Optimizing CNN Memory Efficiency: A Continual Learning Solution for Plant Stress Classification

by

Ahmed Shakib Reza
23341130
Farah Ulfat Zaima
21301138
Mysha Maliha Annisa
21101243
Tazkera Sattar
21101282

A thesis submitted to the Department of Computer Science and Engineering
in partial fulfillment of the requirements for the degree of
B.Sc. in Computer Science

Department of Computer Science and Engineering
Brac University
October 2024

© 2024. Brac University
All rights reserved.

Declaration

It is hereby declared that

1. The thesis submitted is my/our own original work while completing degree at Brac University.
2. The thesis does not contain material previously published or written by a third party, except where this is appropriately cited through full and accurate referencing.
3. The thesis does not contain material which has been accepted, or submitted, for any other degree or diploma at a university or other institution.
4. We have acknowledged all main sources of help.

Student's Full Name & Signature:

Ahmed Shakib Reza 23341130	Farah Ulfat Zaima 21301138
Mysha Maliha Annisa 21101243	Tazkera Sattar 21101282

Approval

The thesis/project titled "Optimizing CNN Memory Efficiency: A Continual Learning Solution for Plant Stress Classification" submitted by

1. Ahmed Shakib Reza (23341130)
2. Farah Ulfat Zaima (21301138)
3. Mysha Maliha Annisa (21101243)
4. Tazkera Sattar (21101282)

Of Summer, 2024 has been accepted as satisfactory in partial fulfillment of the requirement for the degree of B.Sc. in Computer Science on October 17, 2024.

Examining Committee:

Supervisor:

Dr. Muhammad Iqbal Hossain
Associate Professor
Department of Computer Science and Engineering
BRAC University

Co-Supervisor:

Mr. Rafeed Rahman
Lecturer
Department of Computer Science and Engineering
BRAC University

Program Coordinator:

Dr. Md. Golam Rabiul Alam
Professor
Department of Computer Science and Engineering
BRAC University

Head of Department:

Dr. Sadia Hamid Kazi
Associate Professor
Department of Computer Science and Engineering
BRAC University

Abstract

In nature, plants encounter a wide variety of stress-inducing elements that can create unfavorable conditions or impede their metabolic processes and hinder growth. Both biotic, living organisms, and abiotic, non-living elements, can contribute to this stress in plants. Bangladesh, being an agricultural nation, has a lot of difficulties in detecting symptoms of these stresses. Our study focuses on recognizing and classifying stress in four popular plants that are consumed both domestically and internationally, including eggplant, bitter gourd, snake gourd and ash gourd. In recent times, Convolutional Neural Network (CNN) is widely and successfully used to classify health conditions from leaf images. But the CNN models in use today are vulnerable to a phenomenon known as catastrophic forgetting. Because of this, currently, there is no CNN model that can perform well in several tasks at once, without forgetting the previous tasks. This thesis is to empower a CNN model with Continual Learning (CL), that can learn new tasks continuously without forgetting its prior knowledge. In this study, we evaluate and demonstrate the issue of forgetting prior knowledge, that can be eradicated by using Continual Learning (CL) approaches like Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF), while being able to perform multi-task and achieving significantly better performance than conventional approach. To simulate a scenario similar to continuous data flow, the dataset is divided into four different tasks. Where without the Continual Learning (CL) methods the CNN model EfficientNet-B0 suffers from catastrophic forgetting achieving accuracy of 42%, 55%, 67%, and 97% for task-1, task-2, task-3, and task-4 respectively after training on task-4. After applying Continual Learning (CL) a drastic improvement of 7% to 18% in accuracy was achieved throughout different tasks. Elastic Weight Consolidation (EWC) achieves 60%, 65%, 66%, and 98% for task-1, task-2, task-3, and task-4 respectively after training on task-4 . Learning without Forgetting (LwF) achieves 60%, 61%, 55%, and 75% for task-1, task-2, task-3, and task-4 respectively after training on task-4. As Continual Learning (CL) approach preserves the knowledge of previous tasks while only training on the current task, this method can significantly reduce the training time and memory issue of CNN models while training on multiple tasks.

Keywords: Machine Learning; Deep Learning; Convolutional Neural Network (CNN); Continual Learning; Plant stress; Classification; Multi-task; Catastrophic Forgetting

Acknowledgement

To begin with, we are thankful to Almighty Allah for giving us the strength, endurance, and knowledge to accomplish this thesis. Next, we are extremely grateful to our supervisor, Dr. Muhammad Iqbal Hossain, who has always been supportive and provided thoughtful advice which has helped us to successfully complete the thesis. Furthermore, we are also very thankful to our co-supervisor, Mr. Rafeed Rahman, who has provided us with constant support, guidance and feedback which has been necessary throughout our thesis. Lastly, we are also thankful to our parents, whose kind support and prayers have made it possible for us to complete this thesis.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Plants are essential to our existence for a number of reasons, including their profound effects on the world's economy and food production. Their principal role as a food source contributes to global nutritional demands, making them a significant tool for our existence. Furthermore, plants are vital to the economy because they are the basis of many other businesses, such as forestry, agriculture, and pharmaceuticals. In addition to providing food and economic benefits, plants maintain biodiversity and produce oxygen, which both contribute to environmental balance. Essentially, the importance of plants goes way beyond their biological existence, impacting many facets of human health as well as the health of the world. Vegetable plants offer a supply of nutrient-dense, fresh food. They provide a firm supply of fiber, vitamins, and minerals—all of which are necessary for the consumption of a balanced diet. They offer a range of health-promoting antioxidants, including beta-carotene and lycopene. A variety of vegetables guarantee a well-rounded diet. Due to their high vitamin, mineral, and fiber contents, they help make nutrient-dense foods more accessible and leads to lower the rate of malnutrition. Since, plants play a vital role in generating income and enhance nutritional status, vegetable crops are essential for guaranteeing food security and reducing poverty, particularly in developing nations. Producing vegetables on a small scale can help reduce poverty and provide job opportunities. When compared to cereal crops, vegetable cultivation can significantly increase farm profits for smallholder farmers, making it a valuable source of revenue. Vegetables and other horticultural crops have been vital to Bangladesh's agricultural growth and economic advancement. Increased export revenue can result from the inclusion of vegetable crops in the export basket. An important part of Bangladesh's economy is the export of vegetables. Bangladesh exported vegetable goods worth $159 million in 2021 [18].More than 54 different varieties of vegetables are exported from the nation, including radish, bottle gourd, bitter gourd, egg plant, okra, French bean, green chili and green papaya [18]. Vegetable plants also contribute to the cultural heritage and traditions of various societies, reflecting their culinary and medicinal uses. Plants in agricultural environments encounter a range of obstacles from both biotic (living) and abiotic (non-living) stresses. Abiotic factors include things like soil quality and ambient circumstances, whereas biotic factors include things like pests and illnesses. Effective stress mitigation requires an understanding of the indications of stress, including withering and discolouration. Farming vegetables, is a very traditional process in Bangladesh and around the world. From planting the seed, to picking the full grown vegetable the whole process is manual. Farmers

rely on their experience and eyesight to detect any kind of disease, malnutrition or insect attack. This way of diagnosis is error prone, which makes the life of a farmer much harder. Recent advancements in AI and hardware have made it possible to detect and classify various plant stressors with computer vision, which makes farming more profitable, alongside increases the cost-effectiveness. Computerized systems are nearly perfect in identifying and categorizing these strains. Therefore, concentrating on developing modern, more effective approaches to use AI and ML to solve this issue is essential. In conclusion, computer vision-enabled advanced computerized procedures have the potential to revolutionise Bangladeshi and global vegetable growing practices. As a result, it has become a necessity to carry our research in this field.

1.1 Stress in Plant

Plant stress is defined as any adverse condition that affects a plant's growth, development or metabolism. There are two different types of environmental factors by which plant stress is caused. Firstly, there are biotic stressors which include living creatures like insects, bacteria and fungi. Microorganisms play the most important role in causing biotic stress through diseases (viral, fungal, and bacterial). According to a study, pests and pathogens result in 37% of the global agricultural yield loss, while insects contribute 13% [20]. This means that biotic stress, is an external biological stress that affects nearly every plant group [9]. Secondly, abiotic stress factors in plants are caused by non-living elements that include drought, salinity, extreme hot and cold temperatures, mineral unavailability or presence toxicity and many others. As a result, abiotic stress factors are also impacting the plant health which has a reduction on the overall crop yields [14]. Water stress is a significant abiotic stress that impacts a plant's overall health and growth. It can be either a result of excessive flooding or extreme weather conditions like droughts. Water also serves as an important carrier for the various nutrients and metabolites that plants need to complete their various developmental processes. Water stress can be observed in plants by its physical appearance such as wilting [9]. Therefore, in simple terms, plant stress is any inappropriate condition that can hinder a plant's growth or threaten its life caused by both biotic and abiotic factors.

1.2 Problem Statement

In every food cycle, plants are the primary source of food. Additionally, plants release oxygen, which is essential to life on this planet. Therefore, to keep the entire ecosystem going, we need to save plants whenever possible. As mentioned, plants are the basis of any food web, a plentiful supply of food is provided by vegetable plants. Therefore, it is very crucial to make sure that the food which is being consumed is fresh, nutritious and resistant to insects. Early detection of any downside while these vegetable plants grow, is one of the very first steps of ensuring wholesome, nutrition rich and pest-proof food. However, detecting them manually can be time consuming and might have errors. It also takes years to build the expertise in order to detect the actual cause of any plant disease. To solve this issue, many recent studies and researches have taken place using the aid of Machine Learning to perfectly predict

the diseases, nutrition deficiencies and pest attacks these vegetable plants encounter. Many Machine Learning and Deep Learning models have been built and they have shown surprising accuracy. DenseNet-121 was able to give a higher classification accuracy of 99.81% compared to few other CNN models to detect crop diseases [15]. PlantDet, which was built on the basis of InceptionResNetV2, Efficient NetV2 and Xception gives an accuracy of 98.53% [22]. However, these models were built for a limited range of plants. Some experiments were executed where different types of tasks were carried out, but this was only possible as they required different models for each task. One of such studies was conducted where they established three different models as a way of performing 3 separate tasks [16]. Hence, very few studies have been done to detect different types of plant stress with a single model where the model performs equally well for different types of plants and different variants of stress detection. These existing studies have either given a unsatisfactory prediction accuracy or taken longer to detect compared to the models which are trained for a specific task. A multi-task 2 stage continual learning approach provided a maximum accuracy of 94.47% in DenseNet-121 [23]. Since 94.47% accuracy is much less compared to other studies which used separate models, therefore, more studies need to be done to develop generalized models which can be trained for multiple tasks.

Over the past ten years, Bangladesh's fruit and vegetable exports have dropped by 68.7%, from $239.19 million in FY14 to $74.93 million in FY23, due to poor agricultural methods and inadequate infrastructure [18]. Despite these challenges, a variety of fruits, vegetables and betel leaves are typically exported from Bangladesh [18]. As Bangladesh has a wide variety of vegetables available, it lacks enough computerized systems to detect stress for every variety of vegetable plants. Therefore, a system for processing data automatically which is able to predict different vegetable plant's healthfulness, nutrient deficiencies and vulnerability to insect impact, at the same time has become a necessity. The amount of vegetables produced would be greatly increased by a system which can determine whether the leaf is healthy or unhealthy by studying leaf images, determine the type of nutrient that is deficit in that leaf and finally decide the type of illness it possesses. Moreover, the proposed model should be capable to recognising the kind of plant stress in less time and with less training data. Additionally, they would be immune to Catastrophic Forgetting. The proposed model should also be able to determine plant stress correctly, giving a satisfactory percentage of test accuracy. As a result, this problem statement focuses on building one particular computerized system which is able to perform multiple tasks with a much more efficient and higher classification accuracy and immune to Catastrophic Forgetting. To conclude, in this problem statement, a Machine Learning model is suggested which is able to accurately and efficiently detect plant stress on a wide variety of vegetables that are commonly available in Bangladesh and around the world, creating an easy and reliable access to fresh and nutritious food.

1.2.1 What is Catastrophic Forgetting?

Catastrophic Forgetting occurs during fine-tuning an already trained neural network. This event takes place when significant parameters in a network are adjusted to accommodate newly added data while compromising the model's potential to

deal with old data. This makes multiclass classification difficult in the real world. Therefore, in order to overcome catastrophic forgetting, a model must continuously learn and evolve through tasks. This can be achieved through a continual learning approach.

1.3 Research Objective

The primary objective of this research is to address the issue of catastrophic forgetting in Convolutional Neural Networks (CNN) and improve the memory efficiency of these models in the context of plant stress classification. Specifically, the study aims to develop and evaluate a continual learning-based solution capable of classifying different types of stress in multiple plant species while preserving the performance of previously learned tasks. This will be achieved by:

1. Applying continual learning techniques such as Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF) to CNN models, allowing them to continuously acquire new knowledge without losing previously acquired information.

2. Dividing the dataset into tasks that simulate real-world scenarios of continuous data flow, enabling the evaluation of the proposed methods under conditions that mirror dynamic agricultural environments.

3. Optimizing the EfficientNet-B0 architecture for multi-task classification, ensuring high accuracy across different plant stress detection tasks while mitigating the computational overhead typically associated with re-training large models for each new task.

The study's ultimate goal is to demonstrate a significant reduction in catastrophic forgetting and an increase in classification accuracy for plant stress detection using continual learning, thus contributing to more robust and efficient AI systems in agriculture.

1.4 Thesis Structure

The thesis is structured to overcome catastrophic forgetting using continual learning approaches like Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF) for plant stress classification, with chapters covering literature review, dataset description, methodology, results and concludes with key findings, limitations and future works.

- **Chapter 2** highlights related methodologies and research for plant stress classification using various machine learning models, with a particular emphasis on overcoming the problem of catastrophic forgetting. To overcome the problem of catastrophic forgetting, two key models and approaches are used: Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF), and the chapter establishes the theoretical background for the proposed strategies.

- **Chapter 3** provides a detailed description of the dataset used in this research. The chapter also includes the data collection process and details about the different types of plant stressors. The dataset includes eight different plant leaf images, each classified into different stress conditions.

- **Chapter 4** describes the methodology used in this study, focusing on dataset pre-processing and the use of continual learning strategies to improve the memory efficiency of CNN models. The data pre-processing includes several augmentation techniques to ensure balance of the dataset. Then, the dataset is divided into four tasks. The chapter also discusses the architectural models, such as EfficientNet-B0, as well as the changes made to accommodate continual learning methods like EWC and LwF.

- **Chapter 5** evaluates the performance of the proposed models. Metrics like accuracy, precision, and recall are determined for each task, with the goal of examining the influence of continual learning on preventing catastrophic forgetting. Results are compared, revealing that EWC and LwF strategies outperform traditional methods.

- **Chapter 6** concludes the paper with a summary of the results, highlighting how well continual learning works to prevent catastrophic forgetting in CNN models used to classify plant stress. The chapter also includes limitations and suggestions for future improvements in our research.

Chapter 2

Literature Review

2.1 Related Works

Shovon et al., proposed an ensemble model named PlantDet, based on Inception-ResNetV2, Efficient NetV2, and Xception [22]. The model includes additional dense layers to each of the basis models, a dropout mechanism, batch normalization layers, PReLU activation function, global average pooling layer and L2 regularizes. The study concentrated on the two categories of betel leaf, such as healthy and unhealthy classes, and the five most common illnesses that affect rice leaves: bacterial leaf blight, leaf blast, leaf scald, brown spot and narrow brown spot. The model yielded the following results on the rice leaf dataset: 98.35% recall, 98.42% F1, 99.71% specificity, 98.53% accuracy and 98.50% precision. On the betel leaf dataset, accuracy was 97.50%, precision was 99.00%, recall was 97.50%, F1 was 98.14% and specificity was 99.00%. The proposed model maintains a high performance in both datasets while handling underfitting and overfitting issues. However, the model is heavy and complex, and noise is introduced into the model as a result of its intricate architecture. It takes a long time as well. Additionally, the proposed model lacks interpretability as it is an ensemble model.

Shoaib et al., proposed an architecture, based on a recently developed CNN. This architecture was trained on more than 18,000 segmented and non-segmented leaf images of tomato, using a supervised learning for detecting and recognizing distinct diseases of tomato utilising InceptionNet (V1, V2 andV3) [16]. In their research work, they experimented on three different classification methods. Firstly, they explored binary classification which included healthy and diseased leaf classes. Secondly, they experimented on five categories of diseased and one healthy class of leaves i.e. six separate classes. Finally, they experimented on a total of ten classes i.e. nine class of diseased and one class of healthy leaf. For segmentation they experimented with U-Net and modified U-Net. The ability of the modified U-net to distinguish leaf images from the backdrop was better. Furthermore, InceptionNet1 outperformed other designs in terms of extracting the most important properties of images. The proposed model outperformed many currently existing models in accuracy and precision. However, dataset which was used in this study only includes a small number of certain tomato breeds and was gathered from a particular area. Due to its lack of generalization, this model might not perform as well in other breeds or the aforementioned breeds from other locations. Furthermore, the study

could not have looked into alternative lightweight CNN architectures that might be appropriate for more real-time and portable applications.

Chowdhury et al., also worked on the PlantVillage dataset, focusing on 3 types of classification: binary class classification (healthy and unhealthy class), six-class classification (healthy and five class of diseased leaves) and ten-class classification (healthy and nine class of unhealthy leaves) [12]. Their proposed model is based on EfficientNet. EfficientNet-B0, EfficientNet-B4 and EfficientNet-B7 were employed. Additionally, global average pooling was applied in the final convolution layer to improve accuracy and reduce overfitting. A dense layer of size 1024 with 25% dropout was added after global average pooling. Lastly, a Softmax layer provides the likelihood prediction for identifying diseased tomato leaf. In this work, two segmentation model: U-Net and Modified U-Net were examined. Here, the modified U-Net model produced scores of 98.66%, 98.5%, and 98.73% for accuracy, IoU and dice. For binary class EfficientNet-B7 performed better, displaying 99.95% accuracy. In six class classification EfficientNet-B7 also performed better with 99.12% accuracy. EfficientNet-B4 outperformed with 99.89% accuracy for ten classes. Nevertheless, this dataset is restricted to a certain breed and geographic area. Other light CNNs could have been investigated for portable applications as well.

