Preprints.org

Article

Not peer-reviewed version

Resource-Efficient Continual Learning for Medicinal Plant Identification: A Periodic Retraining Approach for Edge-Deployed Agricultural IoT Applications

Trien Phat Tran * , Fareed Ud Din , Ljiljana Brankovic , Cesar Sanin , Susan M. Hester

Posted Date: 29 May 2026

doi: 10.20944/preprints202605.2084.v1

Keywords: agricultural IoT; edge AI; edge intelligence; continual learning; instance-incremental learning; medicinal plant classification; periodic retraining; catastrophic forgetting; OTA model updates; crowdsourced sensing

Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC, OpenAlex.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Disclaimer/Publisher's Note: The statements, opinions, and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions, or products referred to in the content.

Article

Resource-Efficient Continual Learning for Medicinal Plant Identification: A Periodic Retraining Approach for Edge-Deployed Agricultural IoT Applications

Trien Phat Tran [1,*], Fareed Ud Din [1], Ljiljana Brankovic [1], Cesar Sanin [1] and Susan M Hester [2,3]

[1] School of Science and Technology, University of New England, Armidale, NSW 2351, Australia
[2] UNE Business School, University of New England, Armidale, NSW 2351, Australia
[3] CEBRA, The University of Melbourne, Parkville, VIC 3010, Australia
* Correspondence: trien.tran@myune.edu.au

Abstract

Smartphone-based plant identification increasingly serves as the edge tier of agricultural Internet of Things (IoT) systems, where models must adapt to crowdsourced data under bandwidth, memory, and energy constraints. No prior work has systematically investigated continual learning at the scale of thousands of fine-grained medicinal plant species from crowdsourced images, nor how retraining frequency affects the cost–performance trade-off in an IoT model-lifecycle setting. We evaluate three continual learning strategies, naïve fine-tuning, experience replay, and Learning without Forgetting, under periodic retraining schedules (updating every K increments), tested on 2,719 species ($\geq$25 images each) from the Viet Medi Species 2026 dataset (310,647 images; 4,799 species total). All three strategies exhibit negative forgetting (performance improvement rather than degradation) in the instance-incremental setting, with naïve fine-tuning and LwF showing the strongest gains. Periodic retraining with $K = 2$ halves retraining operations while maintaining comparable performance. A baseline MobileNetV2 model achieves 54.07% top-10 accuracy across 2,719 species and has been deployed via TensorFlow Lite (FP16, $\sim$11.5 MB) in the Med Herb Lens Android application. In this regime, naïve fine-tuning offers a favourable cost–performance trade-off and is a reasonable default for instance-incremental agricultural IoT deployments.

Keywords: agricultural IoT; edge AI; edge intelligence; continual learning; instance-incremental learning; medicinal plant classification; periodic retraining; catastrophic forgetting; OTA model updates; crowdsourced sensing

1. Introduction

Recent studies show that deep learning algorithms can correctly identify plant species about 90% of the time [1]; thus far, there are proven applications of deep learning to support species identification in practical agricultural and ecological settings, namely, mobile plant identification systems like Pl@ntNet [2] and smartphone-assisted diagnosis of crop disease [3]. However, accuracy rates vary significantly between datasets and across application scenarios, highlighting the need to evaluate deep learning performance in specific contexts.

One application where accuracy is particularly critical is the identification of medicinal plants. Accurate identification is essential for protecting medicinal plant species, documenting traditional medicines, and providing data for the development of new pharmaceuticals.

The diversity of medicinal plants in Vietnam offers a unique opportunity to apply deep learning-based species identification. Nguyen Quoc and Truong Hoang [4] found that deep learning architectures can achieve very good results on the identification of Vietnamese medicinal plants with an accuracy of 88.26% for Xception using the VNPlant-200 dataset [5] consisting of 20,000 images over

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

200 species of plants. More recent application of Vision Transformers to VNPlant-200 has achieved over 99% accuracy on this dataset [6], demonstrating that contemporary deep learning architectures can achieve expert-like accuracy in the classification of botanic specimens from Vietnam. However, it is important to note that these high accuracies were achieved on a comparatively small classification problem (200 species); classification difficulty increases substantially as the number of visually similar species grows, and top-1 accuracy alone becomes a less informative metric at larger scales. In such contexts, ranked candidate lists and top-k accuracy, where classification is considered correct if the correct species appears among the first k candidates, better reflect practical identification workflows.

Previous research identified a number of challenges in implementing deep learning models in field deployment settings, including rapidly changing environmental conditions, limited connectivity in rural areas, and restrictions in energy and computational capabilities, necessitating adaptive modelling alternatives rather than static solutions [7]. Since training data increments continuously via crowdsourced submissions rather than as a complete dataset, trained models must be created that can continue to gain knowledge without losing any previously acquired knowledge.

Such smartphone-driven, crowdsourced species identification is increasingly understood as a form of agricultural and ecological IoT sensing rather than a standalone mobile-app problem. GPS-enabled smartphones running on-device classifiers act as edge nodes in distributed sensing networks, capturing image, location, and timestamp metadata and feeding it back to centralised systems for aggregation, model retraining, and over-the-air (OTA) redistribution [8,9]. Framing identification systems this way makes explicit the constraints that govern their lifecycle—bandwidth, on-device memory, intermittent connectivity, and energy—and motivates a model-update strategy that respects those constraints. The remainder of this paper accordingly treats Med Herb Lens not as a single mobile app but as the edge tier of an agricultural IoT pipeline whose update cadence is set by the periodic retraining strategy under investigation. We note upfront that the cloud retraining trigger is currently invoked manually rather than by an automated event-driven trigger; the edge and gateway tiers are operationally deployed (Section 6.3.3), and the empirical parameters identified in this study are intended to inform the operational settings of the automated trigger when it is deployed.

The challenges of deploying deep learning models in field settings require models that can adapt continuously. Continual learning — the ability to learn continuously — has thus emerged as an active research area for addressing these deployment challenges [10]. However, a major challenge in continual learning is catastrophic forgetting, as identified by McCloskey and Cohen [11], whereby a neural network rapidly forgets previously learned information when learning from new data. Thus, this has driven research into the different forms of integrating new knowledge (more plasticity) with previously learned knowledge (more stability; i.e. memory) [12].

The most recent comprehensive surveys illustrate that there are three primary ways to mitigate catastrophic forgetting: (1) regularisation-based methods (restricting how weight updates occur); (2) replay-based methods (methods to keep examples of previous data); and (3) architecture-based methods (whereby parts of a model are given to different tasks) [13,14]. Van de Ven and Tolias [15] propose a framework for categorising continual learning, identifying three scenarios: task-incremental, domain-incremental and class-incremental. This framework was subsequently established as foundational to continual learning research [16].

Despite much progress made within the field, continual learning has not yet been fully developed for the application of large-scale fine-grained botanical classification. The majority of continual learning studies have considered learning from benchmark datasets that contain between tens and hundreds of classes [13], whereas many plant identification application systems must identify thousands of visually similar plant species. Moreover, most contemporary work in continual learning is focused on class-incremental scenarios, where the number of classes changes over time, rather than instance-incremental scenarios, where the number of instances changes — a pattern that more closely resembles crowdsourced plant observation data collection.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Given these three broad categories and the instance-incremental setting of our study, we selected three continual learning strategies that span the spectrum of forgetting mitigation. The first, naïve fine-tuning, applies no explicit forgetting mitigation and serves as a lower-bound baseline — the model is simply updated on each new data increment using standard backpropagation. The second, experience replay, is a replay-based approach that maintains a memory buffer of randomly sampled examples from prior increments and interleaves them with new data during training [17]. The third, Learning without Forgetting (LwF), is a regularisation-based method that uses knowledge distillation from a frozen copy of the previous model to preserve learned representations without requiring access to stored exemplars [18]. Architecture-based methods were excluded from this comparison because they are designed primarily for task-incremental settings where new output heads are added over time, and they scale poorly to thousands of classes in an instance-incremental scenario where the class set remains fixed [19,20]. This selection thus allows a direct comparison between no mitigation, data-centric mitigation, and model-centric mitigation under identical periodic retraining conditions.

From a practical perspective, designers of agricultural IoT pipelines for plant identification must make key choices about how often models are retrained and redistributed, and how compute, memory, and bandwidth are budgeted across edge, gateway, and cloud tiers. Online learning approaches that update models continuously demand a constant supply of compute and connectivity. A periodic batch retraining approach—updating the model at fixed intervals (e.g., every two weeks)—offers more predictable resource usage and a cleaner mapping onto OTA update cycles, while still allowing new knowledge to enter the model promptly. The appropriate balance is determined by application requirements, the rate at which observations arrive, the connectivity profile of deployed edge devices, and the available cloud compute budget—none of which have been systematically studied for fine-grained plant identification at scale.

Motivated by the gaps above, this paper investigates one main research question:

RQ: How can continual learning be deployed efficiently and reliably as the model-update component of an agricultural IoT pipeline for large-scale fine-grained medicinal plant identification?

We address this through three specific sub-questions:

RQ1: Among continual learning strategies that span the spectrum of forgetting mitigation—naïve fine-tuning, experience replay, and Learning without Forgetting—which is most effective for instance-incremental learning at the scale of thousands of fine-grained species?

RQ2: How does the periodic retraining frequency (parameterised by the retraining period K) affect the trade-off between model quality and the operational costs of an IoT update pipeline (cloud compute, OTA bandwidth, per-device update energy)?

RQ3: What end-to-end edge–gateway–cloud architecture connects on-device inference, crowd-sourced observation upload, periodic retraining, and OTA model distribution into a coherent model lifecycle under intermittent connectivity, and what practical guidelines—covering strategy selection, retraining frequency, memory budgeting, and connectivity-aware OTA scheduling—follow for adaptive learning systems in agricultural IoT settings?

Table 1 summarises how this work differs from prior studies in medicinal plant classification. While existing studies focus on static, single-round training with small to moderate species counts, our work introduces a periodic retraining framework for continual learning at a scale not previously examined in the medicinal plant literature, coupled with practical deployment as the edge tier of an agricultural IoT pipeline.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Table 1. Comparison of this work with representative prior studies on medicinal plant classification. Key differentiators of our study are highlighted: large-scale species count, continual learning with periodic retraining, and edge IoT deployment.

