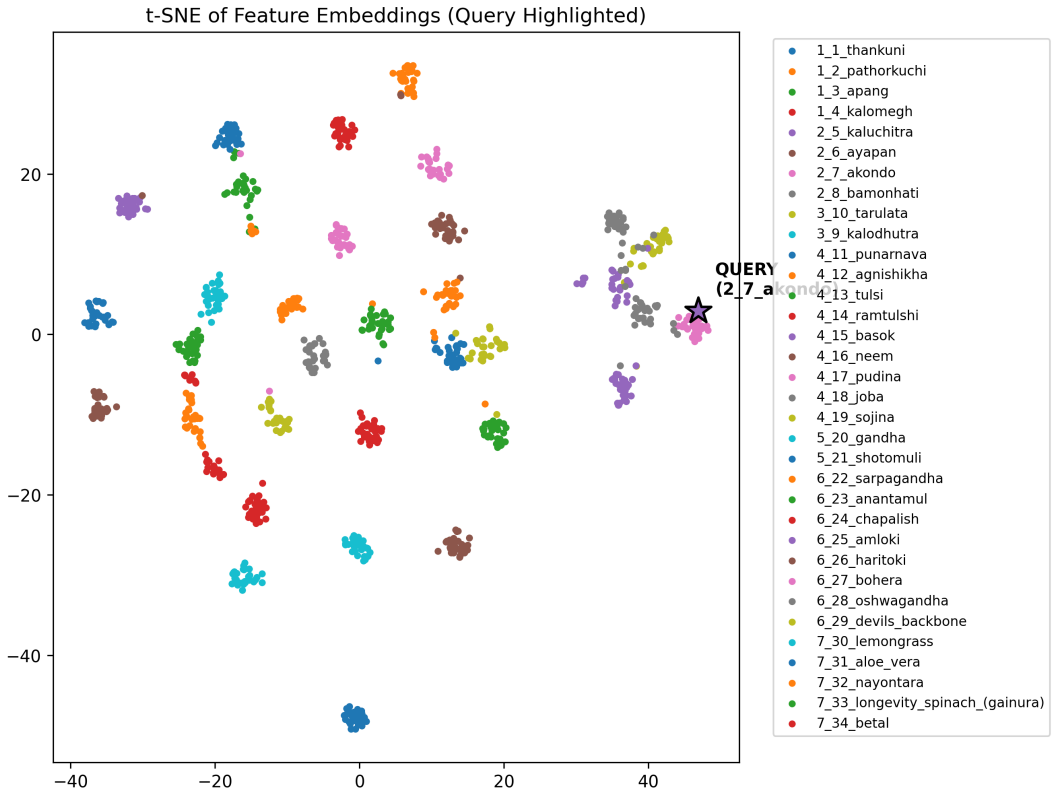


Figure 5.27: t-SNE plot diagram explanation by marking exact position.

5.4.4 knowledge-based query generator from JSON (G)

The knowledge-based query generator (G) is a deterministic retrieval tier which takes the MediNet-XG predicted class and a user query and returns field-precise botanical responses using only the curated JSON knowledge base. The rank-conscious label, after classification, is converted to a canonical plant key, which indexes a structured record, fields of which include scientific name, family, morphology, therapeutic uses, active compounds, toxicity levels, contraindications, and pregnancy-lactation safety. The input query is tokenized and compared to pre-defined sets of keywords coded in such a way that they are hard-wired to specific fields such that, e.g., use-related terms route to traditional and modern uses, safety-related terms route to toxicity, contraindication and pregnancy-safety fields whereby the system selects and concatenates only those JSON values and does not generate new content or alter the underlying facts. On unsupported or out-of-scope queries, a controlled fallback message will be

generated stating that the information requested is not present in the knowledge base to maintain the transparency of all decisions and provide the ability to audit them and keep them within the already tested medicinal plant data limitations.

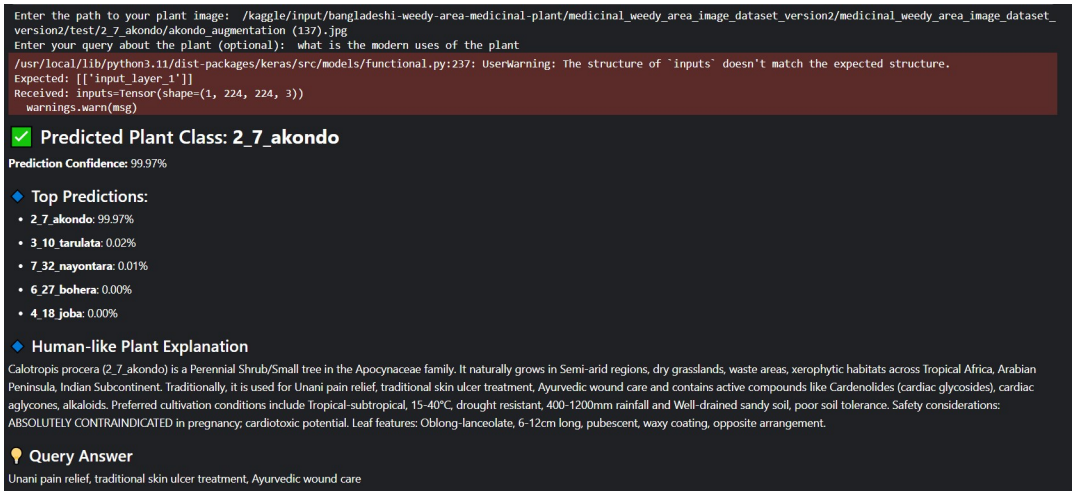


Figure 5.28: Generating the knowledge-based query answer.

