Democratic and Popular Algerian Republic

Ministry of Higher Education and Scientific Research

Yahia fares university of Medea

Faculty of Sciences

Department of Mathematics and Computer Science

Final Master Project

Branch: computer science

Specialty: Systems Engineering and Web Technologies

Title

Mobile application for plant diseases detection using vision transformer architecture

Prepared and Presented by

- **Chikhi Abla**
- **Fehis Imane**

Supervised by:

- **Mr. Sofiane Toulait**

Academic Year : 2023-2024

Dedication

بسم الله الرحمن الرحيم

First of all, I want to thank God.

To give me strength and courage to lead for this humble act

To my beloved parents, Mom and Dad, who have provided me with unwavering support and confidence on a daily basis. Your constant encouragement and belief in me instilled a love of learning that made this project possible. Words can't express my deepest gratitude.

To my brothers, Khalil, Djahid and Abdelghani , whose unwavering support and infectious laughter have been a constant source of strength. Words cannot sum up my gratitude for your presence in my life.

To my cousin as my sister Nassima.

To my partner, Imane.

To Supervisor Mr. Sofiane Toulait.

To all my friends with whom I shared moments of joy and happiness, thank you for making this journey even more special.

To my whole family, your love is my foundation.

Chikhi Abla.

Dedication

To my incredible parents, your endless support and belief in my abilities have been the driving force behind my achievements. Your love and encouragement have made all the difference, and I am eternally grateful.

To my siblings, Houssem, Riadh, and Ritadje, your support and cheerfulness have been a constant source of strength and joy.

Thank you for always standing by my side and lifting my spirits.

To my partner, Abla, your unwavering love, patience, and encouragement have been my anchor. Your confidence in me has been a guiding light, and I am deeply thankful for your presence in my life.

To my wonderful friends, thank you for the joy, laughter, and companionship we have shared throughout this journey. Your support has been invaluable, making this experience truly special.

To my entire family, your collective love and support have been the foundation upon which I stand. Thank you for believing in me and helping me reach my goals.

With sincere gratitude, Fehis imane

Acknowledgement

We thank God, whose praise, prayer, and peace are bestowed under the guidance of our messengers.

From the door of those who did not thank people, they did not thank God. We extend our sincere thanks and appreciation to Professor Supervisor Mr.TOULAIT Sofiane for his guidance, which he has given us.

We also deeply grateful to all those who have assisted us in this endeavor, both near and far, and to our parents, whose unwavering support provided us with the necessary conditions to complete this work.

Thank you for the jury, Mr. khaldoun Ahmed and Mr.Tiberkak Allal.

Additionally, we would like to express my gratitude to all the teachers who have contributed to our learning and development, as well as to our colleagues.

Abstract

Every year, a significant portion of crops is lost to plant diseases worldwide. Farmers often struggle to accurately identify these diseases manually due to limited knowledge and resources. This project aims to address this issue by developing a mobile application that uses a deep learning model to detect and classify plant diseases from leaf images. Using the PlantVillage dataset, a comparative study evaluated several transfer learning models, including VGG16, DenseNet201, and ResNet50, against the proposed MobileViT-XXS model for vision transformers. The MobileViT-XXS model demonstrated superior performance, achieving a test accuracy of 97%, outperforming the other convolutional neural network models.As a result, a mobile android application was developed using the MobileViT-XXS model for plant disease detection. This application aims to improve crop productivity and the overall economy by enabling farmers to make prompt and precise decisions regarding crop diseases.

Key words:

PlantVillage, android mobile application, deep learning, VGG16, ResNet-50,

Densnet-201, MobileViT-XXS, vision transformer.

الملخص

في كل عام، يتم فقدان جزء كبير من المحاصيل بسبب الأمراض النباتية في جميع أنحاء العالم. غالبًا ما يكافح المزارعون لتحديد هذه الأمراض بدقة يدويًا بسبب محدودية المعرفة والموارد. يهدف هذا المشروع إلى معالجة هذه المشكلة من خلال تطوير تطبيق متنقل يستخدم نموذج التعلم العميق لاكتشاف وتصنيف الأمراض النباتية من صور الأوراق. باستخدام مجموعة بيانات PlantVillage، قامت دراسة مقارنة بتقييم العديد من نماذج تعلم النقل، بما في ذلك VGG16 و DenseNet201 وResNet50 ، مقابل نموذج MobileViT-XXS المقترح لمحولات الرؤية. أظهر نموذج MobileViT-XXS أداءً متفوقًا، حيث حقق دقة اختبار بنسبة 97٪، متفوقًا على نماذج الشبكة العصبية الملتفة الأخرى. نتيجة لذلك، تم تطوير تطبيق Android للجوال باستخدام نموذج MobileViT-XXS للكشف عن أمراض النبات. يهدف هذا التطبيق إلى تحسين إنتاجية المحاصيل والاقتصاد العام من خلال تمكين المزارعين من اتخاذ قرارات سريعة ودقيقة بشأن أمراض المحاصيل.

الكلمات المفتاحية:

PlantVillage ، تطبيق الهاتف المحمول ، التعلم العميق ، VGG16، ResNet-50، Densnet-201، MobileViT-XXS، محول الرؤية.

Résumé

Chaque année, une part importante des cultures est perdue à cause des maladies des plantes dans le monde entier. Les agriculteurs ont souvent du mal à identifier ces maladies manuellement en raison des connaissances et des ressources limitées. Ce projet vise à résoudre ce problème en développant une application mobile qui utilise un modèle d'apprentissage profond pour détecter et classer les maladies des plantes à partir d'images foliaires. En utilisant l'ensemble de données PlantVillage, une étude comparative a évalué plusieurs modèles d'apprentissage par transfert, y compris VGG16, DenseNet201 et ResNet50, contre le modèle MobileViT-XXS proposé pour les transformateurs de vision. Le modèle MobileViT-XXS a démontré des performances supérieures, atteignant une précision de test de 97%, surpassant les autres modèles réseau neuronal convolutif. Cette application vise à améliorer la productivité des cultures et l'économie en général en permettant aux agriculteurs de prendre des décisions rapides et précises concernant les maladies des cultures.

Mots clés :

PlantVillage, android mobile application, deep learning, VGG16, ResNet-50, Densnet-201, MobileViT-XXS, transformateur de vision.

Content

Dedication ..... **ii**
Acknowledgement ..... **iv**
Abstract ..... **v**
List Tables ..... **xi**
List of Figures ..... **xii**
Abbreviations ..... **xiii**

GENERAL INTRODUCTION **1**
Chapter 01: Deep Learning **4**
1. Introduction 5
2. Machine learning 5
3. Deep learning 5
4. Difference between deep learning and machine learning 6
5. Types of Machine Learning 7
5.1 Supervised Learning 7
5.2 Unsupervised Learning 8
5.3 Semi-Supervised Learning 8
6. Artificial Neural Network 9
7. Deep learning concepts 10
7.1. Propagation 10
7.1.1. Forward propagation 10
7.1.2. Backward propagation 10
7.2. Gradient Descent 11
7.2.1. Definition 11
7.2.2. Gradient descent variants 11
7.3. Batch size 12
7.4. Epoch 12
7.5. Training 12
7.6. Validation 12
7.7. Testing 12
7.8. Over fitting 13
7.9. Underfitting 13

7.10. Loss function 14
7.11. Convolutional Neural Network 14
7.12. Recurrent neural network 15
8. Deep Learning Applications 15
9. Conclusion 16
Chapter 02 : Convolutional neural network and transfer learning 17
1. Introduction 18
2. ImageNet 18
3. Important CNN layers 19
3.1. Convolutional Layer 19
3.2. Activation function 20
3.3. Pooling Layer 21
3.4. Dropout 22
3.5. Flatten Layer 22
3.6. Dense layer 23
4. Transfer Learning 24
4.1. Transfer learning architectures 24
5. Some pre-trained models 25
5.1. VGG16 model 25
5.2. ResNet-50 model 26
5.3. DenseNet201 model 26
5.4. MobileNetV2 model 27
6. Conclusion 28
Chapter 03: Materials and experimental results 29
1. Introduction 30
2. The Transformer 30
2.1. Encoder 31
2.2. Decoder 31
3. Key Components of Transformers 31
4. Vision transformers (ViTs) 31
4.1. Vision transformers (ViTs) workflow 32
5. MobileViT overview 32
6. MobileViT architecture 33
6.1. CNN components (MobileNetV2 Blocks) 33
6.2. Transformer components (ViT) 34

7. Why use MobileViT instead of ViT for mobile applications 34
8. Project requirement 35
8.1. Python 35
8.1.1. TensorFlow 35
8.1.2. Keras 35
8.1.3. NumPy 35
8.1.4. Matplotlib 35
8.1.5. Scikit-learn 36
8.2. Google Colaboratory. 36
8.3. Hardware 36
9. Methodology 36
9.1. Dataset overview 36
8.1.1. The PlantVillage dataset 36
8.1.2. Dataset Discussion 37
9.2. Data preparation 38
9.3. Model Description 38
9.5. Results of fine training model 41
9.6. Discussion 42
9.7. Diving Deeper into MobileViT-XXS result 42
10. Conclusion 44
Chapter 04: Model deployment 45
1. Introduction 46
2. Methods of deployment 46
2.2. TensorFlow Lite: 46
3. Environment and development tools 47
3.1. Java. 47
3.2. Xml 47
3.3. Android Studio 47
4. Application Overview 48
5. Conclusion 50
GENERAL CONCLUSION 51
Bibliography 53

List Tables

Table2.1:Activation functions and their mathematical operations. 20

Table3.1:Comparison of MobileViT model variants. 33
Table3.2:Dataset description. 37
Table3.3:The architecture of the pre-trained models..... 39
Table3.4:The architecture of the MobileViT-XXS models. 40
Table3.5: Training parameters of model. 41
Table3.6:Comparison of pre-trained models with MobileViT-XXS model propose. 41

List of Figures

Figure1.1:The relation between AI, ML and Deep Learning 6
Figure1.2:The difference between deep learning and machine learning 7
Figure1.3:Supervised Learning 7
Figure1.4:Unsupervised Learning 8
Figure1.5: Semi-Supervised Learning 8
Figure1.6:Architecture of a neural network 9
Figure1.7:Forward propagation 10
Figure1.8:Backward propagation 10
Figure1.9: A visualisation of the splits 13
Figure1.10: Overfitting 13
Figure1.11: Underfitting 14
Figure1.12:An example of CNN architecture for image classification 14
Figure1.13: Simple Recurrent Neural Network 15

Figure2.1: Convolutional process. 19
Figure2.3: Two different pooling techniques were applied. 21
Figure2.4: An example of two Neural networks: (a): Standard Neural Network - (b): After applying dropout method 22
Figure2.5: Flatten layer. 23
Figure2.6: Dense layer 23
Figure2.7: Basic idea behind transfer learning. 24
Figure2.8: VGG 16 neural network architecture. 25
Figure2.9: ResNet-50 model architecture 26
Figure2.10: DenseNet-201 model architecture 27
Figure2.11: MobileNetV2 neural network architecture. 28

Figure3.1: Transformer architecture. 30
Figure3.2: Architecture ViT model. 32
Figure3.3: Architecture MobileViT model 34
Figure3.5: Sample of images from PlantVillage dataset. 37
Figure3.6: Results of MobileViT-XSS training and validation phases. 42
Figure3.7: The classification report of MobileViT-XXS. 43
Figure3.8: The confusion matrix of MobileViT-XXS model 44

Figure4.1:Architecture of Tensorflow Lite 47
Figure4.2: (a) Icon screen - (b) A welcome screen 48
Figure4.3:Foverview application plantiyScan, Usage tips. 49
Figure4.4: Some examples prediction of plant disease 49
Figure4.5: List of available diseases 50

Abbreviations

- **AI:** Artificial Intelligence.
- **ANN:** Artificial Neural Network.
- **CNN:** Convolutional neural network.
- **CPU :** Central Processing Unit.
- **DenseNet:** Dense Convolutional Network.
- **DL :** Deep learning.
- **VGG:** Visual Geometry Group.
- **LSVRC:** Image Net Large Scale Visual Recognition Challenge.
- **ML:** Machine learning.
- **NLP:** Natural Language Processing.
- **ResNet:** Residual Network.
- **RNN:** Recurrent Neural Network.
- **SGD:** Stochastic Gradient Descent.
- **S :** Small.
- **ViT :** Vision Transformer.
- **XXS :** Extra Extra Small.
- **XS :** Extra Small.

GENERAL INTRODUCTION

Agriculture is the backbone of many economies around the world, providing food, raw materials, and employment to a significant portion of the population. However, one of the major challenges that farmers face is the threat posed by plant diseases. These diseases can severely impact crop yields, quality, and, ultimately, the economic stability of agricultural communities. Plant diseases are caused by a variety of pathogens, including fungi, bacteria and viruses. They can lead to substantial losses in crop production, sometimes wiping out entire fields. Effective management and control of plant diseases are crucial for ensuring food security, improving crop productivity, and sustaining agricultural livelihoods.

The problem is that it is very difficult for the farmers to manually identify these diseases accurately with their limited knowledge.

The integration of deep learning models into mobile applications represents a significant advancement in agricultural technology. These applications provide farmers with easy-to-use tools for real-time disease detection and management. By simply taking a photo of a plant leaf, farmers can quickly receive a diagnosis and recommendations for treatment.

Work organization: This report will contain four chapters:

- **Chapter 01:Deep Learning**
  This chapter introduces the concept deep learning and dissects the fundamental principles underlying neural networks.
- **Chapter 02: Convolution Neural Networks and transfer learning**
  In this chapter, we delve into the main layers of convolutional neural networks (CNN), elucidating their functions and roles in deep learning architectures. Furthermore, we introduce the concept of transfer learning, highlight its fundamental principles, and present some of the most significant pre-trained.
- **Chapter 03: Materials and experimental results**
  In this chapter, we focus on the detection of plant diseases using deep learning models. We will test four different deep learning models, analyze their performance, and discuss the results in detail. The goal is to determine which model performs best for accurately diagnosing plant diseases.