Study	Species	Images	Learning Paradigm	Architecture	Deployment
Nguyen Quoc & Truong Hoang [4]	200	20K	Static (single-round)	Xception	None
Nhut et al. [6]	200	20K	Static (single-round)	ViT / BEiT	None
Nguyen & Ngo [21]	109	28K+	Static (single-round)	MobileNetV2	None
Dat et al. [22]	29	373	Static (single-round)	EfficientNet + MobileNet	None
Zhang et al. [23]	300	52K+	Static (single-round)	Multiple CNNs	None
This work	**2,719**	**310K+**	**Instance-incremental with periodic retraining**	**MobileNetV2**	**Med Herb Lens (Android, edge IoT)**

Specifically, we make the following contributions:

- **A periodic retraining strategy for resource-efficient continual learning** (Algorithm 1): a scheduled update strategy where models retrain every K increments rather than after each data addition, quantifying the trade-off between computational savings and model performance.
- **A systematic investigation of catastrophic forgetting in large-scale instance-incremental learning:** we evaluate naïve fine-tuning, experience replay, and Learning without Forgetting on 2,719 fine-grained species—a scale and learning scenario not previously examined in the medicinal plant classification literature.
- **The largest continual learning study for Vietnamese medicinal plants:** 2,719 species, substantially exceeding the scale of previous studies (typically 100–300 species), with cross-seed consistency checks.
- **Actionable deployment recommendations for edge-deployed agricultural IoT pipelines:** concrete guidelines on strategy selection, retraining frequency, memory budgeting, and connectivity-aware OTA scheduling, informed by the Med Herb Lens Android application currently deployed on Google Play.
- **An end-to-end agricultural IoT model-lifecycle architecture:** a three-tier edge–gateway–cloud pipeline (Section 6.3) that connects on-device TensorFlow Lite inference, crowdsourced observation upload, periodic retraining, and OTA model distribution, mapping the periodic retraining frequency K directly onto operational bandwidth and energy costs.

The remainder of this paper addresses the main **RQ** above as follows. Section 2 reviews related work on continual learning, plant classification, and edge AI / IoT for ecological sensing. Section 3 describes the dataset, the instance-incremental learning setup, the continual learning strategies evaluated, and the evaluation metrics. Section 4 details the experimental setup and implementation. Section 5 presents and analyses the empirical findings, addressing **RQ1** and **RQ2**. Section 6 interprets the results, presents the IoT system architecture and OTA model lifecycle (addressing **RQ3**), discusses limitations, and outlines future work. Section 7 concludes. Together, these sections establish both the empirical basis and the practical deployment guidance for resource-efficient continual learning in agricultural IoT settings.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Algorithm 1 Periodic Batch Retraining Strategy

Require: Data increments $\{D_1, D_2, \ldots, D_R\}$, retraining period K, model θ
Train θ on base data D_0
Record base evaluation accuracy a_0
for $r = 1$ to R **do**
  if $r \mod K = 0$ **then**
    Train θ on increment D_r using selected strategy
    Evaluate θ on held-out base evaluation set
    Record accuracy a_r and compute forgetting $F_r = a_0 - a_r$
  else
    Skip increment D_r (no model update; data discarded; $a_r = a_{r-1}$)
  end if
end for
return Forgetting curve $\{F_1, F_2, \ldots, F_R\}$

2. Related Work

2.1. Continual Learning Paradigms

Continual (incremental) learning refers to approaches that allow a machine learning model to learn new information from incoming sequential datasets while maintaining access to previously learned information [10]. Continual learning aims to solve the problem of catastrophic forgetting, which occurs when the model is trained with new datasets and catastrophically loses performance on previous datasets [11,12].

Van de Ven and Tolias [15] proposed a framework that divides three types of problems into different categories based on the task structure and inferences made during learning. In *task-incremental learning*, task identity is provided at test time, allowing task-specific output heads. In *domain-incremental learning*, the same classes appear across tasks but with distribution shifts. In *class-incremental learning*, new classes are introduced over time, and the model must distinguish among all classes without task identity information. This taxonomy was formalised in subsequent work demonstrating that scenario selection fundamentally affects method effectiveness [16].

The scenario most relevant to crowdsourced plant identification is *instance-incremental learning*, where new training examples arrive for existing classes rather than introducing new categories. Read et al. [24] provided early analysis distinguishing batch-incremental from instance-incremental processing for data streams, though this setting has received less attention in deep learning research compared to class-incremental scenarios.

Recent comprehensive surveys have systematically organised the continual learning literature. De Lange et al. [13] conducted an extensive experimental comparison of 11 methods and 4 baselines across multiple datasets, which introduced the stability-plasticity trade-off framework. Masana et al. [14] evaluated 13 class-incremental methods investigating domain shift effects. Wang et al. [25] provided theoretical foundations alongside method categorisation, while Zhou et al. [26] offered a detailed analysis of approaches that are data-centric, model-centric, and algorithm-centric. These surveys consistently group catastrophic-forgetting mitigation strategies into three families—regularisation-based, replay-based, and architecture-based—which we summarise in turn below.

Regularisation-based methods limit weight modifications to preserve knowledge important to previous data. Kirkpatrick et al. [27] introduced Elastic Weight Consolidation (EWC), which uses the Fisher information matrix to identify and protect important parameters; Zenke et al. [28] proposed Synaptic Intelligence (SI), which estimates parameter importance online during training. Li and Hoiem [18,29] introduced Learning without Forgetting (LwF), which uses knowledge distillation from a frozen teacher network to preserve previous behaviour without requiring access to prior training data.

Replay-based methods maintain a memory of examples from previous training phases and interleave them with new data. Shin et al. [30] proposed deep generative replay using GANs to synthesise samples from previous task distributions. Rebuffi et al. [31] introduced iCaRL, combining nearest-

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

mean-of-exemplars classification with herding-based exemplar selection and knowledge distillation; notably, the iCaRL paper explicitly states that "finetuning always achieves the worst results" in class-incremental scenarios—a finding that contrasts with our instance-incremental results. Lopez-Paz and Ranzato [32] proposed Gradient Episodic Memory (GEM) and introduced formal metrics for forward and backward transfer evaluation, while van de Ven et al. [33] demonstrated that brain-inspired generative replay effectively prevents catastrophic forgetting across all three scenarios.

Architecture-based methods allocate distinct model components to different tasks; representative examples include Progressive Neural Networks [19] and PackNet [20]. As discussed in Section 1, these methods are designed primarily for task-incremental settings and scale poorly to thousands of fixed classes, so they are not considered further in this study.

2.2. Plant Classification with Deep Learning

Deep learning methods and advancements have enabled an incredible evolution in plant classification; comprehensive literature reviews have described this rapid evolution [1]. A systematic review of literature was conducted by Mulugeta et al. [1], which included 31 studies from 2018 through 2022, applying PRISMA guidelines. In the review, the authors found that 96.8% of studies classified plants based on leaf organs, 83.8% of studies used transfer learning, and 64.5% of studies used CNNs as their primary classification algorithm.

Mohanty et al. [3] completed baseline studies to compare AlexNet and GoogLeNet with transfer learning from the ImageNet dataset and demonstrated >99% accuracy using the PlantVillage dataset (54,306 total images) containing images of 38 different plant species, thereby demonstrating the effective use of transfer learning for plant disease classification and identification of plant species.

Picek et al. [34] benchmarked current state-of-the-art methods for CNNs and Vision Transformers using PlantCLEF datasets; for example, ViT-Large achieved an accuracy of 91.15% on PlantCLEF 2017, and the authors reported that retrieval-based classification approaches outperformed classification approaches by 0.28% - 10.25%.

Specifically regarding Vietnamese medicinal plants, Nguyen Quoc and Truong Hoang [4] completed a comparative study of several architectures (VGG16, ResNet-50, InceptionV3, DenseNet-121, Xception, MobileNet) and reported that Xception achieved the highest accuracy on VNPlant-200 (88.26%). Their VNPlant-200 companion dataset paper [5] is the first publicly available large-scale dataset for Vietnamese medicinal plants with 20,000 images of 200 species.

Additional work on Vietnamese plant classification includes Nguyen and Ngo [21], who achieved 83.9% accuracy with MobileNetV2 on 109 Vietnamese species, and Dat et al. [22], who proposed a multimodel CNN combining EfficientNet and MobileNet with joint multiloss functions for Vietnamese herb identification.

2.3. Edge AI and IoT for Ecological Sensing

Edge AI—the practice of running inference directly on resource-constrained sensing devices rather than sending data to the cloud—has emerged as a central paradigm for ecological and agricultural monitoring under field conditions. Vuilliomenet et al. [9] reviewed 82 studies published between 2017 and 2025 and identified four broad system types for biodiversity monitoring: (1) TinyML on low-power microcontrollers for single-taxon detection, (2) single-board-computer-class edge devices for multi-species classification, (3) distributed edge AI, and (4) cloud AI used for retrospective processing. The smartphone-based deployment investigated in this paper sits in the second of these tiers—edge devices with sufficient compute for multi-species classification but constrained relative to cloud GPUs. Earlier surveys of agricultural deep learning [7] similarly identified unreliable connectivity, limited power, and the need for real-time processing as the dominant practical constraints, motivating model architectures designed for on-device inference.

Lightweight architectures targeting field deployment have been a particularly active area. Guan et al. [35] proposed Dise-Efficient, achieving 99.80% accuracy on PlantVillage with a 13.3 MB model. Zhou et al. [36] developed REM-ShuffleNetV2 (4.40M parameters, 96.72% accuracy on a field crop

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

disease task). Our use of MobileNetV2 with FP16 TensorFlow Lite quantisation (∼11.5 MB) sits within this same lightweight-edge regime.

Closer to the present work, Lourenço et al. [37] surveyed on-device edge learning for IoT data streams, explicitly connecting continual learning to the constraints of TinyML and IoT edge deployments. They emphasise that data architecture (batch versus stream) and network capacity (cloud versus edge) jointly determine which continual-learning algorithms are feasible—an observation that directly motivates our investigation of periodic retraining as a middle ground between continuous online updates and infrequent full retraining. The optimal balance between online learning and batch retraining depends on application requirements, data arrival rates, and available computational budget, and these considerations are precisely what an IoT model-lifecycle framing makes tractable.