5.5 Comparative Analysis

5.5.1 Dataset Comparison and Analysis

MediNet-XG had also been evaluated on four publicly available datasets of leaf lesions on beans[29], mango[30], tomato[31], and wheat[32] leaves on Kaggle, assessed through cross-dataset generalization. The datasets are different classification problems with varying numbers of classes (3 in bean, 8 in mango, 10 in tomato, and 3 in wheat), image statistics, and acquisition conditions and, therefore, represent a stress test of the extent to which the learned representations can generalize outside of Bangladeshi weedy medicinal plants[1]. Following the training of the model on each dataset using the same pipeline yields final training accuracies of bean, mango, tomato and wheat at 89.58, 99.39, 98.85, and 92.38 percent respectively and validation accuracies of 81.43, 98.25, 98.30 and 91.90 percent respectively as summarized in Table 5.10.

The difference between training and validation values on bean and wheat (e.g., validation accuracies of about 80–90% and losses between 0.27–0.43 with such significantly higher training accuracy) also suggests a domain mismatch in the case of MediNet-XG on such smaller

or noisier training lesion sets without task-specific regularization. By contrast, mango and tomato datasets are both well-trained and well-validated (97% and 98% respectively) with low validation loss of around 0.03; indicating that identical architecture can effectively enter disease-related visual patterns in case adequate and well-organized data is present. All in all, this comparison illustrates that the MediNet-XG compact architecture can be successfully generalized to external leaf recognition tasks when the data quality is sufficient and when facing more difficult or smaller data sets, the need to vary the data and align it with the domain is significant when it comes to successful transfer. 5.10.

Table 5.10: Comparison of existing datasets with RSBdSL38.

Dataset	**Dataset Size**	**No. of Classes**	**Val Acc**	**Val Loss**
Bean-leaf-lesions[29]	9,00	3	0.8143	0.4380
Mango-leaf-disease[30]	6,320	8	0.9825	0.0579
Tomatoleaf[31]	7,092	10	0.9885	0.0362
Wheat-leaf-datase[32]t	1,700	3	0.9190	0.1992
Bangladeshi weedy medicinal plants[1]	**17.050**	**34**	**0.9882**	**0.0495**

5.5.2 Models Comparison and Analysis on Bangladeshi weedy medicinal plants[1]

A direct comparison between all the compared architectures points to the trade-off between predictive performance and model complexity on the weed-infested medicinal plant dataset. By making use of task-specific inverted-residual design and selective channel attention MediNet-XG achieves the best accuracy (98.82%), precision (98.75%), recall (98.87%), and F1-score (98.79%) with the lowest footprint (342 KB), proving the idea that a task-specific inverted-residual architecture with channel attention selectivity can be several-fold lighter than more heavy transfer-learning backbones. The second best in overall recognition is DenseNet121, which is more accurate and has a F1-score of nearly 97.9% but has a model size of 26.98 MB, then comes VGG16 and SqueezeNet, which are less accurate and has lower F1-score at around 88-89% but has a much bigger (56.20 MB) and a moderately large (889 KB) parameter count respectively. Conversely, EfficientNetB0TL and MobileNetV2 demonstrate worse per-

formance in this context, with EfficientNetB0TL demonstrating erratic behaviour and huge losses, and MobileNetV2 lower F1-scores despite its much large size, suggesting that even generic lightweight backbones and naive transfer learning are not able to adapt to the dense background clutter and fine-grained leaf morphology of the proposed dataset. In general, the MediNet-XG provides the most desirable balance that is as accurate as large CNNs but needs to consume one or two orders of magnitude less memory, which is essential to identify medicinal plants in real-time on mobile and edge devices.

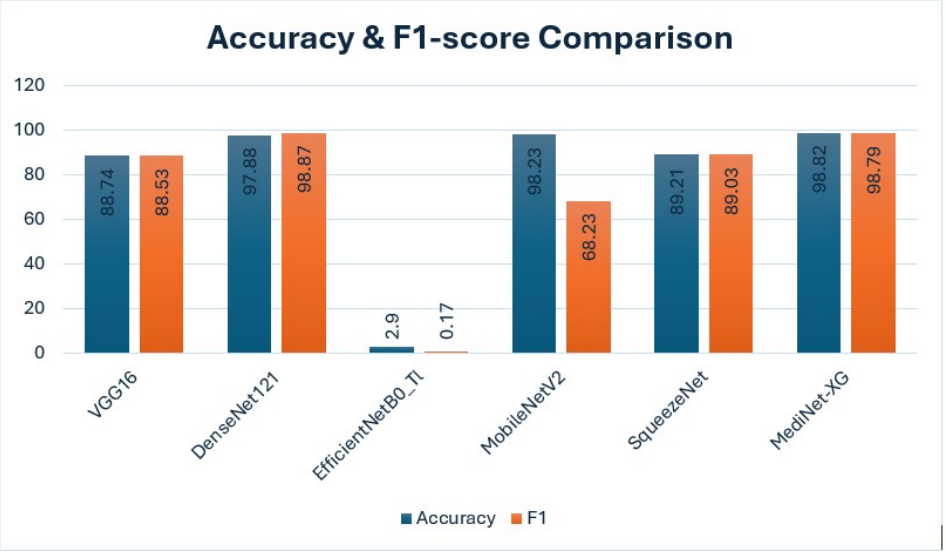


Figure 5.29: Accuracy comparison of various models on Bangladeshi weedy medicinal plants dataset.[1]

