Plant Disease Prediction Using CNN-IDML

A Project Report Submitted

in Partial Fulfillment of the Requirements

for the Degree of

MASTER OF SCIENCE

in

Mathematics and Computing

by

Suvramoy Pal & Chandresh Joshi

(Roll Number: 232123028 & 232123007)

Submitted to the

DEPARTMENT OF MATHEMATICS

INDIAN INSTITUTE OF TECHNOLOGY GUWAHATI

GUWAHATI - 781039, INDIA

April 2025

CERTIFICATE

This is to certify that the work contained in this project report entitled **"Plant Disease Prediction Using CNN-IDML"** submitted by **Suvramoy Pal** (Roll Number: 232123028) and **Chandresh Joshi** (Roll Number: 232123007), to the Department of Mathematics, Indian Institute of Technology Guwahati, towards the partial fulfillment of the Master of Science in Mathematics and Computing, has been carried out by them under my supervision.

It is also certified that this report is a survey work based on the references in the bibliography.

Turnitin Similarity: 8%

Guwahati 781039
April 2025

Dr. Rajen Kumar Sinha
Project Supervisor

ABSTRACT

The primary objective of this project is to develop an efficient and robust system for plant disease prediction using deep learning techniques, addressing the critical challenge of plant disease in agricultural regions. The study utilizes a dataset comprising 21,367 labeled images across five classes, including four disease categories and a healthy class. Two models are proposed: a baseline Convolutional Neural Network (CNN) based on ResNet-50 with Cross-Entropy Loss, and an advanced Introspective Deep Metric Learning (IDML) model incorporating Proxy-Anchor Loss to handle image uncertainty. Data augmentation techniques, including resizing, flipping, rotation, color jitter, and normalization, along with a WeightedRandomSampler, were employed to mitigate class imbalance. Experimental results demonstrate that the IDML model achieves a validation accuracy of 84.46%, surpassing the baseline CNN's 79.30%, with additional metrics (R@1: 84.46%, NMI: 0.4743, RP: 94.49%, MC@R: 90.11%) highlighting its efficacy in retrieval tasks. The model's ability to manage real-world image variability positions it as a promising tool for farmers, enabling rapid disease diagnosis via mobile devices. Future work may explore enhanced architectures or expanded datasets to improve clustering performance.

ACKNOWLEDGEMENT

I, Suvramoy Pal and Chandresh Joshi, would like to express our sincere gratitude to our project supervisor, Prof. Rajen Kumar Sinha, for his invaluable guidance and support throughout this project.

Finally, we are grateful to our families and friends for their constant support and encouragement.

Suvramoy Pal

Roll Number: 232123028

Chandresh Joshi

Roll Number: 232123007

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 What is Plant Disease Prediction?

Plant disease prediction is the method of detecting and classifying plant diseases based on visual symptoms, generally from the images of their leaves. The task is to automate the identification of diseases, e.g., fungal infections, bacterial blights, or viral wilts, by identifying patterns such as spots, discoloration, or wilting in leaf images. In contrast to manual inspection, which relies on human knowledge, automated prediction employs sophisticated technologies to annotate images with precise disease categories or ascertain the plant's health, allowing for faster and more uniform diagnosis.

1.1.1 Why is Plant Disease Prediction Important?

Plant diseases are a serious threat to agriculture, lowering crop yields by as much as 30% and resulting in considerable financial losses and food insecurity, particularly in farming-based areas. Conventional detection systems, whereby farmers or specialists visually inspect plants, are time-consuming, resource-intensive, and miss early-stage diseases that are not yet visible to the

naked eye. This lag may exacerbate crop damage, thus making intervention timely.

Machine learning plant disease prediction resolves these problems by providing an efficient, precise, and scalable solution. It provides early detection, enabling farmers to treat plants before diseases affect large areas. These systems are also beneficial in remote regions with poor expert access, giving farmers the ability to monitor the health of their plants effectively. In addition to agriculture, this technology is useful in ecological research by monitoring plant health in wild ecosystems.

1.1.2 Our Approach

In this project, we create a plant disease prediction system with Convolutional Neural Networks (CNNs) and Introspective Deep Metric Learning (IDML). CNNs are great at handling images by drawing features—such as edges, textures, or sophisticated disease patterns—directly from original pixel values. But certain diseases appear similar, and image quality may differ, making it difficult for CNNs to handle on its own. To enhance precision, we combine CNNs with IDML, which generates a distinct "fingerprint" for every disease, allowing them to be distinguished more easily, and includes a self-check to guarantee predictions are trustworthy.

We train and validate our system on a plant leaf image dataset containing thousands of samples of various disease types and healthy leaves. The dataset consists of varied conditions—such as lighting or angles—to simulate natural scenarios. Every image is tagged with the disease type, serving as the base for our model to learn and make predictions.

1.1.3 Objectives and Challenges

Our overall objective is to develop an accurate and reliable plant disease forecasting system based on CNNs and IDML. We aim to demonstrate that the pairing can deal with difficult cases, such as diseases with similar symptoms, more effectively than conventional techniques. We also intend to render the system transparent, so that users can rely on its forecasts by knowing how it makes its decisions.

Main challenges are:

- Similar Symptoms: Certain diseases resemble each other, threatening to get confused.
- Data Limits: Rare diseases may lack enough images for training.
- Complexity: Processing large image sets demands significant computing power.

To tackle these, we'll refine our model, use tricks like data augmentation to stretch our dataset, and tap into pre-trained models to boost performance. This introduction lays the groundwork for exploring our methods, implementation, and results in later sections, offering a full look at solving plant disease prediction.

