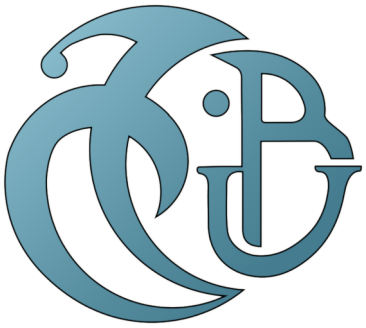
People's Democratic Republic of Algeria

Ministry of Higher Education and Scientific Research

University of Saad Dahleb - BLIDA 1

Faculty of Sciences



Department of Computer Science

Master Thesis Submitted in Partial Fulfillment of The Degree of Master of Science In Computer Science

Specialisation: Computer Systems & Networks

Deep Learning Model For Weeds Detection & Classification

Authors: Ilyes Hider
Idriss Fekir

Defended Publicly on Monday June 26, 2023

Thesis committee:	Dr. Célia Hireche	Jury President	Saad Dahleb University
	Dr. Fatma Zohra Zahra	Examiner	Saad Dahleb University
	Dr. Hayat Daoud	Thesis Advisor	Saad Dahleb University

In memory of Terry Davis, a schizophrenic who wrote an OS in his programming language compiled with his compiler, all from scratch because he thought god told him to.

Rest in Peace, Terry

Abstract

Hunger remains a persistent and serious global issue affecting millions of people worldwide. Despite advancements in agriculture and food distribution, chronic hunger and malnutrition continue to plague communities and nations.

The rise in global population further amplifies the challenge of feeding people adequately. Innovative solutions are being sought, and deep learning techniques offer promising avenues for addressing agricultural problems, particularly weed eradication, which poses a significant threat to crop production.

Smart farming, empowered by advanced technologies like artificial intelligence (AI), presents a more efficient, precise, and sustainable approach compared to traditional agriculture.

In this work, we present an initial effort towards a smart farming solution for weed eradication. Our approach applies transfer learning on DenseNet121 a Deep Convolutional Neural Network (CNN) pretrained on imagenet, trained on a dataset comprising images of eight weed species and various flora. The goal is to detect and classify weed images accurately, serving as a crucial first step towards developing robotic systems that can be deployed in agricultural fields.

By harnessing the power of deep learning, we aim to contribute to the development of effective and automated weed eradication strategies. Despite not contributing much, this research holds significant potential to alleviate the challenges posed by weeds in agriculture and advance the adoption of smart farming practices.

Keywords: Deep Learning, Convolutional Neural Network, Weeds Images, Images Classification.

Résumé

La faim reste un problème mondial persistant et grave qui touche des millions de personnes dans le monde. Malgré les progrès de l'agriculture et de la distribution alimentaire, la faim et la malnutrition chroniques continuent de tourmenter les communautés et les nations.

L'augmentation de la population mondiale amplifie encore le défi de nourrir les gens de manière adéquate. Des solutions innovantes sont recherchées et les techniques d'apprentissage en profondeur offrent des pistes prometteuses pour résoudre les problèmes agricoles, en particulier l'éradication des mauvaises herbes, qui constitue une menace importante pour la production agricole.

L'agriculture intelligente, renforcée par des technologies avancées telles que l'intelligence artificielle (IA), présente une approche plus efficace, précise et durable par rapport à l'agriculture traditionnelle.

Dans ce travail, nous présentons un premier effort vers une solution agricole intelligente pour l'éradication des mauvaises herbes. Notre approche utilise un réseau de neurones à convolution profonde (CNN) formé sur un ensemble de données comprenant des images de huit espèces de mauvaises herbes et de diverses flores. L'objectif est de détecter et de classer avec précision les images de mauvaises herbes, ce qui constitue une première étape cruciale vers le développement de systèmes robotiques pouvant être déployés dans les champs agricoles.

En exploitant la puissance de l'apprentissage en profondeur, nous visons à contribuer au développement de stratégies efficaces et automatisées d'éradication des mauvaises herbes. Bien qu'elle n'apporte pas grand-chose, cette recherche recèle un potentiel important pour atténuer les défis posés par les mauvaises herbes dans l'agriculture et faire progresser l'adoption de pratiques agricoles intelligentes.

Mots-clés: Apprentissage Profond, Réseau neuronal convolutif, Images des mauvaises herbes, Classification des images.

ملخّص

لا يزال الجوع مشكلة عالمية خطيرة ومستمرة تؤثر على ملايين الأشخاص في جميع أنحاء العالم. على الرغم من التقدم في الزراعة وتوزيع الغذاء ، لا يزال الجوع المزمن وسوء التغذية يصيبان المجتمعات والأمم.

يؤدي الارتفاع في عدد سكان العالم إلى تضخيم التحدي المتمثل في إطعام الناس بشكل كافٍ. يجري البحث عن حلول مبتكرة ، وتوفر تقنيات التعلم العميق طرقًا واعدة لمعالجة المشاكل الزراعية ، ولا سيما القضاء على الحشائش التي تشكل تهديدًا كبيرًا لإنتاج المحاصيل.

تقدم الزراعة الذكية ، المدعومة بتقنيات متقدمة مثل الذكاء الاصطناعي ، نهجًا أكثر كفاءة ودقة واستدامة مقارنة بالزراعة التقليدية.

في هذا العمل ، نقدم خطوة أولى نحو حل زراعي ذكي لاستئصال الحشائش. الحل الذي نقدمه هو شبكة عصبية تلافيفية عميقة مدربة على مجموعة بيانات تضم صورًا لثمانية أنواع من الأعشاب ونباتات مختلفة. الهدف هو اكتشاف وتصنيف صور الحشائش بدقة ، وهي بمثابة خطوة أولى حاسمة نحو تطوير أنظمة روبوتية يمكن نشرها في الحقول الزراعية.

من خلال تسخير قوة التعلم العميق ، نهدف إلى المساهمة في تطوير استراتيجيات فعالة وآلية لاستئصال الحشائش. على الرغم من عدم مساهمة هذا البحث كثيرًا ، إلا أن هذا البحث يحمل إمكانات كبيرة للتخفيف من التحديات التي تشكلها الأعشاب الضارة في الزراعة وتعزيز تبني ممارسات الزراعة الذكية.

الكلمات المفتاحيّة : التّعلّم العميق، شبكة عصبيّة تلافيفيّة، صور أعشاب ضارّة، تصنيف الصّور.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
ML	Machine Learning
DL	Deep Learning
CNN	Convolutional Neural Network
SVM	Support Vector Machine
GAN	Generative Adversarial Network
GCN	Graph Convolutional Network
ViT	Visual Transformers
ReLu	Rectified Linear Unit
MANET	Mobile Ad Hoc Network

Introduction

"Whatever you do in life will be insignificant, but it's very important that you do it "
—Mahatma Ghandi, Lawyer, Anti-Colonial Nationalist, and Philosopher

Hunger remains a grave and persistent issue worldwide, posing significant dangers to individuals, communities, and entire nations. Despite advances in agriculture, food production, and distribution, millions of people continue to suffer from chronic hunger and malnutrition.

"Not only are more people in more places around the world going hungry, but the severity of the hunger they face is worse than ever" said *Cindy McCain, WFP Executive Director* [1]

The rise in global population exacerbates the issue of hunger by placing increased pressure on food resources and agricultural systems, Recently the world has surpassed over 8 billion people [2].

Smart farming, enabled by advanced technologies such as IoT, AI, offers improved precision, efficiency, decision-making capabilities, and sustainability, making it a much better and much more effective approach than traditional agriculture.

In this work we make an attempt towards a core component for a smart farming solution to eradicate weeds, through deep learning.
We use apply transfer learning on a Deep Convolutional Neural Network with a dataset comprised of images of 8 weed species and various flora, in order to detect and classify images of weed, this is the first step towards building robotic solutions to be deployed in real fields.

The work is comprised of 7 chapters (Introduction and Conclusion included), the first chapter being this one, chapter I introduces the genral concept of Artifical Intellegence and goes over a brief history, chapter II goes over the state of the art in computer vision with deep learning techniques, specifically with CNNs, and goes over some recent works on weed detection and classification, chapter III describes the employed model, chapter IV goes over an evaluation of the model, and several related models. chapter V goes over some future prospects to move this work forward, and all results are summarized in chapter V.6 (the conclusion).

I

Artificial Intelligence, Machine Learning and Deep Learning

"The development of AI is as fundamental as the creation of the microprocessor, the personal computer, the Internet, and the mobile phone"

—Bill Gate, Founder of Microsoft

Artificial Intelligence is an interdisciplinary field of study concerned with creating *rational* machines. The term *Artificial Intelligence* was coined by *John McCarthy* in *1956*.

Rationality is loosely speaking, *doing the right thing*. More formally, given an *objective* and a set of *rules* a *rational agent* acts to achieve the best outcome (that is attain the objective), and under *uncertainty* the *best possible outcome*.

I.1. A Brief History of Artifical Intelligence

I.1.1. Origins

The origins of AI can be found in ancient philosophical ideas about artificial beings and automata. Thinkers such as *Aristotle* and *Al-Jazari* contemplated the concept of artificial life, sowing the seeds for future exploration. Advancements in the 17th to 19th centuries, including *Blaise Pascal's* mechanical calculator and *Ada Lovelace's* pioneering work on algorithms, set the stage for the emergence of AI as a distinct field [3].

I.1.2. Significant Milestones

1943 Model of an artificial neuron, The first work which is now recognized as AI was done by *Warren McCulloch* and *Walter pitts* in 1943.

1949 Donald Hebb demonstrated an *updating rule* for modifying the connection strength between neurons, termed *Hebbian learning*.

1950 Alan Turing publishes *Computing Machinery and Intelligence* in which he proposed a test. The test can check the machine's ability to exhibit intelligent behaviour equivalent to human intelligence, now known as *The Turing Test*.

1966 Researchers emphasized developing algorithms which can solve mathematical problems. *Joseph Weizenbaum* created the first chatbot in 1966, named *ELIZA*.

1972 WABOT-1, developed by researchers at Waseda University in Japan, one of the earliest humanoid robots. Standing at 1.65 meters tall, it possessed the ability to walk, grasp objects, and communicate using a limited vocabulary, pioneering the integration of mechanical and cognitive capabilities in robotics and laying the groundwork for future advancements in humanoid robot design.

1974-1980 The first AI winter, a period characterized by dwindling funding and waning interest in artificial intelligence due to unfulfilled promises and unrealistic expectations, resulting in a decline in research and development efforts in the field.

1980 AAAI, the first national conference of the American Association of Artificial Intelligence marked a pivotal moment for the field, bringing together researchers, academics, and industry professionals to exchange ideas, present breakthroughs, and establish a collaborative platform that fuelled advancements in AI for years to come.One notable advancement was the development of *expert systems* (systems which utilized knowledge-based rules and reasoning to simulate human expertise in specific domains).

1987-1993 The second AI winter, characterized by a decline in funding and interest in artificial intelligence due to overhyped expectations, lack of practical applications, and failure to deliver on promised breakthroughs, leading to a reduction in AI research and commercial activity until a resurgence in the late 1990s.