Zhe Xu et al., classified ten classes of nutritional deficiencies and compared them with healthy rice plants with full nutrition (total eleven classes) using their own data set [10]. They applied four different architectures: finely tuned version of Inception-V3, a 50 layered ResNet, NasNet-Large and a 121 layered DenseNet. Their work shows novelty in identifying nutrient deficiencies in rice, as most of the existing works mainly focused on image classification with Deep convolutional neural networks (DCNNs). All the DNCCs obtained over 90% accuracy but DenseNet121 performed best with validation accuracy of 98.62% and test accuracy of 97.44%. The effectiveness of the DCNNs was confirmed by comparing the color feature and the oriented gradient histogram with support vector machines. Nevertheless, the research emphasises the possibility of misdiagnosis due to similar symptoms across different nutrient deficits, notably highlighting the difficulties in discriminating between Fe, Mn and Zn deficiencies. The research's dependence on hydroponic experiments and failure to take into account the combinations of many factors is mentioned as an issue, decreasing the the model's robustness. The fact that hydroponic experiments require a lot of time and resources to collect data about is a recognition of this limitation. The necessity of combining studies to better comprehend and identify real-world issues is also mentioned in the study.

Gabriel et al., proposed a novel method to detect insect attack on leaves of plants by using geometric aspects of the leaf and employing digital image processing techniques to construct picture models [13]. There are five main components to the proposed method, such as leaf segmentation, leaf features, leaf models, template matching and leaf prediction. The proposed method is evaluated on 12 crops where it is shown to persist despite noise rotation and image scale. It identifies and draws attention to the leaf areas that insects have attacked, as well as breaking down the shapes left by their bites. Regardless of the type of plant, it can accurately identify locations where insects prey with a precision rate of over 90% on leaves of blueberry,

corn, potato, and soybean. Subsequent investigations should broaden the scope to assess supplementary datasets and employ diverse insect species to cause harm to leaves. Additionally, there is a scope for experimenting with different optimization parameters. Lastly, the suggested approach can only serve as a starting point for categorising insects based on the characteristics of their bites.

J. et al., conducted four experiments and analyzed their relative performance on predicting crop diseases based on the renowned PlantVillage dataset which contains images of 38 different classes of diseased plant leaves [15]. For the experiments, they evaluated different transfer learning techniques using deep CNN (Convolutional Neural Network) models which includes VGG-16, DenseNet-121, ResNet-50 and Inception V4, to classify multi-class plant diseases. Hyperparameters were fine tuned for these pre-trained models, keeping the initial learning rate to be 0.001 for each model. These models ran for 30 epochs, with a ReLu activation, batch normalization and a dropout value of 0.5 epochs. Here, DenseNet-121 outperformed the other three models, giving a categorizing accuracy of 99.81% in the testing phase and an F1 score of 99.8% alongside a validation loss of 0.0154%. Besides, the test accuracy for VGG-16, ResNet-50 and Inception V4 was 82.75%, 98.73% and 97.59% respectively, alongside a test loss of 0.64%, 0.027% and 0.0586% respectively. Moreover, in this paper they also resolve overfitting issues by using data augmentation methods, overcome degradation problems and reduce computational complexity. However, this paper does not deal with complications when gathering data in real-time, and the proposed model is also unable to predict plant diseases from multiple leaves as opposed to just one.

Then Khan et al., proposed a cucumber plant disease prediction and classification technique based on cucumber plant leaves [7]. This method could detect five different cucumber leaf diseases. The proposed technique incorporated five different stages which are contrast enhancement, infected area partitioning, feature derivation, feature selection and lastly categorization. For these stages, the contrast enhancement is used as a preprocessing tool to boost the fraction of distinguishability between the infected area and background, which is further segmented by SHSB (Sharif saliency-base) method. The outcomes of this stage is further enhanced by fusing the suggested saliency method with active contour segmentation. Moreover, VGG-19 and VGG-M models are used for feature extraction. Later, it determines the best features rooting into local entropy, local standard deviation and local interquartile range method. The VGG-19 model utilizes 16 convolutional and Relu layers, fully connected layers (FC6, FC7, FC8) and one softmax function were used for classification. Additionally, the VGG-M model comprised 5 convolution layers, 3 FCL, 7 Relu layers, 1 softmax, 1 norm layer and 1 pooling layer. For strong selection the serial-based approach is used to fuse the VGG-19 and VGG-M feature vectors. Lastly, to sum up the results, the proposed method has given a maximum disease acknowledgement accuracy of 98.4% with an average execution time of 16.51 seconds. However, the drawbacks of this method include its intricate structure which requires a good amount of time for the training phase when a large dataset is used. Additionally, the SHSB method fails during segmentation of poor contrast images.

In their paper, Zhao et al., proposed a two step multi-task continual learning tech-

nique in order to determine 10 different categories containing both healthy and unhealthy leaf samples of three different species [23]. Here, in the first stage, they introduced a feature learning stage which can be scaled, representing the old features as unchanged. A new feature extractor is used to train the new and previously saved data; this is done in order to expand the features. The training is taking place with a cross-entropy loss method. For the second step, they sorted out if the main parameters are being retained, this is done using an auxiliary loss. Moreover, they mitigate the fluctuations in weight parameters. This keeps the old features separable and promotes the model's acquisition of diversity, new ideas, and discriminative features. Later, VGG11, ResNet18 and DenseNet121 models are enhanced with the two stage continual learning technique. This whole scenario is applied so that the model can cut off catastrophic forgetting when a multi domain dataset is used. Furthermore, this method improves the robustness of pre-trained deep learning models. When solely VGG11, ResNet18 and DenseNet121 were used, the accuracy was 82.35%, 83.22% and 88.53% respectively. However, while these models, i.e VGG11, ResNet18 and DenseNet121 were incorporated with the 2 stage continual learning technique, they achieved an accuracy of 89.81 %, 93.84%, 94.47% respectively. Lastly, limitation of the proposed model includes that a much deeper and a large number of environmental factors needs to be considered. Secondly, feature visualization and artificial intelligence interpretability techniques could be combined in order to expand the use of digital image processing and continuous learning in the detection of plant diseases.

Alharbi et al., proposed a continual learning technique consisting of a few-shot learning method which comprises the EfficientNet in order to classify 18 different wheat diseases based on leaf images [17]. Few-shot learning is chosen as it is not data hungry. Alongside, using the complete dataset in order to retrain the model is not required for extending the trained model with incoming classes. Moreover, previous learning is also incorporated for the expansion of trained models. FSL along with EfficientNet contributed in making the network computationally efficient. Attention-based structures are used to decrease false positives even after using less data. Support sets are used to train a model with less data to introduce a new task, while query sets use those knowledge to establish new examples. Hence, this model can solve major problems like catastrophic forgetting and data hungriness. The proposed model has given an accuracy of 93.45% with only 40 images in the query set. The results of the few-shot (5-way-5-shot) technique are accuracy 93.19%, precision 91.35% F1 score of 91.43%. However, even though the model requires limited samples, few-shot learning has a restricted generalization.

Asaari et al., focused on detecting plant stress in crops of eggplant while in the juvenile vegetative phase by using deep learning based detection models [19]. The plants were categorized into early stress, severe stress and healthy. For building a plant stress detection model three deep learning object detection algorithms were used which are Single Shot Detector (SSD), You Only Look Once version-3 (YOLOv3) and You Only Look Once version-4 (YOLOv4). For data preparation, in an open-air garden that receives direct sunlight, a 30cm $\times$ 30cm pot was placed and ten eggplants were planted which was categorized in three groups: early stress with 3 plants, severe stress with 4 plants and healthy with 3 plants. The plants were

grown differently with varying watering conditions since the early stress plants and healthy plants group were watered twice a day comparatively providing less water to the early stress plants group whereas to the the severe stress group the plants were watered only once in a week. The dataset of 335 images for this study was created by capturing the images weekly at various angles.The leaf conditions were categorized using visual inspection, as for healthy plants, a greenish color was observed, and for early stress or severe stress, the severity of symptoms such as powdery mildew or downy mildew, wilting, and yellowing was observed. Data augmentation techniques were applied to further increase the dataset and then the dataset was divided randomly in testing and training sets. The results showed that the mean average precision value for YOLOv3 was 52% and detection speed was 40 msec whereas 83% was achieved for YOLOv4 with 32 msec detection speed and 56% was achieved for SSD with a detection speed of 34 msec. The results clearly show that YOLOv4 performed much better than the other two models. However, the paper aims to find the plant stress conditions at a certain growth stage overlooking other growth stages. Also, the study primarily focuses on water stress conditions, overlooking other factors that might impact the plant health.

Rizvee et al., approach to detect the seven common mango leaf diseases using LeafNet which is a convolutional neural network (CNN) based model [21]. The paper uses MangoleafBD dataset which contains 8 classes including images of healthy leaves of mango and seven mango leaf diseases which are Powdery Mildew, Cutting Weevil, Gall Midge, Anthracnose, Sooty Mould, Bacterial Canker, and Die Back. The images of mango leaves were captured from four orchards by mobile phone cameras in Bangladesh. The dataset initially contains 1800 mango leaves images but after augmentation it contains 4000 mango leaves images in total with each class representing 500 images. The architectures used in this study were AlexNet, VGG16, and LeafNet. LeafNet outperforms state-of-the-art models such as AlexNet and VGG16, with average accuracy of 98.55% , precision of 99.508%, recall of 99.45%, F-score of 99.47%, and specificity of 99.878% in a 5-fold cross-validation. However, the paper focuses on mango leaf diseases for only Bangladesh and only seven mango leaf diseases overlooking other diseases and the diseases that might impact the mango leaves.

Durmus et al., aims to detect diseases on the leaves of tomato plants by using deep learning [3]. It uses a robot with RGB cameras which can navigate autonomously or manually in tomato fields or greenhouses. The PlantVillage dataset has been used in this paper. Images of tomato leaves of ten different classes are used which are septoria leaf spot, bacterial spot, spider mites, late blight, early blight, yellow curl virus, mosaic virus, target spot, and leaf mold, and also healthy images of tomato leaves are included. The study utilizes two models called AlexNet and SqueezeNet which are two different deep learning network architectures. The tomato images of the PlantVillage dataset is used to test and train these two models which are done on Nvidia Jetson Tx1 that is a mobile supercomputer. Comparatively, the performance of AlexNet is slightly better than SqueezeNet as it can be seen from the test set results that the accuracy obtained for AlexNet is 95.65% whereas the accuracy obtained for SqueezeNet is 94.3%. However, the cost could be a concern in developing such a robot as it uses a mobile supercomputer called Nvidia Jetson

Tx1 which is quite expensive.

Barbedo focuses on plant diseases by identifying individual spots and lesions without considering the whole leaf by using deep learning [6]. Image dataset was used. The dataset contains pictures with resolutions ranging from 1 to 24 MPixels that were taken with a variety of sensors, including compact cameras, DSLR cameras, and smartphones. The majority of the disorders are associated with fungi (77%), viruses (8%), various pests (6%), bacteria (3%), phytotoxicity (2%), algae (2%), nutritional deficiencies (1%), and senescence (1%). Manual segmentation was used to eliminate background details and concentrate on the leaves of the plants in the photos. Five different symptom types were taken into account during the segmentation process: isolated spots, powdery spots, widespread lesions, small and large lesions. The Matlab Neural Network Toolbox was used to apply transfer learning to a pre-trained GoogleNet CNN. Due to the superior performance of the GoogLeNet architecture it was chosen for this study. The network was trained using the following parameters: Number of Epochs was 5; Mini Batch Size was 64; Base Learning Rate was 0.001; Momentum was 0.9. The original and divided sets of all crops were used for each training round, and the average time was 13 minutes and 6.5 hours. It took almost 140 hours to train the CNNs using a single Nvidia Titan XP GPU, taking into account the 10-fold cross-validation and the full set of experiments. Even when up to ten diseases were taken into account, no crop had accuracy rates lower than 75%. These results show that plant disease detection and recognition can be achieved with deep learning techniques provided sufficient data is available, even though the dataset fails to include all possible applications.

Ahmed and Reddy utilized deep learning to detect plant leaf diseases using a mobile-based system [11]. The dataset was created with images higher than 96000 for 38 categories of most common plant diseases in 14 species of crops, including cherry powdery mildew, grape leaf blight, corn common rust, apple scab, apple black rot, and many more. For this study, a distributed system was created with parts that run on cloud servers and mobile devices. On the cloud server, a Convolutional Neural Network (CNN) model was developed that can receive images from farmers' mobile devices. The model detects objects, segments them semantically, and displays the disease category, confidence percentage, and classification time for each image. The model obtains an overall classification accuracy of 93.6% and the average prediction time was found to be 0.88 seconds. However, the model sometimes confuses leaves of similar phenology and common physiognomy characteristics of plants, including color, size, and canopy structure.

Author Ferentinos developed an automated method for detecting and diagnosing plant diseases using simple images of healthy and diseased leaves by training and evaluating certain CNN architectures [5]. The researcher used a big collection of 87,848 pictures of different plant leaves – tomato, potato, pumpkin, raspberry, soybean, squash, strawberry, peach, bell pepper, and watermelon for training and testing the CNN models. This dataset has images of both healthy plants and plants that are infected with diseases. The dataset is divided into 58 classes, each representing a pair of a plant and a disease it might have. The researcher collected images in a controlled lab environment (laboratory setups) and in real fields where

plants are grown (cultivation fields). About 37.3% of the images were taken in real cultivation conditions. The dataset covers 25 different plants, including a good mix of images from controlled lab settings and the more complex real cultivation fields. Furthermore, the researcher used five different basic CNN architectures: AlexNet, AlexNetOWTBn, GoogLeNet, Overfeat, and VGG. The Torch7 machine learning computational framework, which is built using the LuaJIT programming language, was used to create these models and their training and testing procedures. The researchers applied three different approaches to train and test the models, (1) For training the models, they used 70,300 images, and the rest 17,548 images to see how well the models could identify diseases in new pictures. Then used a Python script to randomly select images for both training and testing, making sure that the percentages of images taken in lab conditions versus real cultivation conditions were similar in both sets. (2) Just reduced the size of the images to 256x256 pixels and kept the 80/20 split but did not use grayscale versions of the images or separate the leaves from the background and rest the same as the first approach. (3) Picked 12 classes with images from both places (lab and real field) and made two models for each class: (i) One model was trained with lab images and tested with field images. (ii) Another model was trained with field images and tested with lab images. The VGG and AlexNetOWTBn designs were the most successful among the rest of the models with the best success rate and the lowest final average testing error respectively. The VGG model earned the greatest successful classification percentage at 99.53%. Success rates are much lower than when utilizing laboratory and field-condition images. Results indicate improved model performance when trained with field-condition images and asked to identify laboratory-condition images (achieving almost 68% success rates). However, success rates are much lower (approximately 33%) when trained simply on laboratory-conditions photos and asked to identify field-conditions images. Image identification in actual cultivation conditions is more challenging than in lab conditions. On the testing dataset of 17,548 photos, the final model (VGG) had a success rate of 99.53%, which means that 17,466 images were properly classified but failed for the rest because the model misclassified some faulty images. Moreover, the current dataset needs a broader range of plant species and diseases. In addition, the dataset that made up the training set also included the testing dataset that was used to evaluate the models is another critical point that needs fixing. Most importantly, this final model cannot solve catastrophic forgetting.

The paper by Mohanty et al., investigated the feasibility of using deep learning and cell phones to detect plant diseases [2]. Developing a smartphone app for early plant disease detection and management is a top priority, particularly in food-dependent regions. The dataset used in this paper includes 54,306 images of leaves from various plants of apple, blueberry, cherry, corn, grape, orange, peach, bell pepper, potato, raspberry, soybean, squash, strawberry, and tomato. These images represent 38 distinct classes, each of which is a mix of crop type and disease status. This dataset contains images of both healthy and diseased plants clicked in different orientations. The researchers conducted many tests using different versions of the dataset, including color photographs, gray-scaled images, and segmented images, to assess the model's generalizability and ability to train under varied circumstances. In contrast to the dataset used by Ferentinos (2018), PlantVillage is much larger. Several other splits between the dataset's training and testing sets were investigated by the re-

searchers. These included 80/20, 60/40, 50/50, 40/60, and 20/80. In addition, they united all photos of the same plant leaf in the training or testing sets to avoid biases. The researchers used AlexNet and GoogLeNet architectures for the classification. Using the PlantVillage dataset, they compare the two architectures' performance using two methods: first, by training the model from scratch; and second, by modifying already learned models (from the ImageNet dataset, another popular dataset used for such cases) using transfer learning. They set up 60 experimental configurations by varying the parameters: two models (AlexNet or GoogLeNet), training mechanism (transfer learning or training from scratch), dataset type (color, grayscale, leaf segmented), and training-testing set distribution (80/20, 60/40, 50/50, 40/60, 20/80). They have also used the same hyper-parameters (solver type, base learning rate, learning rate policy, etc) for all the experiments. They used their special fork of Coffee (open-source framework for deep learning) for every experiment. The researchers used some criteria to assess the performance, including overall accuracy, mean F1 score, mean recall, and mean precision. In the different experimental configurations, AlexNet was able to get an overall accuracy of 85.53% to 99.34% and GoogLeNet was able to achieve an overall accuracy of 88.23% to 99.34%. When using transfer learning on the colored dataset with an 80-20 train-test set distribution, AlexNet achieved an accuracy of 99.27%, and GoogLeNet 99.34%.Thus GooLeNet is better than AlexNet.Testing the model on modest, validated datasets sourced from reliable internet sources reduces its overall accuracy to around 31%. In addition, diseases that only show up on the top surface of leaves are the only ones the model is presently targeting. Moreover, when evaluated on captured photographs under different settings from those used for training, the model's accuracy is significantly diminished.