Chapter 2

Neural Networks for Plant Disease Detection

2.1 Understanding Artificial Neural Networks

Artificial Neural Networks, or ANNs, are computing systems designed to mimic how the human brain processes information. They consist of layers of nodes, often called neurons, connected in a network. There is an input layer to take in data, one or more hidden layers to process it, and an output layer to give the final result. Each neuron receives inputs, scales them by weights, sums them together, and feeds them through an activation function—such as sigmoid or ReLU—to determine what to output. When training the network, it adjusts these weights to be better predictors using a process called backpropagation.

For forecasting plant diseases, ANNs could be used to check leaf image features such as color patterns or texture features to determine whether a plant is ill. However, this is not a good strategy. These features have to be manually extracted, which is quite labor-intensive and may overlook serious

disease signs, like light spots or irregular shapes. Also, if you directly feed raw image pixels into an ANN, it needs an enormous number of connections, which makes it slow and difficult to train well.

2.2 Why Choose Convolutional Neural Networks?

ANNs are perfect for most tasks, but they are not ideal for dealing with images, particularly for something like predicting plant disease where how things appear really counts. Images are made of pixels arranged in a grid, and nearby pixels often form patterns—like the edge of a leaf or a disease spot—that carry meaning. ANNs do not naturally understand this grid structure. To use an ANN, you would have to flatten the image into a long list of numbers, which breaks these patterns and makes the model overly complex.

Convolutional Neural Networks, or CNNs, are specifically designed for images and address these issues. CNNs have some very important strengths that make them ideal for our project:

- **Recognizing Local Patterns**: CNNs apply tiny filters to image scanning and detect features such as lines, color, or disease markings, tracking their locations in the image. This allows detection of items such as leaf spots or wilting accurately.
- **Utilizing Fewer Resources**: Rather than giving each pixel a weight, CNNs use shared weights among the image, meaning they require less compute power and are simpler to train on large numbers of leaf images.
- **Learning Step-by-Step**: CNNs begin by searching for basic things,

such as edges, and work their way up to complicated patterns, such as the particular appearance of a fungal infection, and so are well suited to distinguishing between diseases.

- **No Manual Work Required**: Unlike ANNs, which require you to specify features ahead of time, CNNs automatically determine what is significant directly from the images and pick up subtle disease hints as a matter of course.

Such strengths make CNNs perfect for our purpose of identifying plant diseases rapidly and accurately even when images are different or diseases appear indistinguishable.

2.3 Building Blocks of Convolutional Neural Networks

CNNs consist of various layers, each performing a particular function to convert an image of a leaf into a prediction of disease. Here is how each component works and why it is important for our project.

2.3.1 Convolutional Layer

The convolutional layer is the core of a CNN. It applies small filters that move over the image to select features, such as the shape of a leaf or a spot of color change. A filter examines a small region at a time and produces a map indicating where specific patterns exist. One filter might detect yellowing edges, for instance, and another textured lesions due to a virus.

In mathematical terms, given an image which is W pixels wide, H pixels high, and has C colours (e.g., red, green, blue), a filter of size $F \times F$ traverses

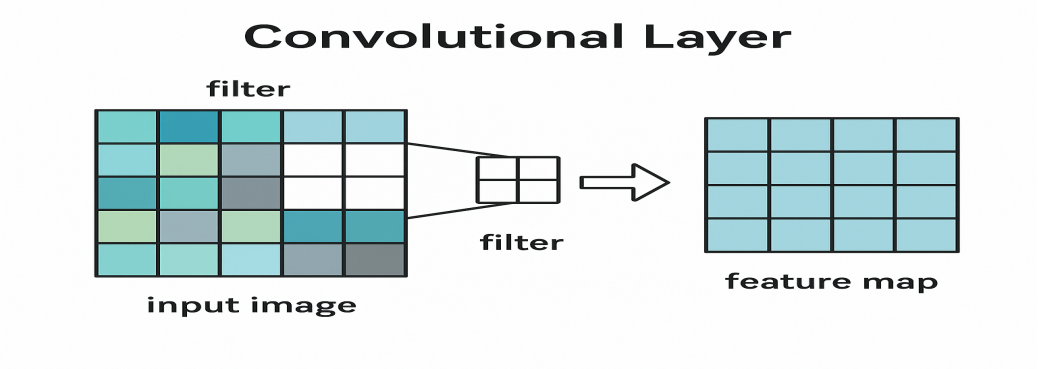

Figure 2.1: Illustration of a convolutional layer transforming an input image into a feature map using a filter.

it with a step size S and additional border P. Its output is a new map of size:

$$W_{\text{out}} = \lfloor (W - F + 2P)/S \rfloor + 1$$

This assists our model to concentrate on disease symptoms without having to process each pixel individually, which is quick and efficient for analyzing leaves.

2.3.2 Pooling Layer

Pooling layers reduce the feature maps to lighten the model and prevent overfitting. The most popular type, max pooling, examines a small region—a 2×2 square, for example—and retains only the maximum value, stepping over with a stride size. This reduces the map's size but retains the most critical bits.

For an input map of W by H, with a pooling area F and step S, the output size is:

$$W_{\text{out}} = \lfloor (W - F)/S \rfloor + 1$$

In our case, pooling helps highlight key disease features, like the brightest

disease spots, while ignoring small distractions, like leaf veins or shadows.

2.3.3 Padding

Padding inserts zeros on the boundaries of an image or feature map prior to applying filters. If you do not use padding, the map decreases in size every time you apply a filter, and you may lose valuable details towards the borders. With padding, you can maintain the map at the same size or have some control over how it reduces.

For plant illnesses, padding is essential to capture symptoms that appear on the edge of the leaf, such as browning tips, so nothing is overlooked.

2.3.4 Activation Layer

Activation layers introduce flexibility to the model by allowing it to learn intricate patterns. Activation layers take the figures from a filter or pooling step and convert them. Popular alternatives are:

- **ReLU**: Converts negative figures to zero and retains positive ones, which allows the model to learn more quickly.
- **Sigmoid**: Squashes values between 0 and 1, suitable for yes-or-no choices.
- **Softmax**: Converts outputs to probabilities for selecting one disease from several.