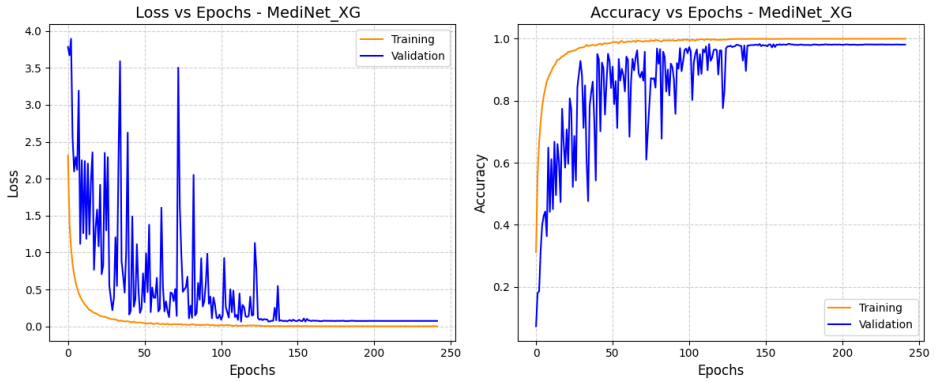


Figure 5.30: Aucc and Loss of MediNet-XG Model.

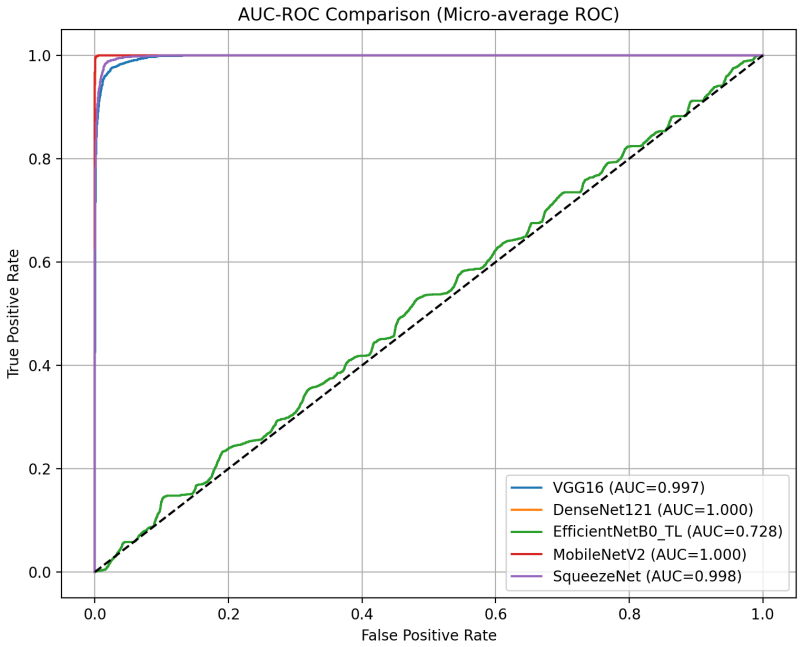


Figure 5.31: AUC-ROC comparison of various models on RSBdSL38 dataset.

5.5.3 Comparative analysis with reviewed (journals) methods

Table 5.X will provide a summary of how the proposed MediNet-XG framework integrates with and builds on the current research in medicinal plant and weed recognition, as well as explainable plant phenotyping in general. Current classifiers of Bangladeshi medicinal plants typically work using controlled leaf images of small numbers of plants and heavy CNN or ensemble implementations which are black box classifiers that do not inherently have safety-conscious knowledge retrieval. Studies of global herbal and phenotyping, and weed detection have been shown to perform well on curated or crop-targeted datasets, with some integrating Grad-CAM, but most do not aim at weedy medicinal flora, do not integrate vision with a structured botanical knowledge base, and are not optimized to ultra-low model size at the edge. Conversely, MediNet-XG is targeted at 34 Bangladeshi weedy medicinal species collected under field contamination and integrates a lightweight custom CNN, and required Grad-CAM and t-SNE explainability, and a retrieval-based JSON knowledge layer, which recalls structured botanical, therapeutic, and safety information. The framework has thus met three features that are not all simultaneously covered in earlier literature, namely field imagery of medicinal weeds but not controlled or non-medicinal plants, enforceable explainability in

both spatial and feature-space views, and knowledge-grounded, non-free-form responses that can be used under safety-critical conditions.