The main objective of the paper by Krishnaswamy Rangarajan and Purushothaman was to provide farmers with the necessary tools to detect and control eggplant crop diseases via the use of cutting-edge technologies [8]. Careful curation went into creating a complete dataset that showcases five main eggplant diseases (Epilachna beetle, Cercospora leaf spot, Little leaf disease, Tobacco Mosaic Virus, and Two-spotted spider mite) via pictures (using different smartphone cameras) of leaves taken in controlled lab settings as well as in the wild. To test how well various models classified diseases, the researchers utilized VGG16, AlexNet, GoogLeNet, ResNet101, and DenseNet201. Specifically, they found that a remarkable 99.4% accuracy rate in disease categorization was achieved by combining VGG16 with RGB photos taken in real fields. Furthermore, the researchers explored feature extraction from various parts of the VGG16 model, which led to the discovery that certain features were quite good at disease categorization. Regardless of these developments, this research has demonstrated the limitations of proper disease classification in a controlled laboratory setting, with some diseases posing their unique hurdles. Both farmers and academics involved in plant disease detection may benefit greatly from the insights provided by this research, which constitutes a substantial contribution. In addition to outlining possible future research options in this vital and ever-changing agricultural subject, it suggests a practical instrument that farmers may use to detect eggplant diseases.

In his paper, author Wang introduced a new method called ViT-TV for multi-disease

image recognition in crops, Total Variation(TV) distance loss to facilitate continual learning [24]. This allows models to learn new tasks without forgetting their previous data. Moreover, the authors used a dataset named PlantDiseaseCL (created specifically for evaluating continual learning methods) which includes a total of 30,863 images (taken from the internet) of various crop diseases, like those affecting corn, apple, potato, and pepper plants, to train the models in a hope of accurate identification and distinguishment of these diseases over time. The authors used some advanced models like Efficientnet-Lite0, Regnetx-02, ConvNeXt-S, and ViT-S/16 to compare the accuracies of the models. To provide predictions on new disease data that closely match the real distribution while keeping consistency with previous disease data, the authors aim to alter the model parameters to minimize the TV distance. Even without sample playback, the ViT-TV model may learn new crop disease detection tasks by optimizing this composite loss, which retains knowledge of earlier tasks. In other words, by measuring the difference between attention distributions of old and new tasks, ViT-TV minimizes the total variation in distance loss, ensuring consistency of attention and thus prevention of catastrophic forgetting during learning. The ViT-S/16 model, along with other models, was evaluated for its classification capabilities using joint learning experiments. To train the model, they adjusted its parameters so that the difference between the expected and actual results was as little as possible. In this process, an optimization method was used – AdaMax optimizer. In addition, they used the ViT-S/16 model to perform a number of class incremental learning tests on PlantDiseaseCL. Partitioning the whole dataset into training and testing sets and then breaking it down into 3-step and 5-step learning procedures were all part of the incremental learning procedure. After each training step, the model would assess the Average Accuracy of all learned categories on the testing set. Both the 3-step and 5-step learning procedures were designed to evaluate their suggested ViT-TV approach in comparison to other continual learning methods such as Baseline: Fine Tuning and Freezing, Exemplar Replay Approach: EEIL, iCaRL, and IL2M, Zero-Exemplar Approach: LwF, EWC, etc. A 95.38% accuracy rate in joint training was achieved by ViT-S/16 with ViT-TV compared to other models. In continual learning, compared to other methods of continual learning, ViT-S/16 with ViT-TV average accuracy in 3-step learning is 70.77%, and in 5-step learning, 56.61% which is much higher. In continual learning, the average accuracy is not that high, thus needing more improvement in the architecture and training of the model. In addition, the dataset should have a few more plants to work with.

2.2 Elastic Weight Consolidation (EWC)

Elastic Weight Consolidation (EWC) [4] is an algorithm that is designed to focus to overcome the problem of catastrophic forgetting in neural networks, which arises when a neural network trained on multiple tasks in a sequential manner tends to forget what it has learned previously while learning new ones. The inspiration of EWC is drawn from the mechanisms of synaptic consolidation observed in biological brains, which involves certain synapses becoming less flexible and more stable over time to retain the knowledge learned previously.

In neural networks, when a task is learned it involves adjustments to the network's weights and biases in order to minimize a loss function. When the network is trained on a new task, EWC introduces a constraint on the update of important weights to protect the performance on previously learned tasks. This constraint makes sure that the weights are not updated much, ensuring the preservation of knowledge from prior tasks. This is accomplished by adding a quadratic penalty to the loss function during training, which functions as a sort of "spring" that pulls these parameters, that is, the weights, back towards their initial values. The importance of each parameter is measured using the Fisher Information Matrix. It gives an indication of how significantly a parameter affects the network's output. The weights having higher Fisher Information values are regarded as more important, and the weights are updated with stronger penalties.

Neural network training can be viewed from a probabilistic approach to help explain this constraint preference and identify the most crucial weights for a given task. The conditional probability $p(\theta|D)$ can be calculated by applying the Bayes' rule from the parameters prior probability $p(\theta)$ and the data's probability $p(D)$ by using this equation (2.1):

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}|\theta) + \log p(\theta) - \log p(\mathcal{D}) \tag{2.1}$$

The network parameters are represented by θ, whereas the dataset is represented by D. $p(D|\theta)$ is equal to the negative of loss function. The split of data is assumed to be into two different tasks, D_A for task A and D_B for task B. In order to incorporate the prior from the previous task and concentrate on the new one, the posterior probability can be rearranged as follows:

$$\log p(\theta|\mathcal{D}) = \log p(\mathcal{D}_B|\theta) + \log p(\theta|\mathcal{D}_A) - \log p(\mathcal{D}_B) \tag{2.2}$$

This equation (2.2) demonstrates that $p(\theta|D_A)$ contains the information about task A and helps in identifying crucial parameters for the task. Using this, the EWC algorithm regularizes the parameters according to their significance for task A by incorporating a penalty term into the loss function for task B as demonstrated below:

$$\mathcal{L}(\theta) = \mathcal{L}_B(\theta) + \frac{\lambda}{2}\sum_i F_i(\theta_i - \theta^*_{A,i})^2 \tag{2.3}$$

In the equation above, the loss for the new task B is $L_B(\theta)$, and λ determines the relative importance of the old task A and new task B. The index i determines each parameter's label, F_i represents the Fisher Information value for parameter i, and $\theta^*_{A,i}$ represents the optimal parameter value for task A.

EWC ensures neural networks' ability to remember and apply knowledge from prior tasks by preserving important weights, and hence promotes continual learning.

2.3 Learning without Forgetting (LwF)

Learning without Forgetting [1], LwF is a way to train Convolutional Neural Networks (CNNs) to learn new tasks without losing success on tasks they have already learned (old tasks), and it does this without needing access to the original training data for the old tasks. This way of doing things works especially well when it

is not possible to store and retrain all past data because of privacy or space problems.

LwF only uses the new task data when it is being trained. It tries to get the best results on both the new task and the old tasks while keeping the network's responses the same. The main idea is to keep the network's current abilities by making sure that the outputs for the old tasks stay the same while the new task is being trained. The first step in the process is to record how the network that has already been trained on old tasks reacts to the new task images. Then, these outputs(probabilities) are used as a guide to make sure that the results from the first tasks are kept while the network is trained on the new ones. Then there are two loss functions to be minimized.

LwF Algorithm:

A pre-trained network, which has already learned to carry out certain tasks, is the starting point of this algorithm. The network consists of shared parameters (θ_s), original task parameters (θ_o), and new task parameters (θ_n).

Keeping track of the first outputs of the task is the initial stage. The outputs (i.e., the probabilities assigned to each class for the previous tasks) of the pre-trained network are recorded for each new task image. These pre-recorded replies make sure the network doesn't lose its memory for the previous tasks.

The network's output layer is enhanced by adding more nodes (neurons) to manage the additional classes when additional tasks are added. Using new weights θ_n, these additional nodes are linked to the network's final shared layer. Two stages comprise training.

- The warm-up stage is where shared parameters (θ_s) and old task parameters (θ_o) are locked/frozen. Training is limited to the new task parameters (θ_n) until they effectively master the new task.
- The joint optimization: At this point, the network is trained to excel at both the old and new tasks at the same time, and all of the parameters (θ_s, θ_o, and θ_n) are no longer locked.

Different loss functions are used in the training process to balance the performance across tasks.

1. New task loss: The goal of the new task loss is to make sure that the model's predictions about the new tasks are accurate. To achieve this, cross-entropy loss/multinomial logistic loss (a standard loss for multiclass classification problems) is being calculated,

$$\mathcal{L}_{\text{new}} = - \, y_n \log \hat{y}_n \tag{2.4}$$

   where $\hat{y}_n$ is the predicted probabilities from the model and y_n is the one-hot ground truth label of the new task.

2. Old task loss: The goal of this loss is to make sure that the model's predictions for the old tasks are similar to its original predictions and it is named as Knowledge Distillation Loss,

$$\mathcal{L}_{\text{old}} = -\sum_{i=1}^{L} y'^{(i)} \log \hat{y'}_o^{(i)} \tag{2.5}$$

where l is the number of labels, y_o stands for the original probabilities that were recorded by the network that had already been trained, and $\hat{y}_o$ stands for current probabilities from the model for the old tasks. Both y_o and $\hat{y}_o$ are softened using a parameter T, temperature.

$$y_o'^{(i)} = \frac{(y_o^{(i)})^{1/T}}{\sum_j (y_o^{(j)})^{1/T}} \quad , \quad \hat{y_o}'^{(i)} = \frac{(\hat{y_o}^{(i)})^{1/T}}{\sum_j (\hat{y_o}^{(j)})^{1/T}} \tag{2.6}$$

3. The new task loss, the old task loss, and a balancing weight λ_o make up the overall loss.

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{new}} + \lambda_o \mathcal{L}_{\text{old}} \tag{2.7}$$

here λ_o is set to 1 as default however we may change it to put more emphasis on the old tasks or the new ones.

To minimize the loss functions during training, we can use optimization methods such as Adam or stochastic gradient descent (SGD) and using regularization methods like weight decay may help keep the model from becoming too used to the new job. This is because weight decay adds a penalty to the loss function, which encourages the use of simpler weight values.

$$\theta_s^*, \theta_o^*, \theta_n^* \leftarrow \arg\min_{\theta_s, \theta_o, \theta_n} \Big(\lambda_o \mathcal{L}_{\text{old}}(Y_o, \hat{Y}_o) + \mathcal{L}_{\text{new}}(Y_n, \hat{Y}_n) + \mathcal{R}(\hat{\theta}_s, \hat{\theta}_o, \hat{\theta}_n) \Big) \tag{2.8}$$

Here R stands for regularization which avoids overfitting.
To sum up, LwF provides an effective way to increase CNN capabilities without the downsides of catastrophic forgetting and the need for massive amounts of data storage. LwF preserves performance on previous tasks while learning new tasks by using Knowledge Distillation loss in addition to the cross-entropy loss.

Chapter 3

Dataset Description

The dataset [20] used in this paper is a secondary dataset which was published by researchers from Islamic University of Technology (IUT) and Bangladesh Agricultural Research Institute (BARI), Gazipur. In their dataset Orka et al., offers significant contributions, including the largest number of classes in an agro-domain dataset and it is the first agricultural multi-label classification problem. The dataset contains leaf image samples of 8 different major Bangladeshi crops. The image samples are taken using an iPhone 13 Pro Max with a 12 MegaPixel wide camera having an aperture lens of f/1.5 in order to assure the prevention of overexposure and distortion of the leaves appearance due to computational software. The aspect ratio was kept to be 1:1, ensuring the image size to be 3024 x 3024 pixels. The images were captured in natural light so that the trained algorithms on the dataset generalize.

The dataset consists of 4749 images with 57 distinct classes. Initially, it consisted of 5000 images with 110 classes but to ensure balanced class representation and prevent skewed predictive accuracies, classes with fewer than 10 samples were removed. The dataset contains 8 different crops namely ash gourd, bitter gourd, bottle gourd, cucumber, eggplant, ridge gourd, snake gourd and tomato with primary classes as healthy, disease, insect and nutritional deficiency and with secondary classes for disease, insect and nutritional deficiency. More details about the dataset are illustrated in the figures below:

Primary plant class graph:

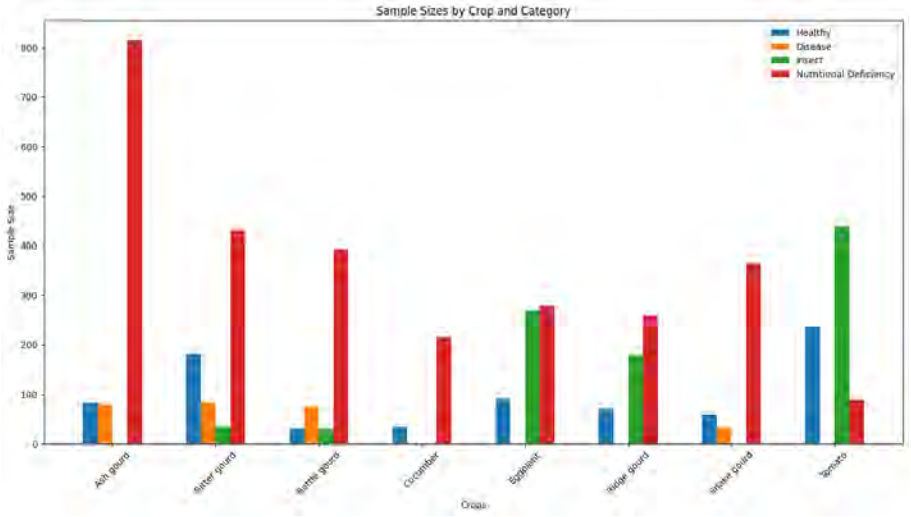


Figure 3.1: OLID I Dataset

Some samples of the dataset:

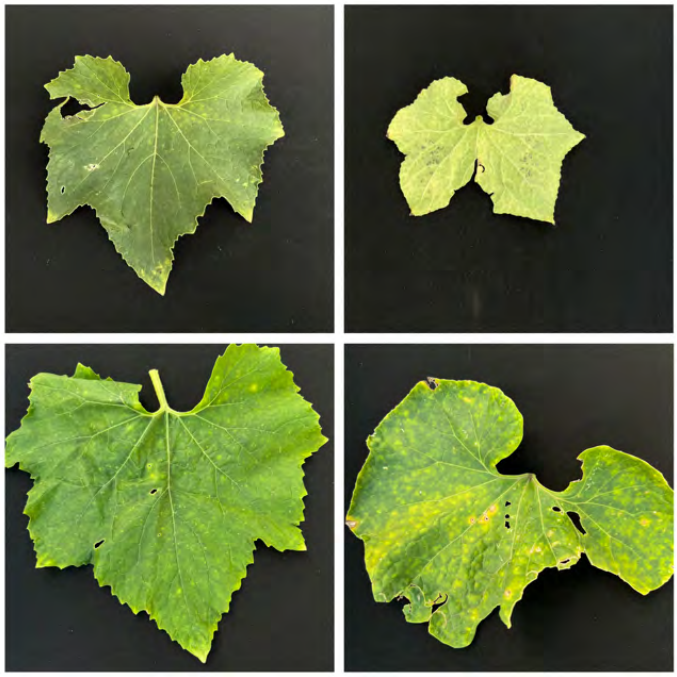


Figure 3.2: Samples of Ash gourd

Figure 3.3: Samples of Tomato

Figure 3.4: Sample of Bottle gourd

Secondary plant class pie charts:

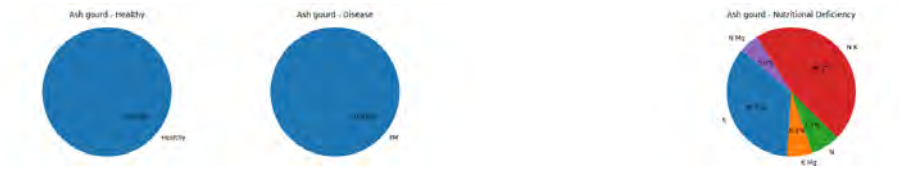


Figure 3.5: Ash gourd

Fig 3.5 demonstrates that for ash gourd, there are 3 primary classes which are healthy, disease and nutritional deficiency. The primary class, healthy, has 83 samples without any secondary class making it a 100%. It has one secondary class for disease which is powdery mildew (PM) which has 79 samples and no other secondary classes thus making it a 100%. Since it doesn't have any primary class for insects, no pie chart is generated for insects. For nutritional deficiency, there are 5 secondary classes, potassium deficiency (K) which contains 293 samples, potassium and magnesium deficiency (K Mg) which contains 53 samples, nitrogen deficiency (N) which contains 61 samples, nitrogen and potassium deficiency (N K) which contains 386 samples and nitrogen and magnesium deficiency (N Mg) which contains 42 samples.

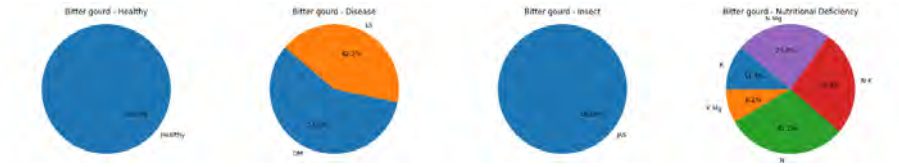


Figure 3.6: Bitter gourd

Fig 3.6 shows that for bitter gourd, there are 4 primary classes, which include healthy, disease, insect and nutritional deficiency. The primary class, healthy, has 181 sample images and it does not have any secondary class, making it a 100%. The primary class disease has 83 sample images of which the secondary class Downy Mildew (DM) has 48 samples and Leaf Spot (LS) has 35 samples. The primary class insect has one secondary class, Jassid (JAS), having a sample size of 35. Finally, the primary class nutritional deficiency has 488 samples, of which secondary classes Potassium deficiency (K), Potassium and Magnesium Deficiency (K Mg), Nitrogen deficiency (N), and Nitrogen and Magnesium Deficiency (N Mg) have 55, 40, 147, 128, and 116 image samples respectively.