1993-2011 Significant advancements, including the emergence of machine learning algorithms like support vector machines and neural networks, the development of natural language processing techniques, the rise of intelligent systems in various domains such as finance and healthcare, and the birth of practical applications like recommendation systems, speech recognition, and computer vision.

1997 IBM's Deep Blue, a supercomputer, defeated the reigning world chess champion *Garry Kasparov*, showcasing the potential of AI in complex decision-making tasks and marking a significant milestone.

1998 LeNet, developed by Yann LeCun in the 1990s, was one of the pioneering convolutional neural networks *(CNNs)* for image recognition tasks. It consisted of multiple layers of convolutional and pooling operations, followed by fully connected layers, and laid the foundation for modern deep learning in computer vision, leading to advancements in areas such as object recognition, handwriting recognition, and other image classification tasks [4].

2002 Roomba, an autonomous robotic vacuum cleaner by *iRobot*. It uses sensors and algorithms to navigate and clean floors effectively, offering a convenient and hands-free solution for household cleaning tasks. Roomba's success has contributed to the growing adoption of robotic technologies in the home automation industry.

2011 IBM's Watson, an AI-powered supercomputer system, gained international attention by winning the quiz show *Jeopardy!*. Watson's victory was attributed to its ability to process and understand natural language, analyse vast amounts of data, and generate accurate answers quickly, showcasing the potential of AI in surpassing human performance in knowledge-intensive tasks.

2012 Google Now, a virtual assistant by Google, aimed to provide personalized information to users through *predictive analytics*. By analysing user data such as search history, location, and calendar events. It offered proactive notifications and recommendations for weather updates, traffic conditions, upcoming appointments, and more, enhancing the user's overall digital experience with relevant and timely information, showcasing the potential of machine learning algorithms in analysing user data and providing personalized experiences.

2018 IBM's Project Debater an AI system developed to engage in persuasive debates with humans. Using natural language processing, machine learning, and knowledge retrieval techniques made headlines by engaging in a live debate with human debaters. It showcased its ability to understand and respond to arguments on a given topic, presenting coherent and persuasive points.

2023 GPT-4, is released, a *generative pre-trained transformer*, can comprehend and generate human-like text across various applications, including language translation, content creation, chatbots, and more, revolutionizing the field of natural language processing and demonstrating the potential of large-scale language models, and it can understand code, and accepts images as input [5].

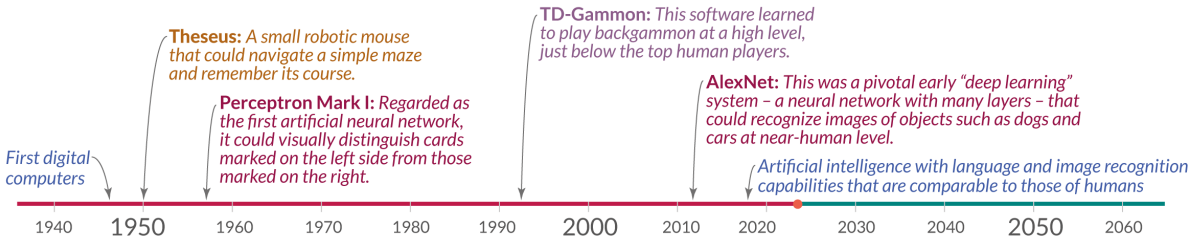


Figure I.1: A timeline of notable AI systems, source:[6]

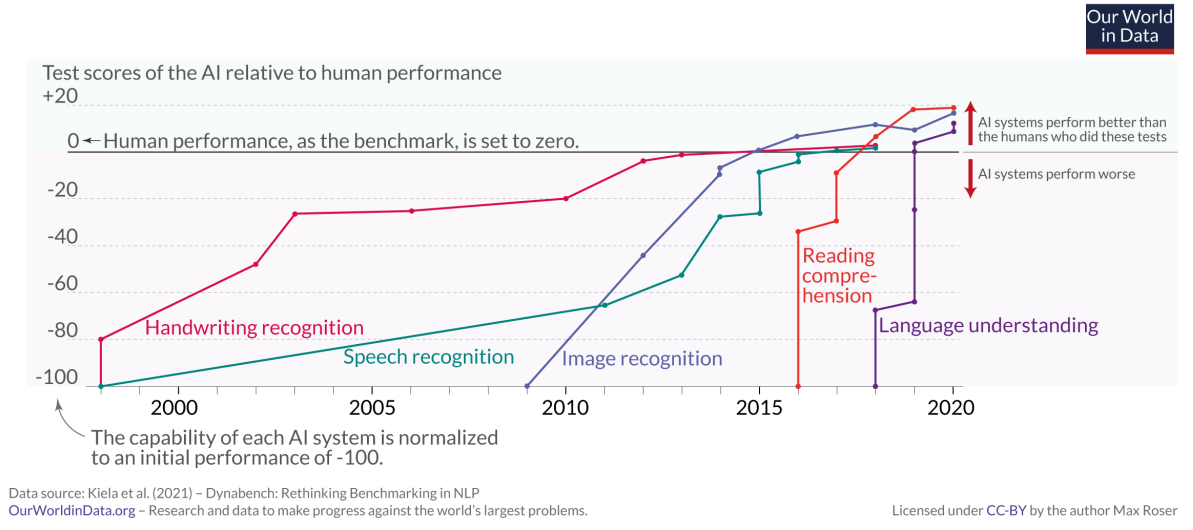


Figure I.2: Language and image recognition systems over the years, source:[6]

As I.1 clearly shows, the AI systems are improving at an almost exponential rate, especially in the last 2 decades. Surprisingly, in some areas even human performance is surpassed as shown in I.2

I.1.3. Classical Artificial Intelligence Approches

Before the advent of machine learning, classical approaches to artificial intelligence (AI) focused on *rule-based systems* and *symbolic reasoning*.

These approaches aimed to explicitly encode human knowledge and reasoning rules into computer programs to simulate intelligent behaviour.

Common techniques included expert systems, knowledge representation and reasoning, natural language processing, and search algorithms such as depth-first search and breadth-first search.[7]

While these classical AI approaches showed promise in solving specific problems, they often struggled with handling complex and uncertain real-world data, leading to the emergence and dominance of machine learning-based approaches in recent years.

I.1.4. Increase of The Need For More Computational Power

The computational requirements of machine learning have increased dramatically across three distinct eras. The handcrafted features era, the era of classic machine learning, and the deep learning era. The field has seen significant increases in computational demands as machine learning models transitioned from manually engineered features to more data-driven and complex deep learning architectures.[8]

I.2. Machine Learning

machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and models that allow computers to learn and make predictions or decisions without being explicitly programmed. The emphasis is on data to enable machines to automatically improve their performance on a specific task or problem[7].

- ***Task*** The specific problem or task that the machine is aiming to solve or learn from. This can range from simple classification tasks, such as identifying spam emails, to more complex tasks like speech recognition or autonomous driving.

- ***Experience*** The data or examples that the machine uses to learn and improve its performance on the given task. Experience can come in the form of labelled data, where each example is already associated with the correct output, or unlabelled data, where the machine must infer patterns and relationships by itself.

- ***Performance Measure*** A measure or metric that quantifies how well the machine is performing on the task. This can be accuracy, error rate, precision, recall, or any other relevant measure depending on the specific task.

I.2.1. Machine Learning Approaches

- ***Unsupervised Learning*** refers to learning from unlabelled data without explicit output labels. Clustering algorithms, such as k-means and hierarchical clustering, aim to group similar instances together. Dimensionality reduction techniques, like Principal Component Analysis (PCA), aim to capture the most important features or reduce the dimensionality of the data.

- ***Supervised Learning*** refers to the process of training a machine learning model using labelled data, where each input instance is associated with the correct output or target value.

- ***Decision Tree Learning*** is a popular machine learning approach that builds a tree-like model of decisions and their possible consequences. It uses a top-down recursive strategy to partition the data based on feature values, aiming to maximize information gain or minimize impurity measures. Decision trees are interpretable and can handle both categorical and continuous features.

- ***Instance-Based Learning*** also known as lazy learning, involves storing and generalizing from specific training instances. Instead of constructing a general model, instance-based learning directly compares new instances to stored instances to make predictions. Common algorithms in this category include k-nearest neighbours (k-NN) and case-based reasoning.

- ***Bayesian Learning*** involves the application of Bayesian inference to machine learning problems. It uses probability theory and Bayes' theorem to update beliefs about hypotheses based on new evidence. Bayesian learning can handle uncertain or incomplete data and provides a principled way to combine prior knowledge and observed data.

- ***Artificial Neural Networks*** Neural networks are composed of interconnected nodes (neurons) organized in layers. They are capable of learning complex patterns and relationships from data. Two common types are feedforward neural networks and recurrent neural networks including single-layer perceptrons and multi-layer perceptrons (MLPs). Training neural networks often involves *backpropagation*, which adjusts the weights based on the error *gradient* to minimize the *loss function*.

- ***Support Vector Machines*** (SVMs) are a class of supervised learning algorithms used for classification and regression tasks. SVMs find a hyperplane that separates data points of different classes with the maximum margin. They can handle high-dimensional data and have effective generalization capabilities.

- ***Reinforcement Learning*** involves an agent learning to interact with an environment to maximize rewards or minimize penalties. The agent learns through trial and error, receiving feedback in the form of rewards or punishments. Q-learning and policy gradient methods are commonly used in reinforcement learning.

These are some of the machine learning approaches discussed in [7], refer to it for a much more comprehensive and thorough coverage of the field and its various methodologies.

I.3. Deep Learning

Deep learning is a subfield of machine learning that focuses on designing and training artificial neural networks with multiple layers. These deep architectures enable the learning of hierarchical representations from raw data, allowing the models to automatically discover complex patterns and features. Deep learning models excel at tasks such as image and speech recognition, natural language processing, and other problems involving large-scale data and high-dimensional inputs. By leveraging the power of deep neural networks, deep learning has achieved remarkable breakthroughs in various domains, pushing the boundaries of AI research and applications.[9]

I.3.1. The Rise of Deep Learning

Traditional machine learning algorithms struggled to achieve high performance because of limited and handcrafted features. Data-driven approaches, particularly with the availability of large-scale datasets, enabled the success of deep learning. Methods, such as convolutional neural networks, overcome the limitations of manually engineered features by learning hierarchical representations directly from data.[10]

The combination of GPUs (Graphics Processing Units) and modern machine learning frameworks has played a pivotal role in making deep learning feasible and accessible to non-computer scientists.

GPUs, with their parallel processing capabilities, greatly speed up deep learning computations. By harnessing the massive parallelism of GPUs, deep learning models can process large amounts of data efficiently, reducing training time from weeks to hours or even minutes.

Modern machine learning frameworks, such as TensorFlow, PyTorch, and Keras, provide high-level abstractions and intuitive APIs that abstract away the complexities of deep learning. These frameworks enable non-computer scientists to build, train, and deploy deep learning models without extensive programming expertise.