2.3.5 Fully Connected Layer

Fully connected layers appear last to render the final decision. They receive all the features identified previously, aggregate them, and produce scores for

every potential disease or "healthy." It is like a judge considering all the evidence to determine what is wrong with the plant.

We do one better in our project by using IDML, which refines this area. Rather than simply choosing a disease, IDML lumps similar patterns of diseases together to make it less difficult to differentiate between them when symptoms are indistinguishable from one another, as we saw in Chapter 1.

2.3.6 How It All Fits Together

Every layer contributes to our system. Convolutional layers discover disease hints, pooling layers tidy them up, padding ensures everything is in sight, activation layers interpret complicated patterns, and fully connected layers (with IDML's assistance) provide the diagnosis. This arrangement allows us to deal with difficult cases, such as diseases that appear alike or images captured under varying light, propelling us toward a dependable tool for farmers.

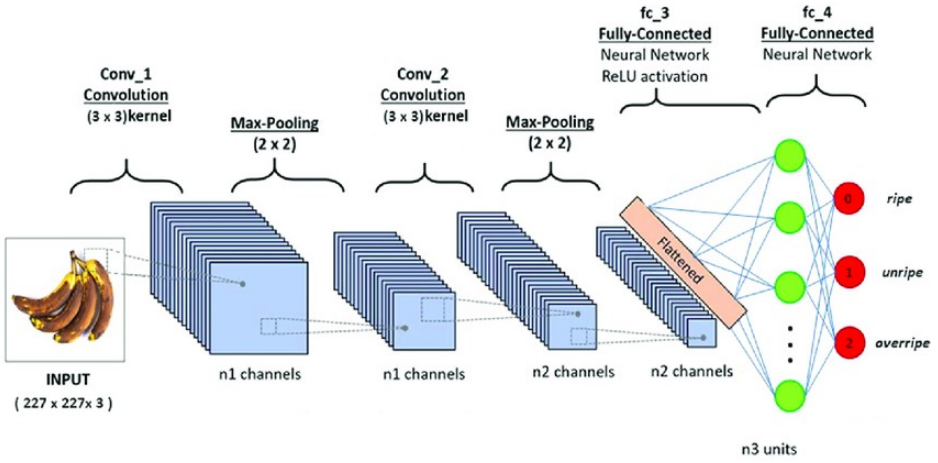

Figure 2.2: The architecture of Convolutional Neural Networks showing the flow from input to output through convolution, pooling, and fully connected layers.

Chapter 3

Introspective Deep Metric Learning for Plant Disease Prediction

3.1 Introduction to Deep Metric Learning

Deep metric learning is an approach to train models to learn to measure similarity between images by embedding them in a space where similar images are near each other and dissimilar ones are far from each other. In classic methods, each photo is mapped to one point, referred to as an embedding, and the distance between the points indicates how similar or dissimilar the images are. For plant disease prediction, this might be classifying leaf images with the same disease, such as bacterial blight, but distinguishing them from healthy leaves or leaves with other diseases.

But standard metric learning has an issue: it takes the position that all images are ideal and forgets about uncertainty. Leaf pictures frequently have problems—out-of-focus areas, shadows, or signs that resemble one another

among diseases, such as wilting or yellowing. These uncertainties may mislead the model to make extremely sure decisions, resulting in errors. For instance, an out-of-focus leaf may be incorrectly diagnosed as healthy because the model does not consider the ambiguity of the picture. This is particularly important within our project, where differentiation of diseases with alike symptoms.

3.2 Why Introspective Deep Metric Learning?

To manage uncertainty, we employ Introspective Deep Metric Learning, or IDML, which is an extension of standard metric learning but includes a method for accounting for how uncertain an image is. IDML is ideal for our project since it addresses the concerns outlined in Chapter **??**, such as variable lighting or overlapping signs of disease, by not requiring the model to make hard decisions on ambiguous images. Here's why IDML excels:

- **Catching Uncertainty**: IDML assigns each image two components—a semantic embedding for its primary features, such as leaf color or lesion form, and an uncertainty embedding to indicate how uncertain it is, such as if it's fuzzy or has conflicting symptoms.
- **Intelligent Similarity**: Rather than evaluating only primary features, IDML considers both features and uncertainty to determine the similarity of two leaves, preventing errors on fuzzy images.
- **Improved Training**: By considering uncertainty, IDML trains the model more meticulously, prioritizing good images over bad ones first and avoiding overfitting on noisy ones.

- **Fits Our Needs**: For diseases in plants, where one leaf could reflect multiple problems or be difficult to categorize with shadows, IDML's conservative method enhances precision.

These advantages make IDML a powerful option for our system, allowing us to accurately classify leaf images even when they're not pristine.

3.3 How IDML Works

IDML works differently from the way we have represented and matched images before to account for uncertainty, employing a unique method for measuring similarity. Below is a description of its major components and how they aid in predicting plant diseases.

3.3.1 Representing Images

In normal metric learning, an image receives one embedding—a set of numbers representing its features. IDML has two embeddings:

- **Semantic Embedding**: This captures the leaf's primary characteristics, such as the spots pattern or edge coloration. It's what we match to determine whether two leaves share the same disease.
- **Uncertainty Embedding**: It captures how much the image is unclear. Like, a leaf with weak symptoms or poor light receives a high uncertainty score, cautioning the model to not be so confident.

For a leaf image, a neural network (such as the CNNs of Chapter 2) produces both embeddings. This two-path approach allows our system to recognize not only what the leaf exhibits but also the reliability of that information.