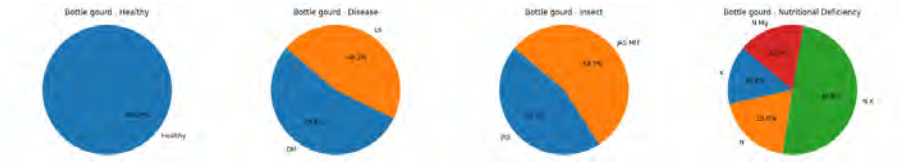


Figure 3.7: Bottle gourd

Fig 3.7 demonstrates that for bottle gourd, there are 4 primary classes which are healthy, disease, insect and nutritional deficiency. The primary class, healthy, has 31 samples without any secondary class making it a 100%. For disease, there are 2

secondary classes, downy mildew (DM) with 28 samples and leaf spot (LS) with 28 samples. For insect, there are 2 secondary classes, Jassid (JAS) with 24 samples and Jassid and Mite (JAS MIT) with 29 samples. For nutritional deficiency, there are 4 secondary classes: potassium deficiency (K) which contains 30 samples, nitrogen deficiency (N) which contains 39 samples, nitrogen and potassium deficiency (N K) which contains 102 samples, and nitrogen and magnesium deficiency (N Mg) which contains 34 samples.

Figure 3.8: Cucumber

Fig 3.8 shows that for cucumber, there are 2 primary classes which are healthy and nutritional deficiency. The primary class, healthy, has 34 samples without any secondary class making it a 100%. Since it doesn't have any primary class for disease and insect, no pie charts are generated for those. For nutritional deficiency, there are 3 secondary classes: potassium deficiency (K) which contains 50 samples, nitrogen deficiency (N) which contains 89 samples, and nitrogen and potassium deficiency (N K) which contains 76 samples.

Figure 3.9: Eggplant

Fig 3.9 describes that for Eggplant, there are 3 primary classes which are healthy, insect, and nutritional deficiency. The primary class, healthy, has 92 samples without any secondary class making it a 100%. Since it doesn't have any primary class for disease, no pie chart is generated for disease. For insect, there are 5 secondary classes: Epilachna Beetle (EB) with 74 samples, Flea Beetle (FB) with 36 samples, Jassid (JAS) with 34 samples, Mite (MIT) with 75 samples, and Mite and Epilachna Beetle (MIT EB) with 95 samples. For nutritional deficiency, there are 3 secondary classes: potassium deficiency (K) which contains 106 samples, nitrogen deficiency (N) which contains 67 samples, and nitrogen and potassium deficiency (N K) which contains 106 samples.

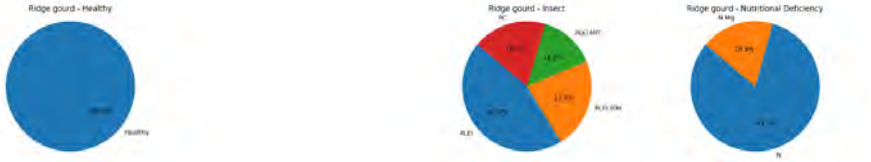


Figure 3.10: Ridge Gourd

Fig 3.10 shows that for Ridge Gourd, there are 3 primary classes, which include healthy, insect, and nutritional deficiency. The primary class, healthy, has 59 sam-

ple images and it has one secondary class making it a 100%. The insect class has 4 secondary classes: Pumpkin Leaf Eating Insect, Pumpkin Leaf Eating Insect with Insect Egg Mass, Pumpkin Leaf Eating Insect with Mite, and Pumpkin Caterpillar, each class having an image sample of 80, 40, 25, and 33 respectively. Additionally, the primary class Nutritional deficiency has subclasses Nitrogen Deficiency and Nitrogen and Magnesium Deficiency, with each secondary class having an image sample size of 152 and 34 respectively.

Figure 3.11: Snake Gourd

Fig 3.11 demonstrates that for Snake Gourd, there are 3 primary classes, which include healthy, disease, and nutritional deficiency. The primary class, healthy, has 59 sample images and it has one secondary class making it a 100%. The primary class Disease has one secondary class, Leaf Spot, of 33 image samples, making it a 100%. Finally, the primary class nutritional deficiency has 364 samples, of which secondary classes Potassium deficiency, Potassium, Nitrogen deficiency, and Nitrogen and Potassium Deficiency have 56, 102, and 206 image samples respectively.

Figure 3.12: Tomato

Fig 3.12 demonstrates that for Tomato, there are 3 primary classes, which include healthy, insect, and nutritional deficiency. The primary class, healthy, has 236 sample images and it does not have any secondary class, making it a 100%. The primary class insect infestation has 439 samples, of which there are 3 secondary classes: Leaf Miner, Mite, and Jassid, with Mite having an image sample size of 207, 200, and 32 respectively. Now, for the primary class Nutrition deficiency, it contains 153 images, of which the secondary class Potassium deficiency has 36 samples, Nitrogen deficiency has 47 samples, and Nitrogen and Potassium Deficiency has a sample size of 70.

Chapter 4

Methodology

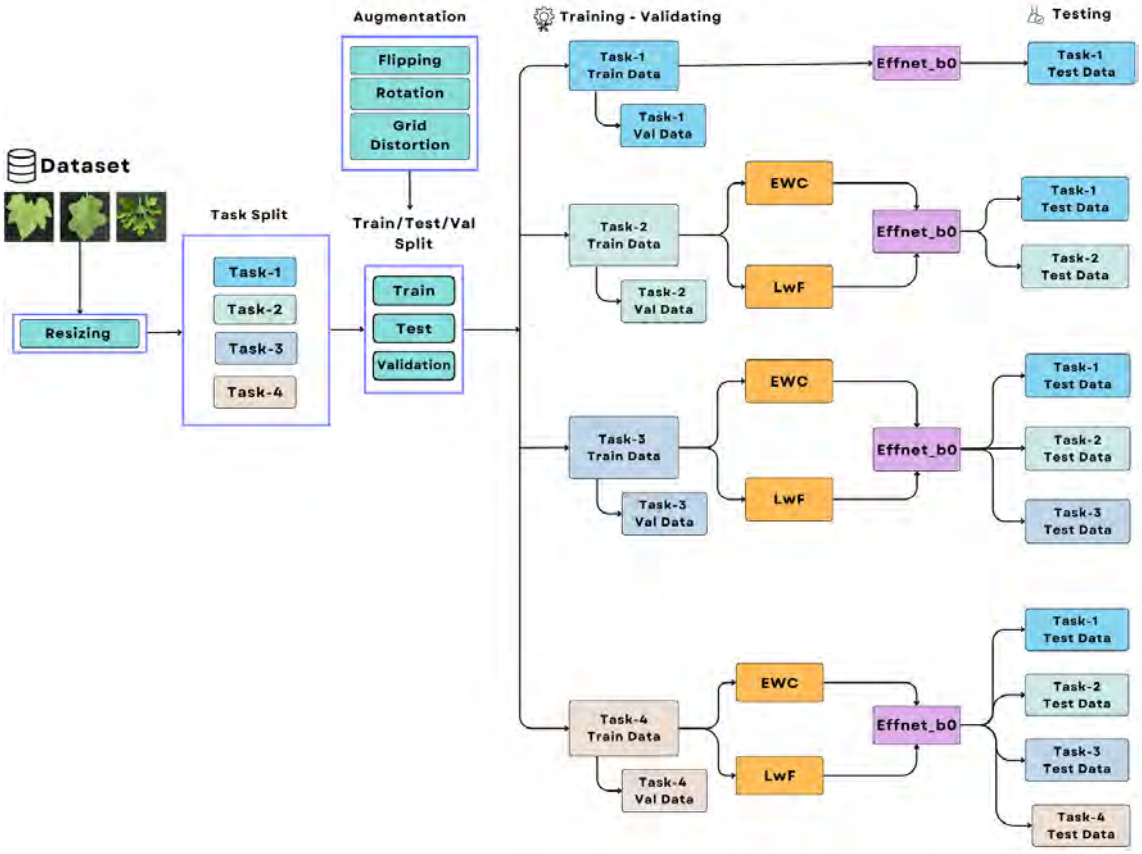


Figure 4.1: Methodological workflow of the study

Figure 4.1 provides a systematic overview of the methodology employed in this thesis, which incorporates continual learning techniques. Initially, the raw images of the dataset were pre-processed, in order to balance the dataset. Next, four types of plants were selected to further carry our study. Later, the data was divided into tasks to simulate the continual learning process. Finally, a base CNN model EfficientNet-B0 was used for training, incorporating continual learning methods (i.e., EWC and LwF) to evaluate the performance.

4.1 Dividing Dataset into Four Tasks

The dataset used in this study is composed of images from eight distinct plant types. For the purpose of this study, four different sorts of plants were selected: Ash Gourd, Bitter Gourd, Eggplant and Snake Gourd. For each selected species, the leaf images were classified into three health conditions: healthy, nitrogen deficiency and nitrogen-potassium deficiency. These health conditions represent the three target classes for each plant types.

The division of the dataset into four tasks is as follows:

- **Task 1**: Classify Ash Gourd leaf images into the three target classes: healthy, nitrogen deficiency and nitrogen-potassium deficiency.
- **Task 2**: Classify Bitter Gourd leaf images into the same three target classes.
- **Task 3**: Classify Eggplant leaf images into the same three target classes.
- **Task 4**: Classify Snake Gourd leaf images into the same three target classes.

By structuring the dataset into these four tasks, we create a continual learning scenario, where the model sequentially encounters new tasks. This approach allows us to study the model's ability to retain previously learned knowledge while adapting to new tasks. Each task focuses on classifying the health conditions of a different plant species, providing a clear progression for the continual learning process. After dividing into four tasks we split each task dataset into train, test and validation sets respectively into 80%, 10% and 10%.

4.2 Data Pre-processing

The images of the leaves were taken against a uniform black satin cloth backdrop to ensure consistency. With this option, segmentation—usually a necessary step to separate the target item (the leaf) from complicated backgrounds, can now be discarded. Consequently, there is no need to do any pre-processing to eliminate background noise.

This dataset is very well-organized, with photos sorted into folders according to their correct annotations. Therefore, eliminating the need for lengthy preprocessing to clean or rearrange the data. Hence, the data loading and preparation process for machine learning models gets simplified.

Visual examination and thorough laboratory analysis were used to annotate images in the collection. This two-step process made sure that the labels were exact and of high quality, so they did not need to be cleaned up or re-labeled as often before they were used.

One major problem with this dataset was that it was imbalanced, which could create biases between features, resulting in poor performance. Therefore, image augmentation was performed to enhance the dataset. Image augmentation, is a way to extend

the dataset by changing versions of images in the dataset. This helps machine learning models do better by giving them more training data. The augmentation was performed by resizing, horizontal flipping, vertical flipping, random rotation and grid distorting the images in order to extend and balance the dataset.

For our thesis four different plants were selected: Ash Gourd, Bitter Gourd, Eggplant, and Snake Gourd. For each plant, three classes were taken: healthy, nitrogen deficiency, and nitrogen-potassium deficiency. Originally Ash Gourd had 83 images in healthy class, 61 images in nitrogen deficiency, 386 images in nitrogen-potassium deficiency class. Bitter Gourd had 181 images in healthy class, 147 images in nitrogen deficiency, 128 images in nitrogen-potassium deficiency class. Eggplant had 92 images in healthy class, 67 images in nitrogen deficiency, 106 images in nitrogen-potassium deficiency class. Snake Gourd had 59 images in healthy class, 102 images in nitrogen deficiency, 206 images in nitrogen-potassium deficiency class. Through augmentation techniques, the images were augmented to increase data points for our thesis. For each class, i.e healthy, nitrogen deficiency and potassium-nitrogen deficiency of all the four plants, a total of 1500 images were created.

Performed Augmentation

- **Resizing images:** To make sure the consistency of input data, we resized all the images to 480×480 pixels. In doing so, it enhances the model's image-processing capabilities.
- **Horizontal Flip:** This augmentation method arbitrarily reversed an image along the horizontal axis (left to right) with a 50% chance. Such augmentation helped our models learn to identify an object even when that object seems mirrored.
- **Vertical Flip:** This augmentation has a 50% chance of arbitrarily flipping an image upside down along the vertical axis in our dataset images. This guaranteed that our models could withstand orientation shifts, much like horizontal flipping.
- **Random Rotation:** This augmentation randomly rotates an image by 90 degrees with a 50% probability in our dataset images. It helped our models enhance their ability to generalize by observing an object from different rotational angles.
- **Shift, Scale and Rotate:** This applies a combination of shifting, scaling and rotation to the image of the dataset with a 40% chance.
  - **Shifting:** This involves making small adjustments to an image along the x or y axes.
  - **Scaling:** This changes the size of the object in an image by zooming in or out.
  - **Rotation:** This method rotates an image in any direction, up to a maximum of 15 degrees.

  By introducing variations to the positions, sizes, and angles of the objects in an image, our models become better equipped to handle these changes.

- **Grid Distortion:** This process distorts an image by dividing it into a grid and randomly shifting the grid lines with a 30% probability. It somewhat distorts an image and helped our models in capturing aberrations and distortions seen in real-world images.

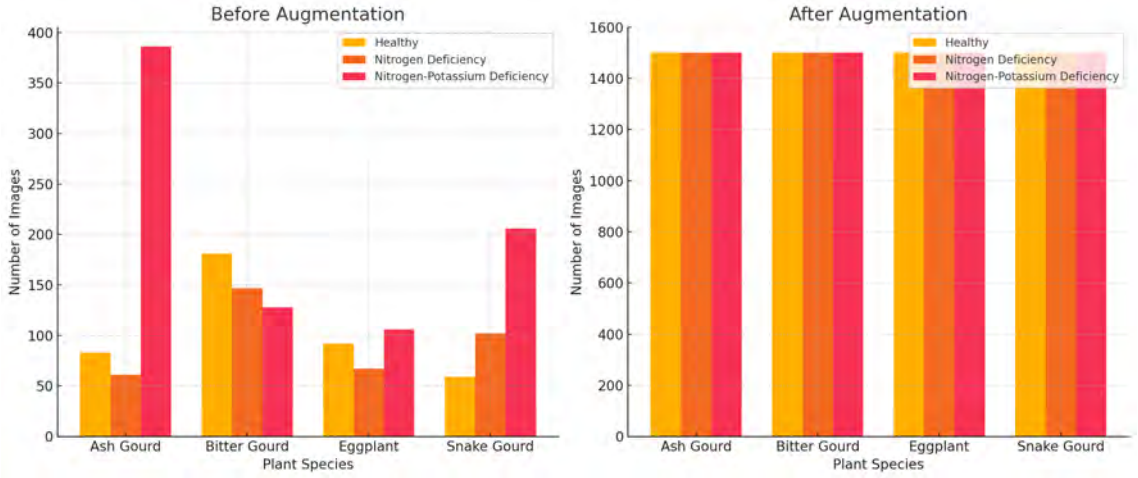


Figure 4.2: Comparing Image count before and after Augmentation

Figure 4.2 shows how the number of data points were increased by augmentation. After augmentation the train set of each tasks were increased to 1200, subsequently test set was increased into 150 and validation 150. In short each task has 1500 images for each class.

4.3 Image Classification Model: EfficientNet-B0

In this thesis, the base model EfficientNet-B0 was utilized to classify the class of stresses of each plant. EfficientNet-B0 is a modern convolutional neural network (CNN) architecture developed for efficient image classification. In order to balance accuracy and computing efficiency, it simultaneously optimizes the network's depth, width and resolution using a compound scaling method.

The EfficientNet-B0 model utilizes a set of pre-defined transformations on input images to ensure compatibility with its architecture, leveraging pre-trained weights on the ImageNet dataset. The transformation pipeline includes resizing the images to 256 pixels on the shorter side, followed by a central crop to 224x224 pixels, which matches the input size required by the model. Additionally, the images are normalized using the mean and standard deviation values ([0.485, 0.456, 0.406] for the mean, and [0.229, 0.224, 0.225] for the standard deviation), derived from the ImageNet dataset, ensuring that the pixel intensities align with the distribution seen during pre-training. This preprocessing ensures that the model can effectively generalize and extract relevant features from the input images, which is critical for image classification tasks.

The architecture EfficientNet-B0:

1. **Initial Convolutional Layer:** The initial convolutional layer takes RGB images (3 channels) as input and performs a 2D convolution with 32 filters, a kernel size of 3×3, a stride of 2, and padding of 1. With the help of 32 filters, this layer produces feature maps with reduced spatial dimensions while capturing low-level features. The convolution operation is followed by batch normalization and a SiLU (Sigmoid Linear Unit) activation function. The output of the convolutional layer is normalized by batch normalization (BatchNorm2d), which speeds up convergence during training, and SiLU adds non-linearity, which improves feature learning.

2. **First MBConv Block:** The second layer consists of a Mobile Inverted Bottleneck Convolution (MBConv) block. It starts with a 32-channel, 3×3 kernel depthwise convolution that works on each channel separately. While preserving performance, depthwise separable convolutions greatly lower computational complexity. Following the convolution, the SiLU activation function and batch normalization are used to add non-linearity and normalize the activations, respectively.

   The next step is the Squeeze-and-Excitation (SE) mechanism, which reduces the number of channels from 32 to 8 by applying global average pooling on the feature maps before passing them via a fully connected (FC) layer. After the channels are restored to 32 by the second FC layer, the feature maps are reweighted according to their global importance using a sigmoid activation function. After being scaled, the output is run through a 1×1 convolution, which lowers the channel number to 16. In order to further regularize the network, the block is dropped during training with a specific probability using Stochastic Depth, which has a probability of 0.0.