Chapter 04: Model deployment

- In this chapter, we focus on the deployment of deep learning model for plant disease detection on a mobile application. Based on the performance analysis from the previous chapter, we chose the most accurate and efficient model for deployment.

Chapter 01: Deep Learning

1. Introduction

Artificial intelligence (AI) is one of the most controversial topics in computer science today, and there does not seem to be any consensus on how artificial intelligence and human intervention differ. AI refers to the process of transferring data, information and human intelligence to machines. Its primary goal is to develop stand-alone machines capable of thinking and acting like humans. These machines imitate human behavior and solve complex problems through learning and problem-solving techniques, simulating natural intelligence.

2. Machine learning

Machine learning (ML) is an artificial intelligence (AI) branch that enables systems to learn from experience and get better over time without the need for explicit programming [1]. It focuses on creating programs that autonomously access and learn from data using sophisticated statistical methods. This allows machines to find patterns, forecast outcomes, and extract knowledge from large datasets, making ML a powerful tool for solving complex problems and making data-driven decisions [2].

In simple words, ML teaches the systems to think and understand like humans by learning from the data and gradually improving its accuracy.

3. Deep learning

Deep learning (DL) is a subset of machine learning techniques that use artificial neural networks (ANNs), inspiration from the organization and operation of the human brain. It aims to learn hierarchical data representations, allowing AI systems to effectively analyze diverse and unregulated data types such as text, sound and images to produce accurate insights and predictions. Unlike traditional machine learning methods based on pre-processed data, deep learning models can learn directly from primary data without human intervention [3].

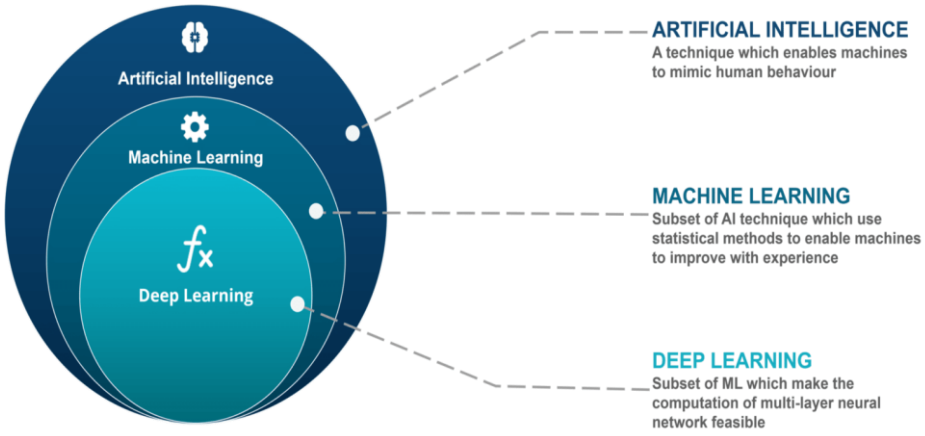


Figure1.1: The relation between AI, ML and Deep Learning

4. Difference between deep learning and machine learning

Machine learning and deep learning are both subfields of artificial intelligence (AI) focused on creating systems that can learn from data and make decisions. While they share common goals, they differ significantly in their approaches, complexity, and applications [4].

The differences between machine learning and deep learning consist of:

- DL models have longer training times and higher computation costs compared to ML models.
- ML uses structured data whereas DL uses unstructured data.
- While ML just needs a small quantity of data, DL requires a vast amount of data.
- Output of DL information in the form of images, music, videos, and even text, whereas ML produces only amounts of information.
- ML can train on a CPU (central processing unit), DL needs a specialized GPU (graphics processing unit) to train.
- ML Requires more human intervention to correct and learn. DL Learns on its own from environment and past mistakes. Take a look for (Figure 1.2).

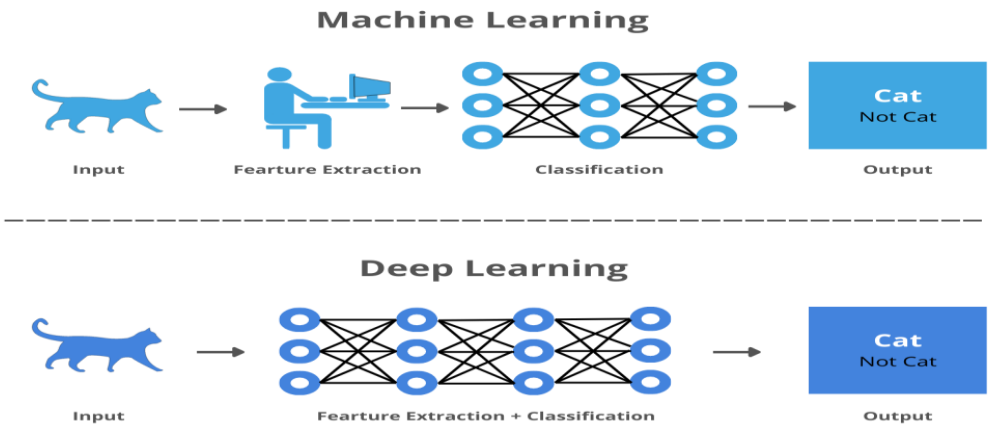


Figure1.2: The difference between deep learning and machine learning.

5. Types of Machine Learning

There are several types of machine learning, each with special characteristics and applications. Some of the main types of machine learning algorithms are as follows:

1. Supervised Machine Learning.
2. Unsupervised Machine Learning.
3. Semi-Supervised Machine Learning.

5.1 Supervised Learning

Supervised learning is a category of machine learning that uses labeled datasets to train algorithms that enable us to forecast future or invisible data [5]. Supervised learning can be separated into two types of problems in data mining: classification and regression. As show in (Figure1.3).

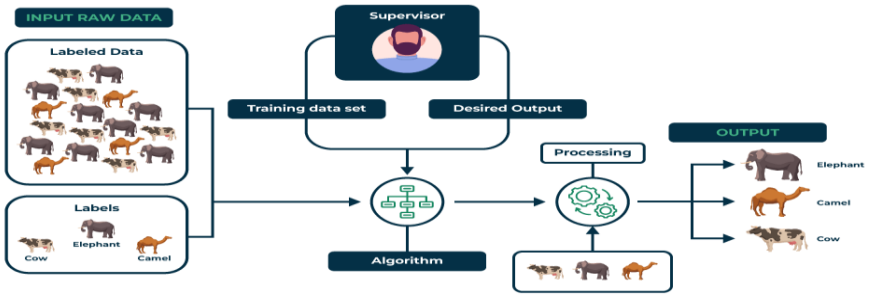


Figure1.3: Supervised Learning

5.2 Unsupervised Learning

In unsupervised learning, the algorithm learns from the data itself without being provided with explicit labels or categories (Figure 1.4). Its objective is to find relationships and patterns within the data using techniques such as clustering, dimensionality reduction, or association rule learning. This allows the algorithm to autonomously identify similarities, differences, and relationships within the data, even when no labeled information is available [5].

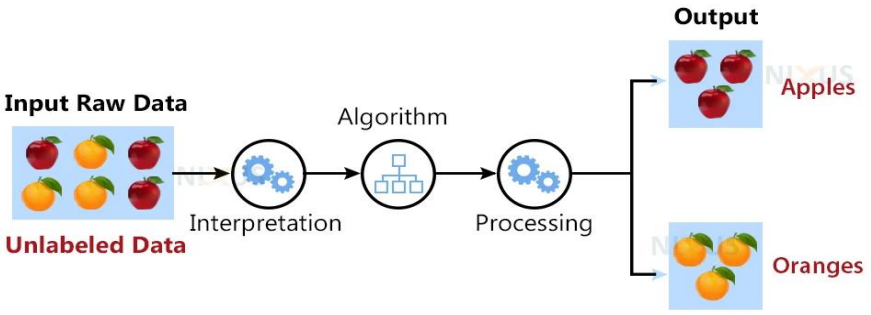


Figure1.4: Unsupervised Learning.

5.3 Semi-Supervised Learning

Semi-supervised learning is a machine learning technique that uses both labeled and unlabeled data, bridging the gap between supervised and unsupervised learning. It's valuable when acquiring labeled data is challenging or costly. By leveraging unsupervised methods to infer labels for unlabeled data, semi-supervised learning enhances model training efficiency, especially in scenarios like image datasets where manual labeling is impractical [5].

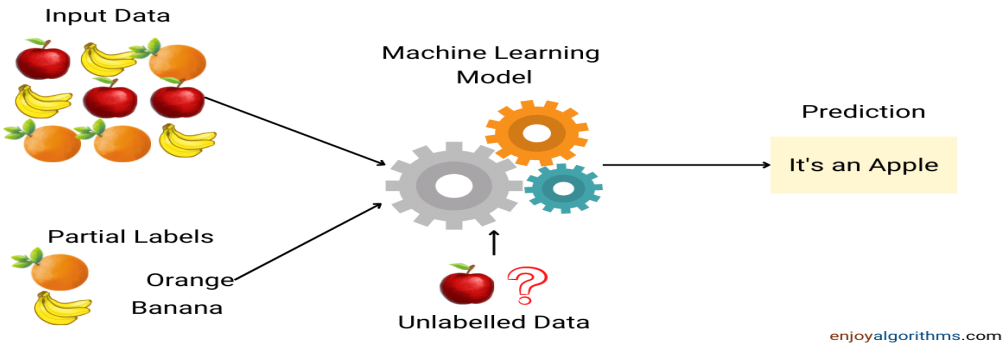


Figure1.5: Semi-Supervised Learning.

6. Artificial Neural Network

An Artificial Neural Network (ANN) is a computer system based on the functioning of the human brain, inspired by biological neural networks .It is made up of layers of networked nodes, or neurons. These nodes process information, and during training, the network modifies the connection strengths (weights) to learn from the data. This allows the network to identify patterns, anticipate outcomes, and perform a variety of machine learning and artificial intelligence tasks [6].

Layers of connected nodes make up artificial neural networks. Every node has a threshold and matching weight, and they are all connected.
Deep learning would not be possible without neural networks. The neural network's depth is the only factor that influences the algorithm's learning depth.

We distinguish three types of neuron layers [7]:

- **Input Layer**: this layer comes first. The data will be received by this layer and then forwarded to the other network layers.

- **Hidden Layer**: is the second kind of layer. In a neural network, there can be one or more hidden layers. Neural networks exhibit exceptional performance and complexity due to the presence of hidden layers. These layers are pivotal for the network's performance and complexity, as they undertake tasks such as data transformation and automatic feature generation simultaneously.

- **Output Layer**: is the final category of layers. The outcome or problem's outcome is carried by the output layer. The input layer receives raw images, and the output layer produces the desired results.

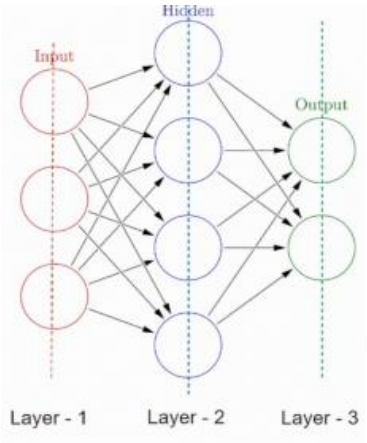


Figure1.6: Architecture of a neural network.

7. Deep learning concepts

7.1. Propagation

7.1.1. Forward propagation: also known as feedforward, is the process of computing and storing the intermediate variables, including the final output, of a neural network as the data moves through the network from the input layer to the output layer [8].

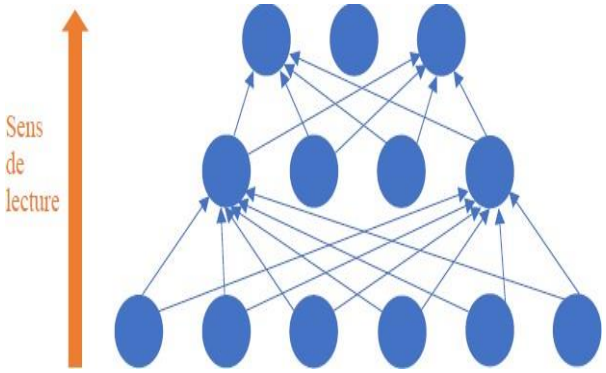


Figure1.6: Forward propagation

7.1.2. Backward propagation: Often abbreviated as "backprop", is a fundamental algorithm in training neural networks. It's used to calculate the gradient of the loss function with respect to the weights of the network. This gradient is then used to update the weights of the network in order to minimize the loss function, thus improving the network's performance [8].

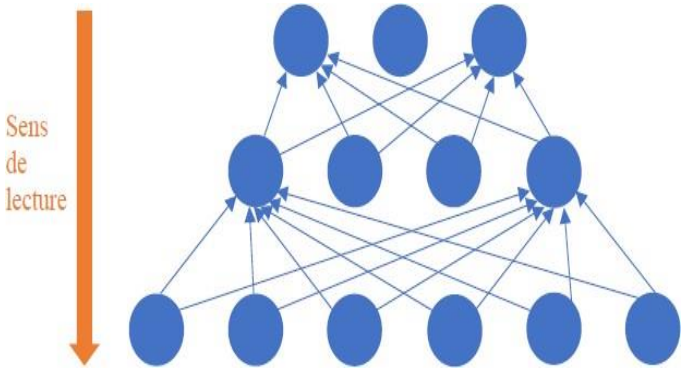


Figure1.7: Backward propagation.

7.2. Gradient Descent

7.2.1. Definition

Gradient descent is an iterative optimization algorithm used to minimize the cost or loss function of a model. It is a fundamental technique used when training a machine learning model to find the optimal parameters (weights and biases) in neural networks to improve the model's performance [9]. Gradient descent is the most widely used neural network optimization algorithm [10].