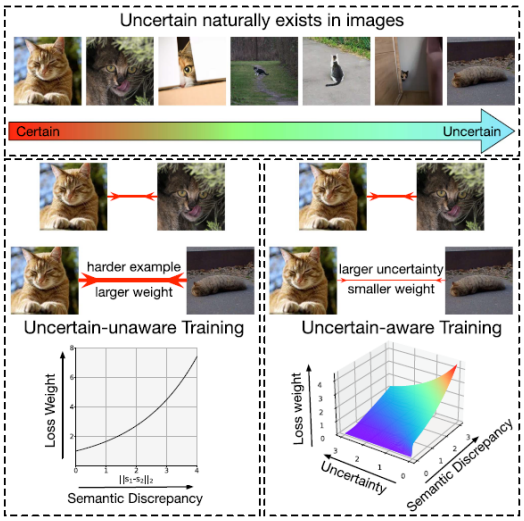

Figure 3.1: Motivation of the proposed IDML framework. Uncertainty naturally exists in images due to occlusion, scale, pose, low resolution, or semantic ambiguity. Most existing deep metric learning methods only consider the semantic distance of an image pair and determine its loss weight solely based on the semantic discrepancy. However, we argue that the model should focus more on certain samples and put less weight on the more uncertain ones. Our IDML achieves this by using both a semantic embedding and an uncertainty embedding to describe an image and employing an introspective similarity metric to compute an uncertainty-aware distance.

3.3.2 Measuring Similarity

IDML employs a novel introspective similarity measure between images, accounting for both their primary characteristics and their uncertainty. Here's the mechanism:

- **Semantic Distance**: This measures the difference between two leaves' semantic embeddings, reflecting how similar their disease patterns are. It's calculated as the Euclidean distance, as the length of a line connecting two points.

- **Uncertainty Level**: This sums up the uncertainty embeddings of the two images to determine how fuzzy the pair is. If the two leaves are

fuzzy, then the uncertainty is high.

- **Merging Them**: The metric combines these two. When uncertainty is high in relation to the semantic distance, IDML decreases the similarity score, stating, "These might not be as different as they seem." Formally, for two images having semantic embeddings s_1 and s_2, and uncertainty embeddings u_1 and u_2, the similarity metric takes the form of:

$$D = \|s_1 - s_2\| \cdot \exp\left(-\frac{\|u_1 + u_2\| + \gamma}{2 \cdot \|s_1 - s_2\|}\right)$$

Here, $\|s_1 - s_2\|$ is the semantic distance, $\|u_1 + u_2\|$ is the uncertainty sum, γ is a small constant to include caution, and $\exp(\cdot)$ mollifies the distance when uncertainty is high. This prevents our model from overcommitting to classify vague leaf pairs, such as two leaves with similarly appearing spots under different illumination.

3.3.3 Handling Uncertain Images

Leaf images often come with uncertainty due to real-world issues, like clouds covering part of the leaf or early-stage diseases with vague signs. IDML shines here by using data augmentation—tweaking images to create variations, like blurring or mixing two leaves' features—to mimic these problems during training. For example, mixing a healthy leaf's image with one showing blight creates a hybrid image that's harder to classify. IDML classifies such blended images as having multiple potential labels (e.g., both "blight" and "healthy"), allowing the model to learn to deal with uncertainty without assigning a single solution.

This is important to our project, since farmers may capture images under challenging conditions. Through training on these difficult instances, IDML

prepares our system to be more resilient, detecting disease even in subpar images.

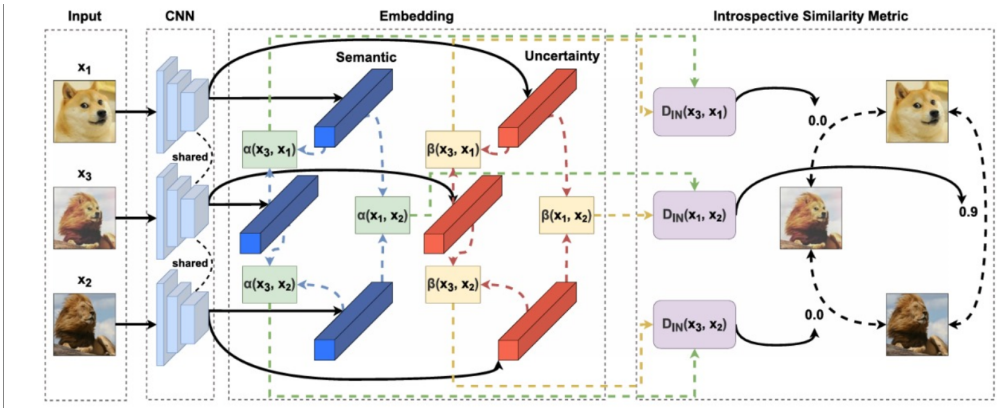

Figure 3.2: Illustration of the proposed IDML framework.We use a convolutional neural network to encode each image with two separate embeddings: a semantic embedding, which encodes the meaningful aspects of the image, and an uncertainty embedding, which measures the uncertainty or ambiguity of the image representation. Semantic difference is calculated by the distance (e.g., Euclidean distance) between the semantic embeddings of two images and represents their similarity. To include uncertainty, we calculate an uncertainty measure by aggregating (e.g., summing) the uncertainty embeddings of the two images. The introspective similarity metric then applies this uncertainty measure to modulate the semantic discrepancy, thereby reducing its impact when uncertainty is great. This yields a more conservative similarity judgment, especially for images with indeterminate features.

3.4 Proxy-Anchor Loss with IDML

To learn our IDML model, we employ the Proxy-Anchor loss, a strong method of structuring the embedding space. While other approaches would compare each image pair to one another, Proxy-Anchor relies on substitute points, or proxies, to stand in for each disease class, such as one proxy for "healthy" and another for "fungal infection." Training is thus quicker and more efficient, particularly for our large plant disease dataset with numerous classes.