2.4. Summary of Research Gaps

The literature reviewed above reveals several key gaps that this work addresses. First, while deep learning has been extensively applied to plant classification, existing studies on medicinal plants are limited to small-scale datasets (typically 80–300 species) and employ static, single-round training without consideration of how models should be updated as new data arrives. Second, the continual learning literature has focused predominantly on class-incremental scenarios using benchmark datasets with tens to hundreds of classes, leaving instance-incremental learning at the scale of thousands of fine-grained species largely unexplored. Third, to our knowledge, no prior work has systematically evaluated the trade-off between retraining frequency and computational cost for plant identification systems, despite this being a critical deployment consideration for resource-constrained edge IoT deployments. Finally, while several Vietnamese medicinal plant datasets exist, none have been coupled with a continual learning evaluation pipeline and a deployed edge-tier IoT application. This study addresses all four gaps through a large-scale empirical investigation of periodic retraining strategies on 2,719 Vietnamese medicinal species, yielding both novel findings (negative forgetting in instance-incremental learning) and practical IoT deployment guidance.

3. Methodology

This section describes our empirical approach to examining periodic retraining methodologies for large-scale classification of Vietnamese medicinal plants. We will discuss our dataset, define the instance-incremental learning environment, explain the continual learning methodologies assessed, and describe our evaluation methods.

3.1. Dataset Description

The dataset we used in this research was the Viet Medi Species 2026 [38], a large-scale multilingual image dataset tailored for medicinal plants documented in Vietnamese traditional medicine. The dataset comprises 310,647 images from 4,799 accepted species spanning four kingdoms (Plantae, Fungi, Chromista, and Bacteria), integrated with the Global Biodiversity Information Facility (GBIF) taxonomic backbone. To the best of our knowledge, this contains one of the largest fine-grained datasets of images of medicinal plants available for research purposes. Table 2 positions the Viet Medi Species 2026 dataset relative to other prominent plant classification datasets, demonstrating its scale advantage.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Table 2. Comparison of Viet Medi Species 2026 with related plant classification datasets.

Dataset	Species	Images	Viet Names
PlantCLEF 2023 [39]	80,000	4M	0%
VNPlant-200 [5]	200	20K	100%
DIMPSAR [40]	80	12.8K	0%
TCMP-300 [23]	300	52K+	0%
Viet Medi Species 2026	**4,799**	**310K+**	**84%**

The dataset was constructed through a systematic five-stage pipeline described in detail by Tran et al. [38]: (i) *taxonomic normalisation* of species lists from the Vietnamese Medicinal Plant Catalogue (*Danh lục cây thuóc Việt Nam*) using the GBIF Species Match API; (ii) *image acquisition* from GBIF's Occurrence API, capped at 130 images per species to limit class imbalance; (iii) *Vietnamese vernacular name curation* for 4,031 species (84% coverage) through over 320 hours of manual research and validation against authoritative sources, with regional naming variations preserved with cultural attribution; (iv) *quality control and filtering*, retaining only species with at least 25 images for the experiments in this study, yielding 2,719 usable classes (>=25 images each); and (v) *metadata integration* following Darwin Core standards. All curation code is openly available on GitHub, and the complete image archive and metadata are deposited on Kaggle, enabling full reproducibility.

Beyond curation, the dataset offers an extensive increase over currently available Vietnamese plant datasets. The VNPlant-200 dataset [5] contains 20,000 images across 200 species, while our dataset scales to 2,719 usable species classes—over 13 times more classes. This scale substantially exceeds prior Vietnamese medicinal plant datasets and is comparable to recent large-scale efforts such as TCMP-300 for Chinese medicinal plants [23], though it remains well below the diversity of PlantCLEF challenges which have addressed up to 80,000 species [39]. Beyond benchmarking deep learning architectures, this scale also serves as a stress test for edge-deployable models: 2,719 fine-grained classes pose a substantially harder learning problem than the 100–300-class settings typical of prior medicinal plant work, while the deployed model must still fit within the size, memory, and energy budget of a smartphone-class edge device.

The dataset exhibits attributes characteristic of real-world crowdsourced biodiversity collections:

- **Long-tail distribution:** Long-tailed distributions of species frequencies; there are many pictures of common species, while there are almost none of rare species near the minimum threshold level.
- **High visual similarity:** High similarity among visual characteristics of many species of the same genus, requiring very fine disambiguation of characteristics for these species.
- **Intra-class variation:** A wide range of image characteristics of the plants due to the differences in the growing conditions, times of day, and quality of images taken in the field shows the diversity of plant data collected in the field.
- **Vietnamese nomenclature:** Scientific and Vietnamese common names (vernacular) for each species are displayed to promote cultural appropriateness of ecological and biodiversity applications.

Table 3 summarises the dataset characteristics used in our experiments.

Table 3. Dataset statistics overview for Viet Medi Species 2026.

Attribute	Value
Total accepted species	4,799
Usable classes ($\geq$25 images)	2,719
Minimum images per class	25
Maximum images per class	130
Image resolution	Variable (resized to 224$\times$224)
Learning paradigm	Instance-incremental (IIL)

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

3.2. Instance-Incremental Learning Scenario

We formulate the learning problem as instance-incremental learning (IIL), where the class set $\mathcal{C}$ of $C = 2{,}719$ classes remains fixed while new training examples arrive over time. This scenario reflects realistic crowdsourced data collection, where users contribute additional images of known species rather than discovering entirely new species.

Following the incremental learning formalisation of van de Ven et al. [16], we define the following notation. Let R denote the total number of increments. Each increment $r \in \{1, 2, \ldots, R\}$ provides a new batch of training data D_r. We partition each class's images into:

- **Base training set:** 20 images per class for initial model training
- **Increment sets:** up to 10 images per class per increment, arriving sequentially over up to 9 increments, depending on the number of images available for that class beyond the base and evaluation sets
- **Held-out evaluation set:** 5 images per class, fixed throughout for measuring forgetting

This partitioning simulates a deployment scenario where an initial model is trained on available data, then periodically updated as new field observations accumulate. The held-out evaluation set enables consistent measurement of catastrophic forgetting across all experimental conditions. Because the dataset filter retains classes with at least 25 images, a class at the minimum threshold contributes its full 20 base and 5 evaluation images but zero increment images; classes with between 26 and 114 images contribute a proportionally truncated sequence of increments (e.g., a class with 50 images supplies the base set, the evaluation set, and approximately 2–3 increments before its image pool is exhausted). The full 9-increment schedule is therefore exercised only by classes with at least 115 images. Long-tail classes are not excluded by this scheme—they continue to contribute to the held-out evaluation set throughout—but they participate in fewer retraining events than head classes, which mirrors the data-availability profile of real crowdsourced collections.

Instead of retraining the model after each increment of data received, we propose *periodic retraining*, which updates the model once every K increments. At increment r where $(r \bmod K) = 0$, the model trains on that increment's data for a fixed number of epochs. Increments where $(r \bmod K) \neq 0$ are skipped—the model does not update, and the data from these skipped increments is not retained for later use. This strategy reflects practical IoT edge scenarios where devices have limited storage and can only retain the most recent data batch rather than accumulating historical increments. The approach represents a compromise between continuous online learning and infrequent batch retraining. Algorithm 1 presents the procedure.

Using $R = 9$ increments and $K = 2$, this periodic retraining model will perform a total of 4–5 updates instead of 9, creating approximately 50% less training work. During skipped increments, the model weights remain frozen, resulting in zero forgetting (no performance change) until the next retraining event.

3.3. Continual Learning Strategies

We evaluate three continual learning strategies designed to address the effects of catastrophic forgetting. All strategies were chosen based upon their frequency of appearance in recent surveys [13,14]:

3.3.1. Naïve Fine-tuning

In this method, the model is trained on new data from each increment using only standard cross-entropy loss functions, without any explicit efforts to mitigate catastrophic forgetting. Let $\mathcal{D}_r = \{(x_i, y_i)\}_{i=1}^{N_r}$ denote the training set available at increment r, where x_i is an input image, y_i is the corresponding class label, and N_r is the number of samples. Let θ denote the set of trainable parameters of the model f_θ. The cross-entropy loss for a single sample (x, y) is defined as:

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

$$\mathcal{L}_{\text{CE}}(f_\theta(x), y) = -\log\left(\frac{\exp(z_y)}{\sum_{j=1}^{C}\exp(z_j)}\right) \tag{1}$$

where $z = f_\theta(x) \in \mathbb{R}^C$ is the logit vector output by the model for C classes, and z_y is the logit corresponding to the true class y. The parameters θ are updated to minimise the average loss over the training set:

$$\mathcal{L}_{\text{naive}}(\theta; \mathcal{D}_r) = \frac{1}{N_r}\sum_{i=1}^{N_r}\mathcal{L}_{\text{CE}}(f_\theta(x_i), y_i) \tag{2}$$

This baseline is critical for establishing whether sophisticated continual learning methods provide benefits over simple fine-tuning in the instance-incremental scenario.

3.3.2. Experience Replay

Experience replay maintains a memory buffer storing representative examples from previous data, following the approach established in foundational work on replay methods [17]. During training on new increments, samples from the replay buffer are interleaved with current data. We use reservoir sampling to maintain 10 examples per class in the buffer. Let $\mathcal{D}_{\text{new}}$ denote the training samples from the current increment and $\mathcal{D}_{\text{buffer}}$ denote the samples stored in the replay buffer. The combined training set is $\mathcal{D}_{\text{combined}} = \mathcal{D}_{\text{new}} \cup \mathcal{D}_{\text{buffer}}$, and the parameters θ are updated to minimise:

$$\mathcal{L}_{\text{replay}}(\theta; \mathcal{D}_{\text{combined}}) = \frac{1}{|\mathcal{D}_{\text{combined}}|}\sum_{(x,y)\in\mathcal{D}_{\text{combined}}}\mathcal{L}_{\text{CE}}(f_\theta(x), y) \tag{3}$$

3.3.3. Learning without Forgetting (LwF)

LwF employs knowledge distillation [41] to preserve the model's predictions on new data from before the update, following Li and Hoiem [18]. A frozen copy of the model with parameters θ_{old} (the teacher) provides soft targets that regularise the updated model with parameters θ (the student). Let $\mathcal{D}_r$ denote the training set at increment r. The combined loss function is:

$$\mathcal{L}_{\text{LwF}}(\theta; \mathcal{D}_r) = \frac{1}{N_r}\sum_{i=1}^{N_r}[(1-\alpha)\mathcal{L}_{\text{CE}}(f_\theta(x_i), y_i) + \alpha\mathcal{L}_{\text{KD}}(\theta, \theta_{\text{old}}; x_i)] \tag{4}$$

where α controls the balance between new learning and knowledge retention (set to 0.7 in our experiments). The knowledge distillation loss $\mathcal{L}_{\text{KD}}$ is defined using temperature-scaled softmax outputs. Let $z = f_\theta(x)$ and $z_{\text{old}} = f_{\theta_{\text{old}}}(x)$ denote the logit vectors produced by the student and teacher models, respectively. The temperature-scaled softmax function σ with temperature T is:

$$\sigma\left(\frac{z}{T}\right)_j = \frac{\exp(z_j/T)}{\sum_{k=1}^{C}\exp(z_k/T)} \tag{5}$$

where the subscript j denotes the j-th element of the resulting probability vector. The knowledge distillation loss is then the Kullback–Leibler (KL) divergence between the teacher's and student's temperature-scaled outputs:

$$\mathcal{L}_{\text{KD}}(\theta, \theta_{\text{old}}; x) = T^2 \cdot \text{KL}\left(\sigma\left(\frac{z_{\text{old}}}{T}\right) \| \sigma\left(\frac{z}{T}\right)\right) \tag{6}$$

The factor T^2 compensates for the reduced magnitude of gradients when using temperature scaling. Temperature T controls the smoothness of the predicted probability distributions: higher values produce softer targets that encode more information about inter-class similarities, while $T = 1$ reduces to standard cross-entropy. We set $T = 2$ following the recommendation of Hinton et al. [41], which has been shown empirically to balance informative soft targets with training stability across

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

diverse classification tasks. The teacher network is refreshed after each retraining step to serve as the reference for the next update.