7.2.2. Gradient descent variants

Gradient descent is a fundamental optimization algorithm used in machine learning and deep learning to minimize a loss function by iteratively adjusting the parameters of a model. There are several variants of gradient descent, each with unique characteristics suited for different scenarios. Here are some common variants:

- Batch Gradient Descent: this method ensures convergence to the global minimum for convex surfaces by computing gradients for the whole training dataset in a single update.

- Stochastic Gradient Descent (SGD): although it has a high variance, SGD updates parameters for every training example, cutting down on redundancy and enabling faster convergence.

- Mini-batch Gradient Descent: this technique achieves equilibrium by utilizing effective matrix optimizations, which are frequently employed in neural network training, to update parameters for mini-batches of training examples while lowering variance [11].

- Momentum: for neural networks, this technique is employed in the objective function. It enhances both the accuracy and the training speed by summing the computed gradient at the preceding training step, which is weighted via a factor (known as the momentum factor).

7.3. Batch size

Batch size is simply defined as the number of data samples that have to be processed before the internal parameters get update ,in another wordre the number of training samples present in a single (forward + backward pass)[12].

The size batch tree type depending on the number of sample in each batch

- Batch Gradient Descent. Batch Size = Size of Training Set.
- Stochastic Gradient Descent. Batch Size = 1.
- Mini-Batch Gradient Descent. 1 < Batch Size < Size of Training Set.

7.4. Epoch

The number of times the learning algorithm will run through the complete training dataset is defined by the hyper parameter epoch. You can specify an integer value between one and infinity for the number of epochs [12].

7.5. Training

In order to train the model, we use the training set, which is the data used to fit the model. The model sees and learns this data, and accuracy is checked and reported after each iteration. Model performance will be improved with high quality and more diverse training data [13].

7.6. Validation

Validation set is data used to evaluate a model during training; accuracy gets checked and reported after each epoch. The model does not update any weights or biases (learn) based on this data. The validation set is utilized throughout training.

The machine learning engineer monitors the validation error and tunes hyper parameters to optimize the model for better performance [13].

7.7. Testing

Test set is data used to evaluate the final trained model. The machine learning engineer checks the accuracy after the entire training is done. Often, a validation set is used as a test dataset. A testing set is typically a well-structured dataset that includes a wide range of data for scenarios that the model is likely to encounter in real world applications [13].

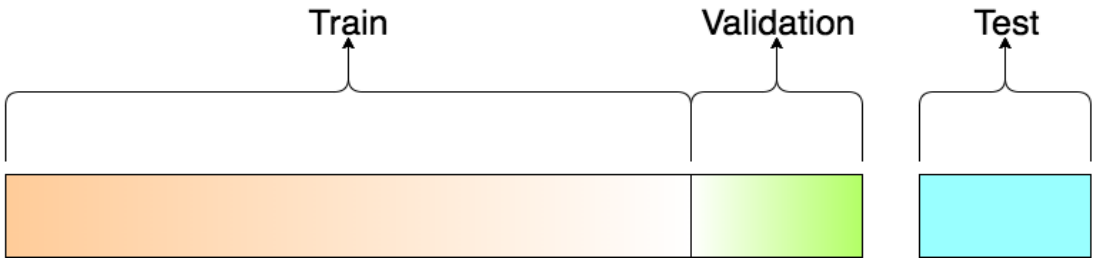


Figure1.8: A visualisation of the splits.

7.8. Over fitting

Unwanted machine learning behavior, known as overfitting, happens when a model produces accurate predictions for training data but not for fresh data. An overfit model can't handle all kinds of new data well and can produce erroneous predictions [14].

The overfitting condition happens when there are significantly more hidden layers than there are problems, which leads to overtraining of the network and negatively affects the network's time complexity of the network and it mostly arises when network matches the data so closely that it loses its generalization ability over the data to be tested [15].

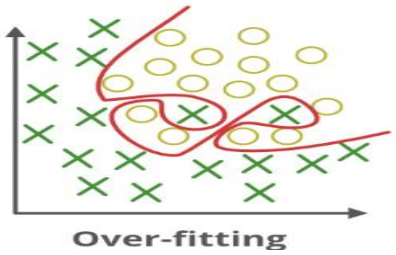


Figure1.9: Overfitting.

7.9. Underfitting

Under fitting, the opposite of overfitting, occurs when a machine learning model is not sophisticated enough to precisely represent the correlations between the characteristics of a dataset and a target variable. An underfitted model frequently performs poorly even on training data, producing problematic or incorrect results on new data or data that it was not trained on [16]. When a network's hidden layer count

falls below the complexity of the problem it is trying to solve a situation known as under fitting condition arises .Another name for this is undertraining. It has a very detrimental effect on the network's efficiency. Under such circumstances, the network's temporal complexity drops dramatically and yields ineffective outcomes [15].

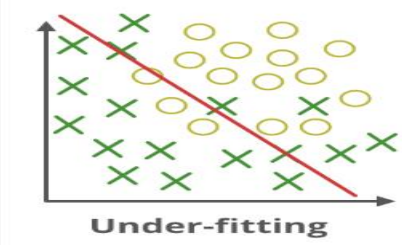


Figure1.10: Underfitting.

7.10. Loss function

A loss function, also called a cost function, is a mathematical function that measures the difference between the predicted output of a machine learning model and the true output, which is used to optimize the model's parameters and assess the model accuracy. The majority of machine learning algorithms optimizes or determines the optimal parameters (weights) for data using some form of loss function. The accuracy of the model is correlated with loss functions [17].

7.11. Convolutional Neural Network

A Convolutional Neural Network (CNN) is a type of deep neural network that is particularly well-suited for analyzing visual data such as images and videos. CNNs have revolutionized the field of computer vision by achieving state-of-the-art performance in tasks such as image classification, object detection, segmentation, and more. They are designed to automatically and adaptively learn spatial hierarchies of features from the input data [18] .As shown in (Figure 1.12).

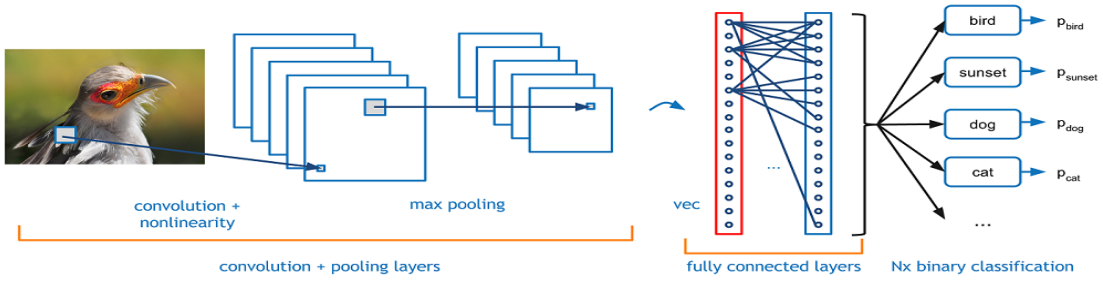


Figure1.11: An example of CNN architecture for image classification.

7.12. Recurrent neural network

Recurrent neural networks (RNN) are a class of neural networks that are trained to process sequential data inputs and transform them into specific sequential data outputs. Sequence data is data such as words, sentences, or time series data in which sequence components are related to each other based on complex semantic and grammatical rules. This type of neural network is used to generate labels for images that have not been tagged [19]. RNN is mainly applied in the area of speech processing and NLP contexts unlike conventional networks.

RNN works by storing the output of a specific layer and feeding it back to the input to predict the output of that layer [20].

The nodes in different layers of the neural network are compressed to form a single layer of recurrent neural networks. A, B, and C are the parameters of the network. a typical unfolded RNN diagram is illustrated in (Figure 1.13).

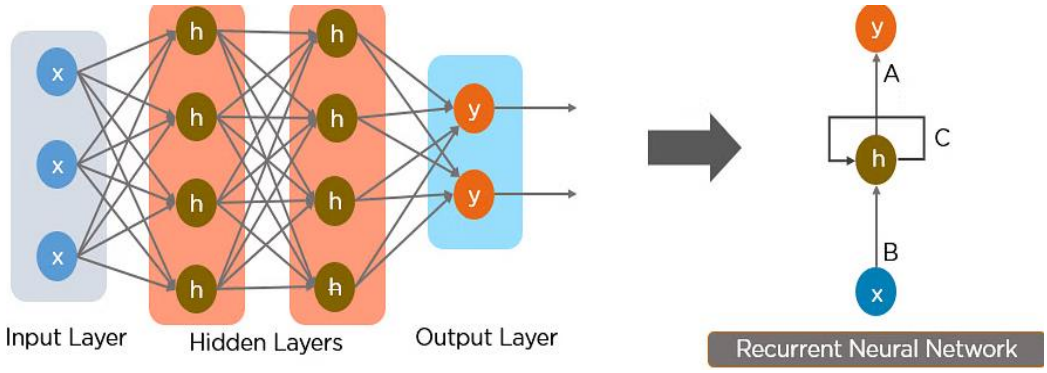


Figure1.12: Simple Recurrent Neural Network.

8. Deep Learning Applications

Deep learning serves as a versatile tool applicable across various domains:

- Self-Driving Cars: Deep learning facilitates decision-making in autonomous vehicles by creating accurate representations of the environment.
- Natural Language Processing (NLP): Transforming language-related tasks such as translation and textual analysis.

- Virtual Assistants: Cloud-based applications capable of understanding and responding to natural language voice commands, exemplified by Amazon Alexa, Cortana, Siri, and Google Assistant.
- Entertainment: Elevating special effects and generating lifelike characters in media.
- Healthcare: Supporting disease diagnosis, prognosis, treatment recommendations, and medical imaging.
- Visual Recognition: Achieving cutting-edge performance in tasks like image classification and object detection.
- Computer Vision: A subset of deep learning enabling automated interpretation of digital images and videos, pivotal in areas such as surveillance and quality control across industries.
- Vocal AI: Converting human speech to text, exemplified by Google's deep learning models capturing voice and providing search results.
- Chatbots: Comprehending and responding to human queries in natural language.
- Advertising: Enhancing user experience by optimizing ad relevance and enhancing campaign effectiveness, potentially reducing acquisition costs while enabling data-driven predictive advertising and real-time bidding [21].

9. Conclusion

In this chapter, we explored the foundations of artificial intelligence, machine learning, focusing particularly on machine learning and deep learning. By understanding the distinctions between these fields and exploring their practical applications. By dissecting the fundamental principles underlying neural networks, training algorithms, and optimization techniques, we've provided readers with a comprehensive understanding of how deep learning operates. From supervised and unsupervised learning to convolutional and recurrent neural networks, we've explored the diverse landscape of deep learning methodologies and their real-world implications. In the next chapter, we will present Convolution Neural Networks (CNN) and transfer learning.

Chapter 02 : Convolutional neural network and transfer learning

1. Introduction

ImageNet has served as a benchmark dataset for evaluating the performance of various computer vision tasks, including image classification, object detection, object localization, and semantic segmentation. Many state-of-the-art deep learning models have been trained and evaluated on subsets of the ImageNet dataset. Deep learning algorithms require huge amounts of data for effective training, and ImageNet provided an ideal training ground for researchers to develop powerful models for image recognition. These pre-trained models then act as a springboard for transfer learning, where the learned features are adapted to new, and specific tasks using smaller datasets. This saves time and resources compared to training a CNN from scratch for each new task.

2. ImageNet

ImageNet is a large visual database designed for use in visual object recognition software research. It was created by researchers at Stanford University and Princeton University and contains millions of labeled images across thousands of categories. ImageNet has been pivotal in advancing the field of computer vision, particularly in the development and benchmarking of deep learning models.

The ImageNet dataset contains of over 14 million images across according to the WordNet hierarchy and more than 20,000 categories, with a typical category consisting of several hundred images. Since 2010, the dataset has been used in the ImageNet Large Scale Visual Recognition Challenge (ILSVRC), a benchmark in image classification and object detection. The publicly released dataset contains a set of manually annotated training images. This contains approximately 1.2 million images for training, 50,000 images for validation, and 100,000 images for testing. Each image in the dataset is associated with a single label indicating the object or scene depicted in the image, with each category having a unique numerical identifier, with the manual annotations withheld. The ImageNet project does not own the copyright to the images, therefore, only thumbnails and URLs of the images are provided. ImageNet's diverse labeled data, combined with its complexity, has enabled the development of robust and accurate image recognition systems, providing researchers with a valuable resource for deep learning models [22].

3. Important CNN layers

Important layers of the convolution neural network (CNN) can vary depending on the network's specific structure and the task at hand. Here are some layers commonly used in many CNN networks:

3.1. Convolutional Layer

A convolution layer is a fundamental component of the CNN architecture that performs feature extraction, which typically consists of a combination of linear and nonlinear operations, convolution operation and activation function.

The convolution operation involves applying small, learnable filters on the input volume, sliding them across the input, computing dot products at each position, and applying a nonlinear activation function to generate activation maps. These maps are then stacked to form output volumes [23] .As shown in below (Figure 2.1)

Key hyperparameters like depth (number of filters), stride (step size of filter movement), and padding (control of output size) influence the properties of the output volume. Stride determines how many steps the filter moves across the input, affecting spatial dimensions, while padding prevents information loss by adding zeros around the input volume's border, maintaining spatial dimensions. Common choices for padding include "valid convolution" (no padding) and "same convolution" (output size matches input size).