Table 5.11: Comparison of MediNet-XG with representative deep-learning studies

Study	**Target domain**	**Modality**	**XAI**	**Acc./F1**	**Remarks**
[3]	BD medicinal (10 spp., controlled leaf)	Image	No	97.62%	Heavy CNN/ensemble, black-box
[5]	Indonesian herbal species	Image	No	∼97%	TL CNN on curated images
[6]	General crops phenotyping	Img + Txt	Grad-CAM	-	XAI used, not medicinal weeds
[7]	Crop vs weeds in wheat fields	Image	Limited	>95%	Field images, no knowledge base
[8]	BD medicinal (curated leaves)	Image	No	∼97%	Heavy dual-attention CNN
[15]	Medicinal vs fruit vs flower	Image	No	>95%	Multistage classifier, no XAI
MediNet-XG	**BD weedy medicinal (34 spp., field)**	**Img + Text**	**Grad-CAM + t-SNE**	**98%**	**∼342 KB, retrieval-based knowledge**

5.6 Summary

The custom MediNet-XG best achieves the overall trade-off between accuracy and efficiency on the 34-class weedy medicinal plant dataset with an approximation of 98.8 percent accuracy and 0.99 F1-score with the footprint of about 342 KB, whilst the larger baselines like VGG16 and DenseNet121 achieve slightly less performance at a much higher memory cost. Grad-CAM analysis and t-SNE analysis prove that for the majority of species, MediNet-XG makes a decision based on the morphology of leaves and builds well-defined clusters of features and the query generator based on JSON allows generating predictions as the structured and safety-aware botanical explanation not resorting to uncontrolled generation of text.

CHAPTER 6
Conclusions and Future Works

6.1 Conclusions

Medicinal weeds, as they are commonly known as growing in crop fields, homesteads, and roadsides constitute an extremely important but little noted source of healthcare, particularly in rural environments where there is a lack of formal medical facilities and professionals knowledgeable in botanical studies. The wrong identification of such vegetation in contaminated, weedy scenes may result in inefficient treatment, ingestion of harmful species and loss of faith in customary remedies making proper classification and clear articulation of classification a necessity and not a choice. Geographic concentration of specialists combined with the dependence on printed floras or informal apprenticeship further limits access of expert level ethnobotanical knowledge, thus making practical, field ready guidance unavailable to many of the potential end users. This gap was filled by construction of the weedy medicinal plant dataset, which recorded 17,050 images of 34 species in realistic Bangladeshi field scenarios, with uncontrolled light sources and high background clutter, and annotated it with a rank-conscious naming scheme that represents both the species itself and the empirical weediness; this dataset is fundamentally different to previous curated-leaf datasets and thus represents a new resource to both computer vision and ethnomedicine. The related JSON body of knowledge is a compilation of detailed and structured data on morphology, uses, active compounds, toxicity and pregnancy safety of authoritative sources, and deterministically linked to each class name, creating a reproducible interface between visual recognition and safe medicinal context. In that context, technology becomes a leveler: a phone with a camera can be converted into a point of care decision support product, as long as the model behind it is correct, understandable and small enough to execute on-phone even without connection to

the Internet. To meet this need, MediNet-XG employs a lightweight, but carefully designed inverted-residual architecture with selective channel attention to learn domain-specific leaf features more efficiently than significantly larger models based on transfer-learning, indicating that a well-designed inverted-residual architecture with channel attention can effectively utilize inverted-residual signals to extract domain-specific leaf features compared to generic transfer-learning models. Combined Grad-CAM and t-SNE pipelines reveal both pixel- and feature-level internal decision logic, and the retrieval based JSON query generator converts each prediction into a constrained, audit-friendly explanation that honors safety limits, both of which meet the transparency and non-hallucination requirements of health-proximate usage. Due to the small size, explainability and integration with a structured knowledge layer, the model can be deployed to IoT and edge devices: medicinal weeds can be detected and placed directly in the fields or villages without depending on remote servers, network connectivity, or continuous oversight by experts. In this regard, MediNet-XG is a new addition not only in having attained the highest accuracy with a ultralightweight architecture, but also that architecture is implemented into a complete, safety-conscious pipeline that embodies real-life limitations on the use of medicinal plants in the use of weed-infested regions.

6.2 Future Recommendations

Further studies must expand the existing medicinal weed benchmarks with more species, growth phases, seasonal factors, and geographically varied collection locations and therefore enhance resilience in ecological diversity and also enhance extrapolation past the current 34-class restriction. Additional development of the architectural design with more sophisticated attention schemes, calibration-sensitive training, and sparse uncertainty sampling would enable the system to provide calibrated confidence and selectively abstain during uncertain situations, which is paramount to safety in health-proximate environments. At the multimodal end, better and more extensively structured knowledge base (with standardized plant ontologies, vernacular synonyms, and more powerful connections to pharmacological evidence) would facilitate refinement and expansion of the structured knowledge base to richer but nevertheless retrieval-constrained explanations, whereas collaborative verification with botanists, ethnomedicine practitioners and rural workers would assist in real deployment requirements

of future iterations of MediNet-XG. The interaction layer can also be extended in the future, a supervised dialogue interface, strictly a retrieval-based interaction, can also accept a more conversational query, supported by other fields filled in through a strictly regulated extraction of peer-reviewed medicinal plant journals and managed pharmacological databases under a restricted expert supervision. With each expansion in scale of the underlying dataset (adding medicinal weeds and non-medicinal look-alikes, region-specific variants), a corresponding expansion of the class set and a corresponding ability to deal with a wider range of real-world plant diversity can be achieved, without affecting the explainable, safety-aware behaviour of MediNet-XG.