3. **Second MBConv Block:** Similar to the first MBConv block, the second MBConv block has an expanded number of channels but functions similarly. First, a 1×1 convolution is used to expand the number of input channels from 16 to 96. A 96-channel input is used for depthwise convolution, which is followed by batch normalization and SiLU activation. Using a similar fully connected structure, the SE block is set up to squeeze the channels from 96 to 4 before expanding them back to 96. A 1×1 convolution is used to compress the final output to 24 channels. To further improve the model's robustness, stochastic depth is used with a probability of 0.0125.

4. **Third MBConv Block:** The third MBConv block improves the model's capacity even more. The input is expanded from 24 to 144 channels in the first pointwise convolution, and then depthwise convolution with a stride of 1 and a kernel size of 3×3. The feature maps are compressed from 144 to 6 channels and then expanded back to 144 by the SE block. With the stochastic depth probability set to 0.025, the output has been reduced to 24 channels. This block helps to extract mid-level information from the input.

5. **Fourth MBConv Block:** The fourth block expands the input channels to 144, followed by a depthwise convolution with a larger kernel size of 5×5 and

a stride of 2, lowering spatial resolution. The SE block starts with global average pooling and then compresses and expands (144→6→144). The channels are reduced to 40 using a 1×1 convolution, and the block is further regularized by stochastic depth, which has a probability of 0.0375. More abstract representations of the input image are extracted with the aid of this block.

6. **Fifth MBConv Block:** The fifth MBConv block expands channels from 40 to 240, followed by depthwise convolution with a 5×5 kernel and stride of 1. The channels are reduced from 240 to 10 by the SE block using global pooling and a bottleneck structure. They are then expanded again to 240 before the output is reduced to 40 channels by a final pointwise convolution. The network is further regularized using stochastic depth with a probability of 0.05.

7. **Sixth MBConv Block:** The sixth block follows the same approach, expanding from 40 to 240 channels first, then using a 3×3 kernel and stride of 2 for depthwise convolution. The feature maps are compressed and expanded in the traditional way (240→10→240) by the SE block. The output channels are reduced to 80 via a pointwise convolution, and a 0.0625 probability of stochastic depth is employed. Because there are more channels in this block, it can capture more complex features.

8. **Seventh MBConv Block:** The seventh MBConv block increases the input from 80 to 480 channels, then performs depthwise convolution and SE block mechanisms (480→20→480). With a probability of 0.075, stochastic depth is applied once the output has been reduced to 80 channels. This layer helps with higher-level feature extraction, allowing the network to detect fine-grained patterns.

9. **Eighth MBConv Block:** The eighth MBConv block has a 3×3 depthwise convolution and an input expansion from 80 to 480 channels, which is similar to the previous ones. After compressing and expanding these channels (480→20→480) using the SE block, the output channels are reduced to 80 using a pointwise convolution. With a 0.0875 probability, stochastic depth is used.

10. **Final MBConv Blocks:** In the final MBConv block series, input channels are increased to 672, followed by a 5×5 depthwise convolution with a stride of 2 to further lower spatial resolution. After the SE block compresses and expands (672→28→672), the output is reduced to 112 channels by the final pointwise convolution. Regularization is achieved by applying stochastic depth, which has a probability of 0.1.

    Following a similar pattern, depthwise convolutions and SE operations are performed after the number of channels increases from 112 to 672 in subsequent MBConv blocks. After reducing the final output channels to 112, stochastic depth is gradually applied with probabilities of 0.125 and 0.1125. These blocks extract deep, high-level representations from the input image, which is important for accurate classification.

4.4 Applying Continual Learning in CNN Architecture

4.4.1 Elastic Weight Consolidation (EWC)

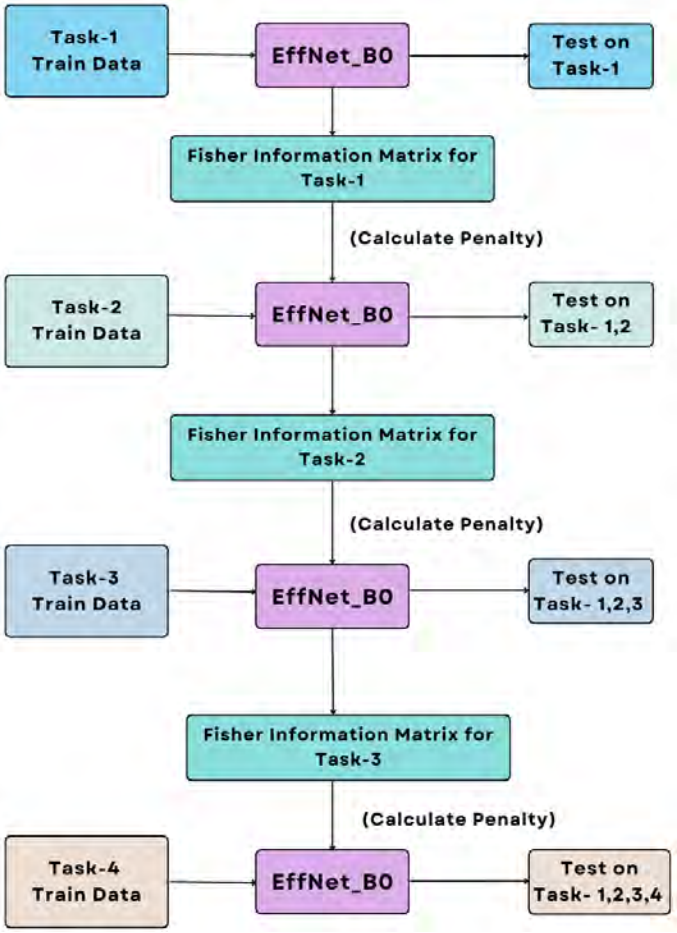


Figure 4.3: Training CNN using EWC

From figure 4.3 we can see that, Task-1 was trained without applying any continual learning method. After that, a penalty was computed for training Task-2 using Task-1 data with EWC. This penalty is added to the training loss while working on Task-2 to determine the optimal weights for both Task-1 and Task-2. For Task-3, a penalty was derived based on Task-2 to find the best weights for Task-1, Task-2 and Task-3, once more leveraging EWC. Finally, the model was trained on Task-4, applying the penalty calculated from Task-3, to achieve the best weights for all four tasks using EWC.

EWC was implemented in three main steps. Firstly, on the training, the Fisher Information Matrix was computed by estimating the diagonal using gradients of the loss function relative to the parameters of the model. Next, after each task is trained, we stored the mean parameters of the model. Furthermore, a regularization term was added to the loss function when new tasks were trained. This term penalized severe discrepancies between the current model parameters and the stored means as

weighted by the Fisher Information matrix. The regularization term was determined as follows:

$$\text{Loss} = \text{Loss}_{\text{new}} + \frac{\lambda}{2} \sum_i F_i(\theta_i - \theta_{i,\text{old}})^2$$

where λ represents the regularization strength, F_i represents the Fisher Information matrix, the current parameters of the model are represented by θ_i, and $\theta_{i,\text{old}}$ represents the parameters from old tasks.

The performance of the model for each task was measured using the test datasets' average loss and accuracy. The implementation of EWC was beneficial in retaining performance on tasks that were previously learned while allowing the model to learn new tasks. A substantial decrease in catastrophic forgetting is shown by our experimental results.

4.4.2 Learning without Forgetting (LwF)

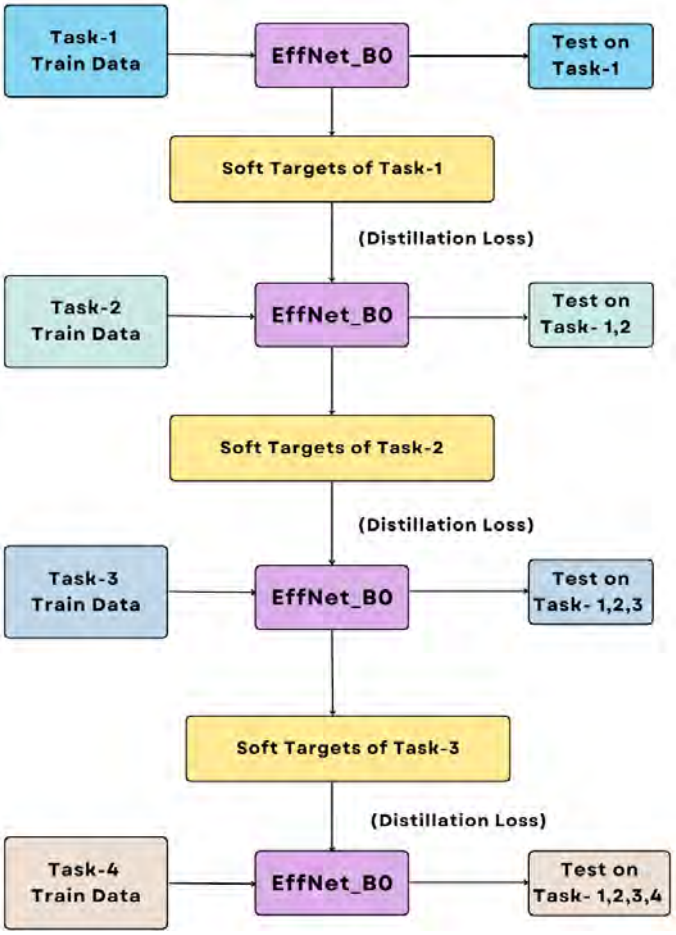


Figure 4.4: Training CNN using LwF

Figure 4.4 illustrates that our model underwent training on each task in sequential order using the Categorical Cross Entropy Loss function and the Adam optimizer with a learning rate of 0.001. Training our model with the first task data and saving

soft targets were the first steps in Task 1. Then, using soft targets from Task 1, our model was trained for Task 2. In Task 3, soft targets from Task 2 were used for training. Lastly, in Task 4, soft targets from Task 3 were used for training. To implement LwF, the training data was fed into the trained model to generate soft targets and then the distillation loss was calculated by scaling the softmax outputs of the current and old tasks by a temperature parameter and then use KL-Divergence.

Distillation Loss Function:

$$\text{Loss}_{\text{distill}} = \text{KL-Divergence}\left(\text{Softmax}\left(\frac{\text{output}}{T}\right), \text{Softmax}\left(\frac{\text{target}}{T}\right)\right) \times T^2$$

where T is the temperature parameter and KL stands for Kullback-Leibler divergence.

The new task loss and the distillation loss were combined to form the total loss, which was then balanced by a parameter.

Total Loss Function equation:

$$\text{Total Loss} = \alpha \times \text{New Task Loss} + (1 - \alpha) \times \text{Distillation Loss}$$

where α is the balancing parameter and the new task loss is the categorical cross-entropy loss function.

While training, our model was exposed to fresh task data while retaining information via the use of soft targets from earlier tasks. By preserving performance on older tasks while learning new ones, LwF successfully prevented catastrophic forgetting according to the evaluation.

Chapter 5

Results

For the purpose of this thesis, an experiment with multi-class classification on the dataset was carried. Firstly, leaves of four different plants were selected, each representing a different task while each of the tasks consists of 3 classes: task-1 (Ash Gourd: healthy, nitrogen deficiency and nitrogen-potassium deficiency), task-2 (Bitter Gourd: healthy, nitrogen deficiency and nitrogen-potassium deficiency), task-3 (Eggplant: healthy, nitrogen deficiency and nitrogen-potassium deficiency) and lastly task-4 (Snake Gourd: healthy, nitrogen deficiency and nitrogen-potassium deficiency). Additionally, in each of the respective classes of every task, there are three sets of data, train, test and validation. The train sets of each class present in every task, consists of 1200 images, while the test and validation sets contain 150 and 150 images respectively.

5.1 Evaluation Measures

We evaluated the performance of our model in classifying healthy and diseased cases using various evaluation metrics. These metrics provide a comprehensive assessment of the performance of the CL-applied CNN model discussed in our thesis.

Accuracy

The primary metric employed was accuracy, which indicates how accurate all of the model's outputs are:

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{5.1}$$

This measure provides a high-level summary of all classes' performance for the model.

Precision

Calculated for each class, precision evaluates the model's capacity to prevent false positives:

$$\text{Precision} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Positives}} \tag{5.2}$$

High precision indicates that when the model predicts a specific class type, it is likely to be correct.

Recall

Recall, also known as sensitivity, evaluates the model's capacity to identify all positive instances:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}} \tag{5.3}$$

A high recall indicates that all cases of a specific class type may be detected by the model.

F1-Score

The F1-score provides a balanced measure of precision and recall. It measures the harmonic mean of precision and recall.:

$$\text{F1-Score} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{5.4}$$

5.2 Result Analysis

In this experiment, there is one baseline model and two models, empowered with EWC and LwF respectively for multi class classification.

Table 5.1: Multiclass Classification Accuracy on Task-1, Task-2, Task-3, and Task-4 using different training methods

		Task-1 Testing Accuracy	Task-2 Testing Accuracy	Task-3 Testing Accuracy	Task-4 Testing Accuracy
Training without CL	Task-1	97%			
	Task-2	60%	94%		
	Task-3	47%	60%	98%	
	Task-4	**42%**	**55%**	**57%**	**97%**
Training with EWC	Task-1	97%			
	Task-2	60%	93%		
	Task-3	37%	60%	98.5%	
	Task-4	**60%**	**65%**	**66%**	**98%**
Training with LwF	Task-1	98%			
	Task-2	44%	72%		
	Task-3	53%	43%	75%	
	Task-4	**60%**	**61%**	**55%**	**75%**

Table 5.1 demonstrates the summary of the performance of the EWC and LwF methods on four different tasks. There are three main steps in this experiment. In the first step, the experiment demonstrates how catastrophic forgetting occurs in CNN models. In the second step, EWC is applied to the same CNN architecture to mitigate catastrophic forgetting. In the last step, LwF is applied to overcome catastrophic forgetting.

Training Without CL: The base model, EfficientNet-B0, is first trained on the data of task-1 and tested on the same task, achieving 97% accuracy. Afterwards, moving on to task-2, where the CNN model was trained with task-2 data and tested on both task-1 and task-2 data. In this scenario, task-1 shows 60% accuracy, while task-2 achieves 94%. This illustrates a classic case of catastrophic forgetting, as training on task-2 causes a drop in task-1 performance. Next, the model is trained on task-3 data and tested on task-1, task-2, and task-3. Here, task-1 accuracy drops to 47%, task-2 to 60% and task-3 achieves 98%, again showing catastrophic forgetting. Task-1 and task-2 performance degrade when the model is trained on task-3. Finally, the model is trained on the task-4 dataset and tested on all four tasks. Task-1 achieves 42% accuracy, task-2 55%, task-3 57% and task-4 97%, once again demonstrating catastrophic forgetting.

Training with EWC: Elastic Weight Consolidation (EWC) was applied to the base EfficientNet-B0 model to mitigate catastrophic forgetting. Initially, the model was trained on task-1 and achieved 97% accuracy. Subsequently, the model was trained on task-2 data, and when tested on both task-1 and task-2, task-1 retained 60% accuracy while task-2 showed 93%. This demonstrates a reduction in forgetting compared to the model without EWC. Moving on to task-3, after training, the model was evaluated on all three tasks, where task-1 accuracy dropped further to 37%, task-2 retained 60%, and task-3 achieved 98.5%. Finally, the model was trained on task-4, and when tested on all tasks, task-1 showed 60%, task-2 improved to 65%, task-3 retained 66%, and task-4 reached 98%. This shows that EWC helps preserve the knowledge of previous tasks, although some performance degradation still occurs with the introduction of new tasks.

Training with LwF: Later, Learning without Forgetting (LwF) was also applied to the base EfficientNet-B0 model to address catastrophic forgetting. The model was initially trained on task-1, achieving 98% accuracy. After training on task-2, the model was tested on both tasks, where task-1 dropped to 44% accuracy while task-2 achieved 72%. Upon training on task-3, the model was evaluated on all three tasks, resulting in task-1 showing 53%, task-2 43%, and task-3 75% accuracy. Finally, after training on task-4, the model was tested across all tasks. Task-1 showed 60%, task-2 61%, task-3 55%, and task-4 achieved 75%. While LwF helps retain performance on previous tasks to some extent, the model still experiences a notable performance drop when new tasks are introduced.

Performance while training Without CL:

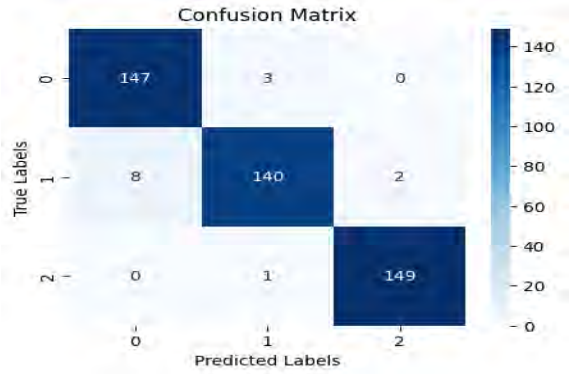
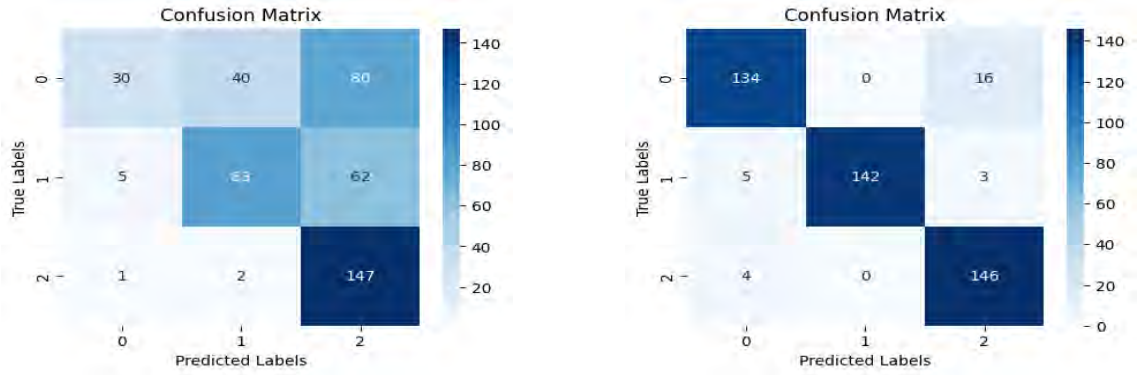


Figure 5.1: Without CL Task 1 (Test on task 1)

Figure 5.1 illustrates the confusion matrix of test results on task-1 while training on task-1.