3.4.1 Proxy-Anchor Loss

In standard Proxy-Anchor loss, every class comes with a proxy embedding, and the goal is to draw images of that class near to its proxy and away from proxies of other classes. The corrected Proxy-Anchor Loss with IDML, denoted as $J_{PA}(y_{IN}, L; C_{IN})$, is defined as:

$$J_{PA}(y_{IN}, L; C_{IN}) = \frac{1}{|P^+|} \sum_{p \in P^+} \left[\log \left(1 + \sum_{i: l_i = l_{p^+}} e^{-\alpha(C_{IN}(x_i, p^+) - \delta)} \right) \right]$$

$$+ \frac{1}{|P|} \sum_{p \in P} \left[\log \left(1 + \sum_{i: l_i \neq l_p} e^{\alpha(C_{IN}(x_i, p) + \delta)} \right) \right]$$

Where:

- $|P^+|$ is the number of positive proxies (proxies for the same class as the sample).
- $|P|$ is the total number of proxies across all classes.
- $p \in P^+$ iterates over positive proxies.
- $p \in P$ iterates over all proxies.
- $i : l_i = l_{p^+}$ indicates summation over samples x_i where the label l_i matches the label l_{p^+} of the positive proxy.
- $i : l_i \neq l_p$ indicates summation over samples x_i where the label l_i differs from the label l_p of the proxy.
- $C_{IN}(x_i, p)$ is the introspective similarity between sample x_i and proxy p, as defined in the previous section.

- α is a scaling factor (set to 32 in our implementation).
- δ is the margin parameter (set to 0.1 in our implementation), enforcing separation between positive and negative pairs.

This loss function encourages the model to minimize the distance to positive proxies while maximizing the distance to negative proxies, adjusted by the introspective similarity C_{IN} that accounts for uncertainty. The two-line structure improves readability by separating the positive and negative terms.

3.4.2 Incorporating IDML into Proxy-Anchor Loss

IDML modifies the Proxy-Anchor Loss by incorporating its introspective similarity metric C_{IN}, which considers both semantic embeddings and uncertainty embeddings. The introspective similarity C_{IN} is calculated as:

$$C_{\text{IN}} = 1 - (1 - \cos(s, p)) \cdot \exp\left(-\frac{\|u\| + \gamma}{2 \cdot (1 - \cos(s, p))}\right)$$

Where:

- $\cos(s, p)$ is the cosine similarity between the semantic embedding s of the image and the proxy p.
- $\|u\|$ is the magnitude of the uncertainty embedding of the image.
- γ is a small constant for conservatism.

This similarity replaces the Euclidean distance in the standard Proxy-Anchor Loss, ensuring that uncertain images contribute less to the loss, preventing overconfidence. For example, a leaf with fuzzy spots will have a higher $\|u\|$, reducing its impact on proxy updates.

3.4.3 Why Proxy-Anchor with IDML?

Applying Proxy-Anchor with IDML is best suited for our project because:

- **Efficiency**: Proxies minimize the number of comparisons required, so training is faster even with numerous disease types.
- **Uncertainty Handling**: IDML's metric prevents uncertain images, such as those with blended symptoms, from biasing the proxies, enhancing robustness.
- **Clear Clustering**: The loss produces compact clusters around every disease proxy, assisting in the differentiation of similar diseases, such as fungal and bacterial diseases, as seen in Chapter 1.

This configuration allows our model to learn a space in which leaf images cluster by disease correctly despite real-world difficulties.

3.5 Using IDML for Plant Disease Prediction

IDML with Proxy-Anchor loss is a natural fit in our Chapter 2 system. Here's how it works:

- **CNN Backbone**: We employ a CNN (similar to those in Chapter 2) to generate features from leaf images, both semantic and uncertainty embeddings.
- **Training**: At training time, the CNN provides embeddings to the Proxy-Anchor loss, which applies IDML's similarity measure to update proxies and embeddings, treating uncertain images with care.

- **Prediction**: To predict a new leaf image, we calculate its semantic embedding and compare against disease proxies with standard Euclidean distance, as uncertainty isn't required at test time.

- **Handling Challenges**: IDML handles problems from Chapter 1, such as symptom overlap, by clustering similar disease patterns with care so a leaf with unclear spots isn't classified in the wrong way.

This strategy renders our system dependable for farmers, detecting diseases in advance even in harsh environments, such as limited lighting or symptom onset.

3.6 Advantage for Our Project

Through IDML integration with Proxy-Anchor loss, our system enjoys multiple benefits:

- **Strongness**: It performs better with blurry or blurred images, vital for actual leaf photos.

- **Precision**: The reflective metric prevents errors on dubious instances, enhancing disease classification.

- **Performance**: Proxies ensure training effectiveness, allowing us to scale up to numerous disease classes.

- **Practicality**: The model performs well under diverse image quality, benefiting farmers in different environments.

In short, IDML elevates our CNN-based system from Chapter 2 to the next level, making it an effective tool for predicting plant diseases accurately and robustly.

Chapter 4

Experimental Results on Cassava Leaf Disease Classification

4.1 Dataset Description

The dataset contains images organized into 5 categories, representing diverse cassava leaf conditions from the Cassava Leaf Disease Classification dataset, obtained from a 2020 Kaggle competition [1]. Its visual diversity and class imbalance require careful preprocessing to support effective disease classification.

4.1.1 Leaves Categories

The 5 categories encompass disease states and healthy conditions, including:

- Class 0 (Healthy): Images of healthy cassava leaves, showing natural green coloration and no visible disease symptoms.

- Class 1 (Cassava Bacterial Blight - CBB): Leaves with water-soaked lesions and angular necrotic spots.
- Class 2 (Cassava Brown Streak Disease - CBSD): Leaves with brown streaks and yellowing, often with root necrosis.
- Class 3 (Cassava Mosaic Disease - CMD): Leaves with mosaic patterns, distortion, and chlorosis.
- Class 4 (Cassava Green Mottle - CGM): Leaves with mottled green patterns and mild chlorosis.