3.4. Evaluation Metrics

We employ metrics focused on quantifying catastrophic forgetting in our instance-incremental learning setting. Let a_r denote the top-1 accuracy evaluated on the held-out base evaluation set after training on increment r. We use top-1 accuracy here (rather than the top-10 metric reported for the full-data baseline in Section 5.4) because forgetting is most clearly observed when the model's single highest-confidence prediction changes; top-1 is also the standard metric in the continual learning literature on which our experimental design is based [16**?**]. Specifically, a_0 represents the base accuracy after initial training on D_0 (the base training set), and a_r for $r > 0$ represents the accuracy after retraining on increment D_r. Under periodic retraining with period K, a_r is only updated when $r \bmod K = 0$; at other increments, $a_r = a_{r-1}$ (i.e., the model is not retrained and performance remains unchanged).

Base evaluation accuracy. After each retraining step, we evaluate accuracy on the fixed held-out base evaluation set (5 images per class). This quantifies how well a model retains its ability to perform on the original data distribution after it has been updated.

Forgetting. Forgetting F_r at increment r is defined as the decrease in base evaluation accuracy relative to the initial baseline:

$$F_r = a_0 - a_r \tag{7}$$

where a_0 is the base accuracy (after initial training on D_0) and a_r is the accuracy at increment r. Positive values of F_r indicate forgetting (i.e., degradation of performance) while negative values indicate improvement. We report both *mean forgetting* $\bar{F} = \frac{1}{R}\sum_{r=1}^{R} F_r$ (averaged across all increments) and *final forgetting* F_R (after the last increment R).

Computational efficiency. We measure wall-clock training time to characterise the computational requirements of each strategy. Combined with the periodic retraining framework (parameterised by K), this enables analysis of efficiency–performance trade-offs.

4. Experimental Setup

MobileNetV2 is chosen for continual learning because its compact architecture is well suited to edge deployment in agricultural IoT settings. The model has ImageNet pre-trained weights with the final fully connected layer replaced with a classifier that defines all 2,719 species classes.

The images are resized to be 224×224 pixels and normalised based on the ImageNet statistics. We apply light data augmentation during training, consisting of random horizontal flips and minor colour jitter to improve generalisation. Table 4 summarises the training hyperparameters.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Table 4. Training hyperparameters.

Parameter	Value
Backbone architecture	MobileNetV2
Pre-training	ImageNet
Optimiser	Adam
Learning rate	0.0001
Batch size	16
Epochs (base training)	8
Epochs (incremental)	3
Early stopping patience	2 epochs
Input resolution	224×224
IIL-specific parameters	
Base images per class	20
Increment images per class	10
Number of increments	9
Evaluation images per class	5
Strategy-specific parameters	
Replay buffer per class	10
LwF distillation weight (α)	0.7
LwF temperature (T)	2.0
Retraining period (K)	2, 5

Our experiments evaluate periodic retraining with $K = 2$, where the model is retrained every 2nd increment. With 9 total increments, $K = 2$ results in 4–5 retraining operations, representing approximately 50% reduction compared to retraining after every increment. The evaluation of each method (naïve, replay and LwF) was done under the same conditions to ensure fair comparison between methods. To validate the reliability of our results, we performed experiments with multiple random seeds (42 and 123). Experiments were conducted on NVIDIA P100 GPU hardware through Kaggle's computing platform, and training times are reported as wall-clock seconds to enable reproducibility assessment.

5. Results

5.1. Overall Performance Comparison

Table 5 presents the main experimental results comparing the three continual learning strategies under periodic retraining with $K = 2$. Note that absolute accuracy in the constrained incremental setting is not directly comparable to full-data training; our focus is on relative forgetting between strategies under identical conditions. Negative forgetting values indicate performance *improvement* over training, rather than degradation.

Table 5. Performance comparison of continual learning strategies with periodic retraining ($K = 2$), averaged across seeds 42 and 123. Negative forgetting indicates improvement rather than degradation. *Base Acc.* is top-1 accuracy on the held-out base evaluation set and reflects the constrained instance-incremental setting (20 images/class for base training); it serves as the reference point for forgetting calculation. See Section 5.4 for the full-data baseline, where top-10 accuracy reaches 54.07% and is the operationally relevant metric for the deployed ranked-candidate identification system.

Strategy	Base Acc.	Mean Forgetting	Final Forgetting	Runtime (s)
Naïve	7.78%	−6.15%	−10.53%	2,267
Replay	7.35%	−4.65%	−5.55%	2,097
LwF	9.67%	−6.12%	−10.12%	2,325

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Table 6 shows the detailed results for each random seed, demonstrating consistency across experimental runs.

Table 6. Per-seed results for $K = 2$ experiments, showing consistency across random seeds.

Seed	Strategy	Base Acc.	Final Forgetting	Runtime (s)
42	Naïve	8.24%	−9.78%	2,176
	Replay	7.48%	−6.49%	2,087
	LwF	9.09%	−11.27%	2,222
123	Naïve	7.31%	−11.28%	2,358
	Replay	7.21%	−4.61%	2,106
	LwF	10.25%	−8.96%	2,427

Several notable findings emerge from these results:

All strategies show negative forgetting (improvement): Contrary to conventional expectations of catastrophic forgetting, all three strategies demonstrate performance improvement over incremental training. Naïve fine-tuning and LwF show similar strong improvements (final forgetting of −10.53% and −10.12% respectively), while replay shows smaller gains (−5.55%). This counterintuitive result suggests that in the instance-incremental learning scenario with fixed classes, new training data reinforces rather than contradicts existing knowledge.

Replay shows the smallest improvement: Experience replay, despite explicitly storing and rehearsing previous examples, shows the smallest performance gains (−5.55% vs −10% for naïve and LwF). This may be attributed to the limited buffer size (10 samples per class), introducing noise rather than useful regularisation when the fundamental learning dynamic is already positive.

LwF achieves the highest base accuracy: Learning without Forgetting achieves the highest initial accuracy (9.67%) and maintains strong improvement comparable to naïve fine-tuning, suggesting that the knowledge distillation mechanism provides useful regularisation during base training.

Computational costs vary: Replay is fastest (2,097s) due to not requiring teacher model forward passes. LwF is slowest (2,325s) due to dual forward passes for distillation. Naïve falls in between (2,267s).

5.2. Forgetting Curves

Figure 1 shows the evolution of forgetting across incremental updates for each strategy. The staircase pattern visible in the chart reflects the periodic retraining schedule, where model updates occur only at every K-th increment.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

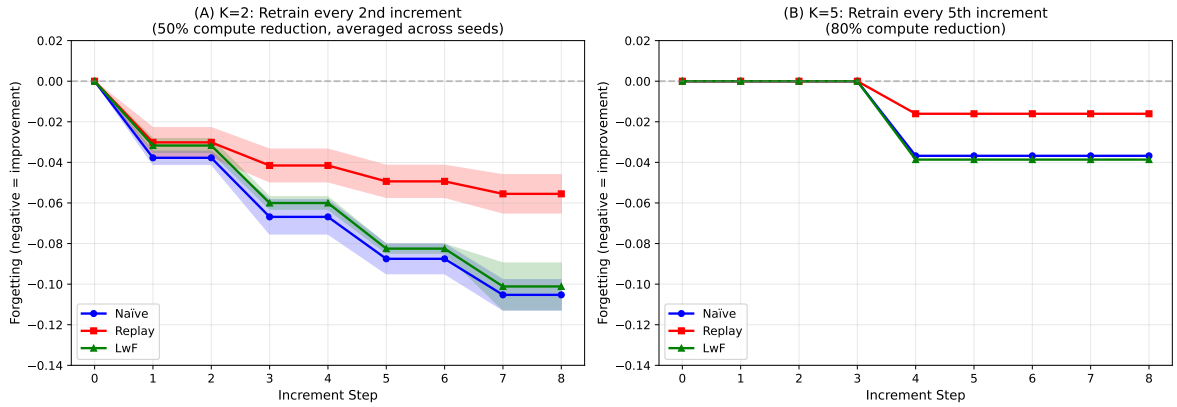

Figure 1. Forgetting curves for three continual learning strategies under periodic retraining with $K = 2$ (model retrained every 2nd increment, achieving 50% compute reduction). Results are averaged across seeds 42 and 123; shaded bands indicate the range between seeds. Negative forgetting values indicate there has been an overall increase in performance instead of a drop. The staircase pattern shows that when there are gaps between retraining, the performance of models does not change. During skipped increments (odd r for $K = 2$), the model maintains frozen weights and exhibits zero forgetting until the next retraining event.

Key observations from the dynamics of forgetting:

Consistent improvement across increments: All of the strategies were providing more and more negative values as we moved through the increments of the training process. This shows that through instance-incremental learning (with fixed classes), the model can continually enhance representation.

Naïve and LwF show the strongest improvement trajectories: Both Naïve fine-tuning and LwF have provided the steepest improvements across the increments (-10% by the last increment). Both of these approaches leverage data in a continuous fashion. Naïve does benefit from working with an unconstrained adaptation on samples, while LwF does benefit from the use of regularisation.

Plateau periods between updates: There are flat periods between increments where no retraining takes place. The flat areas between odd and even increments confirm the fact that changes to the performance are due to whether the model was updated.