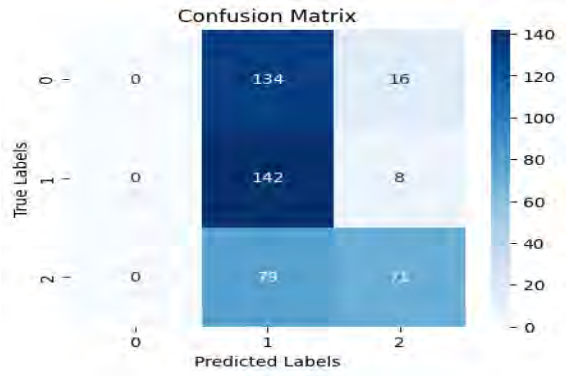


(a) Without CL Task 2 (Test on task 1)

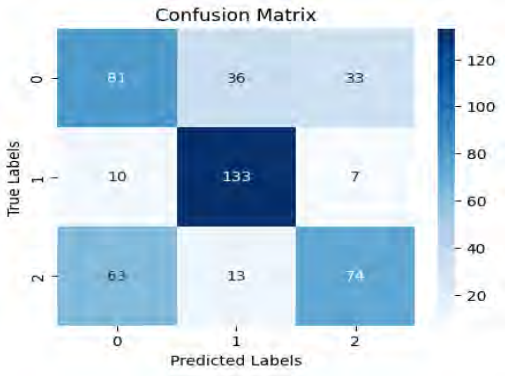
(b) Without CL Task 2 (Test on task 2)

Figure 5.2: Without CL Task 2 results across different tests

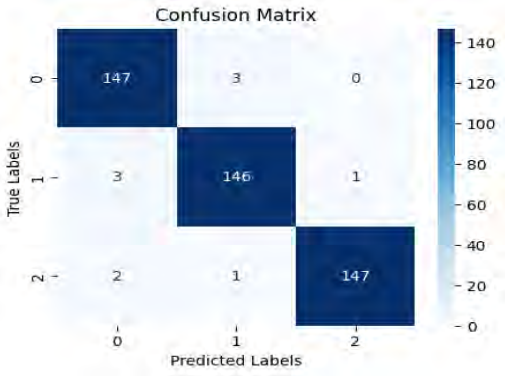
Figure 5.2 illustrates the confusion matrices of test results on task-1, task-2 while training on task-2.



(a) Without CL Task 3 (Test on task 1)



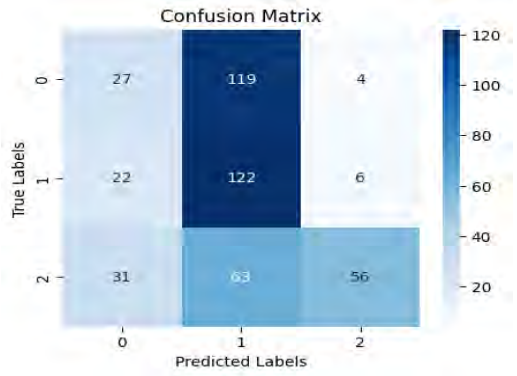
(b) Without CL Task 3 (Test on task 2)



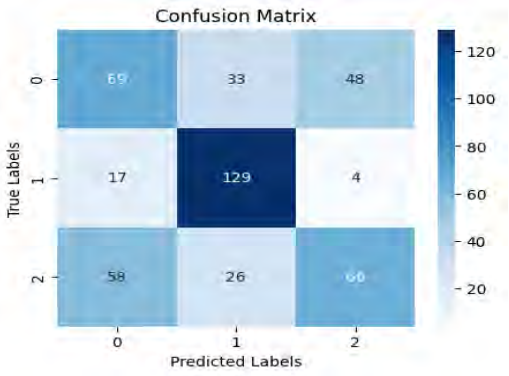
(c) Without CL Task 3 (Test on task 3)

Figure 5.3: Without CL Task 3 results across different tests

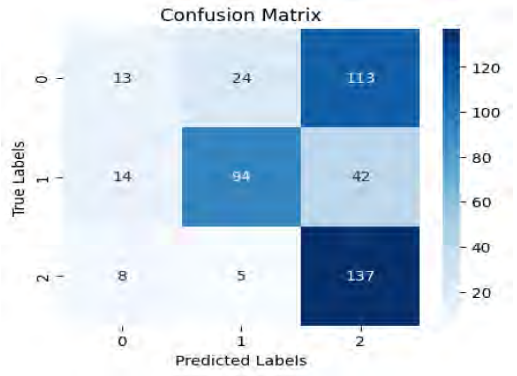
Figure 5.3 illustrates the confusion matrices of test results on task-1, task-2, task-3 while training on task-3.



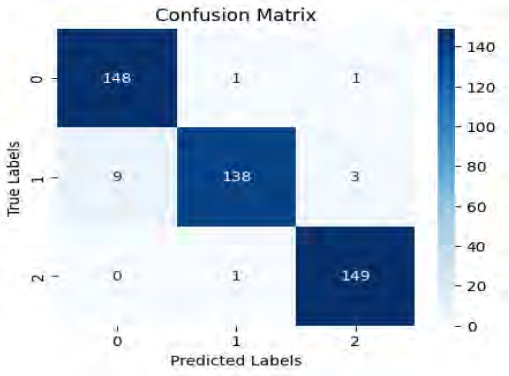
(a) Without CL Task 4 (Test on task 1)



(b) Without CL Task 4 (Test on task 2)



(c) Without CL Task 4 (Test on task 3)



(d) Without CL Task 4 (Test on task 4)

Figure 5.4: Without CL Task 4 results across different tests

Figure 5.4 illustrates the confusion matrices of test results on task-1, task-2, task-3 and task-4 while training on task-4.

Table 5.2: Precision, Recall, and F1 Scores for Task 4 trained model tested on Task 1, Task 2, Task 3, and Task 4 datasets, without CL

Tested on	**Class**	**Precision**	**Recall**	**F1 Score**
Task 1	Class 0	0.337	0.180	0.235
	Class 1	0.401	0.813	0.537
	Class 2	0.848	0.373	0.519
Task 2	Class 0	0.479	0.460	0.469
	Class 1	0.686	0.860	0.763
	Class 2	0.559	0.440	0.493
Task 3	Class 0	0.371	0.087	0.141
	Class 1	0.764	0.627	0.689
	Class 2	0.469	0.913	0.620
Task 4	Class 0	0.943	0.987	0.964
	Class 1	0.986	0.920	0.952
	Class 2	0.974	0.993	0.983

Performance while training With EWC:

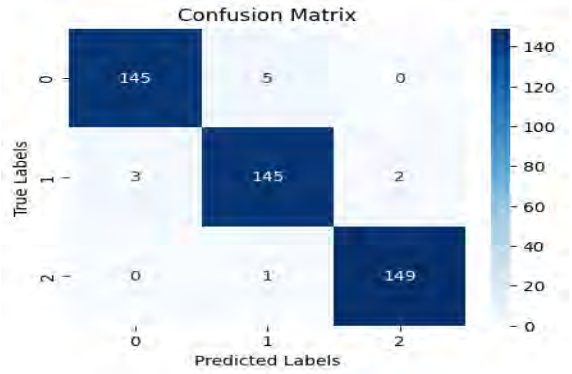
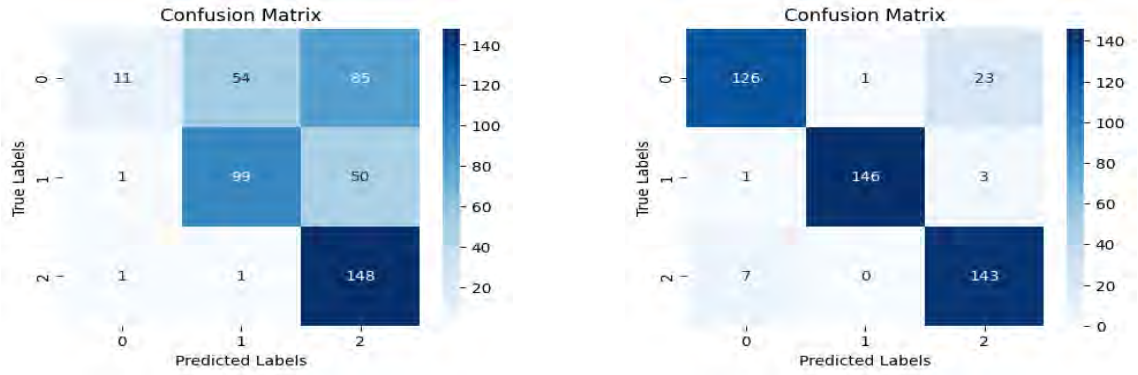


Figure 5.5: With EWC Task 1 (Test on task 1)

Figure 5.5 demonstrates the confusion matrix of test results on task-1 while training on task-1.

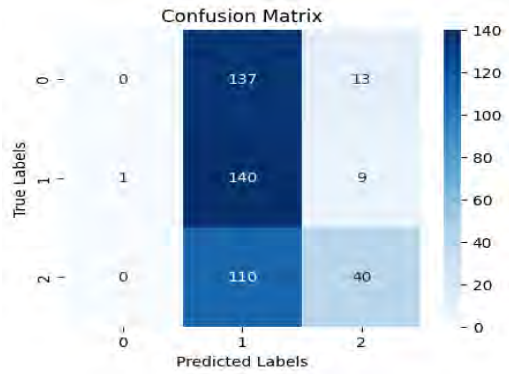


(a) With EWC Task 2 (Test on task 1)

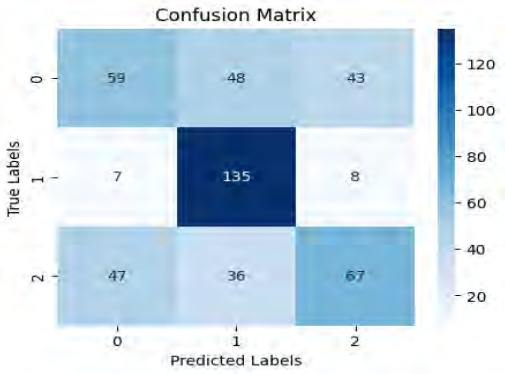
(b) With EWC Task 2 (Test on task 2)

Figure 5.6: With EWC Task 2 results across different tests

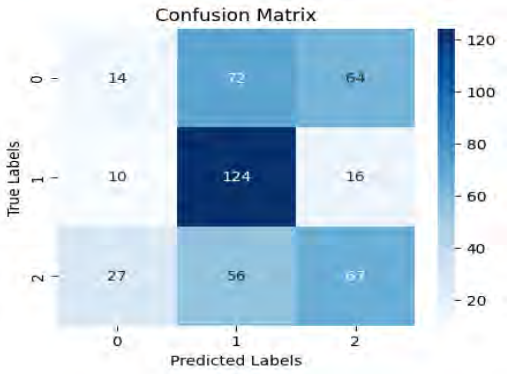
Figure 5.6 demonstrates the confusion matrices of test results on task-1, task-2 while training on task-2 with EWC.



(a) With EWC Task 3 (Test on task 1)



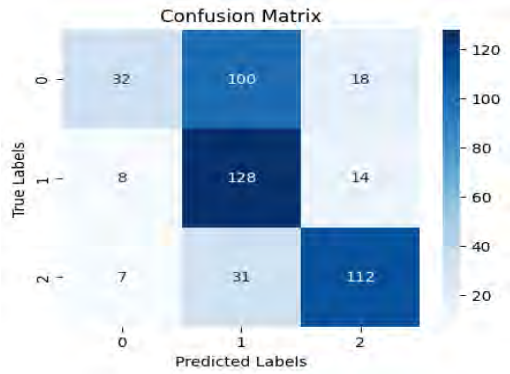
(b) With EWC Task 3 (Test on task 2)



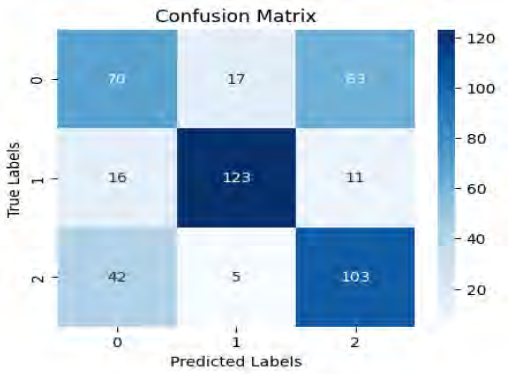
(c) With EWC Task 3 (Test on task 3)

Figure 5.7: With EWC Task 3 results across different tests

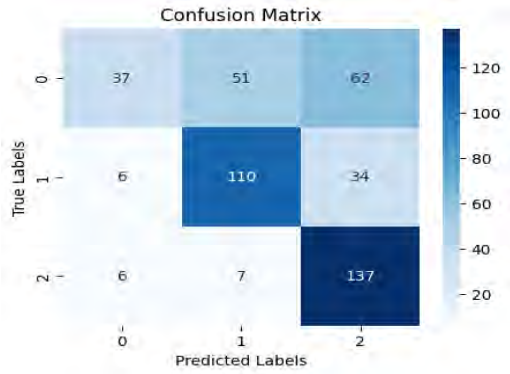
Figure 5.7 demonstrates the confusion matrices of test results on task-1, task-2, task-3 while training on task-3 with EWC.



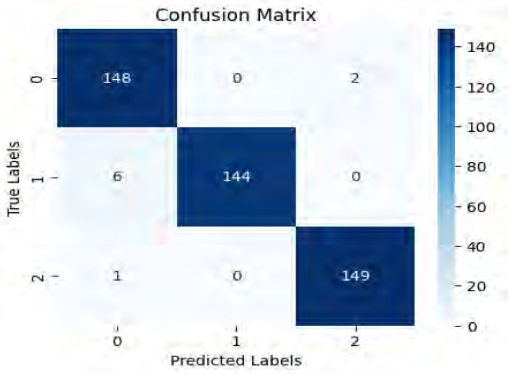
(a) With EWC Task 4 (Test on task 1)



(b) With EWC Task 4 (Test on task 2)



(c) With EWC Task 4 (Test on task 3)



(d) With EWC Task 4 (Test on task 4)

Figure 5.8: With EWC Task 4 results across different tests

Figure 5.8 demonstrates the confusion matrices of test results on task-1, task-2, task-3, task-4 while training on task-4 with EWC.

Table 5.3: Precision, Recall, and F1 Scores for Task 4 trained model tested on Task 1, Task 2, Task 3, and Task 4, applying EWC

Tested on	**Class**	**Precision**	**Recall**	**F1 Score**
Task 1	Class 0	0.681	0.213	0.325
	Class 1	0.494	0.853	0.626
	Class 2	0.778	0.747	0.762
Task 2	Class 0	0.547	0.467	0.504
	Class 1	0.848	0.820	0.834
	Class 2	0.582	0.687	0.630
Task 3	Class 0	0.755	0.247	0.372
	Class 1	0.655	0.733	0.692
	Class 2	0.588	0.913	0.715
Task 4	Class 0	0.955	0.987	0.970
	Class 1	1.000	0.960	0.980
	Class 2	0.987	0.993	0.990

Performance while training With LwF:

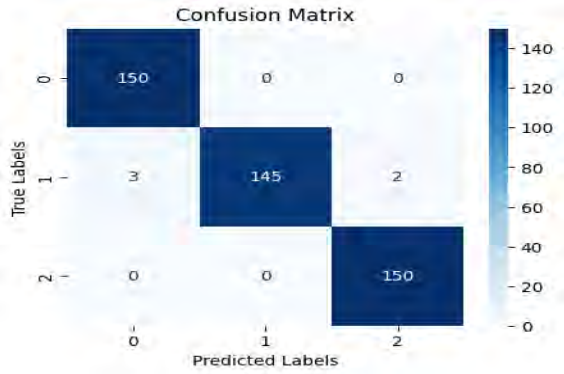
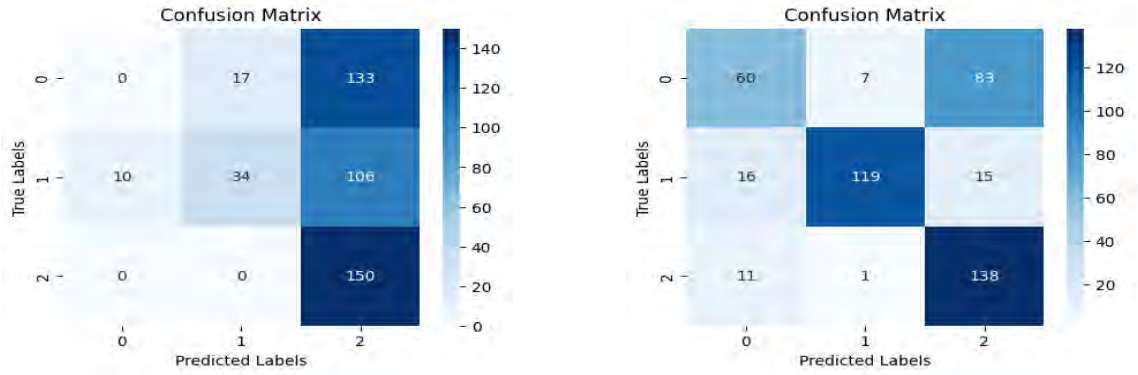


Figure 5.9: With LwF Task 1 (Test on task 1)

Figure 5.9 demonstrates the confusion matrix of test results on task-1 while training on task-1.