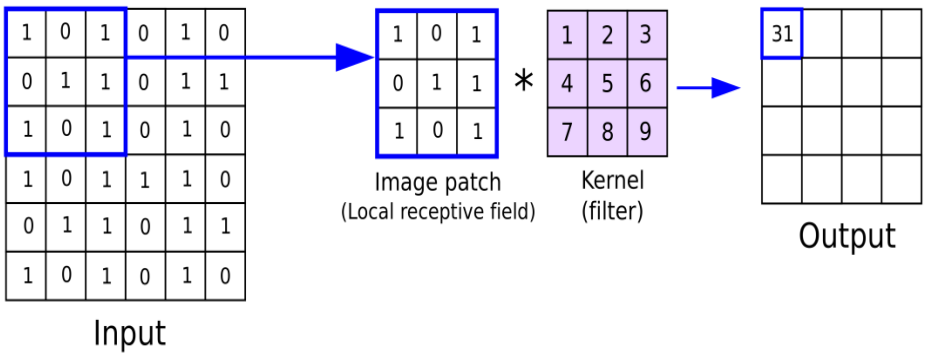
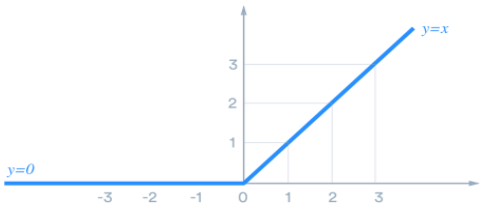
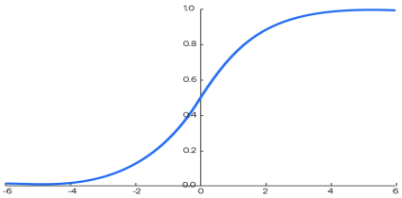
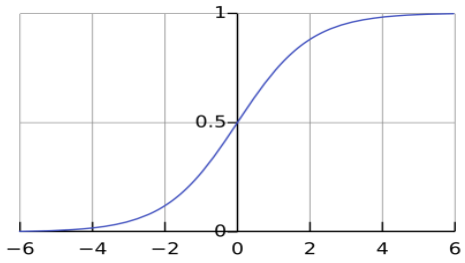
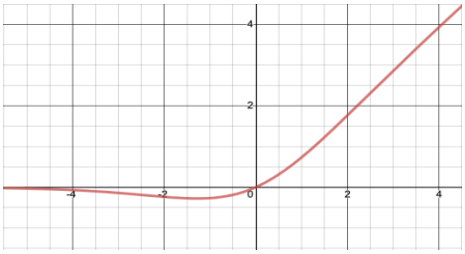


Figure2.1: Convolutional process.

3.2. Activation function

Activation functions are mathematical operations applied to the output of neurons in artificial neural networks. These functions introduce non-linearity into the network, enabling it to learn complex patterns and relationships in the data. Here are some activation functions used in neural networks:

Table2.1:Activation functions and their mathematical operations.

Type of fonction	Mathematical Operations	Plot
Relu	y = max (0, x) [24]	
Softmax	$\sigma(z)=\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$ [25]	
Sigmoid	$\sigma(x)=\frac{1}{1+e^{-x}}$ [25]	
Swish	F (x) = x · Sigmoid (βx) Where β: is a scalable and trainable parameter. [26]	

3.3. Pooling Layer

The pooling layer, also known as (subsampling or downsampling) is often used in a convolutional neural network. Its objective is to gradually reduce the spatial size of representation. Therefore, it reduces the number of features and computational complexity of the network. The assembly layer on each feature map operates independently.

The main reason for the pooling layer is to prevent the model from overfitting. In the field of image processing, pooling does not affect the number of kernels. The hyperparameters of the pooling layer represent the filter size and strides. The most commonly used pooling layer is with filter size 2 and with stride 2.

There are usually four types of pooling: [27]. The most common functions are the following

- **Maximum pooling:** (or max pooling) is the most common approach used in the pooling process that selects the maximum element of the feature map area covered by the kernel.

- **Average pooling:** average pooling involves calculating the average for each patch of the feature map. Below Figure 2.2**,** illustrate the working of average and maximum pooling

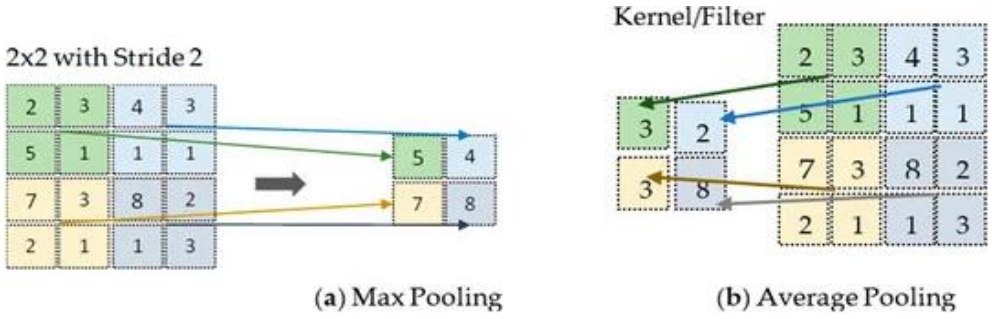


Figure2.2: Two different pooling techniques were applied.

There are other types of pooling layer such as:

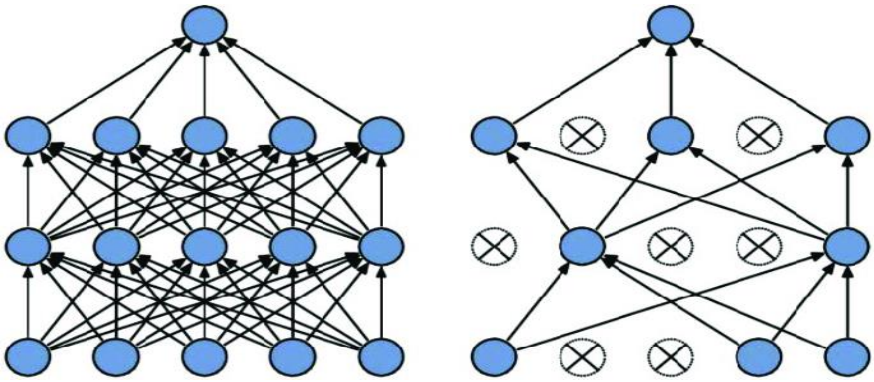
- Minimum pooling: (or min pooling) selects the minimum element from region of the feature map.

- Global pooling: Instead of sampling the input feature map, the global pooling compiles the entire feature map into one value.

3.4. Dropout

Dropout introduces regularization within the network, which ultimately improves generalization by randomly skipping some units or connections with a certain probability. Overfitting occurs when a model learns to memorize the training data rather than generalize well to new, unseen data. Dropout addresses this issue by randomly ignoring ('dropping out') a fraction of neurons during training.

During inference (testing or prediction), dropout is typically turned off, and all neurons are used, but their outputs are scaled by the dropout probability used during training. As a result, the network cannot rely on specific neurons and their connections during training, forcing it to learn more robust and generalizable features. [28] .Figure 2.3 presents an example of applying the dropout method.



(a) (b)

Figure2.3: An example of two Neural networks: (a): Standard Neural Network - (b): After applying dropout method.

3.5. Flatten Layer

After passing the convolution and pooling layers and before entering the fully connected layers, the output of the earlier layers is passed for flattening. This operation does not alter the content of the data; it simply rearranges the data into a format that can be fed into the subsequent fully connected layers This allows neural networks to process the data effectively for tasks like classification, regression, or other forms of analysis [29].

The Flatten layer takes the multi-dimensional input data and reshapes it into a one-dimensional vector.

The operation essentially "flattens" or "unrolls" the input data, stacking all the elements into a single linear sequence, while ensuring that the total number of elements remains unchanged. As shown in below (Figure 2.4).

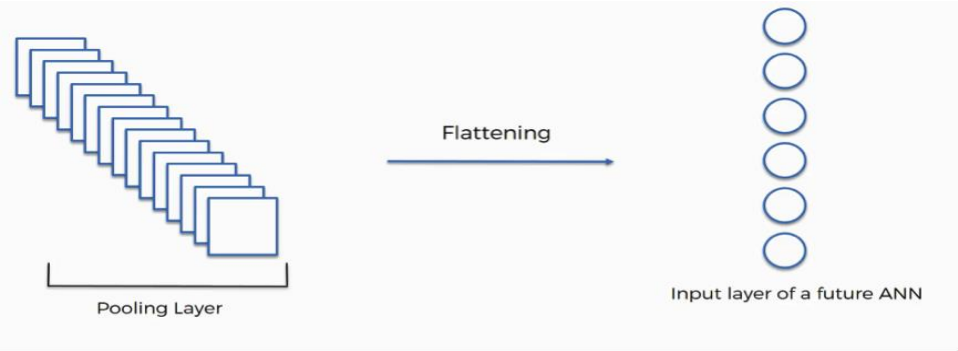


Figure2.4: Flatten layer.

3.6. Dense layer

A Dense layer, also known as a fully connected layer, is a fundamental building block in neural network architectures. It is called "dense" because each neuron in a dense layer receives input from every neuron in the previous layer, forming a fully connected network. Each connection between neurons in adjacent layers has an associated weight, which determines the strength of the connection.

Each neuron in the layer applies an activation function to its weighted sum of inputs and bias term, topically softmax function.

The output of a dense layer is a set of activations corresponding to each neuron, computed based on the weighted sum of inputs, biases, and activation function applied. As shown in (Figure 2.5) [30].

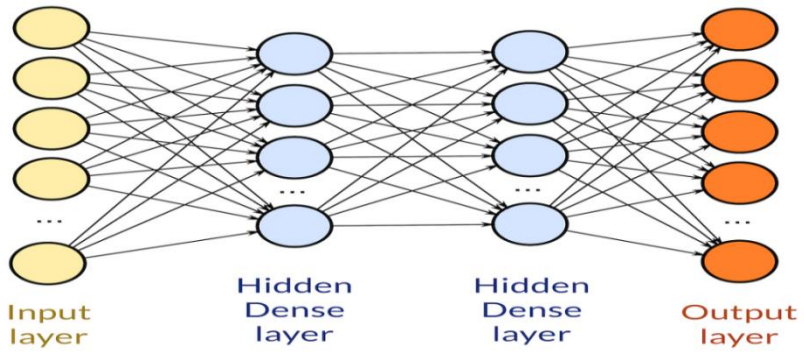


Figure2.5: Dense layer.

4. Transfer Learning

Transfer learning is a machine learning technique where a model trained on one task is reused on a related task. It involves leveraging knowledge gained from solving one problem to improve optimization in another setting. This approach allows models to benefit from pre-existing knowledge, leading to improved performance, faster convergence, and better generalization to new tasks or domains. Transfer learning is typically applied when there is a new dataset smaller than the original dataset used to train the pre-trained model [31].

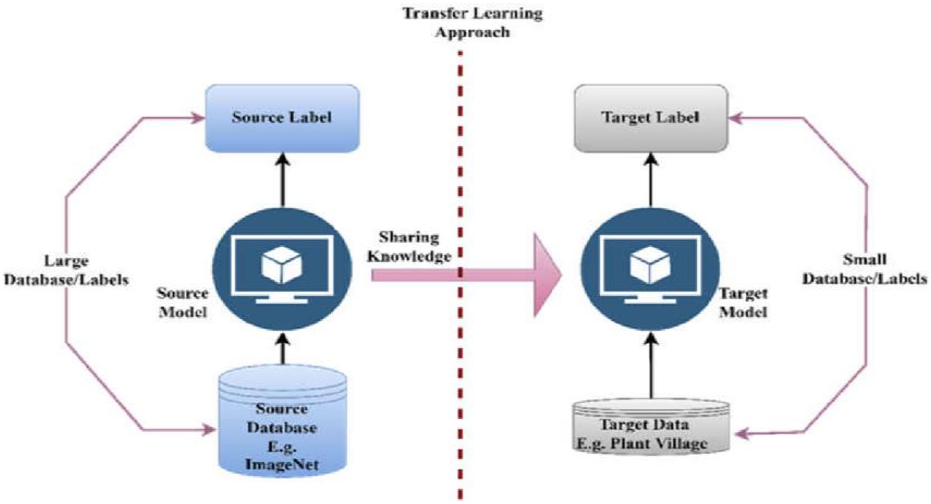


Figure2.6: Basic idea behind transfer learning.

4.1. Transfer learning architectures

Feature Extraction: utilize a pre-trained model to extract features from the data related to the target task, then feed these features into a new classifier trained specifically for the target task.

Fine-tuning: adjust or fine-tune the weights of a pre-trained model during training on the target task dataset, allowing the model to adapt to the nuances of the new task while retaining knowledge learned from the original task.

Domain Adaptation: transfer knowledge from a source domain to a target domain, focusing on aligning the distributions of the source and target domains to improve performance on the target domain.

Multi-task Learning: train a single model on multiple related tasks simultaneously, sharing knowledge across tasks to potentially improve performance on each individual task, often seen as a form of transfer learning when the tasks are related.

5. Some pre-trained models

5.1. VGG16 model

The VGG16 model is a convolutional neural network architecture that was developed by the Visual Geometry Group (VGG) at the University of Oxford. It gained popularity for its simplicity and effectiveness in image classification tasks.

The "16" in VGG16 refers to the fact that it has 16 layers, including 13 convolutional layers and 3 fully connected layers. As shown in (Figure 2.7). It's organized into five blocks, with convolutional layers followed by max-pooling layers. Each convolutional layer uses a 3x3 kernel size and Relu activation. Fully connected layers have 4096 neurons with Relu activation, and the output layer uses softmax for classification. VGG16 has approximately 138 million parameters, enabling it to learn complex features but also demanding significant computational resources. The architecture consists of a series of convolutional layers with small 3x3 filters followed by max-pooling layers to down sample the feature maps [32].