REFERENCES

[1] M. Nur-A-Alam, "Bangladeshi weedy area medicinal plant," https://www.kaggle.com/datasets/mdnuraalambaust/bangladeshi-weedy-area-medicinal-plant, 2024, accessed: 2024-05-22.

[2] 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," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 618–626.

[3] A. H. Uddin, Y.-L. Chen, B. Borkatullah, M. S. Khatun, J. Ferdous, P. Mahmud, J. Yang, C. S. Ku, and L. Y. Por, "Deep-learning-based classification of bangladeshi medicinal plants using neural ensemble models," *Mathematics*, vol. 11, no. 16, p. 3504, 2023.

[4] A. K. Mulugeta, D. Sharma, and A. H. Mesfin, "Deep learning for medicinal plant species classification and recognition: a systematic review," *Frontiers in Plant Science*, vol. 14, 2024.

[5] M. S. I. Musyaffa, N. Yudistira, M. A. Rahman, A. H. Basori, A. B. F. Mansur, and J. Batoro, "Indoherb: Indonesia medicinal plants recognition using transfer learning and deep learning," *Heliyon*, vol. 10, no. 23, 2024.

[6] S. Mostafa, D. Mondal, K. Panjvani, L. Kochian, and I. Stavness, "Explainable deep learning in plant phenotyping," *Frontiers in Artificial Intelligence*, vol. 6, p. 1203546, 2023.

[7] M. Guzel, B. Turan, I. Kadioglu, A. Basturk, B. Sin, and A. Sadeghpour, "Deep learning for image-based detection of weeds from emergence to maturity in wheat fields," *Smart Agricultural Technology*, vol. 9, p. 100552, 2024.

[8] F. H. Bhoyan, M. H. K. Mehedi, M. Ohona, S. Rashid, and M. Mridha, "An efficient dual-attention guided deep learning model with interpretability for identifying medicinal plants," *Current Plant Biology*, p. 100533, 2025.

[9] M. A. Akhtar, "Anti-inflammatory medicinal plants of bangladesh—a pharmacological evaluation," *Frontiers in Pharmacology*, vol. 13, 2022.

[10] J. Hu, K. Kozlov, A. H. Uddin, Y.-L. Chen, B. Borkatullah, M. S. Khatun, J. Ferdous, J. Yang, C. S. Ku, and Y. Por, "Deep-learning-based classification of bangladeshi medicinal plants using neural ensemble models," *Mathematics*, 2023.

[11] S. Islam, M. R. Ahmed, S. Islam, M. M. A. Rishad, S. Ahmed, T. R. Utshow, and M. I. Siam, "Bdmedileaves: A leaf images dataset for bangladeshi medicinal plants identification," *Data in Brief*, vol. 50, p. 109488, 2023.

[12] F. Garibaldi-Márquez, G. Flores, D. Mercado-Ravell, A. Ramirez-Pedraza, and L. M. Valentín-Coronado, "Weed classification from natural corn field-multi-plant images based on shallow and deep learning," *Sensors (Basel, Switzerland)*, vol. 22, 2022.

[13] M. Nur-A-Alam, "MediNet-XG: Final year thesis on Medicinal Plant Classification," https://github.com/Md-Nur-A-Alam/MediNet-XG-Final-year-thesis, 2024, accessed: 2024-05-22.

[14] S. Kalime, P. Sujatha, D. B. Vunnava, S. Sushma, T. K. Sajja *et al.*, "A novel multistage approach for medicinal plant classification with deep learning techniques," *International Research Journal of Multidisciplinary Technovation*, vol. 7, no. 4, pp. 99–114, 2025.

[15] H. Zhao and Y. Wang, "Deep learning–based approaches for weed detection in crops," *Frontiers in Plant Science*, vol. 16, p. 1746406, 2025.

[16] Medicinal Plant Database of Bangladesh, "Medicinal plant database of bangladesh," 2024, accessed: January 15, 2026. [Online]. Available: http://www.mpedb.com.bd/

[17] Bangladesh National Herbarium, "Official website of bangladesh national herbarium (bnh)," 2024, accessed: January 15, 2026. [Online]. Available: http://www.bnh.gov.bd/

[18] Bangladesh Association of Plant Taxonomists, "Bangladesh journal of plant taxonomy," *Bangladesh Journal of Plant Taxonomy*, online ISSN: 2224-7297. [Online]. Available: https://www.banglajol.info/index.php/BJPT

[19] POWO, "Plants of the world online," 2024, accessed: January 15, 2026. [Online]. Available: http://www.plantsoftheworldonline.org/

[20] Wikipedia contributors, "Lanczos resampling — Wikipedia, the free encyclopedia," 2024, [Online; accessed 11-June-2024]. [Online]. Available: https://en.wikipedia.org/w/index.php?title=Lanczos_resampling&oldid=1225247839

[21] S. G. K. Patro and K. K. Sahu, "Normalization: A preprocessing stage," 2015.