Cloud-based platforms and services, such as Amazon Web Services (AWS) and Google Cloud Platform (GCP), offer accessible and scalable infrastructure for deep learning. This eliminates the need for expensive hardware investments, allowing non-computer scientists to leverage powerful GPU resources on-demand.

I.3.2. Key Breakthroughs In Deep Learning

- ***AlexNet (2012)***, by *Alex Krizhevsky et al.* [11], achieved groundbreaking results in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC). It demonstrated the power of deep convolutional neural networks (CNNs) by significantly outperforming traditional computer vision methods.

- ***ResNet (2015), Residual Networks***, by *Kaiming He et al.* [12], addressed the problem of training very deep neural networks. By using skip connections that bypassed layers, they enabled the successful training of networks with hundreds of layers, leading to improved accuracy and performance.

- ***Generative Adversarial Networks (GANs) (2014)***, by *Ian Goodfellow et al.* [13], are a class of neural networks that can generate realistic data samples. They consist of a generator network and a discriminator network that compete against each other. They have been successfully applied to tasks such as image generation, image-to-image translation, and text generation.

- ***DeepMind's AlphaGo (2016)***, a program based on deep neural networks, made significant advancements in the field of reinforcement learning. It defeated the world champion Go player, *Lee Sedol*, demonstrating the power of deep learning and reinforcement learning in complex strategy games [14].

- ***The Transformer architecture (2017)***, by *Vaswani et al.* [15], revolutionized natural language processing by replacing recurrent neural networks with *self-attention mechanisms*. They achieved state-of-the-art performance in machine translation and language generation.

- ***Perfusion (2023)***, by *Tewel et al.* [16], is a text-to-image personalization method. It can portray personalized objects. It allows significant changes in their appearance, while maintaining their identity, using a novel mechanism called *Key-Locking*.

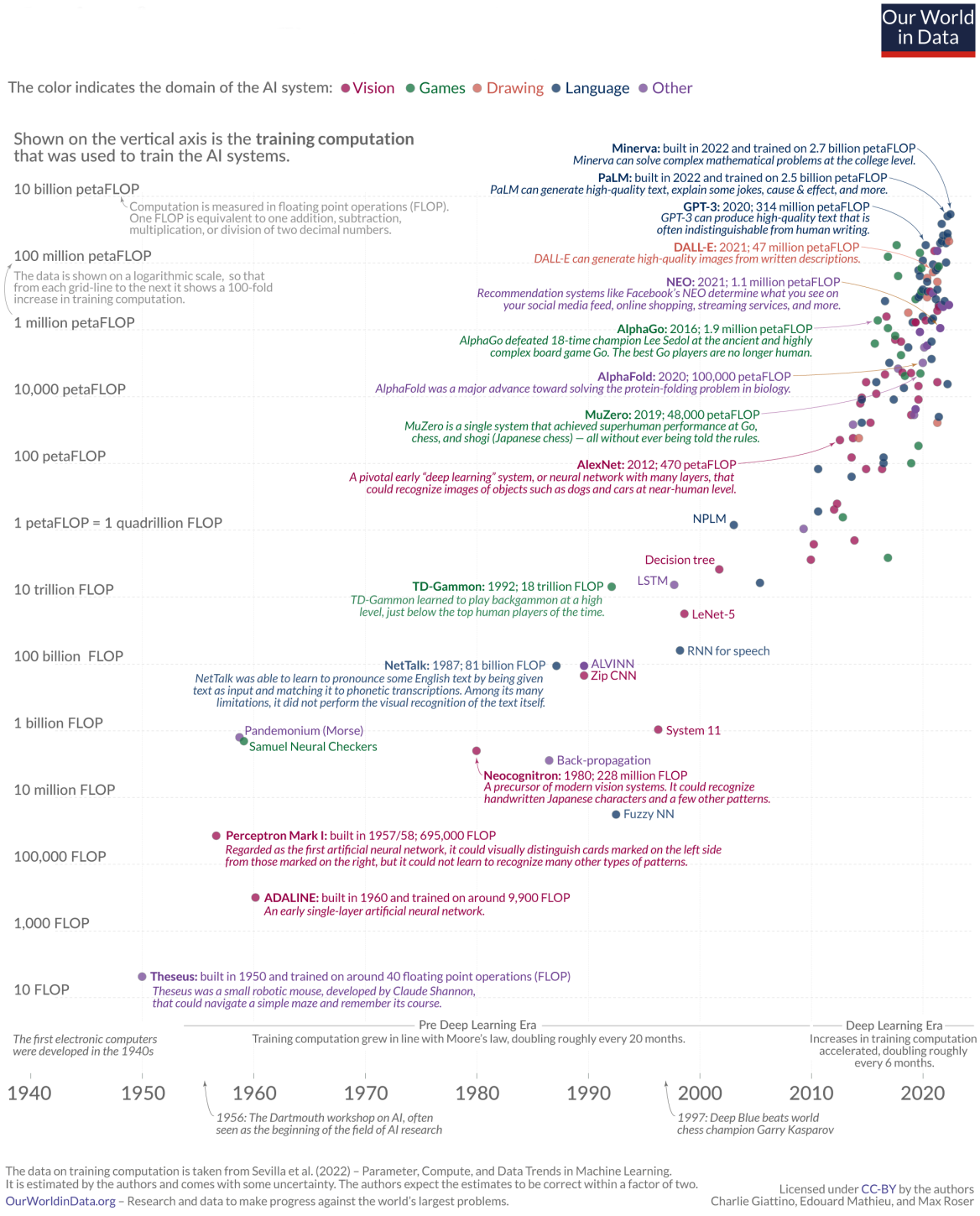


Figure I.3: Rise of computational power (thanks to Moore's law) and its effects on deep learning, source:[6]

I.3 shows that the greatest leaps happened in the last decade due to great hardware improvments.

II

The State of the Art On Image Classification Using Deep Learning

"Every great advance in science and technology has arisen from a deep desire to improve the current state of the art"
—Mae Jemison, Engineer, Physician, and Former Astronaut

II.1. Background

Before deep learning, the most commonly used techniques for image classification were based on handcrafted features and traditional machine learning algorithms. These techniques typically involved extracting features from images using techniques such as edge detection, texture analysis, and color histograms, and then using machine learning algorithms such as support vector machines (SVMs), decision trees, and random forests to classify the images into different categories.

One popular approach was the *Bag of Visual Words (BoVW)* model [17], which involved clustering image features into visual words and then using these words to represent the image. Another approach was the *Scale-Invariant Feature Transform (SIFT)* [18], which detected and described local features in images and matched them across different images for recognition.

These techniques were effective for simple classification tasks but often struggled with more complex and varied images, requiring extensive feature engineering and domain-specific knowledge. The emergence of deep learning has revolutionized image classification by enabling the automatic learning of features directly from raw image data, eliminating the need for extensive feature engineering.

One approach for weed detection was based on handcrafted features, such as color, shape, texture, and size, extracted from the images of plants. These features were used to train classifiers such as *SVM* [19], decision trees [20], and neural networks to distinguish between crops and weeds. However, this approach required a lot of domain-specific knowledge and manual feature engineering.

Another approach was based on segmentation techniques that separated the plants from the background, followed by classification of the segmented regions using features such as color and texture. This approach was effective when the weeds were distinct from the crops, but it was challenging when there was overlap between the two.[21]
In comparison to the conventional computer vision approach in early image processing around two

decades ago, deep learning does not need expertise in particular machine vision areas or special domain knowledge to create handcrafted features.

II.2. Computer Vision In The Era of Deep Learning

Computer vision has undergone a significant transformation in the era of deep learning, with advancements in neural networks and deep learning algorithms revolutionizing the field. Deep learning models have demonstrated remarkable capabilities in various computer vision tasks, such as image classification, object detection, semantic segmentation, and image generation.

II.2.1. Convolutional Neural Networks

Convolutional Neural Networks, are a type of deep learning model designed for analysing visual data. They excel in tasks such as image classification and object detection. CNNs employ specialized layers like convolutional and pooling layers, which extract features and reduce spatial dimensions. These networks leverage the spatial relationships present in images by applying convolutional filters to capture local patterns. Through multiple layers of convolution and pooling, CNNs learn hierarchical representations, enabling them to recognize complex patterns and make accurate predictions. Refer to [9] by *Goodfellow et al.* for a more comprehensive coverage.

CNNs learn feature representations directly from the raw data through a series of learnable convolutional filters. These filters, also known as kernels, scan the input data to detect local patterns such as edges, corners, and textures. By stacking multiple convolutional layers, CNNs can progressively learn complex and abstract features that capture hierarchical representations of the data, thereby eliminating the need for manual feature engineering.

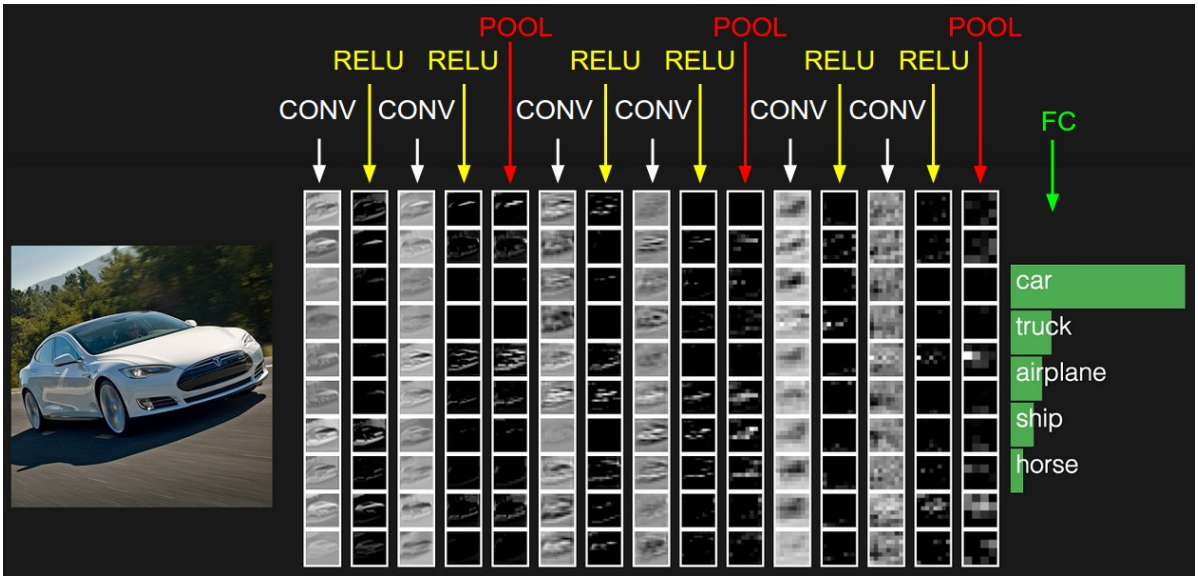


Figure II.1: How a CNN predicts the class of an input image (said differently, what does the cnn perceive), source:[22]

The General Structure of A CNN

- ***Input Layer*** receives the raw input data, usually in the form of images or other multidimensional data. It acts as the entry point for the network and preserves the spatial structure of the input. The input layer's dimensions correspond to the dimensions of the input data, such as the height, width, and depth (number of channels) of an image.