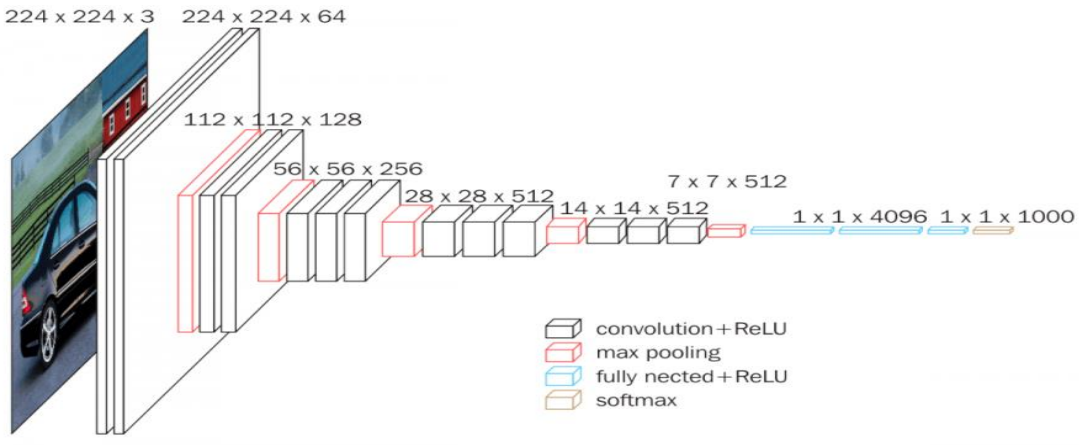
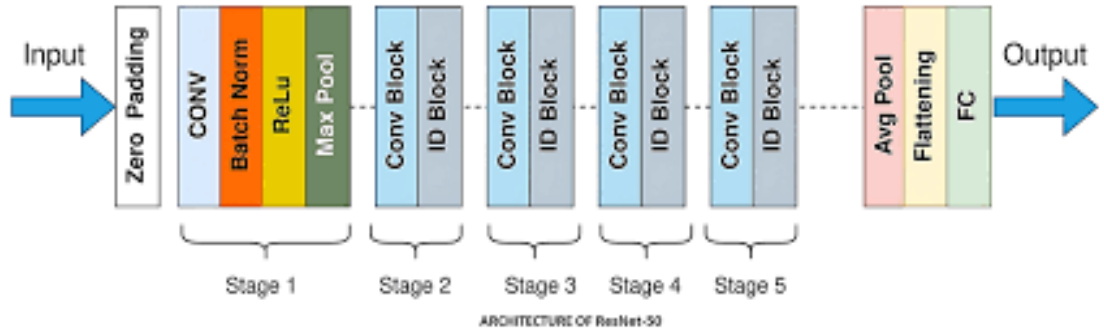


Figure2.7: VGG 16 neural network architecture.

5.2. ResNet-50 model

ResNet50 is another popular convolutional neural network architecture, known for its depth and residual learning approach, which was introduced by Microsoft Research in 2015. "ResNet" stands for "Residual Network".

The key innovation in ResNet is the introduction of residual blocks, which allow for training very deep networks without suffering from the vanishing gradient problem. ResNet-50 is 50 layer architecture. The architecture consists of an initial convolutional layer, max-pooling, residual blocks, a bottleneck architecture global average pooling, a fully connected layer, and an output [33].As shown in (Figure 2.8).



.

Figure2.8: ResNet-50 model architecture.

5.3. DenseNet201 model

DenseNet-201 is a convolutional neural network architecture that is part of the DenseNet family, introduced in the paper "Densely Connected Convolutional Networks" by Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger in 2016. It is characterized by its depth, consisting of 201 layers in total.

The DenseNet-201 architecture incorporates dense connectivity, where each layer receives feature maps from all preceding layers and passes its own feature maps to all subsequent layers. This dense connectivity pattern facilitates feature reuse and enhances gradient flow throughout the network, leading to better feature propagation and representation learning.

To manage computational complexity, DenseNet-201 utilizes bottleneck layers, which reduce the number of input feature maps before passing them through dense blocks. These bottleneck layers help maintain representational capacity while reducing the computational cost of the network [34].

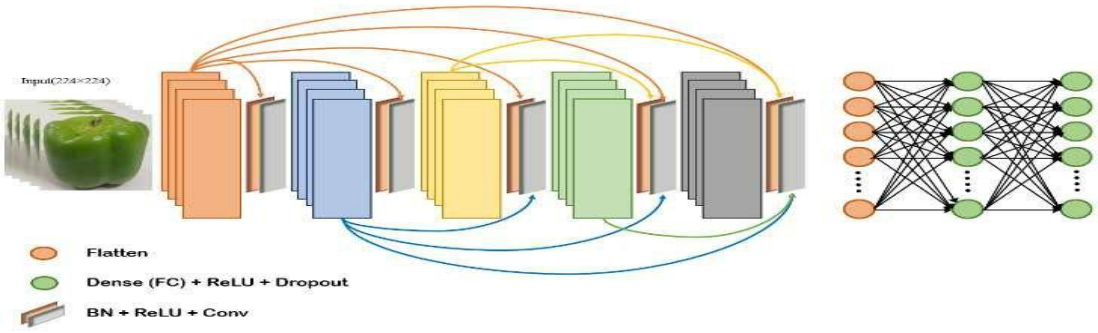


Figure2.9: DenseNet-201 model architecture.

5.4. MobileNetV2 model

MobileNetV2 is a mobile-friendly convolutional neural network architecture designed for efficient deployment on mobile and embedded devices. It was introduced by researchers at Google in their paper titled "MobileNetV2: Inverted Residuals and Linear Bottlenecks" in 2018.

Depthwise Convolution: In a standard convolutional layer, each filter operates on the full depth of the input feature maps. Depthwise convolution, on the other hand, applies a separate convolutional filter to each input channel independently.

Pointwise Convolution: After the depthwise convolution, MobileNet applies a 1x1 pointwise convolution to mix and adjust the features. This step helps to increase the model's representational power by allowing the network to learn linear combinations of the depthwise features.

Batch Normalization and Swish: Batch normalization and the swich activation are typically applied after each convolutional operation to stabilize training and introduce non-linearity [35].

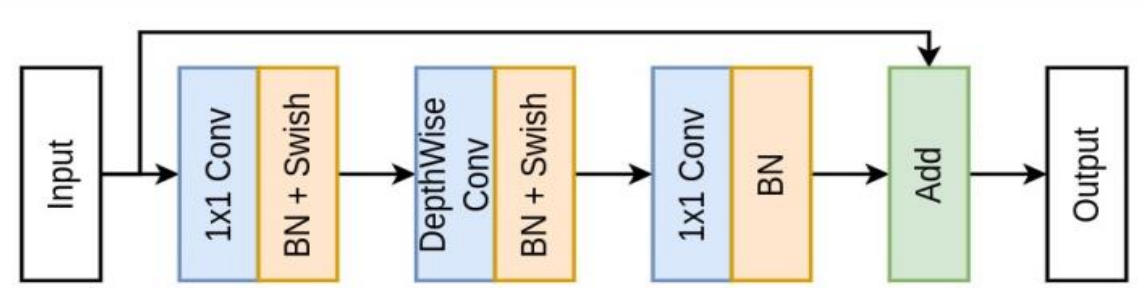


Figure2.10: MobileNetV2 neural network architecture.

6. Conclusion

In this chapter, we provided an overview of ImageNet's database, highlighting its importance in training machine learning models given the wide variety of images. We discussed the main layers of convolutional neural networks, explaining their functions and roles in deep learning structures. In addition, we introduced the concept of transport learning, the most important of its fundamentals and presented some of the pre-trained model.

Chapter 03: Materials and experimental results

1. Introduction

In recent years, deep learning models have made significant strides in the field of image classification, enabling automated systems to accurately identify and classify objects. This research focuses on leveraging advanced models, specifically transformers and their adaptations for vision tasks, to enhance image classification performance. The PlantVillage dataset, sourced from Kaggle, will be utilized to demonstrate these capabilities by classifying healthy and diseased plant leaves.

2. The Transformer

The transformer model was primarily developed based on the multi-head attention mechanism proposed in 2017, with the aim of processing sequential data its outstanding performance. The transformer structure consists of both an encoder and a decoder [36].

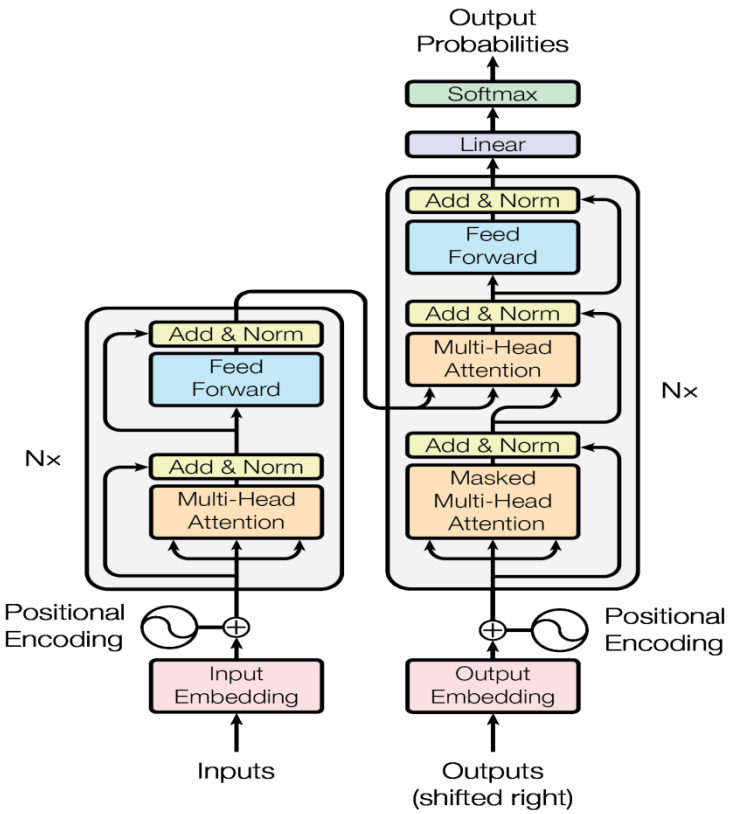


Figure3.1: Transformer architecture.

2.1. Encoder

Processes the input sequence to generate a set of high-level representations. Key components include input embeddings, positional encodings, multi-head self-attention mechanisms, feedforward neural networks, and layer normalization. Captures the semantics of the input sequence by attending to all other tokens and learning complex relationships [36].

2.2. Decoder

The decoder uses a masked multi-head self-attention mechanism, allowing each position to attend to all positions up to the current one. The masking ensures that a position cannot attend to future positions, preventing information leakage during training. Mathematically, this is achieved by applying a mask that sets the attention scores of future tokens to negative infinity, effectively ignoring them in the attention computation [36].

3. Key Components of Transformers

- **Self-Attention Mechanism:** This allows the model to weigh the importance of different parts of the input data (e.g., words in a sentence) when making predictions, effectively capturing long-range dependencies and contextual information.
- **Feed-Forward Neural Networks:** Applied to the output of the attention layers, these enhance the representation capability.
- **Positional Encoding:** Since Transformers don't inherently process sequences in order, positional encoding is used to retain the order information in the input sequence.

4. Vision transformers (ViTs)

Proposed by Dosovitskiy et al. in "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale" (2020), Vision Transformers adapt the Transformer encoder architecture for image classification tasks [37].

4.1. Vision transformers (ViTs) workflow

Image patch embedding: Instead of processing words, ViTs process images by splitting them into fixed-size patches. Each patch is then flattened and projected into a high-dimensional space to form embeddings.

Positional embedding: Similar to NLP, positional embeddings are added to these patch embeddings to retain spatial information.

Transformer encoder: The sequence of patch embeddings, with positional encodings, is passed through a stack of Transformer encoder layers, leveraging the self-attention mechanism to learn global dependencies across the entire image [37].

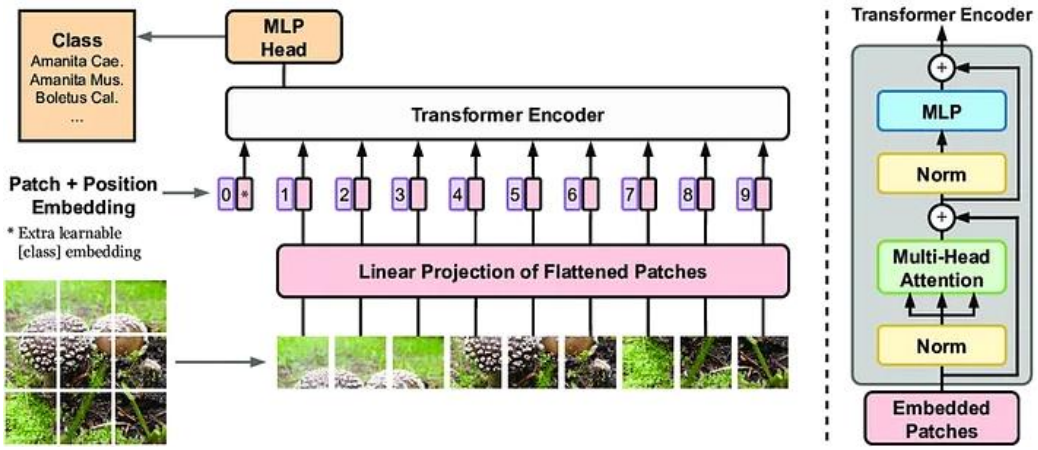


Figure3.2: Architecture ViT model.

5. MobileViT overview

MobileViT is a lightweight transformer model designed to combine the strengths of convolutional neural networks (CNNs) and vision transformers (ViTs) to achieve efficient and effective performance on mobile devices. This hybrid approach allows MobileViT to leverage the local feature extraction capabilities of CNNs and the global context understanding of transformers, making it particularly suitable for vision tasks on resource-constrained devices. Three different network sizes (S: small, XS: extra small, and XXS: extra extra small) are designed. The differences between MobileViT-XXS, MobileViT-XS, and MobileViT-S primarily lie in their model sizes and complexity, which are tailored to various computational and performance requirements.

Table3.1:Comparison of MobileViT model variants.