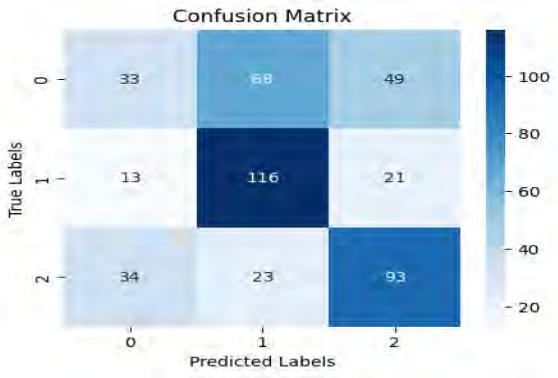


(a) With LwF Task 2 (Test on task 1)

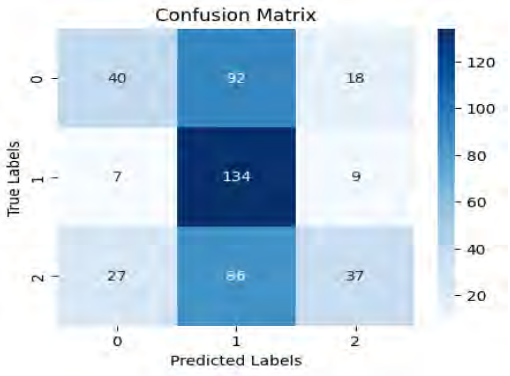
(b) With LwF Task 2 (Test on task 2)

Figure 5.10: With LwF Task 2 results across different tests

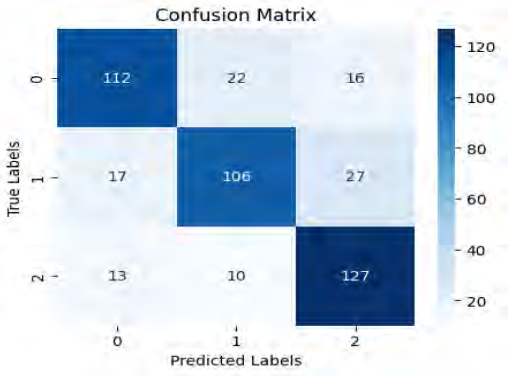
Figure 5.10 demonstrates the confusion matrices of test results on task-1, task-2 while training on task-2 with LwF.



(a) With LwF Task 3 (Test on task 1)



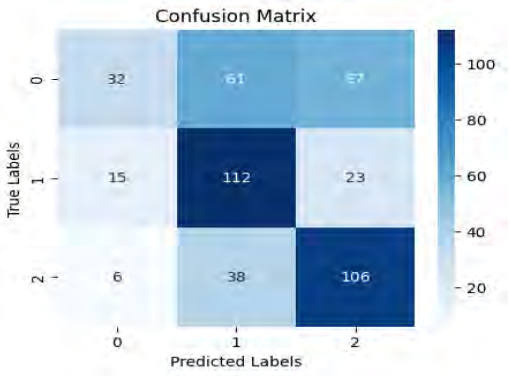
(b) With LwF Task 3 (Test on task 2)



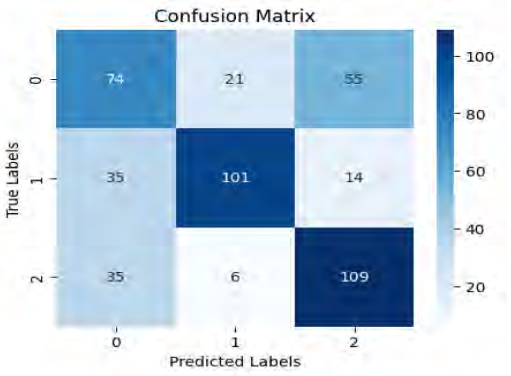
(c) With LwF Task 3 (Test on task 3)

Figure 5.11: With LwF Task 3 results across different tests

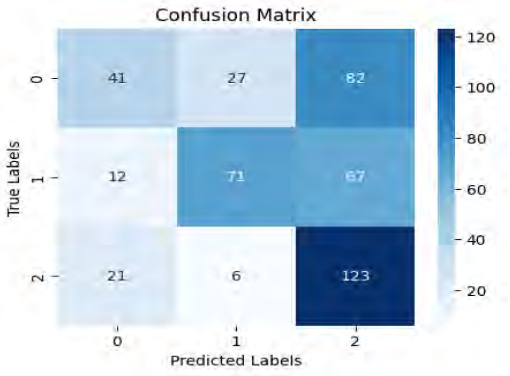
Figure 5.11 demonstrates the confusion matrices of test results on task-1, task-2 and task-3 while training on task-3 with LwF.



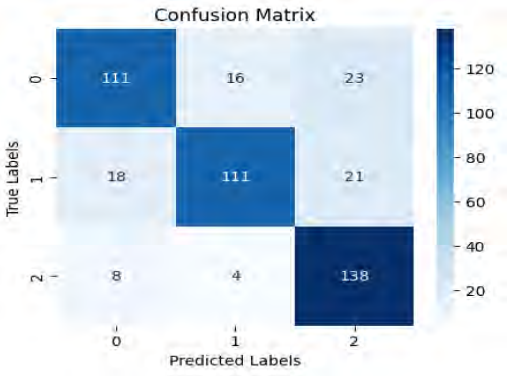
(a) With LwF Task 4 (Test on task 1)



(b) With LwF Task 4 (Test on task 2)



(c) With LwF Task 4 (Test on task 3)



(d) With LwF Task 4 (Test on task 4)

Figure 5.12: With LwF Task 4 results across different tests

Figure 5.12 demonstrates the confusion matrices of test results on task-1, task-2, task-3 and task-4 while training on task-4 with LwF.

Table 5.4: Precision, Recall, and F1 Scores for Task 4 trained model tested on Task 1, Task 2, Task 3, and Task 4, applying LwF

Tested on	**Class**	**Precision**	**Recall**	**F1 Score**
Task 1	Class 0	0.604	0.213	0.315
	Class 1	0.531	0.747	0.620
	Class 2	0.570	0.707	0.631
Task 2	Class 0	0.514	0.493	0.503
	Class 1	0.789	0.673	0.727
	Class 2	0.612	0.727	0.665
Task 3	Class 0	0.554	0.273	0.366
	Class 1	0.683	0.473	0.559
	Class 2	0.452	0.820	0.583
Task 4	Class 0	0.810	0.740	0.774
	Class 1	0.847	0.740	0.790
	Class 2	0.758	0.920	0.831

Table 5.2, 5.3 and 5.4 it is clear shows that there is a massive improvement in precision, recall and f1 score for Task 4 trained model tested on Task 1, Task 2, Task 3, and Task 4, after applying CL methods.

5.3 Discussion

It can be observed that EWC mitigates catastrophic forgetting quite well, achieving 60%, 65%, 66%, and 98% for task-1, task-2 task-3, and task-4 respectively after training on task-4. This represents a significant improvement over the results without CL, where the accuracies were 42%, 55%, 67%, and 97% for task-1, task-2, task-3, and task-4 respectively after training on task-4. The weights of previous tasks were successfully preserved while training only on task-4, resulting in a 7% to 18% improvement across different tasks. Since EWC aims to preserve common features across all tasks, if more diverse data were available, it would be possible to train on them and potentially discover more common features.

LwF also helps in addressing catastrophic forgetting, achieving 60%, 61%, 55%, and 75% for task-1, task-2, task-3, and task-4 respectively after training on task-4. Although these results show improvement compared to the case without CL, where the corresponding values were 42%, 55%, 67%, and 97%, the overall performance drop for earlier tasks, which remains higher with EWC. LwF works by retaining previous knowledge while learning new tasks, and with more diverse and extensive data, it might be possible to further minimize forgetting and improve task performance.

Both EWC and LwF provides stability across tasks by preserving the knowledge from previous tasks and mitigating catastrophic forgetting. But EWC performs well than Lwf. For the EWC method a lower end GPU was used (Google colab's T4) but for Lwf it was not sufficient, a GPU with higher GPU RAM (Google colab's L4 with High RAM) was used. In conclusion, EWC performed better both in performance measures and hardware than LwF.

Training Without CL

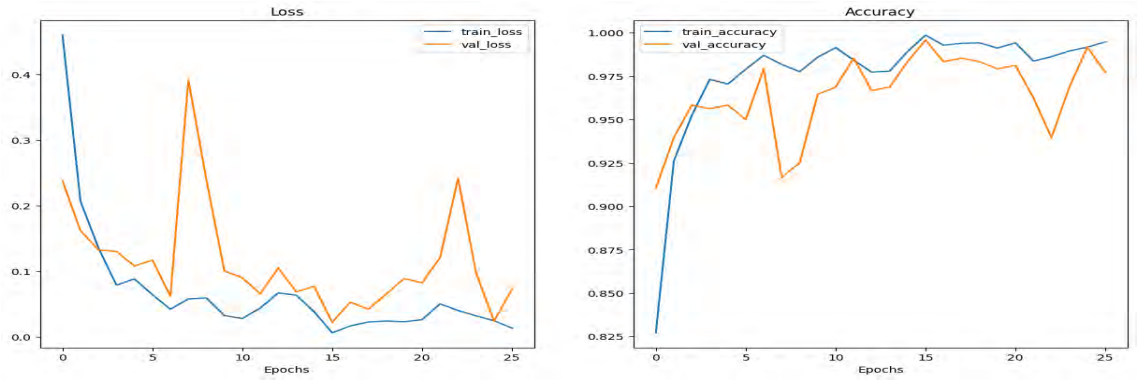


Figure 5.13: Without CL Task 1 (Training/Validation Loss and Accuracy Curve)

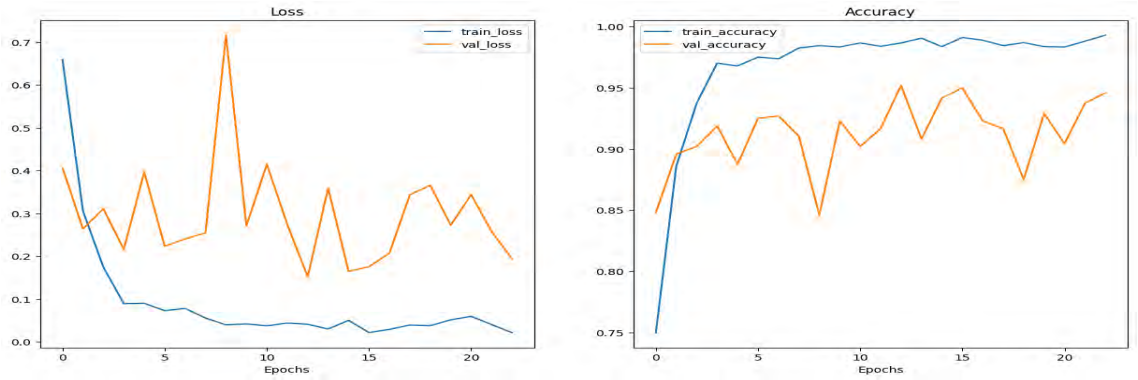


Figure 5.14: Without CL Task 2 (Training/Validation Loss and Accuracy Curve)

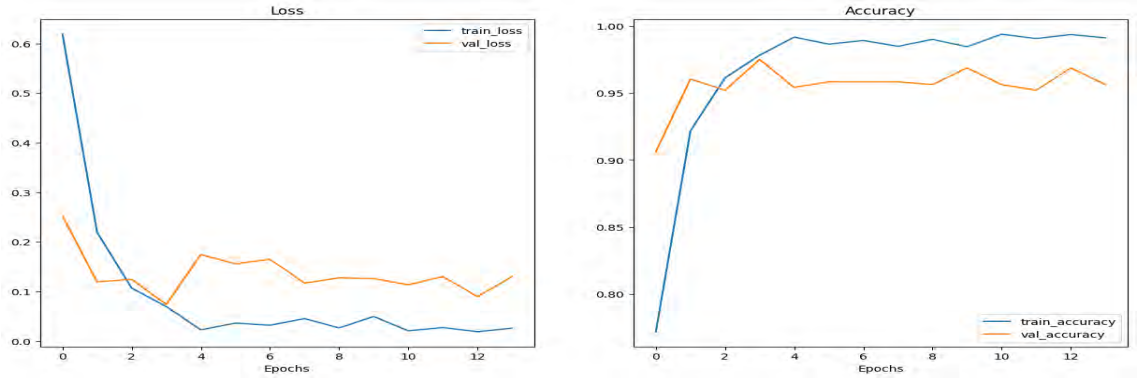


Figure 5.15: Without CL Task 3 (Training/Validation Loss and Accuracy Curve)

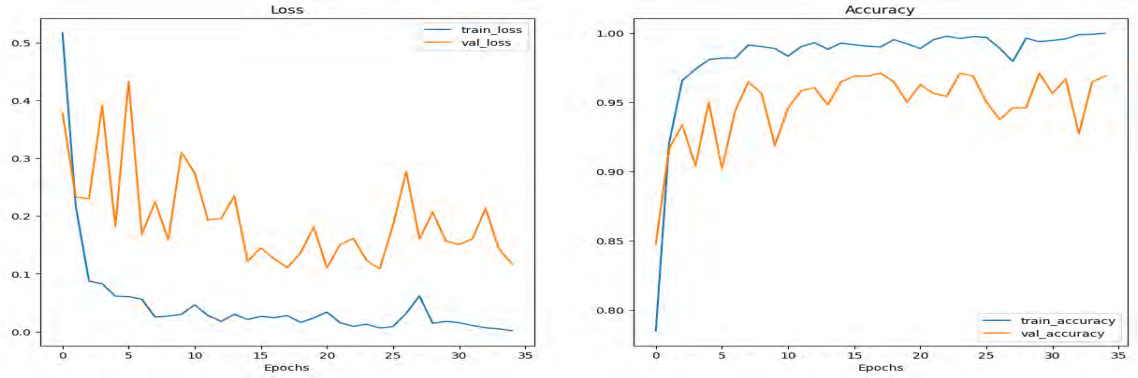


Figure 5.16: Without CL Task 4 (Training/Validation Loss and Accuracy Curve)

Figure 5.13, 5.14, 5.15 and 5.16 illustrates the Training vs Validation curve of loss and accuracy over four training phases throughout the four different tasks without applying CL. Which shows it neither overfits or underfits, as earlystopping was used.

Training With EWC

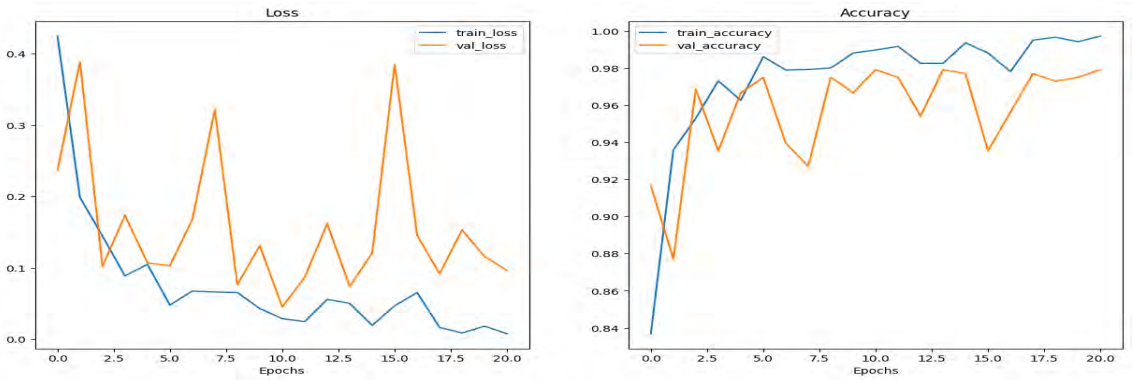


Figure 5.17: With EWC Task 1 (Training/Validation Loss and Accuracy Curve)

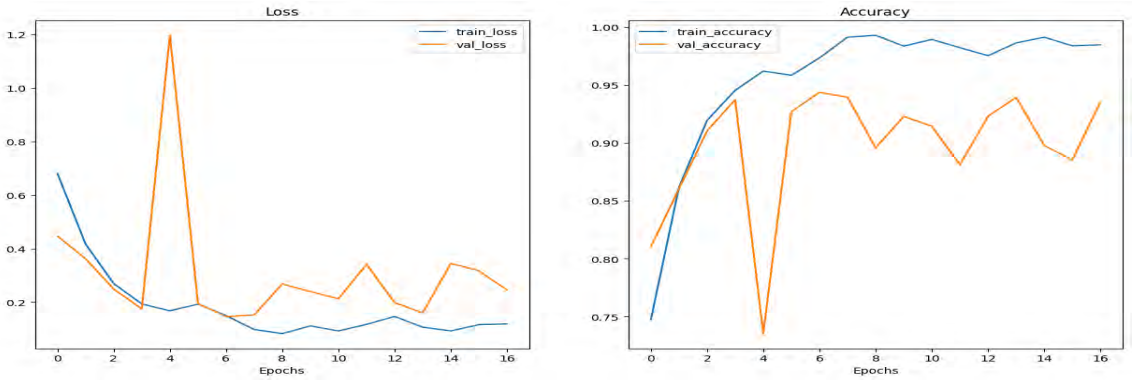


Figure 5.18: With EWC Task 2 (Training/Validation Loss and Accuracy Curve)

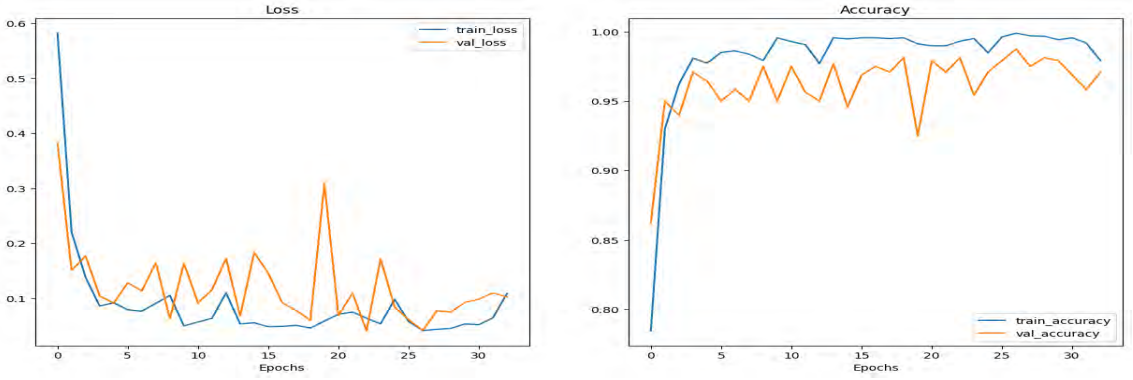


Figure 5.19: With EWC Task 3 (Training/Validation Loss and Accuracy Curve)

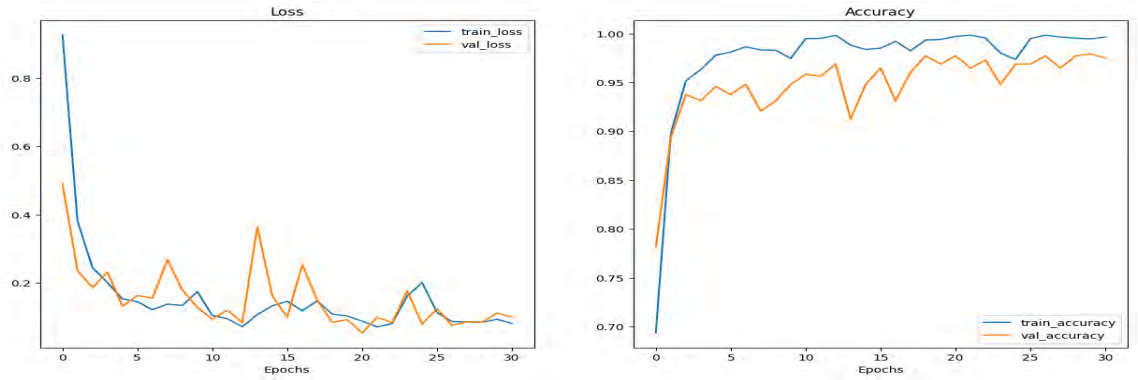


Figure 5.20: With EWC Task 4 (Training/Validation Loss and Accuracy Curve)

Figure 5.17, 5.18, 5.19 and 5.20 illustrates the Training vs Validation curve of loss and accuracy over four training phases throughout the four different tasks while applying EWC. Which shows it neither overfits or underfits, as earlystopping was used.