- ***Convolutional Layers*** are responsible for extracting meaningful features from the input data. They apply convolutional filters to scan the input data and perform element-wise multiplications

and summations to produce feature maps. These layers help capture spatial hierarchies and local patterns, enabling the network to learn and recognize complex features in the data, as shown in conv layers in II.1.

- ***Activation Functions*** Activation functions in a CNN introduce non-linearities to the network, allowing it to model complex relationships between inputs and outputs. Common activation functions in CNNs include *ReLU (Rectified Linear Unit)*, which introduces sparsity (that is, when the input to ReLU is negative, it outputs zero, effectively sparsifying the activation values. This can be beneficial in reducing the computational load and improving the efficiency of the network by eliminating unnecessary calculations) and non-linearity, and softmax, which normalizes outputs into probabilities for multi-class classification tasks. In II.1 after each ReLU layer some unnecessary features from the input are omitted.

- ***Pooling Layers*** are used to reduce the spatial dimensions of the feature maps while preserving important information. They achieve this by aggregating local regions of the feature maps, such as taking the maximum (max pooling) or average (average pooling) value within each region. They help to downsample the feature maps, making the network more computationally efficient and providing a form of spatial invariance, allowing the network to focus on the most salient features.

- ***Fully Connected Layers*** also known as dense layers, connect every neuron in the current layer to every neuron in the subsequent layer. These layers help in capturing high-level abstractions and making predictions based on the extracted features from earlier layers. Fully connected layers are typically employed towards the end of a CNN architecture to map the learned features to the desired output, such as class probabilities in classification tasks.

- ***Output Layer*** The output layer in CNNs is the final layer that produces the desired output based on the learned representations from previous layers. It typically consists of one or more neurons, depending on the task, such as a single neuron for binary classification or multiple neurons for multi-class classification. The activation function used in the output layer depends on the task, such as *sigmoid* for binary classification or *softmax* for multi-class classification, to produce the final output predictions.

II.2.2. The AlexNet Architecture

One of the fundamental breakthroughs in computer vision with deep learning was the development of Convolutional Neural Networks (CNNs). CNNs have shown outstanding performance in image classification tasks, surpassing traditional methods by a large margin.

The seminal paper by *Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton* [11] introduced the AlexNet architecture, which won the *ImageNet Large Scale Visual Recognition Challenge* in 2012 and paved the way for the deep learning revolution in computer vision.

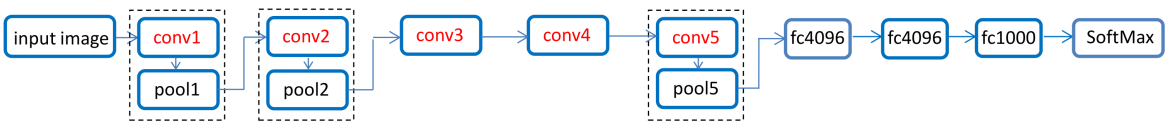


Figure II.2: The Architecture of AlexNet, source:[23]

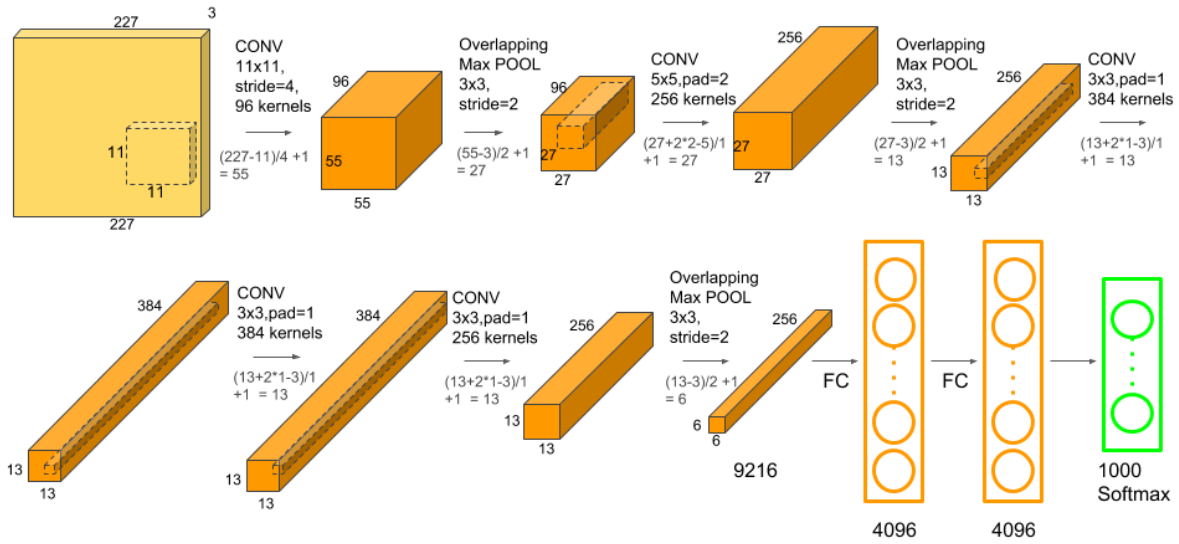


Figure II.3: A More Detailed view of The Architecture of AlexNet, source:[24]

II.2.3. The Inception Architecture

The GoogLeNet architecture (also known as Inception-v1), proposed by *Christian Szegedy et al.* in [25], introduced the concept of inception modules and showed improved performance with reduced computational complexity. The original Inception is 22 layers deep, with 27 pooling layers included. There are 9 inception modules stacked linearly in total. The ends of the inception modules are connected to the global average pooling layer. After that, came Inception-V2 [26], Inception-V3 [27] in 2016.

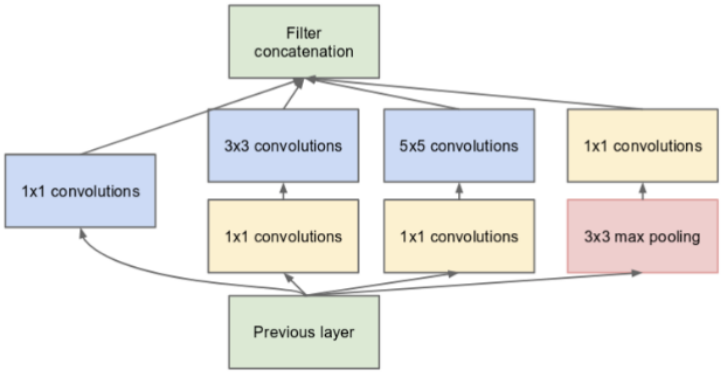


Figure II.4: The Improved Inception Module, source:[24]

II.2.4. The VGGNet Architecture

Following the success of AlexNet, deeper and more sophisticated CNN architectures were introduced. For instance, the Visual Geometry Group Network (VGGNet) architecture presented in [28] by *Karen*

Simonyan and Andrew Zisserman at the Visual Geometry Group at the University of Oxford, demonstrated the benefits of increasing network depth.

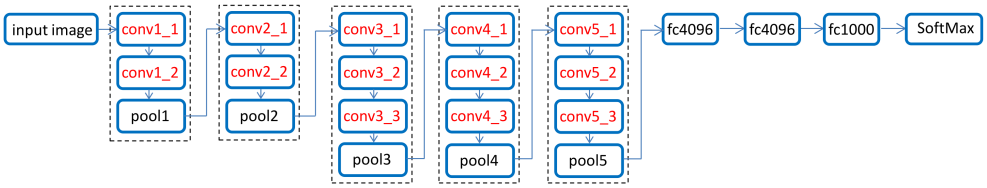


Figure II.5: The Architecture of VggNet16, source:[28]

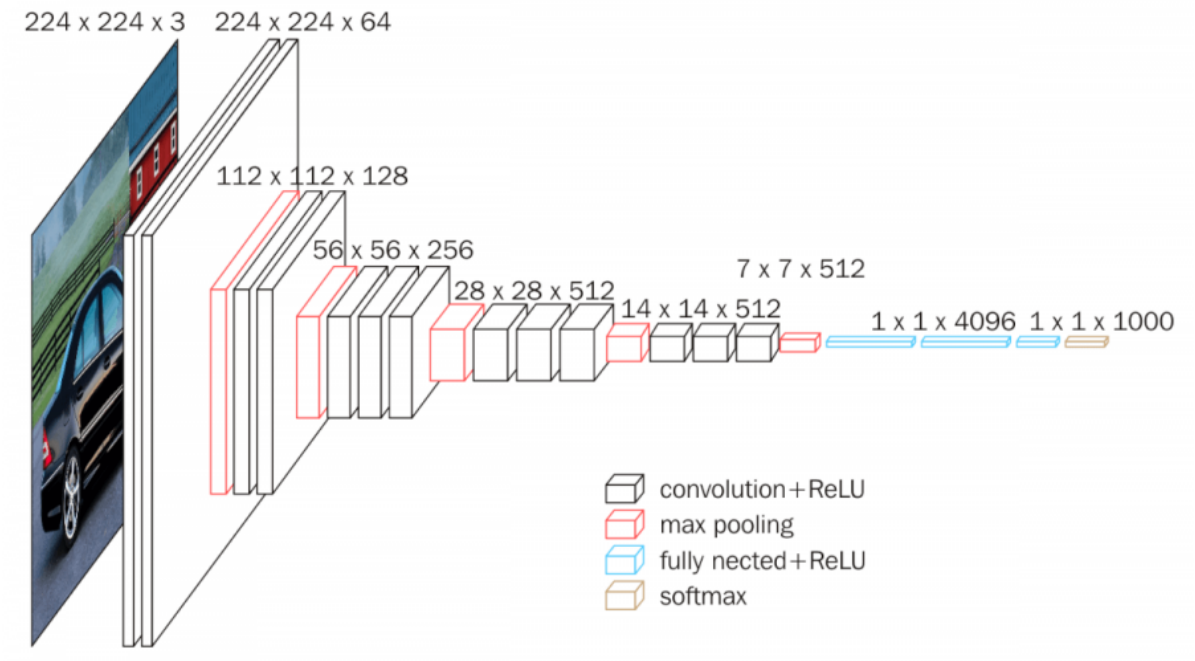


Figure II.6: A More Detailed view of The Architecture of VggNet16, source:[24]

II.2.5. The ResNet Architecture

The introduction of residual connections in the ResNet architecture, as described by *Kaiming He et al.* in [12], allowed for training even deeper networks and achieved remarkable accuracy on challenging datasets.

ResNet aimed at solving *the degradation problem*, when training deep networks there comes a point where an increase in depth causes accuracy to saturate, then degrade rapidly. It uses a *residual mapping*, the process of learning the residual (difference) between the desired output and the current input, allowing the network to focus on the residual learning and ease the training process by using shortcut connections in the network.

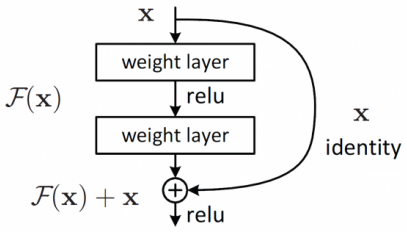


Figure II.7: The Building block of a Residual network, source:[24]

II.2.6. The DenseNet Architecture

Another influential architecture is DenseNet, presented by *Gao Huang et al.* in [29], which emphasized the dense connectivity pattern among layers, that is, that is, the layers in the network receive feature maps from all the preceding layers which improves feature reuse and gradient flow.