Forgetting magnitude depends on training amount, not skip duration: An important observation is that different retraining frequencies show similar forgetting magnitudes after their respective first retraining events. For instance, $K = 2$ at $r = 2$ and $K = 5$ at $r = 5$ both exhibit approximately -4% forgetting despite $K = 5$ skipping training across more increments. This occurs because our periodic retraining strategy trains only on the current increment's data at each retraining event, not on accumulated data from skipped increments. Thus, both $K = 2$ at $r = 2$ and $K = 5$ at $r = 5$ train on a single increment, yielding comparable model updates. The key differentiation lies in computational efficiency: $K = 5$ achieves 80% training cost reduction by skipping 4 increments, while $K = 2$ achieves 50% reduction by skipping 1 increment. This design reflects IoT edge constraints where devices have limited storage and process only the most recent data batch rather than accumulating historical increments.

5.3. Computational Efficiency Analysis

Table 7 compares the computational efficiency of periodic retraining ($K = 2$) against the theoretical baseline of retraining after every increment.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Table 7. Computational efficiency of periodic retraining with $K = 2$. The number of retraining operations is reduced from 9 to 4–5, achieving approximately 50% compute reduction. $K = 1$ (retraining after every increment) runtime could not be measured due to memory constraints.

Configuration	Retrain Ops	Compute Reduction	Naïve Runtime
$K = 1$ (every increment)	9	– (baseline)	– (OOM)
$K = 2$ (every 2nd)	4–5	∼50%	2,267s

The periodic retraining strategy achieves substantial efficiency gains. With $K = 2$, the number of retraining operations is reduced by approximately 50%, with a wall-clock runtime of 2,267 seconds for naïve fine-tuning. This compute reduction is beneficial for field deployments where the availability of computing resources is restricted and scheduling changes to existing models must accommodate other operational requirements.

5.4. Baseline Model Performance

To contextualise the continual learning results and demonstrate practical deployment viability, we trained a baseline MobileNetV2 model on the full dataset without incremental constraints [42]. MobileNetV2 is used consistently across all experiments to ensure fair comparison between continual learning strategies and the baseline. Table 8 presents the test set performance.

Table 8. Baseline MobileNetV2 model performance on held-out test set (no incremental learning).

Metric	Value
Number of Classes	2,719
Test Set Size	60,633
Top-1 Accuracy	26.68%
Top-5 Accuracy	45.75%
Top-10 Accuracy	54.07%
Macro F1-Score	0.2591
Weighted F1-Score	0.2645

The accuracy of the baseline is much greater than that of the continual learning experiments (26.68% versus 7–10% for top-1) due to differences between training on full data versus conducting instance-incremental training. It is important to contextualise these absolute accuracy values against the scale of the classification task. Prior Vietnamese plant studies report 88–99% accuracy on VNPlant-200 [4,6], but that dataset contains only 200 species. Our task spans 2,719 fine-grained species—over 13 times more classes—where many species within the same genus share highly similar visual features. At this scale, the random-chance baseline is approximately 0.037% (1/2,719), meaning the 26.68% top-1 accuracy represents a 720-fold improvement over random guessing.

Moreover, for large-scale species identification, top-k accuracy is the more operationally relevant metric because practical identification workflows present users with a ranked shortlist of candidate species rather than a single prediction. This ranked candidate paradigm is well established in large-scale plant identification systems such as Pl@ntNet [2] and the PlantCLEF evaluation campaigns [39], where mean reciprocal rank and top-k retrieval metrics are standard. Our model's 54.07% top-10 accuracy means that the correct species appears in the candidate list more than half the time, providing substantial practical value for field users who can apply their own domain knowledge to select from the shortlist. This baseline has been successfully implemented to provide identification of Vietnamese medicinal plants within the Med Herb Lens Android app.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

6. Discussion

6.1. Interpretation of Results

The observation of negative forgetting across all strategies can be attributed to several factors specific to the instance-incremental setting. Unlike class-incremental learning, where new output neurons introduce competing gradients, instance-incremental learning keeps the architecture fixed; new instances reinforce existing decision boundaries rather than displacing them. With fixed classes, each increment adds training data for already-known classes, allowing per-class representations to improve rather than competing for finite model capacity. With 2,719 classes, every mini-batch contains diverse gradients, providing implicit multi-task regularisation that promotes class-invariant features and discourages overfitting to recent instances. An important caveat is that the base model was trained on only 20 images per class for 8 epochs, achieving 7–10% top-1 accuracy—far below the 26.68% top-1 accuracy of the full-data baseline. Under these conditions, the model has substantial capacity for improvement from any additional training data, regardless of strategy. The observed negative forgetting may therefore partly reflect continued convergence on a data-starved model rather than a property unique to instance-incremental learning. Both explanations likely contribute; disentangling them would require a control experiment using a fully converged base model, which we identify as an important direction for future work.

The smaller benefit of experience replay relative to the other two strategies, in turn, can be understood through three observations. In an instance-incremental setting, representations are improving naturally as new data arrives. The replay buffer, however, holds a fixed selection of earlier examples that quickly fall behind the model's evolving feature space. Sampling from such a buffer reintroduces stale representations of the same classes and can slow, rather than reinforce, the model's improvement trajectory. With 10 samples per class across 2,719 classes (about 27,000 examples total), the buffer captures only a small fraction of the within-class visual diversity. Doubling the buffer to 20 samples per class triggered out-of-memory errors on the 16 GB GPU—a concrete illustration of the scalability challenges replay-based methods face at fine-grained scale. The replay mechanism adds computational overhead (buffer management, additional training samples), but in the instance-incremental scenario where forgetting is not the primary challenge, this overhead does not translate to performance gains.

Finally, LwF achieves the highest base accuracy among the three strategies, suggesting that the knowledge distillation mechanism provides beneficial regularisation during base training. However, its incremental improvement trajectory is comparable to naïve fine-tuning, while incurring additional computational overhead from dual forward passes for teacher–student distillation. In instance-incremental scenarios where the dominant dynamic is improvement rather than forgetting, this overhead is not clearly justified, and naïve fine-tuning offers a simpler and equally effective alternative for resource-constrained deployments.

6.2. Practical Deployment Guidelines

Based on our experimental findings, we offer the following recommendations for deploying continual learning as the model-update component of edge-deployed agricultural IoT pipelines, with particular attention to the resource constraints typical of field and edge settings:

1. **Consider naïve fine-tuning first:** For instance-incremental learning scenarios where new images arrive for existing species, simple fine-tuning is effective and shows strong improvement over time. Avoid the complexity of replay buffers or distillation unless class-incremental updates (new species) are anticipated. Naïve fine-tuning also avoids the additional memory overhead of maintaining replay buffers or teacher models, which is critical when GPU memory is limited (e.g., 16 GB on our NVIDIA P100).
2. **Use periodic retraining to manage resources:** Retraining every K ($K = 2$ to 5) new data increments reduces the computational cost of retraining by 50–80% while maintaining comparable performance improvements over time relative to retraining after every increment. In cloud and edge environments, this translates into fewer GPU hours per update (lower energy and monetary

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

cost) and proportionally smaller cumulative OTA bandwidth across the device fleet. Updates can be scheduled during off-peak windows or while devices are connected to power and Wi-Fi.

3. **Avoid oversized replay buffers at scale:** For thousands of fine-grained classes, large replay buffers may exceed memory constraints. Our experiments showed that increasing buffer size from 10 to 20 samples per class caused out-of-memory errors with 2,719 classes on 16 GB GPU memory. Developers should budget memory requirements as approximately $B \times C$ images (where B is buffer size per class and C is the number of classes) and verify feasibility before deployment.
4. **Monitor improvement continuously:** Maintain a held-out evaluation set drawn from the original data to track model behaviour as time progresses. In instance-incremental settings, the expectation is performance improvement after retraining rather than forgetting; an automated monitor can use this to gate OTA distribution, releasing a new model only when its evaluation accuracy meets or exceeds the previous version, and avoiding unnecessary cloud retraining cycles when sufficient new data has not yet accumulated.
5. **Match update frequency to deployment realities:** The choice of K determines update cadence and, by extension, OTA bandwidth, per-device update energy, and how quickly new data influences predictions. We discuss specific cadence implications in Section 6.3.3; our experiments with $K = 2$ demonstrate a reasonable balance between responsiveness and resource economy for crowdsourced agricultural IoT settings, particularly in rural deployments where cellular connectivity is intermittent.

6.3. IoT System Architecture and Model Lifecycle

The continual learning experiments described above are intended not as standalone results but as the model-update component of an end-to-end agricultural IoT system. This subsection situates the work within that system, describes the three-tier architecture that connects edge inference to periodic retraining, and discusses the over-the-air (OTA) update pathway and its associated connectivity assumptions.

6.3.1. Three-Tier Edge–Gateway–Cloud Architecture

Figure 2 illustrates the three-tier architecture connecting on-device inference, crowdsourced observation upload, and cloud-side periodic retraining. This structure is consistent with the edge–cloud sensing systems reviewed by Vuilliomenet et al. [9] for biodiversity monitoring and with the on-device continual-learning patterns surveyed by Lourenço et al. [37] for IoT data streams.

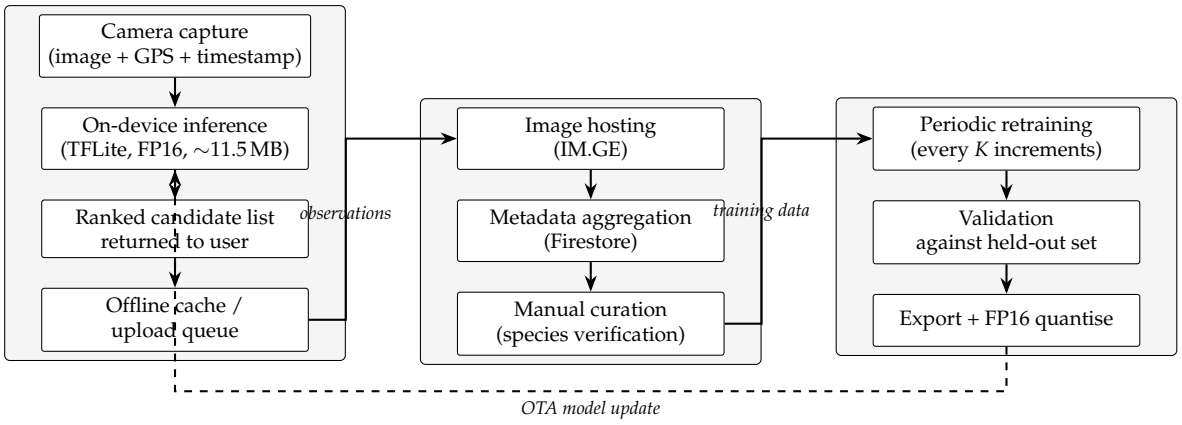

Figure 2. Three-tier edge–gateway–cloud architecture for the agricultural IoT pipeline supporting Med Herb Lens. The edge tier performs on-device inference and queues observations for upload; the gateway tier aggregates uploaded images and metadata; the cloud tier performs periodic retraining (every K increments, per Algorithm 1), validates the new model, and distributes it back to edge devices via OTA update (dashed arrow). The continual learning experiments in Section 5 characterise the cloud-tier retraining step.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

The edge tier corresponds to user smartphones running the Med Herb Lens application. Each device performs on-device inference using a TensorFlow Lite (FP16-quantised, ∼11.5 MB) MobileNetV2 model, returns a ranked candidate list to the user, and—when an image is contributed back—caches the image together with its GPS coordinates and timestamp in a local upload queue. The cache decouples observation capture from network availability, which is essential in rural agricultural settings.