Model	Parameters	Intended Use	Performance
MobileViT-XXS	Approximately 1.3 million parameters.	Designed for extremely resource-constrained environments like low-power edge devices.	Provides good performance while being highly efficient in terms of computational resources.
MobileViT-XXS	Approximately 2.3 million parameters.	Suitable for moderately resource-constrained environments.	Offers better performance than MobileViT-XXS with a reasonable increase in computational requirements.
MobileViT-XXS	Approximately 5.6 million parameters.	Aimed at environments with more substantial computational resources.	Provides the highest performance among the three variants, suitable for more demanding applications.

- ✓ In our project, choise the MobileViT-XXS model. It is the smallest in computational power and is suitable for highly constrained devices.

6. MobileViT architecture

The architecture of MobileViT integrates components from MobileNetV2 and vision transformers, combining their advantages to form a lightweight and low-latency network [38].

6.1. CNN components (MobileNetV2 Blocks)

Local feature extraction: MobileNetV2 blocks are used in the initial stages of MobileViT to efficiently extract local features from the input images. MobileNetV2 is known for its depthwise separable convolutions, which significantly reduce the number of parameters and computational cost while maintaining good performance.

Inverted residuals: The inverted residual structure of MobileNetV2, along with linear bottlenecks, helps in preserving the representational power of the network while reducing the model's size and computational requirements [38].

6.2. Transformer components (ViT)

Global context understanding: Vision transformers are integrated into the architecture to capture global dependencies and contextual information across the entire image. This is achieved by using multi-head self-attention mechanisms that allow the model to attend to different parts of the image simultaneously [38].

Flexibility and scalability: The transformer components in MobileViT add flexibility to the model, enabling it to learn complex relationships between distant pixels, which are crucial for tasks requiring a high-level understanding of the visual scene [38].

We explained previously architecture of MobileViT, as illustrated (Figure3.3).

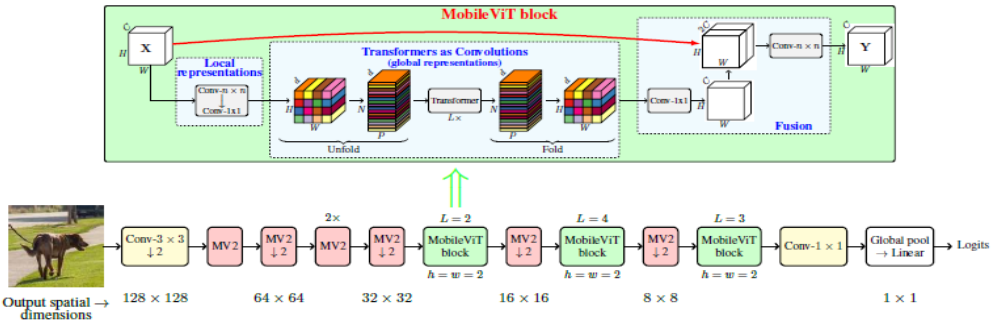


Figure3.3: Architecture MobileViT model.

7. Why use MobileViT instead of ViT for mobile applications

- ViTs have high computational demand, whereas MobileViTs are optimized with lightweight convolutions and Transformers for better efficiency.
- ViT high memory requirements. MobileViT Reduced memory footprint, efficient management.
- ViT high energy consumption, impractical for battery-powered devices. MobileViT energy-efficient.
- ViT have higher latency and slower inference, while MobileViTs provide low latency and faster inference, making them ideal for real-time mobile applications.

To summarize:

MobileViT addresses the limitations of ViT by being more efficient in terms of computational demand, memory usage, and energy consumption. This makes

MobileViT a better choice for tasks that require real-time performance on mobile and embedded devices.

8. Project requirement

8.1. Python

Python is an interpreted, high-level, general-purpose programming language. Created by Guido van Rossum and first released in 1991, Python's design philosophy emphasizes code readability with its notable use of significant whitespace. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects [39].

Python libraries used for this project:

8.1.1. TensorFlow: is an open-source library developed by the Google Brain team for machine learning and artificial intelligence applications. It provides a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML, and developers easily build and deploy ML-powered applications [40].

8.1.2. Keras: is a high-level neural networks API, written in Python, designed for ease of use and fast experimentation. It can run on top of several backends, including TensorFlow, CNTK, and Theano. Developed with a focus on enabling rapid prototyping, Keras simplifies the process of building and training deep learning models [41].

8.1.3. NumPy :(Numerical Python) is a fundamental package for scientific computing in Python. It provides support for arrays, matrices, and a large collection of mathematical functions to operate on these data structures. NumPy is essential for numerical calculations and serves as the foundation for many other scientific libraries in Python [42].

8.1.4. Matplotlib: is a comprehensive library for creating static, animated, and interactive visualizations in Python. It is widely used for generating plots, graphs, and charts [43].

8.1.5. Scikit-learn (sklearn): is a free versatile and easy-to-use machine learning library for Python ,it provides efficient tools for predictive data analysis and reusable in various contexts Built on NumP , SciPy and Matplotlib [44].

8.2. Google Colaboratory: often referred to as Colab, is a cloud-based platform provided by Google that allows users to write and execute Python code in a browser, with no setup required. It's particularly popular among data scientists, researchers, and educators for its ease of use, free access to GPU and TPU resources, and seamless integration with other Google services like Google Drive **[**45].

8.3. Hardware

For our implementation and testing, we worked:

1. Online on Google Colaboratory CPU.

2. On a computer having the following characteristics:

• Operating system: Windows 10 Professionnel;

• Operating system architecture type: 64 bits;

• Processor: Intel(R) Core(TM) i3-6100U CPU @ 2.30GHz 2.30 GHz.

• RAM: 8 GB.

• Graphics card: Intel (R) HD Graphics 520.

9. Methodology

9.1. Dataset overview

This research article will cover model shaping for image classification, utilizing the plant village dataset gathered from Kaggle.

9.1.1. The PlantVillage dataset

The PlantVillage dataset is a widely used collection of over 54,000 RGB images of both healthy and diseased plant leaves. This dataset includes images of 14 different crop species, such as apple, blueberry, cherry, grape, orange, peach, pepper, potato, raspberry, soybean, squash, strawberry, and tomato, and covers 38 different crop species and types of diseases. The images are taken under controlled conditions and

are designed to support classification tasks, aiding in the identification of various plant diseases [46].

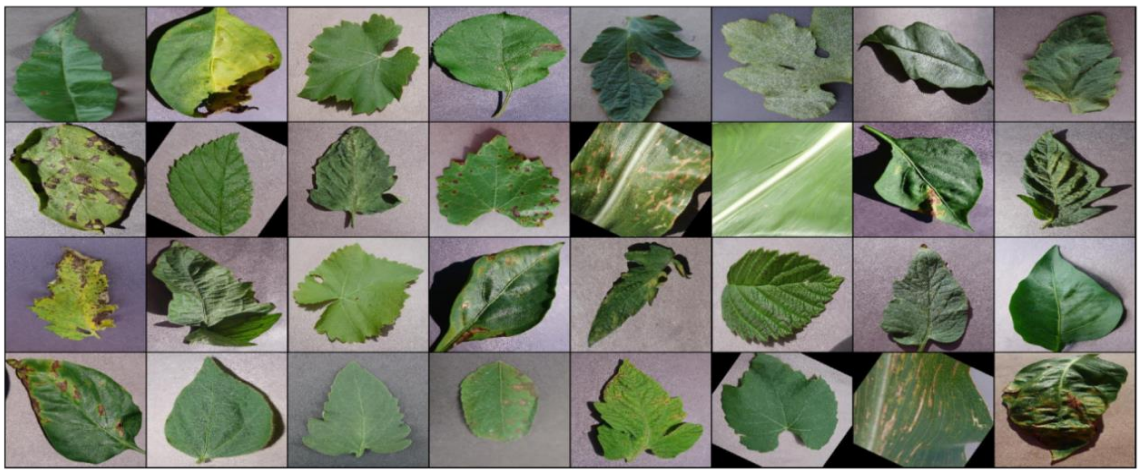


Figure3.4: Sample of images from PlantVillage dataset.

9.1.2. Dataset Discussion

In our project we will work on 3 types of plants (Apple, Corn, and Potato) with 11 classes have total 9205 images. More details illustrated can be seen in (Table 3.2).

Table3.2:Dataset description.

Class	Plant Name	Healthy or Diseased	Disease Name	Images (Number)
C1	Apple	Diseased	Apple_scab	630
C2	Apple	Diseased	Black_rot	621
C3	Apple	Diseased	Cedar_apple_rust	275
C4	Apple	Healthy	-	1645
C5	Corn_(maize)	Diseased	Cercospora_leaf_spotGray_leaf_spot	513
C6	Corn_(maize)	Diseased	Common_rust	1212
C7	Corn_(maize)	Diseased	Northern_Leaf_Blight	985
C8	Corn_(maize)	Healthy	-	1172
C9	Potato	Diseased	Early_blight	1000
C10	Potato	Diseased	Late_blight	1000
C11	Potato	Healthy	-	152
Total	-	-	-	9205

9.2. Data preparation

In a typical deep learning workflow, the initial stage involves preprocessing the data to transform raw data into a format suitable for the neural network.

The input images are resized to match the specific requirements of the chosen pre-trained models.VGG16, ResNet50, and DenseNet201 require images of size 224x224x3 (height, width, and channel).MobileViT-XXS requires images of size 256x256x3.

We partitioned the dataset into three separate subsets: training, validation, and testing, using the **splitfolders** package. The dataset was split with a ratio of 70% for training, 10% for validation, and 20% for testing. Training dataset consists 6440 images; validation consists 918 images and for testing consists 1847 images.

Splitting the data allows for training the model, evaluating its performance on unseen data, and ultimately testing its generalizability.

9.3. Model Description

This research article will test 3 pre-trained models VGG16, ResNet50, DenseNet201 with the proposed MobileViT-XXS model. The project aims to identify the most suitable CNN architecture for plant disease detection. The findings will contribute to the development of automated systems for early disease detection in plants.

9.3.1. Pre-trained models details

VGG16 tuning details

The input image dimensions for the network are 244×244×3. The first two layers have 64 channels with a filter size of 3×3 and a stride of 2. The next two layers in the VGG-16 architecture have 256 channels with 3×3 filters. Following these layers is a max-pooling layer with a stride of 2. After the pooling layer, there are two convolution layers, each with 256 channels and a 3×3 filter size. Subsequently, there are two sets of three convolution layers, each followed by a pooling layer, with 3×3 filters. The network includes one flatten layer, five max-pooling layers, and two dense layers.

ResNet-50 turning details

The input image dimensions for the network are 224 × 224 × 3. The first layer is a convolutional layer with a kernel size of 7×7, a stride of 2, and 64 channels. The next three stages consist of convolution layers with filter sizes of 1×1, 3×3, and 1×1, and with 64, 64, and 256 channels, respectively. These layers are repeated three times. The following stages include the next set of convolution layers, which are repeated four times. Subsequent convolutional blocks are repeated six times.

DenseNet-201 tuning details

DenseNet-201 processes input images with dimensions 224×224×3. The network consists of four dense blocks. In the first dense block, convolution is performed with 1×1 and 3×3 filter sizes, repeated six times. In the second dense block, convolution is performed with 3×3 and 1×1 filter sizes, and the steps are repeated 12 times. In the third dense block, convolution operations with the same filter sizes are repeated 24 times. In the fourth dense block, the steps are repeated 16 times. Transition blocks, which include convolution and pooling layers, are situated between the dense blocks; more details illustrated can be seen in (Table 3.3).

Table3.3:The architecture of the pre-trained models.

Les modèles	VGG-16	ResNet-50	DenseNet-201
Total des couches	16	50	201
Max pool	5	1	4
Couches denses	3	3	-
Drop-out	2	2	-
Couches de Flatten	1	1	-
Tailles des filtres	3×3	3×3	3×3 ,1×1
Strides	2×2	2×2	2×2

9.3.2. Model proposed details

MobileViT-XXS tuning details

The architecture begins with a strided 3×3 standard convolution layer, followed by MobileNetV2 (MV2) blocks and MobileViT blocks. The Swish activation function is used throughout the network. In the MobileViT block, n=3 is used for the convolutional layers. The spatial dimensions of the feature maps are multiples of 2, with height (h) and width (w) set to 2 at all spatial levels. The MobileViT block is designed to model both local and global information in an input tensor with fewer parameters. For an input tensor $X \in R^{HxWxC}$, the MobileViT block first applies an n×n standard convolutional layer, followed by a point-wise (or 1×1) convolutional layer, producing $X_L \in R^{HxWxd}$. The $n \times n$ convolutional layer encodes local spatial information, while the point-wise convolution projects the tensor to a high-dimensional space (d-dimensional, where $d > C$) by learning linear combinations of the input channels, more details illustrated can be seen in (Table 3.4).

Table3.4:The architecture of the MobileViT-XXS models.

Layer	**Output size**	**Output stride**	**Output channels(XXS)**
Image	256 × 256	1	
Conv-3 × 3, ↓ 2 MV2	128 × 128	2	16 16
Conv-3 × 3, ↓ 2 MV2	64 × 64	4	24 24
MV2, ↓ 2 **MobileViT block (L = 2)**	32 × 32	8	48 48(d=64)
MV2, ↓ 2 **MobileViT block (L = 4)**	16 × 16	16	64 64(d=80)
MV2, ↓ 2 **MobileViT block (L = 3)** **Conv-1 × 1**	8 × 8	32	80 80(d=96) 320

9.4. Fine-Tuning of hyper parameters in pre-trained models

When using pre-trained models, we typically don't train the entire model from scratch. Instead, we leverage the pre-trained weights and focus on adjusting the model's final layers to the specific task of detecting plant disease. This process is called fine-tuning; more details illustrated can be seen in (Table 3.5).

Table3.5: Training parameters of model.

MODLE	VGG16, ResNet-50, DensNet-201	MobileViT-XXS
Epochs	20	20
Batch Size	32	32
Activation in middle layers	Relu	Swich
Activation in Final layer	Softmax	Softmax

9.5. Results of fine training model

The following table (3.6) shows the results of the training of different models (VGG16, RESNET-50, Densenet-201, MobileViT-XXS) where the complexity of the model (parameters) and performance (loss and accuracy) in both training and test dataset.