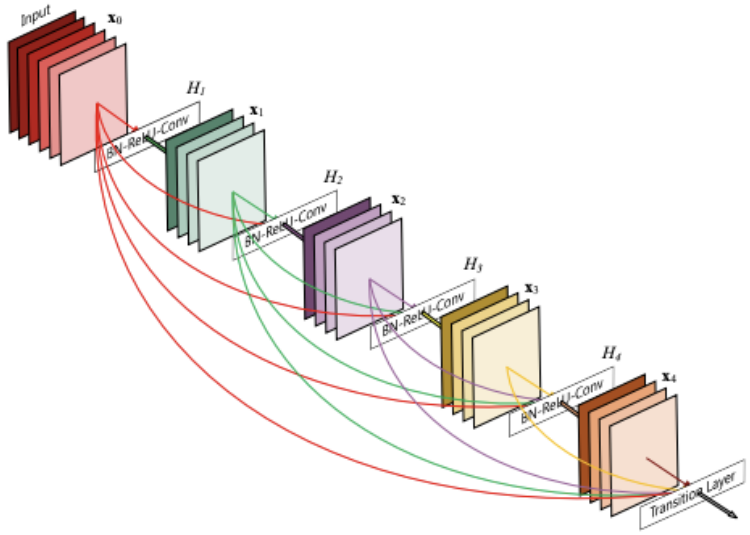


Figure II.8: Dense Connectivity, source:[24]

II.2.7. The MobileNet Architecture

MobileNet is a lightweight and efficient CNN architecture introduced by *Andrew G. Howard et al.* [30] designed for mobile and embedded devices. It utilizes depth-wise separable convolutions, which reduce computational complexity while maintaining accuracy. MobileNet achieves a good trade-off between model size and performance, making it suitable for real-time applications on resource-constrained devices.
MobileNetV2 improved upon the original MobileNet architecture by introducing inverted residual blocks with linear bottleneck layers and linear shortcuts. [31]

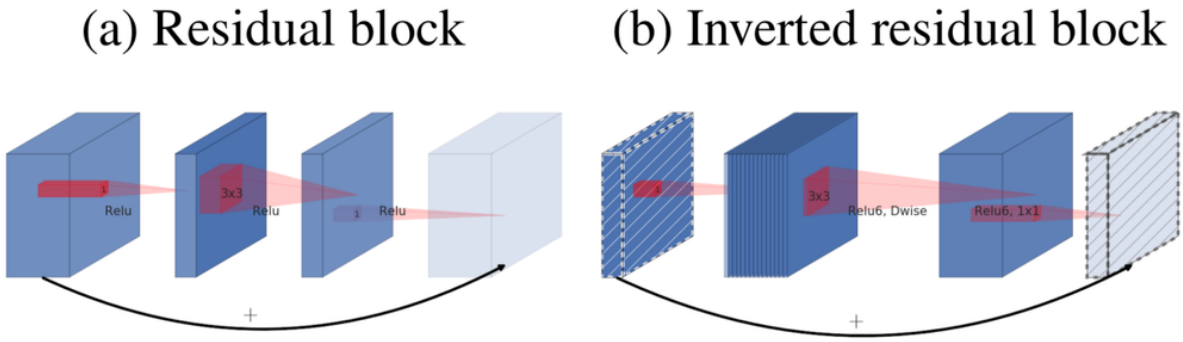


Figure II.9: Inverted Residual Block, source:[24]

MobileNetV3 further improved upon V2 with *Efficient architecture search* and several other techniques [32].

II.2.8. Semantic Segmentation

For semantic segmentation, Fully Convolutional Networks (FCNs) have become a popular choice. Introduced by *Jonathan Long et al.* [33] as a method to perform pixel-wise segmentation by replacing fully connected layers with convolutional layers.

Figure II.10: YOLOv8 [34] (introduced in 2023) by *Ultralytics .Inc* is a lightweight model for segmentation used in many robotics applications and driverless cars, performing segmentation on live cctv footage, source:Ultralytics

II.3. Weeds Detection With Deep Learning

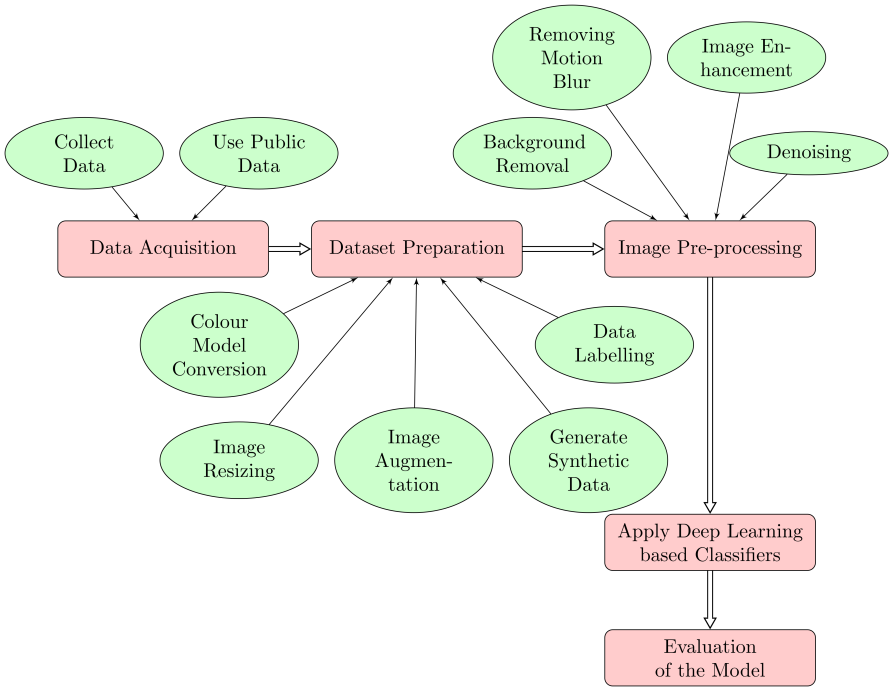


Figure II.11: General data collection and preprocessing workflow of utmost important in Machine Learning, source:[35]

The above figure shows the pipeline from data collection to application and evaluation of a deep learning model.

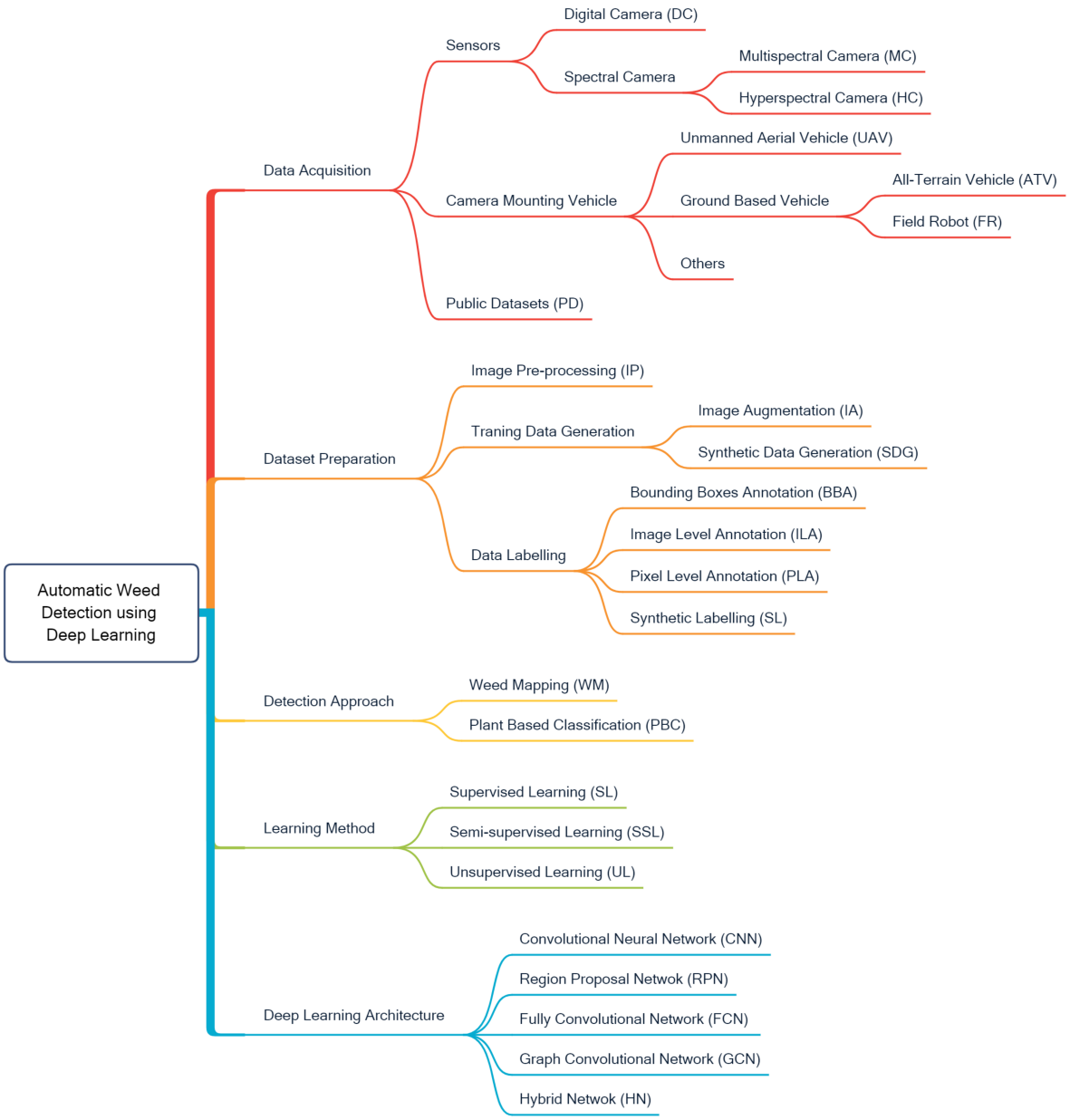


Figure II.12: General view on the methods and techniques used for data collection and model training for weeds detection and classification, source:[35]

The above figure lists most technique for data collection and model training in the context of weeds detection and classification.

We picked the related works from the survey by *Hasan, ASM Mahmudul, et al.* [35] most of the weeds datasets are not very big nor diverse, and are mostly site/weed/crop-specific. There is no Agreed upon benchmark like the ImageNet dataset.

An Attempt to collect and standardise weeds datasets is being carried by *Precision Weed Control Group* at the University of Sydney [36], with the aim of standardizing, structuring, and offering user-friendly search and the convenience of storing datasets together in one place. At the time of writing, they collected 20 datasets.

Here, we Consider only works which used a dataset of RGB images (as opposed to multi-spectral images, which require special insturment to capture, for the interested, *I. Sa et al. (2018)* [37] is an example), and only under-canopy plants (as opposed to e.g. Trees).