Training With LwF

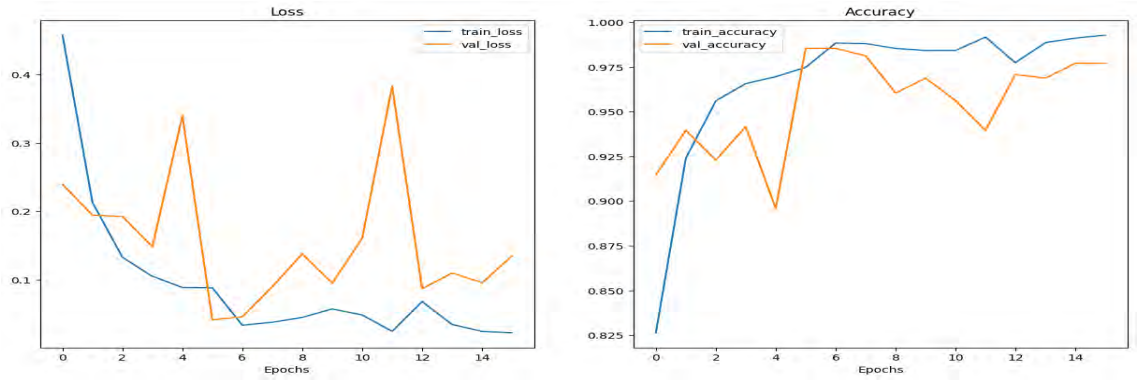


Figure 5.21: With LwF Task 1 (Training/Validation Loss and Accuracy Curve)

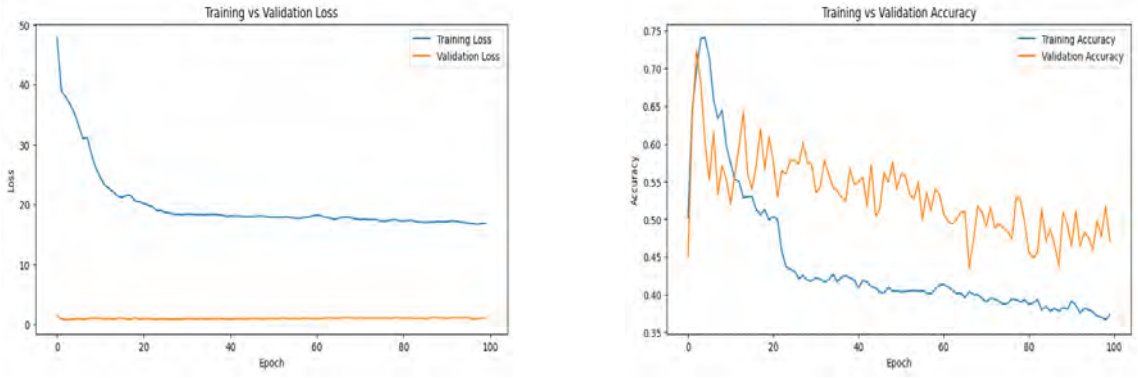


Figure 5.22: With LwF Task 2 (Training/Validation Loss and Accuracy Curves)

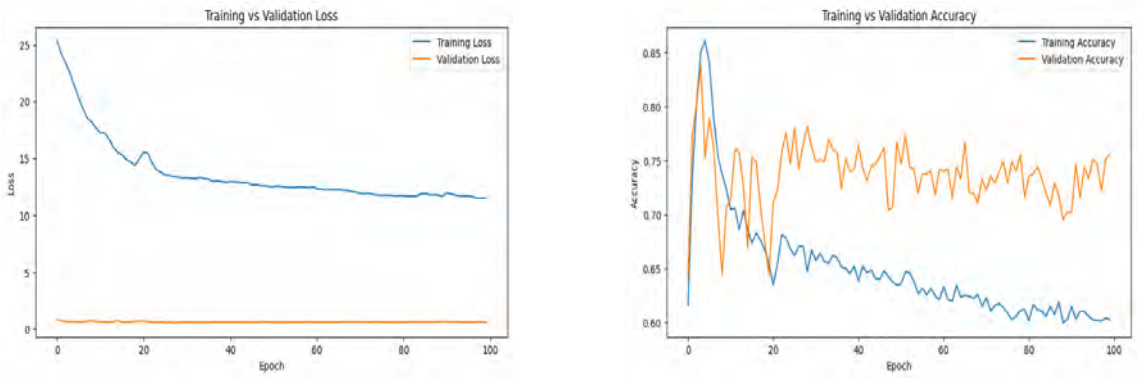


Figure 5.23: With LwF Task 3 (Training/Validation Loss and Accuracy Curves)

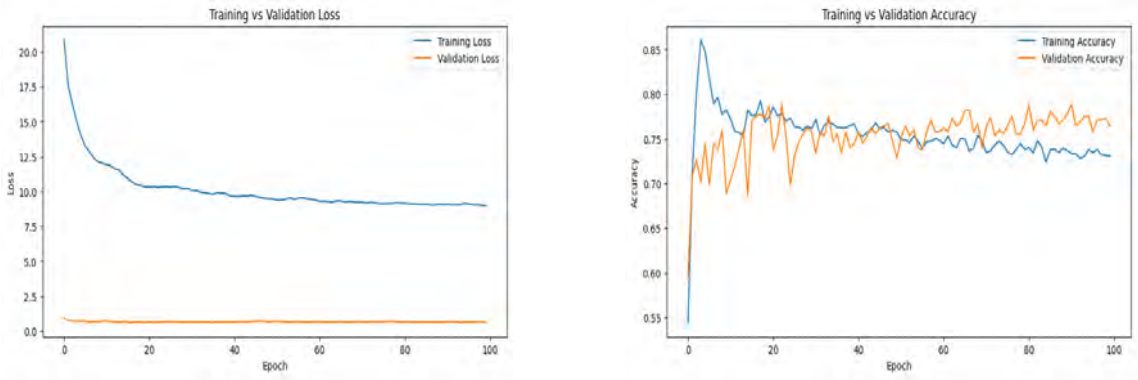


Figure 5.24: With LwF Task 4 (Training/Validation Loss and Accuracy Curves)

Figures 5.21, 5.22, 5.23, and 5.24 illustrate the training vs. validation curves of loss and accuracy over four training phases across the different tasks while applying LwF. In figure 5.21, training on task-1 shows neither overfitting nor underfitting, as early stopping was implemented. However, from the training vs. validation loss curves in figures 5.22, 5.23, and 5.24, it appears that for tasks 2, 3 and 4 the model is underfitting. This is not the case here because LwF preserves knowledge by carrying over the losses of previous tasks to the next task. This serves as a signal to the model that it needs to optimize the loss for all tasks. As a result, the training loss appears higher while the validation loss is lower. Validation is only done here on the current task, to prevent overfiting and underfitting on the current task. Moreover, as we keep training, the training loss keeps getting lower and lower. So, the more we will train the model, the lesser will be the training loss. In reality, the model performs quite well with unseen data.

Chapter 6

Conclusion

This study focuses on addressing the detection of different stresses that plants encounter due to nutrition deficiencies using deep learning techniques which can also overcome catastrophic forgetting. Incorporating Elastic Weight Consolidation (EWC) and Learning without Forgetting (LwF) methods with deep learning, the model can successfully mitigate the problem of retaining knowledge from previously learned tasks besides learning the new tasks. Even though our model has shown a performance gain between 7% to 18%, the overall accuracy of previously learned tasks ranges between 60% and 65%, suggesting the potential for further optimization. The penalty term in EWC requires fine-tuning to improve performance, and LwF, while effective, demands high-performance hardware and a larger number of epochs to minimize training loss as more tasks are added. Therefore, the overall efficiency can be improved by resolving these constraints.

In this thesis, the tasks selected had similar nutritional deficiencies in order to maintain uniform parameters across all tasks. However, to resolve real-world issues, it is necessary to experiment with more tasks and classes to address more plant issues. Moreover, a base CNN model was implemented in this thesis, further experiments with different CNN architectures and new models like Vision Transformers could be integrated in order to enhance the performance of continual learning to detect plant stresses.

This study contributes to addressing one of the major issues farmers encounter, by offering techniques to detect different stresses in some of the regularly consumed vegetable plants. Our findings determine a deep learning method incorporated with continual learning to detect health status from leaf image of plant, yet being able to preserve knowledge from priorly learned tasks. This offers a promising avenue for developing more accurate, efficient, and scalable models for agricultural applications, ultimately helping to optimize crop health and agricultural productivity. Further study in this topic will ensure optimization of the models and extend their application to a broader range of plant varieties and environmental circumstances.

Bibliography

[1] Z. Li and D. Hoiem, "Learning without forgetting," *CoRR*, vol. abs/1606.09282, 2016. arXiv: 1606.09282. [Online]. Available: http://arxiv.org/abs/1606.09282.

[2] S. P. Mohanty, D. P. Hughes, and M. Salath'e, "Using deep learning for image-based plant disease detection," *Frontiers in plant science*, vol. 7, p. 1419, 2016.

[3] H. Durmuş, E. O. Güneş, and M. Kırcı, "Disease detection on the leaves of the tomato plants by using deep learning," in *2017 6th International Conference on Agro-Geoinformatics*, 2017, pp. 1–5. DOI: 10.1109/Agro-Geoinformatics.2017.8047016.

[4] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the National Academy of Sciences*, vol. 114, no. 13, pp. 3521–3526, 2017. DOI: 10.1073/pnas.1611835114. eprint: https://www.pnas.org/doi/pdf/10.1073/pnas.1611835114. [Online]. Available: https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

[5] K. P. Ferentinos, "Deep learning models for plant disease detection and diagnosis," *Computers and electronics in agriculture*, vol. 145, pp. 311–318, 2018.

[6] J. G. Arnal Barbedo, "Plant disease identification from individual lesions and spots using deep learning," *Biosystems Engineering*, vol. 180, pp. 96–107, 2019, ISSN: 1537-5110. DOI: https://doi.org/10.1016/j.biosystemseng.2019.02.002. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1537511018307797.

[7] M. A. Khan, T. Akram, M. Sharif, K. Javed, M. Raza, and T. Saba, "An automated system for cucumber leaf diseased spot detection and classification using improved saliency method and deep features selection," *Multimedia Tools and Applications*, vol. 79, pp. 18 627–18 656, 2020.

[8] A. Krishnaswamy Rangarajan and R. Purushothaman, "Disease classification in eggplant using pre-trained vgg16 and msvm," *Scientific reports*, vol. 10, no. 1, p. 2322, 2020.

[9] P. Singh, A. Sharma, and J. Sharma, "Stress physiology in plants," in Dec. 2020, pp. 175–186, ISBN: 978-81-946901-8-4.

[10] Z. Xu, X. Guo, A. Zhu, *et al.*, "Using deep convolutional neural networks for image-based diagnosis of nutrient deficiencies in rice," *Computational Intelligence and Neuroscience*, vol. 2020, 2020.

[11] A. A. Ahmed and G. H. Reddy, "A mobile-based system for detecting plant leaf diseases using deep learning," *AgriEngineering*, vol. 3, no. 3, pp. 478–493, 2021, ISSN: 2624-7402. DOI: 10.3390/agriengineering3030032. [Online]. Available: https://www.mdpi.com/2624-7402/3/3/32.

[12] M. E. H. Chowdhury, T. Rahman, A. Khandakar, *et al.*, "Automatic and reliable leaf disease detection using deep learning techniques," *AgriEngineering*, vol. 3, no. 2, pp. 294–312, 2021, ISSN: 2624-7402. DOI: 10.3390/agriengineering3020020. [Online]. Available: https://www.mdpi.com/2624-7402/3/2/20.

[13] G. da Silva Vieira, B. M. Rocha, A. U. Fonseca, *et al.*, "Automatic detection of insect predation through the segmentation of damaged leaves," *Smart Agricultural Technology*, vol. 2, p. 100 056, 2022, ISSN: 2772-3755. DOI: https://doi.org/10.1016/j.atech.2022.100056. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2772375522000211.

[14] S. Dey and A. Raichaudhuri, "Abiotic stress in plants," in *Advances in Plant Defense Mechanisms*, J. N. Kimatu, Ed., Rijeka: IntechOpen, 2022, ch. 8. DOI: 10.5772/intechopen.105944. [Online]. Available: https://doi.org/10.5772/intechopen.105944.

[15] A. J., J. Eunice, D. E. Popescu, M. K. Chowdary, and J. Hemanth, "Deep learning-based leaf disease detection in crops using images for agricultural applications," *Agronomy*, vol. 12, no. 10, 2022, ISSN: 2073-4395. DOI: 10.3390/agronomy12102395. [Online]. Available: https://www.mdpi.com/2073-4395/12/10/2395.

[16] M. Shoaib, T. Hussain, B. Shah, *et al.*, "Deep learning-based segmentation and classification of leaf images for detection of tomato plant disease," *Frontiers in Plant Science*, vol. 13, 2022, ISSN: 1664-462X. DOI: 10.3389/fpls.2022.1031748. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpls.2022.1031748.

[17] A. Alharbi, M. U. G. Khan, and B. Tayyaba, "Wheat disease classification using continual learning," *IEEE Access*, vol. 11, pp. 90 016–90 026, 2023. DOI: 10.1109/ACCESS.2023.3304358.

[18] S. Ali, *Bangladesh's fruits, vegetables exports plunge 68.7% in a decade*, https://www.tbsnews.net/economy/bangladeshs-fruits-vegetables-exports-plunge-687-decade-723398, Accessed on 20-01-2024, 2023.

[19] M. S. M. Asaari, S. Shamsudin, and L. J. Wen, "Detection of plant stress condition with deep learning based detection models," in *2023 International Conference on Energy, Power, Environment, Control, and Computing (ICEPECC)*, 2023, pp. 1–5. DOI: 10.1109/ICEPECC57281.2023.10209458.

[20] N. A. Orka, M. N. Uddin, F. M. Toushique, and M. S. Hossain, "Olid i: An open leaf image dataset for plant stress recognition," *Frontiers in Plant Science*, vol. 14, 2023, ISSN: 1664-462X. DOI: 10.3389/fpls.2023.1251888. [Online]. Available: https://www.frontiersin.org/articles/10.3389/fpls.2023.1251888.

[21] R. A. Rizvee, T. H. Orpa, A. Ahnaf, *et al.*, "Leafnet: A proficient convolutional neural network for detecting seven prominent mango leaf diseases," *Journal of Agriculture and Food Research*, vol. 14, p. 100 787, 2023, ISSN: 2666-1543. DOI: https://doi.org/10.1016/j.jafr.2023.100787. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2666154323002946.

[22] M. S. H. Shovon, S. J. Mozumder, O. K. Pal, M. F. Mridha, N. Asai, and J. Shin, "Plantdet: A robust multi-model ensemble method based on deep learning for plant disease detection," *IEEE Access*, vol. 11, pp. 34 846–34 859, 2023. DOI: 10.1109/ACCESS.2023.3264835.

[23] Y. Zhao, C. Jiang, D. Wang, X. Liu, W. Song, and J. Hu, "Identification of plant disease based on multi-task continual learning," *Agronomy*, vol. 13, no. 12, 2023, ISSN: 2073-4395. DOI: 10.3390/agronomy13122863. [Online]. Available: https://www.mdpi.com/2073-4395/13/12/2863.

[24] B. WANG, "Zero-exemplar deep continual learning for crop disease recognition: A study of total variation attention regularization in vision transformers," *Frontiers in Plant Science*, vol. 14, p. 1 283 055,