[22] W. Ge, P. Lalbakhsh, L. Isai, A. Lensky, and H. Suominen, "Comparing deep learning models for the task of volatility prediction using multivariate data," *arXiv preprint arXiv:2306.12446*, 2023.

[23] F. J. P. Montalbo and A. S. Alon, "Empirical analysis of a fine-tuned deep convolutional model in classifying and detecting malaria parasites from blood smears." *KSII Trans. Internet Inf. Syst.*, vol. 15, no. 1, pp. 147–165, 2021.

[24] A. Tragoudaras, P. Stoikos, K. Fanaras, A. Tziouvaras, G. Floros, G. Dimitriou, K. Kolomvatsos, and G. Stamoulis, "Design space exploration of a sparse mobilenetv2 using high-level synthesis and sparse matrix techniques on fpgas," *Sensors*, vol. 22, p. 4318, 06 2022.

[25] N. Radwan, "Leveraging sparse and dense features for reliable state estimation in urban environments," Ph.D. dissertation, University of Freiburg, Freiburg im Breisgau, Germany, 2019.

[26] M. Hasan, M. Fatemi, M. Khan, M. Kaur, and A. Zaguia, "Comparative analysis of skin cancer (benign vs. malignant) detection using convolutional neural networks," *Journal of Healthcare Engineering*, vol. 2021, pp. 1–17, 12 2021.

[27] S. E. Nassar, I. Yasser, H. M. Amer, and M. A. Mohamed, "A robust mri-based brain tumor classification via a hybrid deep learning technique," *The Journal of Supercomputing*, vol. 80, no. 2, pp. 2403–2427, 2024.

[28] L. v. d. Maaten and G. Hinton, "Visualizing data using t-sne," *Journal of machine learning research*, vol. 9, no. Nov, pp. 2579–2605, 2008.

[29] marquis03 (Kaggle), "Bean leaf lesions classification," Kaggle Dataset, 2026, https://www.kaggle.com/datasets/marquis03/bean-leaf-lesions-classification (accessed January 17, 2026).

[30] aryashah2k (Kaggle), "Mango leaf disease dataset," Kaggle Dataset, 2026, https://www.kaggle.com/datasets/aryashah2k/mango-leaf-disease-dataset (accessed January 17, 2026).

[31] sarojbhagat (Kaggle), "Tomatoleaf dataset," Kaggle Dataset, 2026, https://www.kaggle.com/datasets/sarojbhagat/tomatoleaf (accessed January 17, 2026).

[32] olyadgetch (Kaggle), "Wheat leaf dataset," Kaggle Dataset, 2026, https://www.kaggle.com/datasets/olyadgetch/wheat-leaf-dataset (accessed January 17, 2026).

APPENDIX A

Raw data processing:

```
import os, random
import tensorflow as tf
import numpy as np

IMG_SIZE = 224
BATCH_SIZE = 32
SEED = 42
AUTOTUNE = tf.data.AUTOTUNE

DATASET_DIR = "/kaggle/input/bean-leaf-lesions-classification/train"

raw_ds = tf.keras.utils.image_dataset_from_directory(
    DATASET_DIR,
    image_size=(IMG_SIZE, IMG_SIZE),
    batch_size=BATCH_SIZE,
    label_mode="categorical",
    shuffle=True,
    seed=SEED
)

class_names = raw_ds.class_names
num_classes = len(class_names)
```

Data Augmentation:

```
from tensorflow.keras.layers import RandomFlip, RandomRotation, RandomZoom,
    ↪ RandomContrast

augmentation = tf.keras.Sequential([
    RandomFlip("horizontal"),
    RandomRotation(0.1),
    RandomZoom(0.2),
    RandomContrast(0.1)
])

def augment(x, y):
    return augmentation(x), y

aug_ds = raw_ds.map(augment, num_parallel_calls=AUTOTUNE)
```

Train / Validation / Test Split:

```
dataset_size = tf.data.experimental.cardinality(aug_ds).numpy()
train_size = int(0.7 * dataset_size)
val_size = int(0.2 * dataset_size)

train_ds = aug_ds.take(train_size)
val_ds = aug_ds.skip(train_size).take(val_size)
test_ds = aug_ds.skip(train_size + val_size)
```

```
rescale = tf.keras.layers.Rescaling(1./255)

train_ds = train_ds.map(lambda x,y:(rescale(x),y)).cache().prefetch(AUTOTUNE)
val_ds   = val_ds.map(lambda x,y:(rescale(x),y)).cache().prefetch(AUTOTUNE)
test_ds  = test_ds.map(lambda x,y:(rescale(x),y)).cache().prefetch(AUTOTUNE)
```