Models

- ***Espejo-Garcia et al. (2020)*** [38], Used a dataset They created, and made available [39], Their they experimented with several models all of which have a CNN as the feature extractor, then used traditional ML methods as classifiers (e.g. SVM). They reported a micro F1 score of 99.29%.

- ***Umamaheswari and Jain (2020)*** [40], Used a dataset of Carrot images, the weed(s) were not mentioned, their model is an *Encoder-Decoder* architecture based on VGG16 (also known as *SegNet*). They concluded that SegNet-512 is better than SegNet-256 for this task.

- ***Yan et al. (2020)*** [41], Used a dataset comprised of images of Paddy, along with 6 types of weeds (Alternanthera philoxeroides, Eclipta prostrata, Ludwigia adscendens, Sagittaria trifolia, Echinochloa crus-galli, Leptochloa chinensis) the data was collected using a handheld video camera. Their model was an AlexNet serving as the feature extractor and SVM as a classifier. They reported an accuracy of 94.5%.

- ***Kounalakis et al. (2019)*** [42], Used a dataset of Clover and grass images, and Broad-leaved dock weeds, all images are monochrome, because they argued that since weeds are green just like plants, colors do not add new information. Their model was a CNN for feature extraction along with traditional ML classifiers. They concluded that best combination is Resnet50 with L2 Regularized Logistic Regression for classification, they reported a classification accuracy of 96.1%.

Model	Technique	Dataset	Conclusion
Espejo-Garcia et al. (2020)	Several, CNN as a feature extractor, ML (e.g. SVM) as classifiers	Their own	micro-f1 score of 99.21%
Umamaheswari and Jain (2020)	SegNet-512, SegNet-256	Their own, weed species not mentioned	SegNet-512 did better than SegNet-256
Yan et al. (2020)	AlexNet as a feature extractor, SVM as a classifier	Their own, images of Paddy plant, and 6 weed species	SegNet-512 did better than SegNet-256
Kounalakis et al. (2019)	Several, CNN for feature extraction, ML classifiers	Clover and grass images, and Broad-leaved dock weed, all images monochrome	Best combination was, Resnet50 with L2 Regularized Logistic Regression, with a classification accuracy of 96.1%

Table II.1: Summary of similar works

Data Sets

- ***Olsen et al. (2019)*** [43] is the one we worked on. Comprised of 17,509 RGB images of 8 weed species and various plants from across Australia, it does however, suffer from *class imbalance* (a class having significantly more data than the others).

- ***Du et al. (2022)*** [44], interesting dataset, contains 10,000 images of Flax and 14 categories of weeds, taken from various location in the USA and China, but the version they made available needs cleaning.

- ***Plant Seedlings Dataset*** [45], a big dataset comprised of 208,477 images, with 960 unique plants belonging to 12 species at several growth stages, it should be noted however, that they where taken in a laboratory setting where lighting conditions and other factors are ideal.

- ***Espejo-Garcia et al. (2020)*** [38], mentioned above.

Dataset	Image Count	Location	Weed Species
Olsen et al. (2019)	17,509	Various locations across Australia	8 species
Du et al. (2022)	10,000	Usa, China	14 species
Plant Seedlings Dataset	208,477	Denmark	12 species
Espejo-Garcia et al. (2020)	Not Mentioned	Greece	Early crop weed

Table II.2: Summary of notable datasets

II.3.1. Related Works

Our criteria for selecting similar works is that the work had to employ a *deep learning* approach (e.g. a CNN), and it had to use the dataset we used (*DeepWeeds*), otherwise most attempts of comparison will be baseless and meaningless.

Only 2 met these criteria, the paper which introduced the dataset by *Olsen et al.* [43], and the paper by *Hu, Kun, et al.* [46].

In [43], *Olsen et al.* fine-tuned ResNet50, and InceptionV3 (pretrained on ImageNet) on the DeepWeeds dataset, and reported an average accuracy of 95.1% for InceptionV3 95.7% for ResNet50.

In [46], *Hu, Kun, et al.* employed a *Graph Convolutional Neural Network (GCN)* (architecture for processing graph-structured data, enabling effective information propagation and feature learning on graph nodes). They reported an accuracy of 98.1%.

III

The Model

"All models are wrong, but some are useful"
—George E. P. Box, Statistician

III.1. Experimental Setup

Our main training platform was ***Google Colab*** (before we switched to *Kaggle*). Google Colab is a cloud-based platform that provides a free, interactive environment for writing and running Python code, it offers up to 12 hours of free hardware accelerator (GPU, or TPU) usage (which is essential for training CNNs in a reasonable amount of time).

It offers access to powerful GPUs and TPUs, making it suitable for machine learning and data analysis tasks. Colab supports collaboration, allowing multiple users to work on the same notebook simultaneously. It integrates with other Google services, making it easy to import and export data from Google Drive and interact with other Google tools such as google drive.

Our experience with it was suboptimal to say the least, we never got more than 6 hours before being disconnected due to hitting the arbitrary runtime limits. It also has slow I/O when files are stored on Google Drive.

After facing many problems with Colab, we switched to ***Kaggle***, Kaggle is an online platform that hosts data science competitions and provides a collaborative environment for data scientists and machine learning practitioners. It offers datasets, computational resources, and a wide range of tools to explore, analyze, and model data. It also serves as a hub for sharing code. It offers 30 hours/week of hardware accelerators free of charge.

To speed up experimentation we used ***Saturn Cloud*** alongside Kaggle, Saturn Cloud offers easy access to scalable computing resources and integrates with popular data science libraries and tools. It offers 30 hours of free GPU usage per month, ***we do not recommend it***, beside the very little offered time, it suffers from frequents disconnects.

We used ***Keras***, Keras is a high-level neural network library written in Python that provides a API for building and training deep learning models. It allows for easy experimentation and prototyping. Keras supports multiple backend engines, including TensorFlow, Theano, and CNTK.

We used the ***Tensorflow*** backend as the training framework (through Keras), TensorFlow is an open-source deep learning framework developed by Google. It provides a comprehensive ecosystem for building and deploying machine learning models. TensorFlow enables efficient computation on both CPUs and GPUs, facilitating large-scale neural network training.

We used ***Jupyter*** as the coding environment, Jupyter is an open-source platform that enables interactive computing and data exploration. It provides a web-based interface for creating and sharing Jupyter Notebooks, which combine code, visualizations, and explanatory text. Jupyter supports multiple programming languages, including Python, R, and Julia.

We used ***Git*** for version control, to manage the code the models. Git is a distributed version control system created by *Linus Torvalds* in 2006, originally to manage the linux kernel codebase. It allows multiple collaborators to work on a project simultaneously. It tracks changes to files and allows for easy collaboration, branching, merging, and reverting. Git provides a robust history of changes and facilitates efficient code management for software development teams. It is widely used in the industry and has become a standard tool for version control.

III.2. Model Architecture

We experimented with several architectures (namely *DenseNet-121*, *ResNet50V2*, *MobileNetV3 Large*, and *MobileNetV2*), but the main one was *DenseNet-121*.

DenseNet-121 is a convolutional neural network known for its dense connectivity pattern, where each layer is connected to every other layer in a feed-forward manner. It comprises multiple dense blocks, each containing several convolutional layers with batch normalization and ReLU activations. Transition layers with pooling and dimensionality reduction are inserted between dense blocks to control the number of parameters. Refer to subsection II.2.6 for a more thorough discussion of the DenseNet architecture.

We used a pretrained model that was originally trained on the ImageNet dataset (which contains a 1 million images belonging divided into 1000 classes).

We added a ***global average pooling layer***, It computes the average value for each feature map, it captures the overall presence or absence of each feature throughout the image. This reduces the number of parameters in the network and provides a global context, enabling the classifier to focus on essential features rather than specific spatial locations. It also aids in making the network more robust to spatial translations and improves computational efficiency.

We also added a ***dropout layer*** with a probability of *0.5* before the classifier[1] to help reduce overfitting, Dropout is a regularization technique used in neural networks to prevent overfitting. It randomly deactivates a fraction of the neurons during training, forcing the network to learn robust representations that are not overly dependent on specific neurons, improving generalization.
We replaced the last layer (the classifier) with a dense layer comprised of 9 neurons (one for each class) with *softmax* as its activation function.

Softmax is a mathematical function often used as the output activation in *multi-class classification* (where classes are mutually exclusive) tasks. It converts a vector of real numbers into a probability distribution,

[1]This is usually employed when the classifier is comprised of several layers and not just one, because in this case it might cause a wrong prediction, but surprisingly it gave better results than not using a dropout layer

where each element represents the likelihood of belonging to a specific class. Softmax normalizes the input values by exponentiating them and dividing by the sum of the exponentiated values, ensuring that the resulting probabilities add up to one. This activation function provides a useful interpretation of the model's outputs as class probabilities, facilitating the selection of the most probable class prediction.[47]

$$f(s)_i = \frac{e^{s_i}}{\sum_j^C e^{s_j}} \tag{III.1}$$

Mathematically, it is represented by eq IV.1, where s_j are the scores inferred by the net for each class in C. Note that the Softmax activation for a class si depends on all the scores in s.

III.3. Dataset Description

We used the *DeepWeeds* dataset by *Olsen et al. (2019)* [43]. It consists of 17509 RGB images of 8 species of weeds and various flora, all captured in various regions across Australia.