These conditions vary in color, texture, and symptom presentation, necessitating models to distinguish visually similar diseases (e.g., CBSD vs. CMD).

4.1.2 Image Characteristics

Images reflect real-world variability:

- Lighting: From bright outdoor shots of healthy leaves to dimly lit images of diseased leaves taken in shaded areas.
- Backgrounds: Clean leaf surfaces for healthy samples to cluttered field settings with diseased leaves.
- Angles: Top-down views of healthy leaves, side profiles of CBB-affected leaves, and angled shots of CMD leaves.
- Resolution: High-definition healthy leaves to lower-resolution images of CGM affected by mobile camera quality.

This variability supports real-world applications but complicates classification, requiring robust preprocessing.

Figure 4.1: Photos of each classes.

4.2 Data Preprocessing

The preprocessing pipeline standardized the dataset for training, addressing image variability, class imbalance, and computational requirements through five steps: image resizing, data augmentation, normalization, label encoding, and dataset splitting. These steps ensured compatibility with the classification models, preserving the visual integrity of disease symptoms like those of CBB and CMD.

4.2.1 Image Resizing

Images were resized to 224×224 pixels for one model and 48×48 pixels for others to balance visual detail and computational efficiency. Non-square images were scaled to fit within the target resolution, padded with a white background to preserve aspect ratios, ensuring features like the angular lesions of CBB or the mosaic patterns of CMD remained intact.

4.2.2 Data Augmentation

Data augmentation addressed class imbalance (Table 2.1), increasing under-represented categories (e.g., Class 0 from 1087 to 300 images) to 294–300 images, except for Class 3 (13158 images), which retained its original count. Applied during training for most models, augmentation included:

- Rotation: Up to 20 degrees to simulate varied viewpoints (e.g., tilted CBSD leaves).
- Width/Height Shifts: 20% to capture positional variations (e.g., offset CMD patterns).
- Zoom: 20% to mimic different scales (e.g., zoomed CGM leaves).
- Horizontal Flips: 50% probability to reflect mirrored perspectives (e.g., flipped healthy leaves).
- Brightness Adjustments: ±20% to account for lighting variations (e.g., dim diseased leaves).
- Color Jittering: ±10% hue/saturation to handle color variations (e.g., varied CMD chlorosis).

One model used minimal augmentation (rescaling only) for baseline performance. Augmentation enhanced generalization, reducing overfitting for diseases like CGM.

4.2.3 Class Imbalance

Class imbalance varies significantly, with initial counts from 1087 (Class 0: Healthy) to 13158 (Class 3: CMD). Table 2.1 shows initial and augmented counts. Under-represented categories like Class 0 (1087 images) and Class

Table 4.1: Initial and Augmented Image Counts per Category

Category	Augmented Images	Initial Images
Class 0 (Healthy)	1087	300
Class 1 (CBB)	2189	298
Class 2 (CBSD)	2386	297
Class 3 (CMD)	13158	13158
Class 4 (CGM)	2577	299

1 (2189 images) were augmented to 294–300 images, while Class 3 (13158 images) retained its original count with minimal augmentation. Augmentation balanced the dataset, enhancing representation of less common disease states.

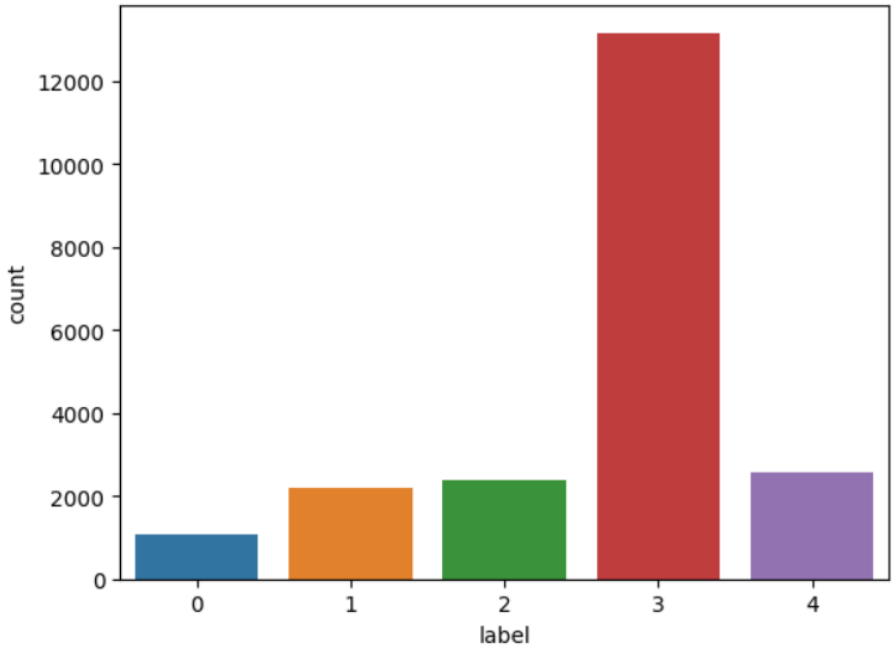

Figure 4.2: Data Imbalance

4.2.4 Normalization

Pixel values were normalized from [0, 255] to [0, 1] by dividing by 255, ensuring consistent inputs. This standardized pixel intensities across varied lighting conditions, focusing models on features like the necrosis of CBSD.

4.2.5 Label Encoding and Dataset Splitting

Labels were encoded as integers (0–4) corresponding to the five classes. The dataset was split into a training (80%) and validation (20%) set, stratified by class to preserve imbalance proportions, enabling the model to generalize to the varying quality of images farmers may encounter.