Build MediNet Model:

```
from tensorflow.keras import backend as K
from tensorflow.keras.layers import Input, Conv2D, DepthwiseConv2D,
  ↪ BatchNormalization
from tensorflow.keras.layers import Activation, Add, Dropout,
  ↪ GlobalAveragePooling2D
from tensorflow.keras.layers import Dense, Multiply, Reshape
from tensorflow.keras.models import Model

def efficient_channel_attention(x, ratio=8):
    ch = K.int_shape(x)[-1]
    se = GlobalAveragePooling2D()(x)
    se = Dense(ch//ratio, activation="relu")(se)
    se = Dense(ch, activation="sigmoid")(se)
    se = Reshape((1,1,ch))(se)
    return Multiply()([x,se])

def inverted_residual(x, f, s, t, attn):
    in_ch = K.int_shape(x)[-1]
    y = Conv2D(in_ch*t, 1, padding="same", use_bias=False)(x)
    y = BatchNormalization()(y)
    y = Activation("relu6")(y)
    y = DepthwiseConv2D(3, strides=s, padding="same", use_bias=False)(y)
    y = BatchNormalization()(y)
    y = Activation("relu6")(y)
    if attn:
        y = efficient_channel_attention(y)
    y = Conv2D(f, 1, padding="same", use_bias=False)(y)
    y = BatchNormalization()(y)
    if s==1 and in_ch==f:
        y = Add()([x,y])
    return y

def MediNet(input_shape, num_classes):
    inp = Input(shape=input_shape)
    x = Conv2D(16, 3, strides=2, padding="same", use_bias=False)(inp)
    x = BatchNormalization()(x)
    x = Activation("relu6")(x)

    x = inverted_residual(x, 24, 2, 2, False)
    x = inverted_residual(x, 32, 2, 3, False)
    x = inverted_residual(x, 64, 2, 4, True)
    x = inverted_residual(x, 96, 1, 6, True)

    x = Conv2D(320, 1, use_bias=False)(x)
    x = BatchNormalization()(x)
    x = Activation("relu6")(x)

    x = GlobalAveragePooling2D(name="embedding")(x)
    x = Dropout(0.2)(x)
    out = Dense(num_classes, activation="softmax")(x)

    return Model(inp, out)

model = MediNet((IMG_SIZE,IMG_SIZE,3), num_classes)
```

Train Model:

```
from tensorflow.keras.optimizers import SGD
from tensorflow.keras.callbacks import EarlyStopping

model.compile(
    optimizer=SGD(0.01, momentum=0.9),
    loss="categorical_crossentropy",
    metrics=["accuracy"]
)

history = model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=50,
    callbacks=[EarlyStopping(patience=10, restore_best_weights=True)],
    verbose=1
)
```

Model Evaluation:

```
from sklearn.metrics import classification_report

y_true = np.concatenate([y.numpy() for _,y in test_ds])
y_prob = model.predict(test_ds)
y_pred = np.argmax(y_prob, axis=1)
y_true = np.argmax(y_true, axis=1)

print(classification_report(y_true, y_pred, target_names=class_names))
```

Grad-CAM:

```
from tensorflow.keras.layers import Conv2D, DepthwiseConv2D
import matplotlib.pyplot as plt

def last_conv(model):
    for l in reversed(model.layers):
        if isinstance(l,(Conv2D,DepthwiseConv2D)):
            return l.name

def gradcam(img, model):
    ln = last_conv(model)
    gmodel = Model(model.inputs,[model.get_layer(ln).output,model.output])
    with tf.GradientTape() as tape:
        conv, pred = gmodel(img)
        cls = tf.argmax(pred[0])
        loss = pred[:,cls]
    grads = tape.gradient(loss, conv)
    w = tf.reduce_mean(grads,axis=(0,1,2))
    cam = tf.reduce_sum(w*conv[0],axis=-1)
    cam = tf.maximum(cam,0)/tf.reduce_max(cam)
    return cam.numpy()
```

t-SNE:

```
from sklearn.manifold import TSNE
import matplotlib.pyplot as plt

embed_model = Model(model.input, model.get_layer("embedding").output)

```

```
features, labels = [], []
for x,y in test_ds.take(10):
    features.append(embed_model.predict(x))
    labels.extend(np.argmax(y.numpy(),axis=1))

X = np.vstack(features)
Z = TSNE(2, perplexity=30).fit_transform(X)

plt.scatter(Z[:,0], Z[:,1], c=labels, s=8)
plt.show()
```

Query answer generator:

```
import json

def load_knowledge_base(json_path):
    with open(json_path, "r", encoding="utf-8") as f:
        kb = json.load(f)
    class_keys = sorted(list(kb.keys()))
    return kb, class_keys

def map_query_to_field(query):
    q = query.lower().strip()
    mapping = [
        (["scientific", "scientific name"], "Scientific Name"),
        (["botanical", "family"], "Botanical Family"),
        (["genus", "species"], "Genus & Species"),
        (["leaf", "morphology", "shape"], "Leaf Morphology"),
        (["life", "lifespan", "life span"], "Life Span"),
        (["habit", "plant habit"], "Plant Habit"),
        (["habitat", "native habitat"], "Native Habitat"),
        (["origin", "geographical", "country"], "Geographical Origin"),
        (["uses", "traditional uses"], "Traditional Uses"),
        (["medicinal", "therapeutic"], "Therapeutic Properties"),
        (["dosage", "dose"], "Standard Dosage"),
        (["toxic", "toxicity"], "Toxicity Levels"),
        (["safety", "pregnancy", "lactation"], "Pregnancy & Lactation Safety"),
        (["interaction", "drug"], "Drug-Herb Interactions"),
        (["soil"], "Preferred Soil Type"),
        (["climate"], "Climate Requirements"),
        (["compound", "active"], "Primary Active Compounds"),
    ]
    for keys, field in mapping:
        if any(k in q for k in keys):
            return field
    return None

def query_answer_generator(predicted_class, confidence, user_query, knowledge_base,
    ↪ confidence_threshold=0.0):
    if confidence < confidence_threshold:
        return {
            "predicted_class": predicted_class,
            "confidence": float(confidence),
            "answer": "Low confidence prediction. Please provide a clearer image."
        }

    info = knowledge_base.get(predicted_class, {})
    if not info:
        return {
            "predicted_class": predicted_class,
            "confidence": float(confidence),
            "answer": "No knowledge base entry found for this predicted class."
        }

    field = map_query_to_field(user_query)
    if field is None:
        return {
            "predicted_class": predicted_class,
            "confidence": float(confidence),
```

```
            "answer": "Query type not supported by the knowledge base mapping."
        }

    value = info.get(field, "")
    answer = value if str(value).strip() else "No information available for this query."
    return {
        "predicted_class": predicted_class,
        "confidence": float(confidence),
        "query": user_query,
        "field_used": field,
        "answer": answer
    }
kb, _ = load_knowledge_base(JSON_PATH)
result = query_answer_generator(pred_class, conf_percent, user_query, kb, confidence_threshold=20.0)
print(result["answer"])
```