Table3.6: Comparison of pre-trained models with MobileViT-XXS model propose.

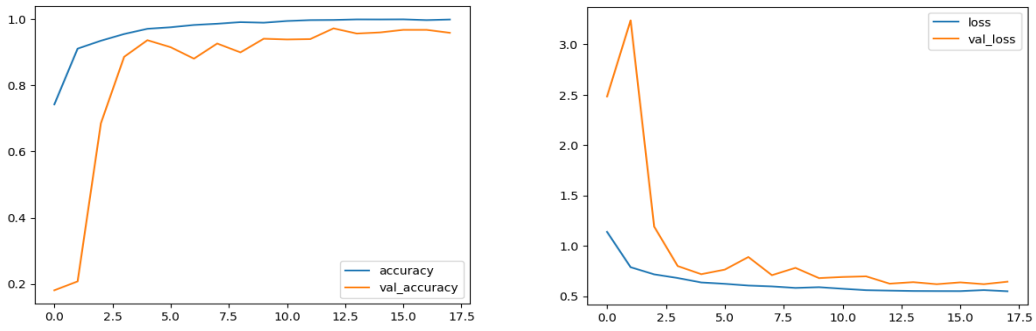
Parameters	VGG16	RESNET-50	Densenet-201	MobileViT-XXS
Number of total model parameters	27.626.823	75.035.531	66.558.539	1.309.547
Training loss	0.028	0.1676	0.1127	0.5484
Test loss	0.1897	0.2127	0.1487	0.6131
Train accuracy	0.9951	0.9740	0.9651	0.9989
Test accuracy	0.9383	0.9426	0.9503	0.9713

9.6. Discussion

The results suggest that MobileViT-XXS is a highly efficient and effective model despite its small size. It outperforms the larger models in terms of both training and test accuracy. This efficiency could be particularly valuable in applications where computational resources are limited or in mobile and edge computing scenarios.

VGG16, ResNet-50 and DensNet-201 are robust models with good performance, but their higher parameter counts and lower efficiency compared to MobileViT-XXS make them less suitable for resource-constrained environments.

It is clear that MobileViT-XXS is demonstrated the best performance among the threemodels on the basis of our quantitative performance evaluation paramaeters. We have plotted training and test accuracy and loss of MobileViT-XXS which was conducted on our training and validaction set. The accuracy and loss graphs are depicted in (Figure 3.6)



(a) Accuracy training and validation

(b) Loss training and validation

Figure3.5: Results of MobileViT-XSS training and validation phases.

9.7. Diving Deeper into MobileViT-XXS result

The classification report summarizes the performance of a model across different classes by evaluating precision, recall, and F1-score. These metrics provide a detailed understanding of how well the model is performing for each specific class.

Precision: The ratio of true positive predictions to the total predicted positives. High precision indicates a low false positive rate.

Recall: The ratio of true positive predictions to the actual positives. High recall indicates a low false negative rate.

F1-Score: The harmonic mean of precision and recall, providing a balance between the two.

	precision	recall	f1-score	support
Apple___Apple_scab	1.00	0.94	0.97	126
Apple___Black_rot	0.97	0.98	0.97	125
Apple___Cedar_apple_rust	1.00	0.98	0.99	56
Apple___healthy	0.97	1.00	0.99	330
Corn_(maize)___Cercospora_leaf_spot Gray_leaf_spot	0.84	0.83	0.83	103
Corn_(maize)___Common_rust_	1.00	1.00	1.00	243
Corn_(maize)___Northern_Leaf_Blight	0.92	0.92	0.92	198
Corn_(maize)___healthy	1.00	1.00	1.00	235
Potato___Early_blight	0.99	0.98	0.99	200
Potato___Late_blight	0.98	0.99	0.99	200
Potato___healthy	0.97	0.90	0.93	31
accuracy			0.97	1847
macro avg	0.97	0.96	0.96	1847
weighted avg	0.97	0.97	0.97	1847

Figure3.6: The classification report of MobileViT-XXS.

The confusion matrix based on the provided classification report. Each cell in the matrix represents the count of actual instances (rows) versus predicted instances (columns).

Diagonal elements (from top-left to bottom-right) indicate correct predictions, while off-diagonal elements indicate misclassifications. Some misclassifications are present, such as cases 4 errure, the actual disease "Apple___Apple_scab" and the predict is "Appel_Black_rot".

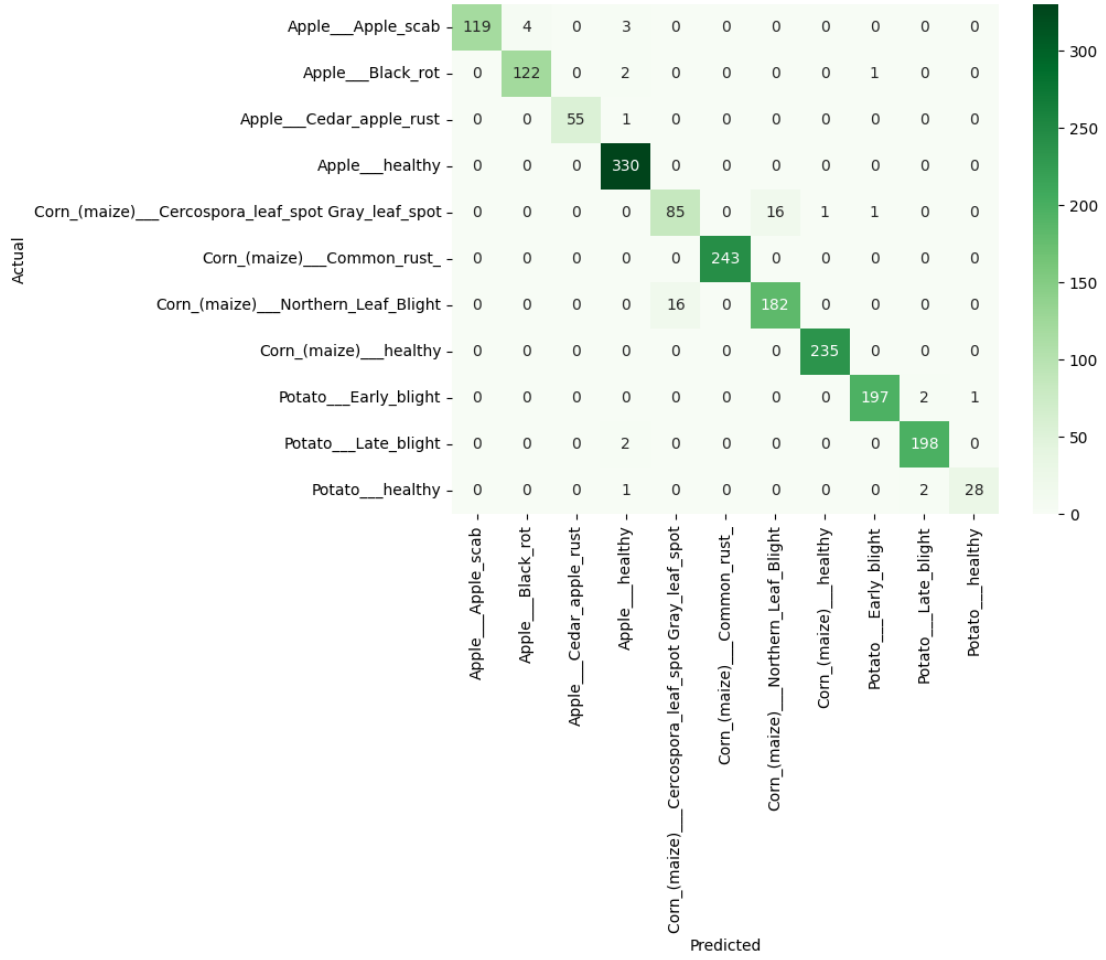


Figure3.7: The confusion matrix of MobileViT-XXS model.

10. Conclusion

In conclusion, MobileViT-XXS demonstrates that smaller, more efficient models can outperform traditional deep learning architectures, especially in scenarios requiring low computational overhead and high accuracy. MobileViT is a relatively new architecture compared to VGG16, ResNet50, or DenseNet201. More research and development might be needed to fully understand its strengths and weaknesses. In the following chapter, we will test the deployment of a mobile application for plant diseases detection using a TensorFlow Lite for MobileVit-XXS model.

Chapter 04:
Model deployment

1. Introduction

Deep learning is more than just building a model and predicting responses to it. We have to make sure the model works properly in production. We need an application that can help anyone (people who can't program) use the model and make predictions. Deploying a mobile application refers to the entire process of getting your app ready for users, as emphasized by Luigi Patruno's assertion that a machine learning model only holds value once it is deployed and utilized effectively.

2. Methods of deployment

There are various tools to deploy like Docker, Kubernetes, Heroku, Django, Flask and TensorFlow Lite.

We chose TensorFlow Lite of deploying the model Mobilevit-XXS for its efficiency and suitability for mobile devices. It allows for smooth disease prediction on smartphones.

2.2. TensorFlow Lite: is a lightweight version of Google's TensorFlow framework, designed specifically for mobile and embedded devices. It allows developers to bring machine learning capabilities to mobile applications, ensuring efficient performance and low latency. Here’s a detailed guide to deploying a Tensorflow Lite model on mobile applications, specifically for Android and iOS platforms [47].

TensorFlow Lite provides three essential components:

Converter: Converts Tensorflow models to Tensorflow Lite format, optimizing them for mobile and embedded devices.

Interpreter: Executes tensorflow lite models on mobile and embedded devices, handling model operations and inferences from input data.

Model Maker: Simplifies the creation of custom deep learning models for specific tasks, automatically selecting architectures, tuning hyperparameters, and converting them to TensorFlow Lite format for mobile and embedded use.

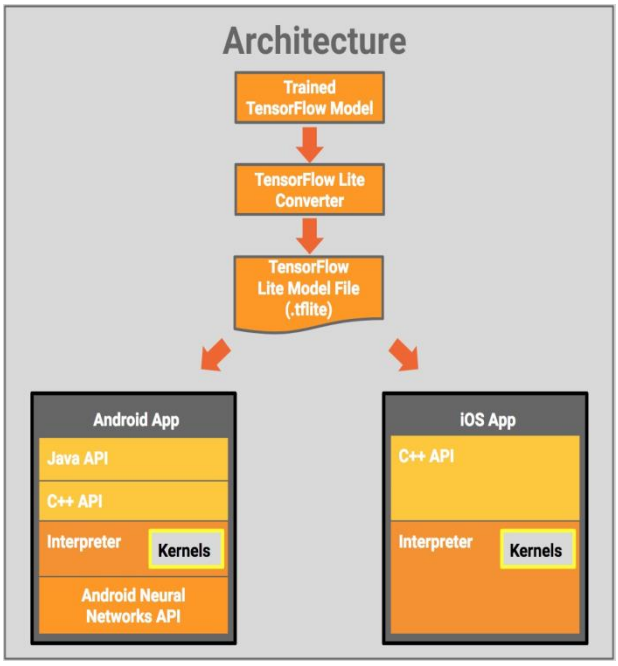


Figure4.1: Architecture of Tensorflow Lite.

3. Environment and development tools

3.1. Java: is an object-oriented programming language and computer platform, used to create software in very diverse environments, such as: web and mobile applications, distributed applications (EJB) [48].

3.2. Xml: In Android Studio, XML (eXtensible Markup Language) is primarily used for defining the layout of the user interface, resource values, and other configuration settings in android applications [49].

3.3. Android Studio: is the official integrated development environment (IDE) for Android apps. Based on IntelliJ IDEA's powerful code development and editing tool. It uses gradle as the basis of the compilation system. Furthermore, Java boasts a rich standard library, the java development kit (JDK), which provides an extensive collection of pre-built classes and packages for diverse programming tasks. [50]

For our project using android studio iguana version 2023.2.1, jdk-16 windows-x64,gradle version 8.4 sdk 34 and emelature (Pixel 3API 28) .

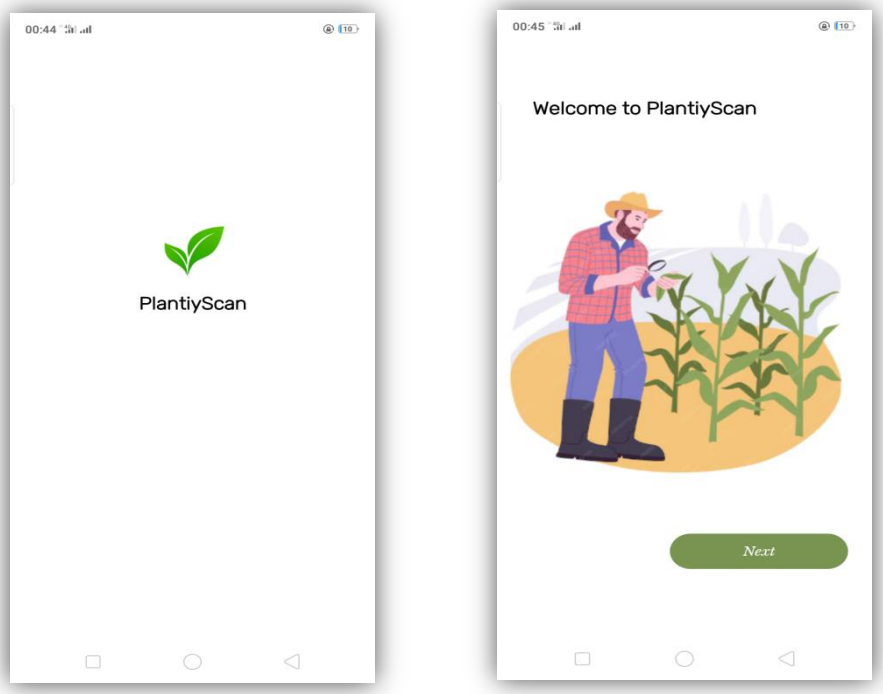
4. Application Overview

PlantityScan is Android application for plant disease detection that helps farmers and gardeners identify and manage plant diseases effectively, leading to healthier crops and improved agricultural productivity.