4.2.6 To Solve Class Imbalance

In addition to data augmentation, a WeightedRandomSampler was utilized for training to further counteract class imbalance. This method uses sampling weights inversely proportional to class frequencies to make underrepresented classes like Class 0 (Healthy) and Class 1 (CBB) more probable during mini-batch creation. When used together with data augmentation, the model sees a more balanced set of all classes per epoch. Such dual strategy not only reduces the dominance bias towards influential classes such as CMD but also improves the generalization capability of the model on different image conditions, including those of low-quality or infrequent disease samples a farmer would typically find in real-world situations.

4.3 Model Architecture

Two models were compared:

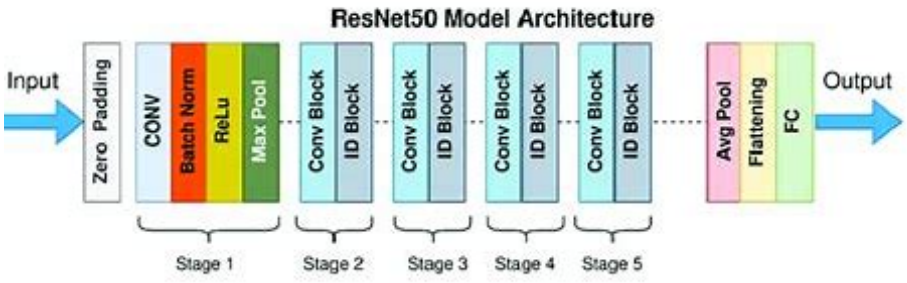

Figure 4.3: ResNet-50 architecture used in both models.

- **Baseline CNN Model**:

  A ResNet-50 CNN, pretrained on ImageNet, trained with Cross-Entropy Loss for multi-class classification. It provides direct class probabilities, optimized to identify the five cassava leaf classes.

- **IDML Model**: A ResNet-50 backbone augmented with Introspective Deep Metric Learning (IDML), employing Proxy-Anchor Loss to produce 512-dimensional semantic embeddings. IDML also generates uncertainty embeddings to measure prediction confidence, enhancing robustness for low-quality or unclear images.

Both models take advantage of ResNet-50's depth and efficiency, making them suitable for deployment on resource-scarce devices such as mobile phones.

4.4 Loss Function

To effectively guide the learning process, different loss functions were employed for the two models, aligned with their respective objectives:

- **Cross-Entropy Loss (Baseline CNN Model)**: The baseline ResNet-50 model used the standard Cross-Entropy Loss, suitable for multi-class

classification tasks. This loss function compares the predicted class probabilities with the true class labels, penalizing incorrect predictions more heavily. It is well-suited for conventional classification pipelines and enables the model to directly predict class labels.

- **Proxy-Anchor Loss (IDML Model)**: The IDML model optimized its embedding space with Proxy-Anchor Loss, a metric learning objective that brings samples closer to their class proxies and pushes them away from proxies of other classes. This loss allows the model to learn semantically informative embeddings, improving classification and retrieval performance. Following are the hyperparameters employed:
  - Scaling factor
    $alpha = 32$
  - Margin = 0.1
  - Embedding size = 512

  Proxy-Anchor Loss works particularly well with situations involving a lot of intra-class variation but limited inter-class difference, which is the nature of cassava leaf diseases.

4.5 Model Training

Model training was conducted with the following setup:

- **Epochs**: 20.
- **Batch Size**: 32.
- **Optimizer**: AdamW, with a learning rate of 0.0003 and a weight decay of 0.0001.

- **Learning Rate Scheduler**: StepLR, which decreases the learning rate by a factor of 0.5 every 5 epochs.
- **Early Stopping**: Activated after 6 epochs with no improvement in validation loss.

The baseline CNN model utilized Cross-Entropy Loss, while the IDML model employed Proxy-Anchor Loss with parameters $\alpha = 32$, margin=0.1, and embedding size 512. Training was performed on GPUs, with inference optimized for CPUs or GPUs to align with real-world deployment limitations.

4.6 Evaluation Metrics

The primary metric, according to the competition's requirements, is classification accuracy, defined as the fraction of correctly predicted images. For the IDML model, additional metrics were calculated to evaluate its metric learning performance:

- **R@1 (Recall at 1)**: Fraction of samples where the top-1 predicted class is correct.
- **NMI (Normalized Mutual Information)**: Measures clustering quality of embeddings by performing K-means with 5 clusters.
- **RP (Retrieval Precision)**: Precision of retrieving the correct class among the top-2 predictions.
- **MC@R (Mean Class Precision at R)**: Mean precision over all classes for top-2 retrieval.

These measures offer a complete evaluation of both classification and retrieval performance, essential for real-world applications.

4.7 Results

The performance of both models was evaluated based on their validation accuracy, with the IDML model demonstrating superior results. Below, we first present the validation accuracy for both models, followed by the additional metrics for the IDML model only, as it outperformed the baseline.

4.7.1 Baseline CNN Model Validation Accuracy: 79.30%

Trained with Cross-Entropy Loss, the baseline ResNet-50 model achieves a commendable validation accuracy, reflecting good performance in cassava leaf disease classification. However, it lacks the sophisticated retrieval capabilities provided by metric learning methods.

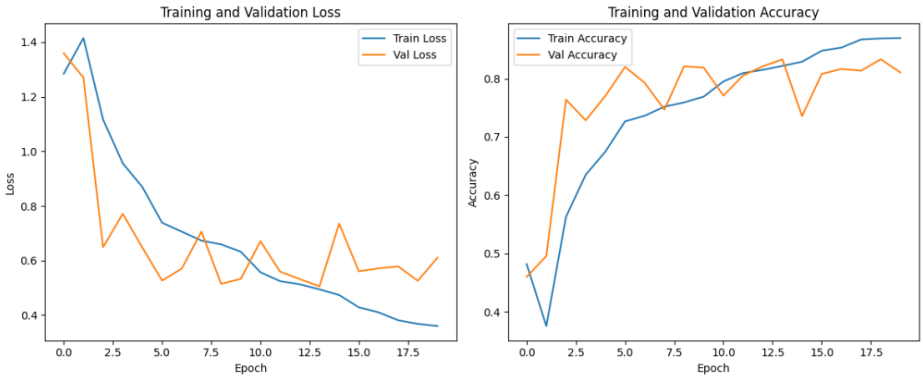

Figure 4.4: Baseline cnn model accuracy & loss graph.