The gateway tier handles aggregation. Image data are uploaded to a hosted image service (currently IM.GE), and structured metadata (image identifier, GPS, timestamp, user-supplied label corrections) are written to a cloud document store (Firestore). At present, a manual curation step verifies species labels before observations enter the training pool; we discuss the implications of this in Section 6.3.3 below.

The cloud tier performs the periodic retraining studied in this paper. At each scheduled retraining event, the new accumulated training data is incorporated into the model using one of the continual learning strategies evaluated in Section 5; the candidate model is validated against the held-out evaluation set; and, on passing validation, it is exported, FP16-quantised, and pushed to edge devices as an OTA update. This pathway turns the algorithmic findings of this paper—that periodic retraining with $K = 2$ preserves accuracy at substantially reduced compute—into operational savings in cloud GPU hours, OTA bandwidth, and edge-device update frequency.

6.3.2. Edge Tier: Med Herb Lens Deployment

The baseline model described in Section 5.4 has been deployed in the Med Herb Lens application, currently published on Google Play. The app is built on the prototype architecture described in our earlier work [43], which combines on-device TensorFlow Lite inference with a Firebase-backed knowledge base and offline-capable contribution queue. FP16 post-training quantisation reduces the deployed model to approximately 11.5 MB while preserving inference accuracy, allowing the model to ship within an Android application bundle of practical size for distribution over mobile networks. The user takes a photograph of a plant; the model returns a ranked list of candidate species with confidence scores. This ranked-candidate paradigm aligns with the 54.07% top-10 accuracy of the underlying baseline, meaning the correct species appears within the top ten candidates more than half the time, and the user applies their own botanical knowledge to make the final selection.

On-device benchmark: To characterise the edge tier empirically, we benchmarked the deployed model on three Android devices spanning low-end to mid-range hardware, representative of the device profile typical of rural Vietnamese deployments. For each device we measured cold-start time (median of five app launches, from icon tap to first prediction), per-image inference latency (median over 50 inferences after a 10-image warmup), peak RAM during inference (Android Studio Profiler live telemetry), and battery cost normalised to a per-100-inferences figure derived from the number of consecutive inferences required to register a 1% battery drop with airplane mode enabled and screen brightness at 50%. Table 9 summarises the results.

Table 9. Edge-tier on-device benchmark for the deployed Med Herb Lens model (MobileNetV2, FP16 TensorFlow Lite, 11.5 MB on disk) on three representative Android devices. Cold-start times are medians of five launches; inference latency is the median over 50 inferences after a 10-image warmup; peak RAM is measured via Android Studio Profiler; battery cost is normalised to a per-100-inferences figure derived from the number of inferences required to register a 1% battery drop with the screen on, airplane mode enabled, and brightness at 50%.

Device	SoC	RAM	Android	Cold start (s)	Inference latency (ms)	Peak RAM (MB)	Battery / 100 inf. (%)
OPPO A60	Snapdragon 680	8 GB	14	6.58	201.4	286.3	0.045
Xiaomi Redmi Note 12	Snapdragon 685	8 GB	15	6.90	883.9	423.8	0.077
Realme C21-Y	Unisoc T610	4 GB	11	9.03	1029.7	327.8	0.040

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Three observations follow from the benchmark: First, on-device inference is operationally feasible across all three devices: even on the lowest-spec hardware (Realme C21-Y with 4 GB RAM and an entry-level Unisoc T610 SoC), inference completes in roughly one second per image, which is acceptable for an interactive identification workflow where the user composes a photograph and waits for a ranked list. Second, peak inference-time RAM stays well below 500 MB on all devices, comfortably within the working-set budget of even 4 GB-class phones. Third, the per-100-inferences battery cost is small in absolute terms—between 0.04% and 0.08%—meaning that a typical day of intensive field use (e.g., 100–200 identifications) consumes well under 1% of a 5000 mAh battery on inference alone, with the remaining energy headroom available for camera capture, display, and observation upload. The faster per-inference time on the OPPO A60 relative to the Xiaomi Redmi Note 12, despite the SoCs being closely matched on synthetic benchmarks, is most plausibly attributable to differences in the default TensorFlow Lite delegate selected by each vendor's OS image (e.g., NNAPI versus CPU-only fallback); we did not control for delegate selection in this study.

Scope of the benchmark: Table 9 characterises the baseline MobileNetV2 model currently deployed in the Med Herb Lens application (Section 5.4). Because the periodic retraining strategy investigated in this work preserves the MobileNetV2 architecture and the FP16 quantisation pipeline, the inference profile reported here applies equally to any model produced by the retraining loop; the strategy modifies *which* weights are deployed, not the architectural envelope within which inference occurs. End-to-end deployment of a periodically retrained model is identified as a near-term engineering step (Section 6.5).

Positioning relative to other edge ML systems: Table 10 situates this work among representative edge-deployed plant and ecological classification systems and on-device continual learning research. The dimensions chosen highlight the combination of properties that distinguishes the present work: large class count, deployed (rather than benchmark-only) edge model, agricultural IoT framing, and inclusion of a continual learning pipeline.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Table 10. Positioning of the present work alongside representative edge-deployed plant/ecological classification systems and on-device continual learning research. "Deployed" indicates whether the system was shipped beyond a research prototype to a real edge target; "CL pipeline" indicates whether the work includes a continual learning component for model updates over time.

System	Domain	Classes	Architecture	Model size	Edge platform	Deployed	CL pipeline
Guan et al. [35]	Plant disease (PlantVillage)	38	Dise-Efficient (custom)	13.3 MB	Lightweight (general)	No	No
Zhou et al. [36]	Field crop disease	8	REM-ShuffleNetV2	4.40 M params	Lightweight (general)	No	No
Mohanty et al. [3]	Plant disease (PlantVillage)	38	AlexNet / GoogLeNet	—	Smartphone (proposed)	No	No
Pl@ntNet [2]	General plant ID	1000s+	CNN (cloud-assisted)	N/A (cloud)	Smartphone	Yes	No
Pellegrini et al. [44]	Object recognition (CORe50)	50	MobileNetV1 + AR1*	—	Smartphone (Android)	Yes (research)	Yes (on-device)
Tran et al. (KES 2025) [43]	Vietnamese medicinal plants (prototype)	7	EfficientNetB0	<10 MB	Smartphone (Android)	Yes	No
This work	**Vietnamese medicinal plants**	**2,719**	**MobileNetV2 (FP16)**	**11.5 MB**	**Smartphone (Android)**	**Yes (Google Play)**	**Yes (periodic retraining)**

The combination of properties in the bottom row—a deployed edge-tier model serving thousands of fine-grained classes, paired with a continual learning pipeline designed for the IoT model lifecycle—is, to the best of our knowledge, not jointly achieved by any prior plant identification or on-device continual learning system. Pellegrini et al. [44] is the closest analogue in deploying continual learning on a real Android device, but their study uses a 50-class object-recognition benchmark and focuses on personalisation rather than agricultural IoT. Pl@ntNet operates at large class scale but is fundamentally a cloud-assisted system rather than an edge-resident model, and it does not employ continual learning for its on-device component.

6.3.3. OTA Model Updates and Connectivity Assumptions

The cloud-to-edge pathway shown in Figure 2 is the operational expression of our periodic retraining strategy. Each retraining event yields one OTA model update; the choice of K therefore directly determines update frequency and, by extension, the bandwidth, energy, and storage costs incurred across the device fleet.

For the deployed FP16 model (∼11.5 MB per update), the choice of K has significant operational implications. With $K = 2$ and an accumulation rate of one increment per fortnight, an OTA push occurs roughly once per month. Aggregated over a notional fleet of N active devices, this scales to bandwidth and energy budgets that are linear in N and inversely proportional to K. Periodic retraining with larger K values would translate into proportionally smaller cumulative OTA bandwidth and per-device energy expenditure for model updates—a non-trivial concern in rural deployments where users may be on metered cellular connections.

The architecture is also explicitly designed for intermittent connectivity. The edge-tier upload queue (Figure 2) caches observations when the device is offline and synchronises them once a network

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

connection is available. OTA updates are similarly opportunistic: on the receiving side, the application checks for a new model on launch when a network is reachable and downloads it preferentially over Wi-Fi to avoid imposing cellular costs on the user. These design choices follow established patterns for IoT edge devices operating under unreliable connectivity [37] and reflect the rural-deployment assumptions identified by earlier surveys of agricultural deep learning [7].

Field deployment also introduces data-quality considerations distinct from algorithmic accuracy. Edge-captured images vary significantly under natural lighting and weather, and contributing users may photograph plants at different growth stages, from varying angles, and at varying spatial scales (whole plant, leaf cluster, individual leaf). Geotagging quality depends on the device's location subsystem, and timestamp metadata can be coarsened by user privacy settings. The periodic retraining loop is precisely the mechanism by which the model adapts to these field-realistic conditions over time, rather than remaining frozen on the comparatively uniform GBIF reference imagery used for the initial baseline.

It is important to be precise about the current state of automation. The pipeline shown in Figure 2 is fully designed and the edge and gateway tiers are operationally deployed: the application is published on Google Play, image uploads to IM.GE function end-to-end, and metadata aggregation in Firestore is active. The cloud retraining step itself, however, is currently performed manually using the Kaggle notebook released alongside this paper [42]—contributed images are downloaded from IM.GE, manually verified, integrated into the training set, and a retrained model is exported and uploaded for OTA distribution. The application has not yet been promoted to the Vietnamese user community, so observation upload volume is presently low, and full automation of the retraining trigger has not yet been required. Closing this gap—an automated retraining trigger that fires every K accumulated, verified increments—is a near-term engineering objective that the present empirical study is intended to support.