APPENDIX B

Complex Engineering Problems (CP) and Complex Engineering Activities (CA) Analysis

Title: Bangla Sign Language Recognition: Dataset Innovation and Methodologies.

Attainment of Complex Engineering Problem (CP)

S.L.	CP No.	Attainment	Remarks
1.	P1: Depth of Knowledge Required	Yes	K3 (Engineering Fundamentals): python describe in Chapter 4 Art No: 4.2.2.
			K4 (Engineering Specialization): knowledge about Deep learning describe in Chapter 3. Art No: 3.3
			K5 (Design): Methodology flow chart describe in Chapter 3 Art. No: 3.5.
			K6 (Technology): Tensorflow, Numpy, Pandas, etc. Described on Chapter 4 Art. No: 4.2.
			K8 (Research): Related work describe in Chapter 2
2.	P2: Range of Conflicting Requirements	Yes	Learn about Bangla Sign Language describe in Chapter 1 Art. No: 1.2 and knowledge about deep learning describe in Chapter 3. Art No: 3.3.

3.	P3: Depth of Analysis Required	Yes	Build new dataset about Bangla Sign Language described in 3 Art. No: 3.2.
4.	P4: Familiarity of Issues	Yes	Study about Bangla Sign Language as a CSE students describe in Chapter 1 and Chapter 3.
5.	P5: Extent of Applicable Codes	Yes	Follow standards methodology describe on Chapter 3
6.	P6: Extent of Stakeholder Involvement and Conflicting Requirements	Yes	Involves Post Office, All of the people who are involved are digital Documents processing Chapter 1 Art. No: 1.6
7.	P7: Interdependence	Yes	Collecting Dataset describe in Chapter 3 Art. No: 3.2. Apply different augmentations also describe in Chapter 3 Art. No: 3.2.

Mapping of Complex Engineering Activities (CA)

S.L.	**CA No.**	**Attainment**	**Remarks**
1.	A1: Range of resources	Yes	Involves Post Office, All of the people who are involved are digital Documents processing Chapter 1 Art. No: 1.6
2.	A2: Level of interaction	Yes	There are several issues come when we work this Thesis describe in Chapter 1 Art. No: 1.5
3.	A3: Innovation	Yes	Build new dataset and build model for detect Bangla Handwritten City Name describe in Chapter 3 and chapter 4.
4.	A4: Consequences for Society and the Environment	Yes	Involves Schools, All of the people who are involved are mute and deaf Chapter 1 Art. No: 1.4.

5.	A5: Familiarity	Yes	Build a new Bangla Sign Language Dataset and recognition model describe in Chapter 3 and chapter 4.

BIOGRAPHY

of

Md Nur A Alam

Md Nur A Alam, born to Shah Alam and Nur Jahan Khatun, is a B.Sc. student in Computer Science and Engineering at Bangladesh Army University of Science and Technology (BAUST), Saidpur. He completed his Secondary School Certificate with a GPA of 4.82 from Satkhira Government High School, Satkhira, and his Higher Secondary Certificate with a GPA of 4.25 from Satkhira Government College, Satkhira. He is currently maintaining a CGPA of 3.96, with his thesis work ongoing. His academic interests include Machine Learning, Natural Language Processing, and Internet of Things (IoT). He has developed **MediNet-XG: A Lightweight Explainable Multimodal AI for Medicinal Plant Identification from Weed-Infested Areas** as part of his research work. Md. Nur A Alam is skilled in Python and competitive programming. He has demonstrated strong leadership skills as the President of the BAUST Computer Club, BAUST Mathematics Club, and the CSE Society of BAUST. His achievements include becoming a **two-time regional champion (Rangpur Region) in the National Undergraduate Mathematics Olympiad** and a **two-time champion in intra-university programming contests**. He aspires to pursue a career as a researcher focused on real-world industrial applications of Computer Science and Engineering.

Thesis:

- **Title:** MediNet-XG: A Lightweight Explainable Multimodal AI for Medicinal Plant from weed-infested area.

Md Nur A Alam
+8801307-631378
mdnuralam2812@gmail.com
Satkhira Sadar, Satkhira