To develop the "PlantityScan" application for detecting plant diseases using Android Studio and Java, we'll outline the structure of the application based on the provided requirements. Implementing the application, including the five main interfaces: Welcome, Usage Tips, Plant Selection, Prediction, and Disease List. The application features.

1. **Icon screen:** A clean and simple background, application icon representing the type of image classification the app performs like leaf in our application with the app name position below the icon. as show in (Figure 4.2a)

2. **A welcome screen:** Click the "Next" button to go to the next page, as show in (Figure 4.2b).



(a) (b)

Figure4.2: (a) Icon screen - (b) A welcome screen.

4. **Page Home: overview application plantiyScan, Usage tips:** Short Description of application plantiyscan and briefly explain how to use it.

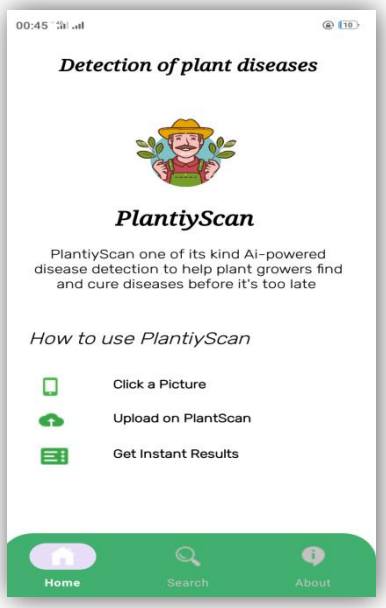


Figure4.3: Foverview application plantiyScan, Usage tips.

5. **Page Search: capture or Select an Image.**

- Use the in-app camera feature to take a photo of the affected plant.
- Alternatively, select an existing image from your device's photo gallery.
- Disease prediction: get the prediction (disease) with confidence.

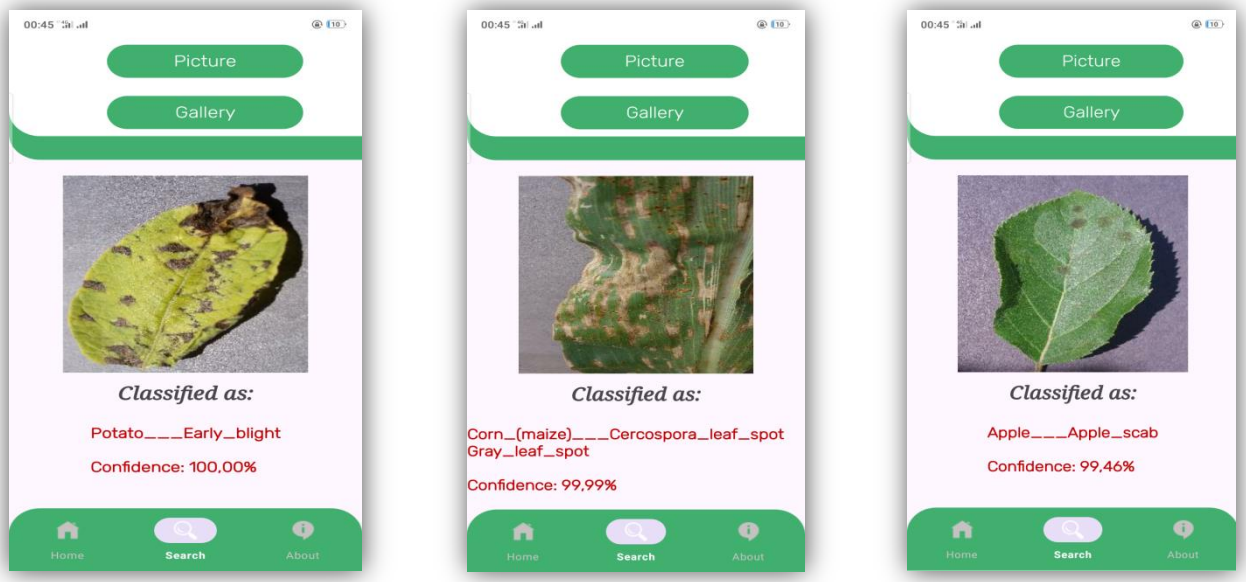


Figure4.4: Some examples prediction of plant diseases.

6. **Page about:** it shows list of available diseases.

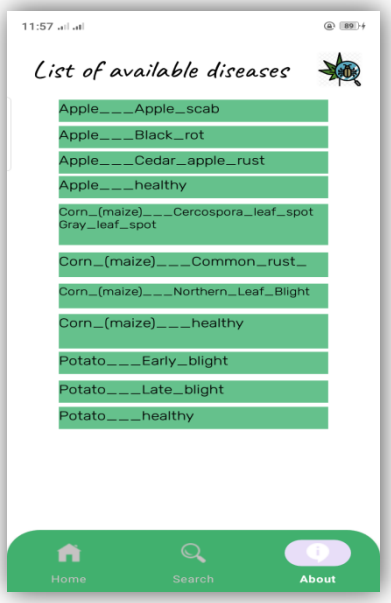


Figure 4.5: List of available diseases.

5. Conclusion

By leveraging TensorFlow Lite and the MobileViT-XXS model, PlantiyScan offers real-time disease detection on mobile devices. This empowers farmers and gardeners, regardless of their technical background, to diagnose plant issues efficiently. The application is designed for ease of use, allowing users to capture images, receive predictions, and gain insights into potential plant diseases without complex interactions. Its goes beyond the realm of machine learning model development by focusing on deployment. By making the application available to users, it has the potential to significantly impact agricultural practices. Early and accurate disease detection can lead to timely interventions, improved crop yields, and increased food security.

GENERAL CONCLUSION

The project on plant disease detection using vision transformers (ViTs), particularly the MobileViT-XXS model, shows significant potential in accurately identifying and classifying plant diseases. By combining the strengths of convolutional neural networks (CNNs) and ViTs, it aims to create a lightweight and low-latency network suitable for mobile vision tasks. This network can greatly aid farmers and researchers in the early detection and management of plant diseases, reducing economic losses and improving crop health.

While still in its early stages, the research demonstrates that further advancements in image classification through the fusion of CNNs and ViTs could yield even more effective solutions. This integration represents a significant advancement in agricultural technology, laying a solid foundation for future developments in accessible and innovative plant disease management.

In the future, this system could be integrated into mini-drones for real-time detection of plant diseases in cultivated areas. By using GPS for precise navigation and geotagging images, the drone can locate sick trees accurately. Equipped with high-resolution cameras and various sensors, the drone can capture detailed images and data of the crops from above.

Bibliography

Bibliography

[1] Das, S., Dye, A., Pal, A., & Roy, N. (2015). Applications of artificial intelligence in machine learning: review and prospect. *International Journal of Computer Applications*, 115(9). Volume 115 – No. 9, April 2015.

[2] Abu-Naser, S. S., & Abunasser, B. S. (2023). The Miracle Of Deep Learning In The Holy Quran. *Journal Of Theoretical And Applied Information Technology*, *101*, 17. 15th September 2023. Vol.101. No 17.

[3] What is Deep Learning? https://aws.amazon.com/what-is/deep-learning.

[4] https://www.turing.com/kb/ultimate-battle-between-deep-learning-and-machine-learning-

Deep Learning vs Machine Learning: The Ultimate Battle-Major Accessed:

21/02/2024.

[5]Types of Machine Learning- https://www.geeksforgeeks.org/types-of-machine-learning/ Accessed: 21/02/2024

[6] https://www.analyticsvidhya.com/blog/2021/09/introduction-to-artificial-neural-networks Accessed: 22/02/2024

[7] https://www.analyticsvidhya.com/blog/2021/03/basics-of-neural-network/- Deep Learning 101: Beginners Guide to Neural Network. Accessed: 22/02/2024

[8] Deep learning - Forward Propagation, Back Propagation, and Computational Graphs. https://ja.d2l.ai/chapter_deep-learning-basics/backprop.html Accessed: 23/02/2024

[9] Gradient Descent in Machine Learning: A Basic Introduction. WHAT IS DESCENT IN MACHINE LEARNING? https://builtin.com/data-science/gradient-descent Accessed: 23/02/2024

[10] Ruder, S. (2016). An overview of gradient descent optimization algorithems. arXiv preprint arXiv:1609.04747.

[11] Chandola, Y., Virmani, J., Bhadauria, H., Kumar, P. (2021). Deep Learning for Chest

Radiographs: Computer-Aided Classification. Netherlands: Elsevier Science.

[12] Brownlee, J. (2018). What is the Difference Between a Batch and an Epoch in a Neural Network. *Machine learning mastery*, *20*.

[13] Training vs Testing vs Validation Sets. https://www.geeksforgeeks.org/training-vs-testing-vs-validation-sets/ Accessed: 24/02/2024.

[14]. https://aws.amawon.com/What is /overfitting- What is overfitting? Overfitting in machine learning explained

[15] Uzair, M., & Jamil, N. (2020, November). Effects of hidden layers on the efficiency of neural networks. In *2020 IEEE 23rd international multitopic conference (INMIC)* (pp. 1-6). IEEE. DOI: 10.1109/INMIC50486.2020.931819

[16]What is Underfitting | DataRobot Artificial Intelligence. Underfitting What does Underfitting Mean? https://www.datarobot.com/wiki/underfitting/

[17] Importance of Loss Function in Machine Learning. towardsdatascience.com was first indexed by Google in October 2016 https://towardsdatascience.com/importance-of-loss-function-in-machine-learning Accessed: 26/02/2024

[18] https://www.geeksforgeeks.org/introduction-convolution-neural-network/

Accessed: 27/02/2024

[19] https://aws.amazon.com/what-is/recurrent-neural-network.Recurrent Neural Networks Explained . Accessed: 27/02/2024

[20] Power of Recurrent Neural Networks (RNN): Revolutionizing AI. What Are Recurrent Neural Networks (RNN)? https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn. Accessed: 27/02/2024

[21] Top 26 Applications of Deep Learning in 2024 Across Industries- https://www.knowledgehut.com/blog/data-science/deep-learning-applications Accessed: 28/02/2024

[22] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009, June). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp.248-255).Ieee**.** DOI :10.1109/CVPR.2009.5206848.

[23] Bačanin Džakula, N. (2019). Convolutional neural network layers and architectures. In Sinteza 2019-International Scientific Conference on Information.

[24] A Practical Guide to ReLU. https://medium.com/@danqing/a-practical-guide-to-relu /Accessed: 02/03/2024.

[25] Cost, Activation, Loss Function|| Neural Network|| Deep Learning. What are these? https://medium.com/@zeeshanmulla/cost-activation-loss-function-neural-network-deep-learning-what-are-these

[26]Ramachandran, P., Zoph, B., & Le, Q. V. (2017). Searching for activation functions. *arXiv preprint arXiv:1710.05941*.

[27] Shyam, R. (2021). Convolutional neural network and its architectures. *Journal of Computer Technology & Applications*, *12*(2), 6-14. DOI (Journal): 10.37591/JoCTA

[28] Ba, J., & Frey, B. (2013). Adaptive dropout for training deep neural networks. *Advances in neural information processing systems*, *26*. Volume 6, Issue 1 March 2018

[29] Jha, D., Yazidi, A., Riegler, M. A., Johansen, D., Johansen, H. D., & Halvorsen, P. Lightlayers: Parameter efficient dense and convolutional layers for image classification, 2021. URL https://arxiv.org/abs/2101.02268.

[30] Hussain, M., Bird, J. J., & Faria, D. R. (2019). A study on CNN transfer learning for image classification. In Advances in Computational Intelligence Systems: Contributions Presented at the 18th UK Workshop on Computational Intelligence, September 5-7, 2018, Nottingham, *UK* (pp. 191-202). Springer International Publishing.

[31] Ganakwar, P., & Date, M. S. (2020). Convolutional neural network-VGG16 for road extraction from remotely sensed images. *Int J Res Appl Sci Eng Technol*, *8*(8), 916-922.

[32] Zakaria, N., & Hassim, Y. M. M. (2024). A Review Study of the Visual Geometry Group Approaches for Image Classification. Journal of Applied Science, Technology and Computing, 1(1), 14-28. DOI:https://doi.org/10.30880/jastec.2024.01.01.003Article

[33] Targ, S., Almeida, D., & Lyman, K. (2016).Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029.

[34]Zhong, Z., Zheng, M., Mai, H., Zhao, J., & Liu, X. (2020, November). Cancer image classification based on DenseNet model. In Journal of physics: conference series (Vol. 1651, No. 1, p. 012143). IOP Publishing. DOI 10.1088/1742-6596/1651/1/012143.

[35] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[36] Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L. C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4510-4520).

[37] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., . & Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems, 30*.

[38] Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., ... & Houlsby, N. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*.

[39] Mehta, S., & Rastegari, M. (2021). Mobilevit: light-weight, general-purpose, and mobile-friendly vision transformer. *arXiv preprint arXiv:2110.02178*.

[39] Python https://www.python.org/

[40] TensorFlow https://www.tensorflow.org/

[41]Keras. https://keras.io/about/

[42]NumPy. https://numpy.org/

[43] Matplotlib: Visualization with Python .https://matplotlib.org/

[44] Scikit-learn. https://scikit-learn.org/stable/

[45] Google colab. https://colab.research.google.com/

[46] Noyan, M. A. (2022). Uncovering bias in the PlantVillage dataset. *arXiv preprint arXiv:2206.04374*.

[47] TensorFlow Lite https://www.tensorflow.org/lite/

[48] What is Java? https://aws.amazon.com/what-is/java/

[49] https://www.geeksforgeeks.org/- A Complete Guide to Learn XML For Android App Development.

[50] Meet Android Studio. https://developer.android.com/studio/.