Beyond direct user identification, the architecture also supports several broader ecological workflows. Extension workers and field surveyors can use the application during biodiversity assessments and contribute locally collected specimens back into the training pipeline; conservation programmes can systematically catalogue medicinal plant populations through aggregated user observations; and herbal supply chains can use it as a secondary check to reduce misidentification at collection points. In each case, verified observations feed back into the periodic retraining loop described in this study, gradually improving model accuracy over time.

6.4. Limitations

This study has several limitations that should be considered when interpreting results, which we group below by methodological scope, computational constraints, and the maturity of the deployed pipeline.

Scope and methodology: Our experiments consider only instance-incremental learning where the class set is fixed; class-incremental scenarios (adding new species over time) present different challenges and may show different patterns, including actual catastrophic forgetting. To maintain consistency across all experiments, MobileNetV2 was used as the sole base architecture, and performance differences in alternative architectures (e.g., Vision Transformers or larger backbones) may yield different results. The experiments also use a partitioned static dataset to simulate incremental arrival, whereas real-world crowdsourced datasets may contain different temporal patterns and distribution shifts in addition to variation in image quality. Additionally, our periodic retraining implementation trains only on the current increment at each retraining event rather than accumulating skipped increments; an alternative cumulative retraining strategy (training on all pending increments when $r \bmod K = 0$) would show different forgetting patterns and may warrant investigation in future work.

Computational constraints: Several of our design choices were driven by the 16 GB VRAM ceiling of the NVIDIA P100 GPU used for experiments. We were unable to test $K = 1$ (retraining every increment), which would have served as the natural baseline for compute reduction claims; with $K = 1$, the cumulative training data across all 9 increments, combined with replay buffer storage and

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

teacher model copies for LwF, exceeded available memory. The reported 50% compute reduction for $K = 2$ is therefore based on the number of retraining operations (4–5 versus 9 for $K = 1$) rather than on measured runtime comparisons against a $K = 1$ baseline. Similar memory pressure prevented systematic testing of larger replay buffers (e.g., 20 samples per class), which itself serves as a concrete example of the scalability challenges replay-based approaches face at fine-grained scales of thousands of classes. We also ran only two random seeds (42 and 123); a third seed (456) produced an out-of-memory error. Although the two valid seeds yielded similar results, adding a third seed would strengthen the statistical claim.

Interpretation of accuracy: The difference in accuracy between the continual learning experiments (7–10% top-1 accuracy) and the baseline model (26.68% top-1 accuracy) demonstrates the constrained nature of instance-incremental learning when only a limited number of images per class are available at each increment. However, these figures should be interpreted in light of the 2,719-class scale (where random chance is 0.037%) and the ranked candidate identification paradigm: the baseline model's 54.07% top-10 accuracy indicates that the correct species appears in the shortlist more than half the time, which is the operationally relevant metric for deployed plant identification systems.

Pipeline maturity and benchmark coverage: As described in Section 6.3.3, the cloud-tier retraining step is currently triggered manually through a Kaggle notebook rather than by an automated event-driven trigger over accumulated edge contributions. The architecture supports full automation, and the periodic retraining strategy evaluated here is the algorithmic basis for that automation, but the end-to-end loop has not yet been exercised under real user load; the findings of this study are intended to inform the operational parameters (K, strategy, validation thresholds) of the automated trigger when it is deployed. Similarly, the on-device benchmark in Section 6.3.2 (Table 9) covers three Android devices spanning low-end to mid-range hardware. While this range is representative of devices typical in rural Vietnamese deployments, it does not include flagship devices, tablets, or non-Android edge platforms; performance on those targets, and the influence of vendor-specific TensorFlow Lite delegate selection, remain to be characterised systematically.

6.5. Future Directions

This work opens several directions for future research.

- **Convergence control experiment:** Investigate whether negative forgetting persists when the base model is trained to convergence on the full available dataset, to disentangle the effects of data reinforcement from continued learning on a data-starved model.
- **Class-incremental scenarios:** Extend the evaluation to class-incremental learning, where new species are added to the system over time and catastrophic forgetting is expected to be more pronounced.
- **Federated continual learning:** Investigate federated continual learning (FCL) for the agricultural IoT setting. The architecture in Figure 2 is currently centralised: contributed images and metadata leave the edge device. A federated formulation would instead train model updates locally on each edge device or on a regional gateway and aggregate only model parameters at the cloud tier, preserving the privacy of contributed images and reducing OTA-direction bandwidth at the cost of higher per-device energy and compute. Recent surveys situate FCL within the IoT edge-AI research frontier [37], and the periodic retraining cadence studied here translates naturally into a federated aggregation cadence.
- **Uncertainty-aware curation:** Integrate uncertainty quantification into the process when determining which images should undergo expert verification prior to being added to training datasets.
- **Multi-modal learning:** Extend the model to incorporate textual descriptions, geographic metadata, and seasonal information alongside images.
- **Memory-efficient replay at scale:** Build replay strategies that scale to thousands of classes without exceeding device memory boundaries.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

- **Systematic study of K:** Systematically investigate the periodic retraining period, including memory-optimised implementations that enable $K = 1$ benchmarking.
- **Wider edge characterisation:** Extend the edge-inference characterisation in Section 6.3.2 to a wider device range (including flagships, tablets, and non-Android targets), and use power-instrumented measurement (rather than battery-percentage proxy) to isolate inference energy from screen and SoC idle costs. Systematic study of vendor-specific TensorFlow Lite delegate behaviour (NNAPI, GPU, Hexagon DSP) is also a natural extension, given the substantial cross-device latency variation observed in this study.

7. Conclusion

This study addressed how continual learning can be deployed efficiently and reliably as the model-update component of an agricultural IoT pipeline for large-scale fine-grained medicinal plant identification. Through systematic experiments on a dataset of 2,719 species, with cross-seed consistency checks and multiple retraining frequencies, we compared naïve fine-tuning, experience replay, and Learning without Forgetting under a periodic retraining strategy designed for computational efficiency, and situated the findings within an end-to-end edge–gateway–cloud architecture for crowdsourced agricultural IoT.

Our key findings indicate that instance-incremental learning exhibits negative forgetting (performance improvement over time) rather than catastrophic forgetting, with naïve fine-tuning and LwF showing the strongest improvement trajectories (final forgetting of approximately -10%). This result suggests that explicit forgetting mitigation techniques may be unnecessary when new data reinforces existing class representations, although the limited initial training data (20 images per class) may also contribute to the observed improvement pattern.

Furthermore, we demonstrated that periodic retraining with $K = 2$ reduces the number of retraining operations by approximately 50% while maintaining competitive performance. In an agricultural IoT model-lifecycle setting, this compute reduction translates directly into proportionally smaller cloud GPU usage, OTA bandwidth, and per-device update energy. A baseline MobileNetV2 model trained on the full dataset achieves 54.07% top-10 accuracy across 2,719 species—meaning the correct species appears in the ranked candidate list more than half the time—with a corresponding top-1 accuracy of 26.68% (a 720-fold improvement over random chance at this scale), demonstrating practical viability for edge deployment through ranked candidate identification. This baseline has been successfully deployed via TensorFlow Lite (FP16, $\sim$11.5 MB) in the Med Herb Lens Android application, which we describe as the edge tier of a three-tier edge–gateway–cloud IoT architecture (Section 6.3); the cloud tier of this architecture is where the periodic retraining strategy investigated here operates.

By addressing the challenges of scalable continual learning in fine-grained plant classification within a deployable IoT pipeline, this work contributes both methodological understanding and actionable deployment guidance—including concrete recommendations on strategy selection, retraining frequency, memory budgeting, and connectivity-aware OTA scheduling—for adaptive machine learning systems in agricultural and field IoT settings. Future work will extend these approaches to class-incremental scenarios where catastrophic forgetting is expected to be more pronounced, to federated continual learning configurations that better suit privacy- and bandwidth-constrained edge deployments, and to multi-modal plant identification systems.

Author Contributions: Conceptualisation, T.P.T., F.U.D., L.B., C.S. and S.M.H.; methodology, T.P.T.; software, T.P.T.; validation, T.P.T.; formal analysis, T.P.T.; investigation, T.P.T.; resources, T.P.T.; data curation, T.P.T.; writing—original draft preparation, T.P.T.; writing—review and editing, T.P.T., F.U.D., L.B., C.S. and S.M.H.; visualisation, T.P.T.; supervision, F.U.D., L.B., C.S. and S.M.H.; project administration, F.U.D. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

Data Availability Statement: The Viet Medi Species 2026 dataset and associated curation code described in Section 3.1 are publicly available. The complete image archive and metadata are deposited on Kaggle (https://www.kaggle.com/datasets/trientran/viet-medi-species-2026), and the curation code is hosted on GitHub (https://github.com/trientran/oriental-herb-lens). The Kaggle code notebooks used to train and evaluate the trained models reported in Section 5 are available at https://www.kaggle.com/code/trientran/continual-learning-vietnamese-medicinal-plants/ and https://www.kaggle.com/code/trientran/species-level-2026. The Med Herb Lens Android application is publicly available on Google Play (https://play.google.com/store/apps/details?id=com.uri.lee.dl&hl=en).