4.7.2 IDML Model Validation Accuracy: 84.46%

Using Proxy-Anchor Loss and IDML, this model achieves a 1.14% gain over the baseline, demonstrating improved generalization to novel validation images. The gain reflects IDML's effectiveness in addressing real-world image variability and uncertainty.

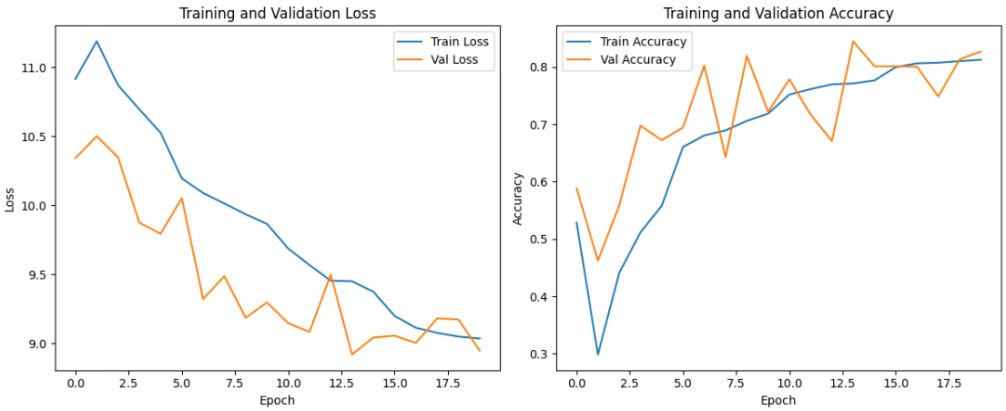

Figure 4.5: Baseline cnn model accuracy & loss graph.

IDML Model Metrics:

The IDML model provides additional metrics that reflect its performance in classification and retrieval tasks. These metrics are shown in Table 4.2 below:

Table 4.2: IDML Model Performance Metrics

R@1	NMI	RP	MC@R
84.46%	0.4743	94.49%	90.11%

- **R@1**: 84.46% corresponds to the validation accuracy, indicating that the top-1 prediction is correct for 84.46% of validation samples.

- **NMI**: 0.4743 shows moderate clustering quality, meaning that embeddings cluster similar classes reasonably well, although distinguishing visually similar diseases remains challenging.
- **RP**: 94.49% indicates high retrieval precision in selecting the correct class among the top-2 predictions, allowing the model to be trustworthy for suggesting multiple plausible diagnoses.
- **MC@R**: 90.11% signifies reliable retrieval precision across all classes, ensuring balanced performance for both underrepresented and typical diseases.

For reference, the baseline CNN model achieved a training accuracy of 73.64% with a training loss of 0.7057 and a validation loss of 0.5704. The IDML model, with a lower training accuracy of 77.11% and higher losses, excels in validation performance due to its metric learning optimization, which cannot be directly compared to Cross-Entropy Loss.

4.8 Predicted Image Examples

Figure 4.6: Predicted images for each cassava leaf disease class. Each image represents a sample prediction from the IDML model for the respective class.

4.9 Challenges and Solutions

Several challenges were encountered during the experiment:

- **Class Imbalance**: The dataset's imbalanced class distribution was addressed using a WeightedRandomSampler and data augmentation to ensure equal representation of minority classes such as CGM and CBSD.
- **Image Quality Variability**: Poor-quality images from mobile cameras were mitigated by IDML's uncertainty modeling, enabling the model to flag uncertain predictions for further review.
- **Metric Learning Difficulty**: The IDML model's higher loss values reflect the complexity of optimizing embeddings, but its superior validation accuracy and retrieval metrics validate the approach.

These solutions align with the objective of creating a practical tool for farmers without access to advanced technology or expert inspections.

4.10 Conclusion

The IDML model, with a validation accuracy of 84.46%, surpasses the baseline CNN model's 79.30%, demonstrating improved generalization for cassava leaf disease classification. Its additional metrics—R@1 (84.46%), NMI (0.4743), RP (94.49%), and MC@R (90.11%)—emphasize its prowess in retrieval tasks, making it highly suitable for real-world applications where farmers can utilize top-2 predictions. The model's capacity to handle real-world image variability positions it as

a promising solution for smallholder farmers in Africa, enabling rapid disease diagnosis using mobile devices. Future research could enhance clustering performance (NMI) by investigating advanced architectures or expanding the dataset with more varied images.

Bibliography

[1] Kaggle, "Cassava Leaf Disease Classification," 2020. [Online]. Available: https://www.kaggle.com/competitions/cassava-leaf-disease-classification

[2] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*, MIT Press, 2016.

[3] C. Wang, W. Zheng, Z. Zhu, J. Zhou, et al., "Introspective Deep Metric Learning," *IEEE Transactions on ...*, 2023. [Online]. Available: https://ieeexplore.ieee.org/abstract/document/10239539

[4] G. Shrestha, M. Das, N. Dey, "Plant Disease Detection Using CNN," *2020 IEEE Applied Signal ...*, 2020. [Online]. Available: https://ieeexplore.ieee.org

[5] A. Babenko, A. Slesarev, A. Chigorin, and V. Lempitsky, "Neural Codes for Image Retrieval," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014, pp. 584–599.