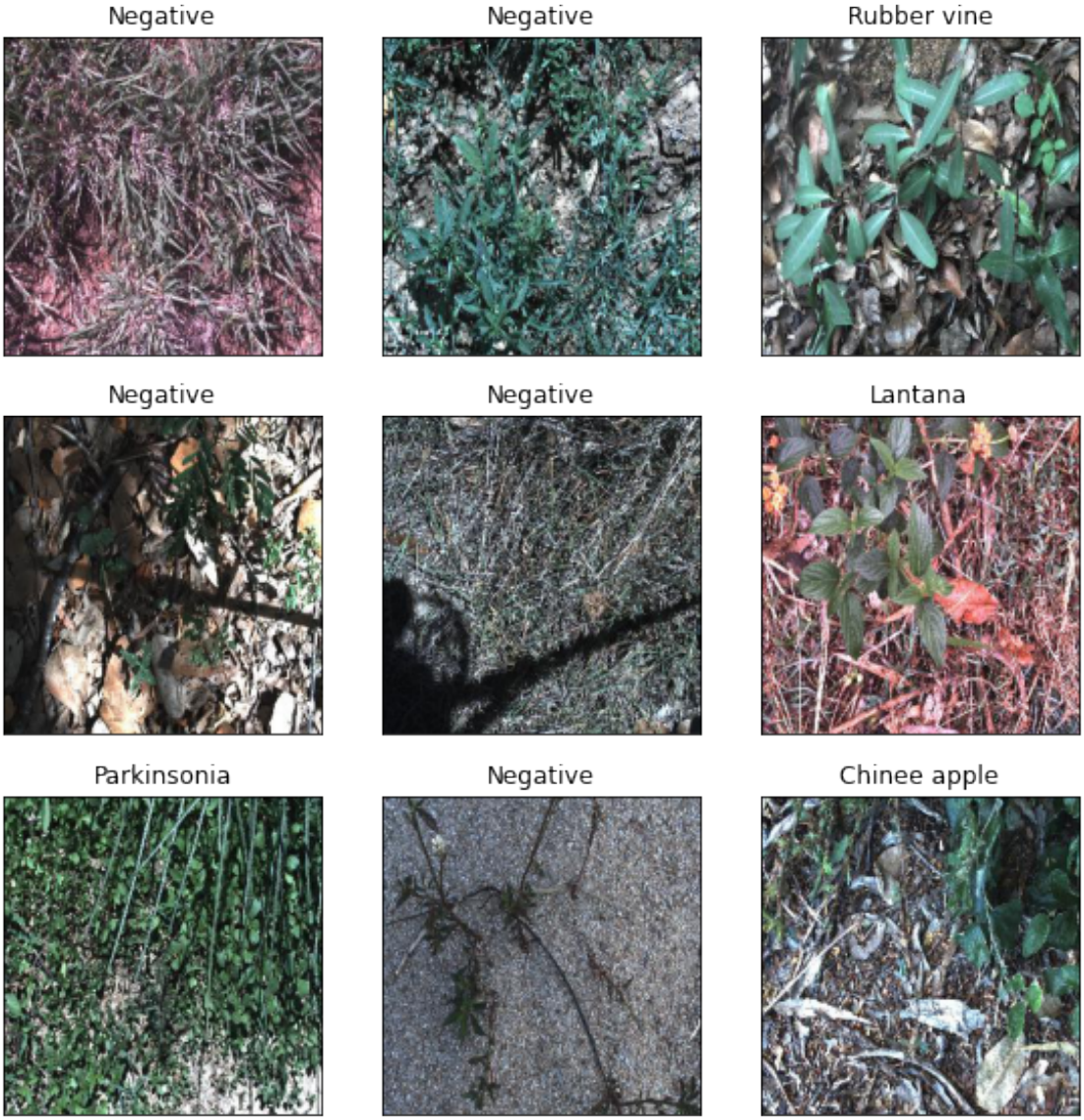


Figure III.1: Sample images from the dataset

Weed Species	Image Count
Chinee Apple	1125
Lantana	1064
Parkinsonia	1031
Parthenium	1022
Prickly Acacia	1062
Rubber Vine	1009
Siam Weed	1074
Snake Weed	1016
Negative	9106
Total	17509

Table III.1: Number of images per weed species

Bigger datasets would have been too slow to train on, given our limited hardware resources, and smaller ones would not have reflected the power of deep learning.

III.4. Training Procedure

Training a model essentially boils down to find the minima of a very big multivariate function (which is the neural network). But, becausse the function is too big, It is not feasible to find the global minima, so we try to reach some optimal local minima, through the *Backpropagation algorithm*.

Backpropagation is a technique used to compute the gradients of the model's parameters with respect to the loss function, enabling efficient optimization of neural networks. It involves a two-step process: the forward pass computes the output of the network given the input, and the backward pass propagates the error gradients from the output layer back to the input layer, updating the parameters using gradient descent or other optimization algorithms. This iterative process allows the network to adjust its parameters to minimize the error and improve its predictive capabilities.

Hyperparameters are parameters whose values are used to control the learning process. By contrast, the values of other parameters (typically node weights) are derived via training. Hyperparameters include the batch size, the optimizer, the learning rate.
Refer to [48] for a more thorough discussion of the mathematical underpinnings of neural networks.

III.4.1. Transfer Learning

Given the small dataset (by deep learning standards), we applied the technique of ***transfer learning*** and finetuned the pre-trained model. Transfer learning refers to using pre-trained models as a starting point and leveraging their learned features for a new task. In transfer learning, the entire pre-trained model or its specific layers are utilized, and only the final layers are retrained on the new task-specific data.

We used a ***batch size*** of 32, (Batch size is the number of training examples utilized in a single forward and backward pass during the training, the higher it is, the faster the training due to parallel computations on the GPU, on the other hand, Smaller batch sizes may provide more accurate gradient estimates).

We used the ***Adam optimizer*** because it fared better than other (such as *stochastic gradient descent (SGD)*),

(An optimizer is an algorithm used to adjust the parameters of a model during training to minimize the loss function, optimizing the model's performance and guiding the learning process).

We used ***an initial learning rate*** of 0.0001 (because we used a pre-trained model, a high learning rate would cause large gradient updates which is not optimal since we want only small adjustments because the model already learned how to recognize many features from the ImageNet dataset).

The learning rate is a hyperparameter which controls the magnitude of parameter updates in a neural network during training, influencing the speed and stability of convergence. It scales the gradients computed during backpropagation and determines the extent to which the network's parameters are adjusted in each optimization step. A higher learning rate may result in faster training but risks overshooting the optimal solution, while a lower learning rate may lead to slower convergence or getting stuck).

We used ***Sparse Categorical Cross Entropy***[2] as the loss function, (the loss function is also known as the objective or cost function, it measures the discrepancy between the predicted outputs of a model and the true values in the training data. It quantifies the error or the degree of mismatch between the model's predictions and the ground truth, allowing it to learn and update its parameters to minimize this error during the training process).

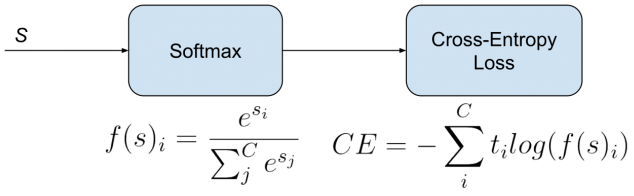


Figure III.2: The categorical cross-entropy function, source:[49]

Mathematically, The outputted loss is the negative average of the sum of the true values multiplied by the log of the predicted values.

We monitored the loss (used it as the *metric* during training) and every-time it decreased the model got saved to disk (as a *.hdf5* file, (It is an efficient file format commonly used to store and organize large amounts of numerical data, including model weights and configurations in deep learning)).

We also halved the learning rate each time the loss did not decrease after 8 consecutive epochs, and stopped training if the loss did not decrease after 16 consecutive epochs (called *Early Stopping*) (not necessarily a proof that the model has converged but we were limited by time and hardware resources).

We set a maximum *epoch* of 70 (but not always reached it due to early stopping). An *epoch* is a single pass of the entire training dataset through a learning algorithm. During one epoch, the algorithm processes and updates the model's parameters using all of the available training examples. The number of epochs determines how many times the algorithm iterates through the entire dataset. Each epoch allows the model to learn from the data and adjust its parameters to improve its performance.

[2] *Sparse* in this context means that the labels are integers, as opposed to *one hot encoded arrays*, but using just Categorical Cross Entropy would work just as well because the integers would be converted to arrays internally.

IV

Evaluation of The Model

"Evaluating and analyzing mathematical models is like polishing a diamond, each facet reveals new insights, illuminating the brilliance of knowledge and empowering us to unravel the mysteries of the universe"

—Terence Tao, Mathematician

IV.1. Evaluation Metrics

- ***Accuracy***, is the ratio of number of correct predictions to the total number of predictions.

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ number\ of\ Predictions} \tag{IV.1}$$

- ***Confusion matrix***, is a table that summarises the predictions of a classification model. Each entry i, j in a confusion matrix is the number of observations actually in group j, but predicted to be in group i.

- ***Precision***, is the number of true positives over the number of true positives plus the number of false positives.[47]

$$Precision = \frac{truepositives}{truepositives + falsepositives} \tag{IV.2}$$

- ***Recall***, identifies how many positive labels the model identified out of all the possible positive labels.[47]

$$Recall = \frac{truepositives}{truepositives + falsenegatives} \tag{IV.3}$$

- ***Normalized confusion matrix***, A normalized confusion matrix, also known as a confusion matrix with row-wise normalization, is obtained by dividing each element in the confusion matrix by the sum of its corresponding row, the recall of each class is the corresponding position in the diagonal.[47]

- ***F1-Score***, is the harmonic mean of precision and recall.[47]

$$F_1 = 2\frac{precision \cdot recall}{precision + recall} \tag{IV.4}$$

IV.2. DenseNet121

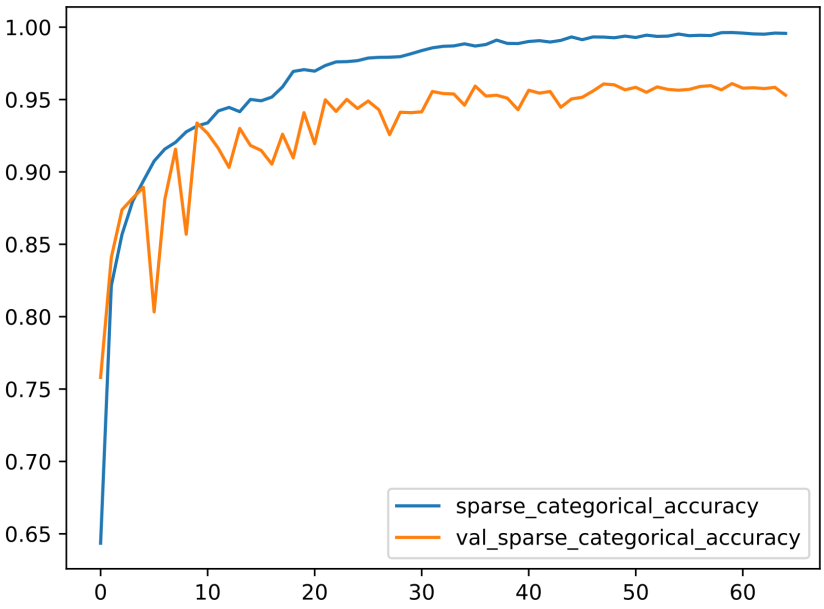


Figure IV.1: Validation accuracy, and Training accuracy for each epoch for DenseNet121

Learning plateaued after epoch 48. Starting from the that epoch with one tenth of the original learning rate improved loss a little bit (about 0.2%). Probably, the model could have improved a bit more if trained for more epochs with a much lower learning rate, as the loss is still a bit higher than we had hoped, at around 0.12.

Weed Species	Precision	Recall	F1-Score	Image Count
Chinee Apple	0.97	0.91	0.94	226
Lantana	0.95	1.00	0.97	213
Parkinsonia	0.95	0.99	0.97	207
Parthenium	0.99	0.92	0.95	205
Prickly Acacia	0.94	0.99	0.96	213
Rubber Vine	0.95	0.95	0.95	202
Siam Weed	0.95	1.00	0.97	215
Snake Weed	0.94	0.95	0.94	204
Negative	0.98	0.98	0.98	1821
Accuracy			0.97	3507
Macro Average	0.96	0.96	0.96	3507
Weighted Average	0.97	0.97	0.97	3507

Table IV.1: Precision, Recall, and F1-Score for Densenet121

The high F1-Score indicate that the model performs well in terms of both identifying true positives and avoiding false positives and false negatives.