Acknowledgments: The authors thank the Global Biodiversity Information Facility (GBIF) and data providers for granting free access to biodiversity occurrence records and imagery utilised in this research. All computational experiments were carried out on the GPU infrastructure of Kaggle (www.kaggle.com). The authors thank the translators and curators who supplied the current names for Vietnamese medicinal plants referenced in the vernacular name annotations for this dataset. During the preparation of this manuscript, the authors used Claude (Anthropic), Grammarly, and QuillBot for the purposes of brainstorming, proofreading, and language refinement. The authors have reviewed and edited the output and take full responsibility for the content of this publication.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Mulugeta, A.K.; Sharma, D.P.; Mesfin, A.H. Deep Learning for Medicinal Plant Species Classification and Recognition: A Systematic Review. *Front. Plant Sci.* **2024**, *14*, 1286088.
2. Affouard, A.; Goëau, H.; Bonnet, P.; Lombardo, J.-C.; Joly, A. Pl@ntNet App in the Era of Deep Learning. In Proceedings of the ICLR 2017 Workshop Track, Toulon, France, 24–26 April 2017.
3. Mohanty, S.P.; Hughes, D.P.; Salathé, M. Using Deep Learning for Image-Based Plant Disease Detection. *Front. Plant Sci.* **2016**, *7*, 1419.
4. Nguyen Quoc, T.; Truong Hoang, V. Medicinal Plant Identification in the Wild by Using CNN. In Proceedings of the International Conference on Information and Communication Technology Convergence (ICTC), Jeju Island, Republic of Korea, 21–23 October 2020; pp. 25–29.
5. Nguyen Quoc, T.; Truong Hoang, V. VNPlant-200: A Public and Large-Scale Vietnamese Medicinal Plant Image Dataset. *Lect. Notes Netw. Syst.* **2021**, *136*, 406–411.
6. Nhut, D.T.N.; Tan, T.D.; Quoc, T.N.; Hoang, V.T. Medicinal Plant Recognition Based on Vision Transformer and BEiT. *Procedia Comput. Sci.* **2024**, *234*, 188–195.
7. Kamilaris, A.; Prenafeta-Boldú, F.X. Deep Learning in Agriculture: A Survey. *Comput. Electron. Agric.* **2018**, *147*, 70–90.
8. Atsumi, K.; Nishida, Y.; Ushio, M.; Nishi, H.; Genroku, T.; Fujiki, S. Boosting Biodiversity Monitoring Using Smartphone-Driven, Rapidly Accumulating Community-Sourced Data. *eLife* **2024**, *13*, RP93694. https://doi.org/10.7554/eLife.93694
9. Vuilliomenet, A.; Jones, K.E.; Wilson, D. Future of Edge AI in Biodiversity Monitoring. *arXiv* **2026**, arXiv:2602.13496. https://doi.org/10.48550/arXiv.2602.13496
10. Parisi, G.I.; Kemker, R.; Part, J.L.; Kanan, C.; Wermter, S. Continual Lifelong Learning with Neural Networks: A Review. *Neural Netw.* **2019**, *113*, 54–71.
11. McCloskey, M.; Cohen, N.J. Catastrophic Interference in Connectionist Networks: The Sequential Learning Problem. *Psychol. Learn. Motiv.* **1989**, *24*, 109–165.
12. French, R.M. Catastrophic Forgetting in Connectionist Networks. *Trends Cogn. Sci.* **1999**, *3*, 128–135.
13. De Lange, M.; Aljundi, R.; Masana, M.; Parisot, S.; Jia, X.; Leonardis, A.; Slabaugh, G.; Tuytelaars, T. A Continual Learning Survey: Defying Forgetting in Classification Tasks. *IEEE Trans. Pattern Anal. Mach. Intell.* **2022**, *44*, 3366–3385.
14. Masana, M.; Liu, X.; Twardowski, B.; Menta, M.; Bagdanov, A.D.; van de Weijer, J. Class-Incremental Learning: Survey and Performance Evaluation on Image Classification. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 5513–5533.
15. van de Ven, G.M.; Tolias, A.S. Three Scenarios for Continual Learning. *arXiv* **2019**, arXiv:1904.07734.
16. van de Ven, G.M.; Tuytelaars, T.; Tolias, A.S. Three Types of Incremental Learning. *Nat. Mach. Intell.* **2022**, *4*, 1185–1197.
17. Rolnick, D.; Ahuja, A.; Schwarz, J.; Lillicrap, T.; Wayne, G. Experience Replay for Continual Learning. *Adv. Neural Inf. Process. Syst.* **2019**, *32*, 348–358.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

18. Li, Z.; Hoiem, D. Learning without Forgetting. *IEEE Trans. Pattern Anal. Mach. Intell.* **2018**, *40*, 2935–2947.
19. Rusu, A.A.; Rabinowitz, N.C.; Desjardins, G.; Soyer, H.; Kirkpatrick, J.; Kavukcuoglu, K.; Pascanu, R.; Hadsell, R. Progressive Neural Networks. *arXiv* **2016**, arXiv:1606.04671.
20. Mallya, A.; Lazebnik, S. PackNet: Adding Multiple Tasks to a Single Network by Iterative Pruning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Salt Lake City, UT, USA, 18–23 June 2018; pp. 7765–7773.
21. Nguyen, V.H.; Ngo, L.H.H. Automatic Plant Image Identification of Vietnamese Species Using Deep Learning Models. *Int. J. Eng. Trends Technol.* **2020**, *68*, 25–31.
22. Dat, T.T.; Le Thien Vu, P.C.; Truong, N.N.; Anh Dang, L.T.; Thanh Sang, V.N.; Bao, P.T. Leaf Recognition Based on Joint Learning Multiloss of Multimodel Convolutional Neural Networks: A Testing for Vietnamese Herb. *Comput. Intell. Neurosci.* **2021**, *2021*, 5032359.
23. Zhang, Y.; Sun, W.; Yang, C.; et al. TCMP-300: A Comprehensive Traditional Chinese Medicinal Plant Dataset for Plant Recognition. *Sci. Data* **2025**, *12*, 1166.
24. Read, J.; Bifet, A.; Pfahringer, B.; Holmes, G. Batch-Incremental versus Instance-Incremental Learning in Dynamic and Evolving Data. In Proceedings of the International Symposium on Intelligent Data Analysis (IDA), Helsinki, Finland, 25–27 October 2012; pp. 313–323.
25. Wang, L.; Zhang, X.; Su, H.; Zhu, J. A Comprehensive Survey of Continual Learning: Theory, Method and Application. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 5362–5383.
26. Zhou, D.-W.; Wang, Q.-W.; Qi, Z.-H.; Ye, H.-J.; Zhan, D.-C.; Liu, Z. Class-Incremental Learning: A Survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2024**, *46*, 9851–9873.
27. Kirkpatrick, J.; Pascanu, R.; Rabinowitz, N.; Veness, J.; Desjardins, G.; Rusu, A.A.; Milan, K.; Quan, J.; Ramalho, T.; Grabska-Barwinska, A.; et al. Overcoming Catastrophic Forgetting in Neural Networks. *Proc. Natl. Acad. Sci. USA* **2017**, *114*, 3521–3526.
28. Zenke, F.; Poole, B.; Ganguli, S. Continual Learning through Synaptic Intelligence. In Proceedings of the International Conference on Machine Learning (ICML), Sydney, Australia, 6–11 August 2017; pp. 3987–3995.
29. Li, Z.; Hoiem, D. Learning without Forgetting. In Proceedings of the European Conference on Computer Vision (ECCV), Amsterdam, The Netherlands, 11–14 October 2016; pp. 614–629.
30. Shin, H.; Lee, J.K.; Kim, J.; Kim, J. Continual Learning with Deep Generative Replay. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 2990–2999.
31. Rebuffi, S.-A.; Kolesnikov, A.; Sperl, G.; Lampert, C.H. iCaRL: Incremental Classifier and Representation Learning. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, USA, 21–26 July 2017; pp. 5533–5542.
32. Lopez-Paz, D.; Ranzato, M. Gradient Episodic Memory for Continual Learning. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 6467–6476.
33. van de Ven, G.M.; Siegelmann, H.T.; Tolias, A.S. Brain-Inspired Replay for Continual Learning with Artificial Neural Networks. *Nat. Commun.* **2020**, *11*, 4069.
34. Picek, L.; Šulc, M.; Patel, Y.; Matas, J. Plant Recognition by AI: Deep Neural Nets, Transformers, and kNN in Deep Embeddings. *Front. Plant Sci.* **2022**, *13*, 787527.
35. Guan, H.; Fu, C.; Zhang, G.; Li, K.; Wang, P.; Zhu, Z. A Lightweight Model for Efficient Identification of Plant Diseases and Pests Based on Deep Learning. *Front. Plant Sci.* **2023**, *14*, 1227011.
36. Zhou, H.; Chen, J.; Niu, X.; Dai, Z.; Qin, L.; Ma, L.; Li, J.; Su, Y.; Wu, Q. Identification of Leaf Diseases in Field Crops Based on Improved ShuffleNetV2. *Front. Plant Sci.* **2024**, *15*, 1342123.
37. Lourenço, A.; Rodrigo, J.; Gama, J.; Marreiros, G. On-Device Edge Learning for IoT Data Streams: A Survey. *arXiv* **2025**, arXiv:2502.17788. https://doi.org/10.48550/arXiv.2502.17788
38. Tran, T.P.; Din, F.U.; Brankovic, L.; Sanin, C.; Hester, S.M. Viet Medi Species 2026: A Web-Accessible Multilingual Dataset for Vietnamese Medicinal Biodiversity. In Proceedings of the Companion Proceedings of the ACM Web Conference 2026 (WWW Companion '26), Dubai, United Arab Emirates, 29 June–3 July 2026; ACM: New York, NY, USA, 2026; 8 pages. https://doi.org/10.1145/3774905.3795608. (Accepted; to appear.)
39. Goëau, H.; Bonnet, P.; Joly, A. Overview of PlantCLEF 2023: Image-Based Plant Identification at Global Scale. In Proceedings of the CLEF 2023 Working Notes, Thessaloniki, Greece, 18–21 September 2023.
40. Pushpa, B.R.; Shobha Rani, N. DIMPSAR: Dataset for Indian Medicinal Plant Species Analysis and Recognition. *Data Brief* **2023**, *49*, 109388.
41. Hinton, G.; Vinyals, O.; Dean, J. Distilling the Knowledge in a Neural Network. *arXiv* **2015**, arXiv:1503.02531.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.

Preprints.org (www.preprints.org) | NOT PEER-REVIEWED | Posted: 29 May 2026 doi:10.20944/preprints202605.2084.v1

42. Tran, T. Species-level-2026: Kaggle Code Notebook. Kaggle, 2026. Available online: https://www.kaggle.com/code/trientran/species-level-2026 (accessed on 10 May 2026).
43. Tran, T.P.; Din, F.U.; Brankovic, L.; Sanin, C.; Hester, S.M. Med Herb Lens: A Prototype AI App for Medicinal Plant Identification. *Procedia Comput. Sci.* **2025**, *270*, 2603–2612. https://doi.org/10.1016/j.procs.2025.09.382
44. Pellegrini, L.; Lomonaco, V.; Graffieti, G.; Maltoni, D. Continual Learning at the Edge: Real-Time Training on Smartphone Devices. In Proceedings of the 29th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (ESANN 2021), Bruges, Belgium, 6–8 October 2021; pp. 23–28. https://doi.org/10.14428/esann/2021.ES2021-136

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.

© 2026 by the author(s). Distributed under a Creative Commons CC BY license.