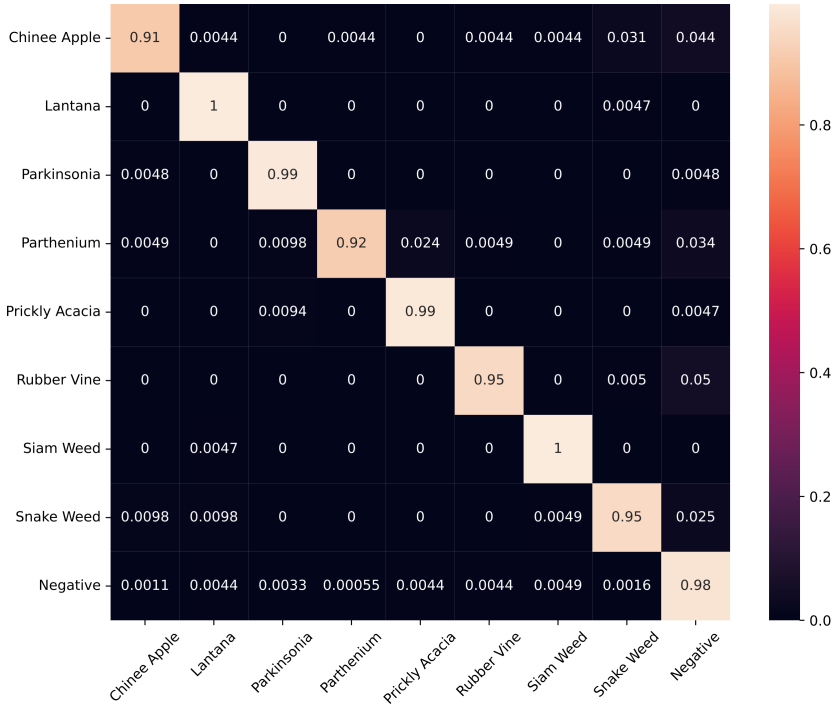


Figure IV.2: Normalized confusion matrix for Densenet121

We can see that the most correctly predicted class was the *Lantana* class, and the worst was *Chinee Apple* (probably due to similarity with the negative class). We also performed 5-fold cross validation with densenet [1], and found the results to be very similar with each fold (with the top accuracy being 96.82%, and the worst being 95.70%), with an average accuracy of all fold being 96% indicating the model's ability to generalize well.

IV.3. ResNet50 V2

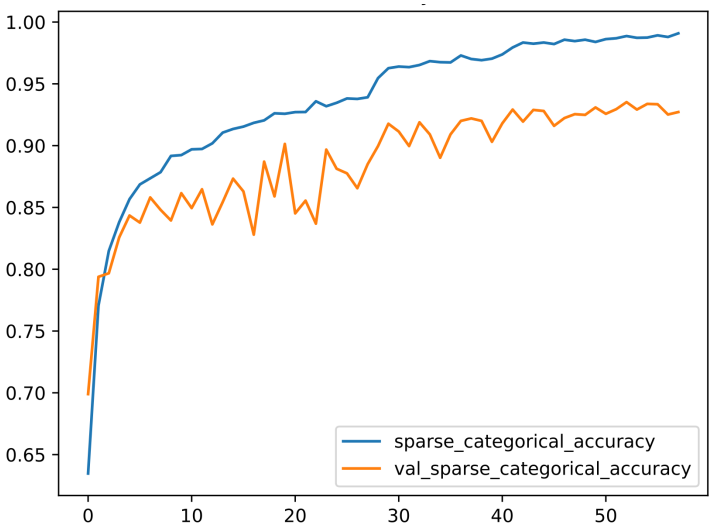


Figure IV.3: Validation accuracy, and Training accuracy for each epoch, for ResNet50v2

[1] all of the code and related things are made available at the GitHub repository https://github.com/user062/Master-Thesis

Learning plateaued after 42 epochs with a validation accuracy of 93%.

Weed Species	Precision	Recall	F1-Score	Image Count
Chinee Apple	0.80	0.88	0.84	226
Lantana	0.89	0.97	0.93	213
Parkinsonia	0.93	1.00	0.96	207
Parthenium	0.92	0.90	0.91	205
Prickly Acacia	0.84	0.93	0.88	213
Rubber Vine	0.94	0.92	0.93	202
Siam Weed	0.95	0.97	0.96	215
Snake Weed	0.83	0.87	0.85	204
Negative	0.97	0.93	0.95	1821
Accuracy			0.93	3507
Macro Average	0.90	0.93	0.91	3507
Weighted Average	0.93	0.93	0.93	3507

Table IV.2: Precision, Recall, and F1-Score for ResNet50v2

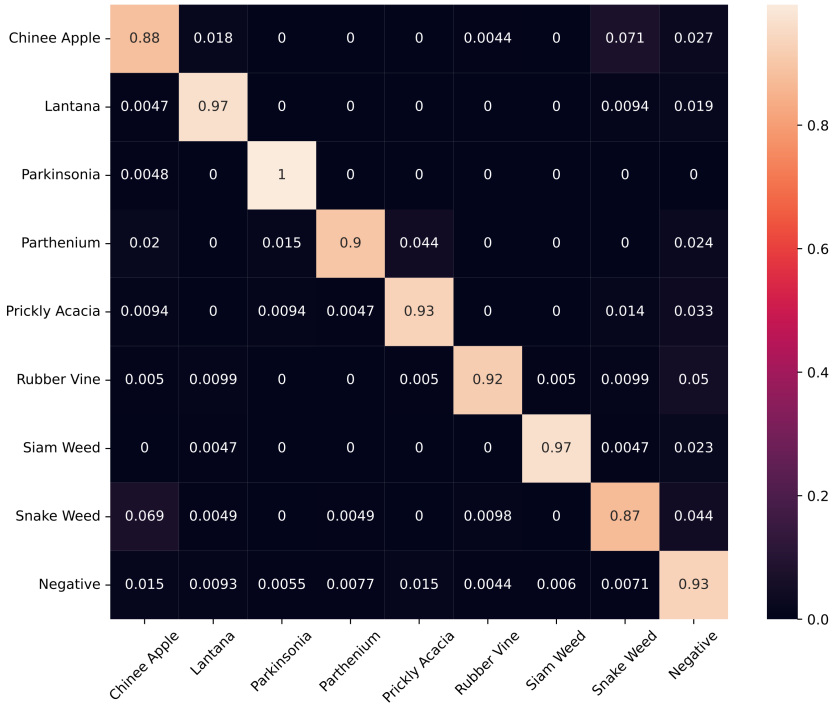


Figure IV.4: Normalized confusion matrix for ResNet50 V2

The least correctly predicted class was *Chinee Apple* class, the same as densenet (albeit desenet predicted better, 0.91% vs 0.88%)

IV.4. MobileNet V3 Large

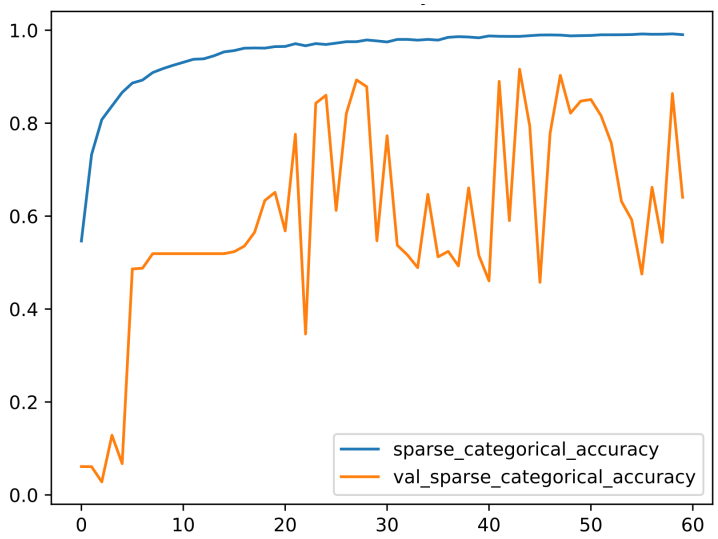


Figure IV.5: Validation accuracy, and Training accuracy for each epoch for MobileNet V3

Learning plateaued after Very Large fluctuations in accuracy throughout training despite the very low learning rate, a better approach with mobilenetv3 might have been to train only the classifier.

Weed Species	Precision	Recall	F1-Score	Image Count
Chinee Apple	0.91	0.77	0.83	226
Lantana	0.91	0.89	0.90	213
Parkinsonia	0.96	0.93	0.94	207
Parthenium	0.89	0.91	0.90	205
Prickly Acacia	0.89	0.87	0.88	213
Rubber Vine	0.97	0.90	0.93	202
Siam Weed	0.95	0.90	0.93	215
Snake Weed	0.94	0.72	0.81	204
Negative	0.91	0.97	0.94	1821
Accuracy			0.92	3507
Macro Average	0.93	0.87	0.90	3507
Weighted Average	0.92	0.92	0.92	3507

Table IV.3: Precision, Recall, and F1-Score for MobileNetV3

Despite the troubled training process, it did perform reasonably well, with an accuracy of 92%. The hardest class to classify was *Snake Weed*.

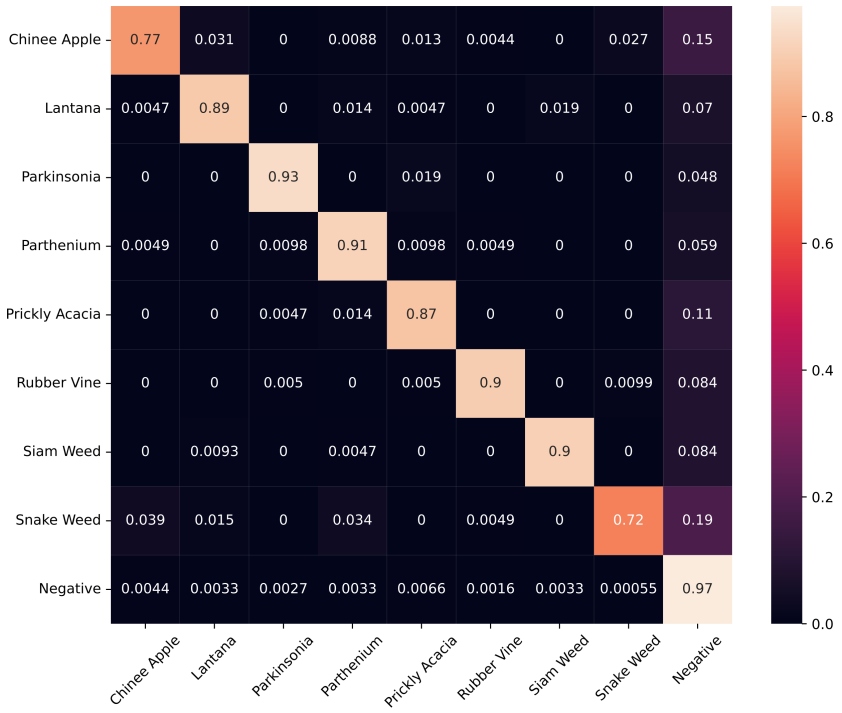


Figure IV.6: Normalized confusion matrix for MobileNet V3

IV.5. MobileNet V2

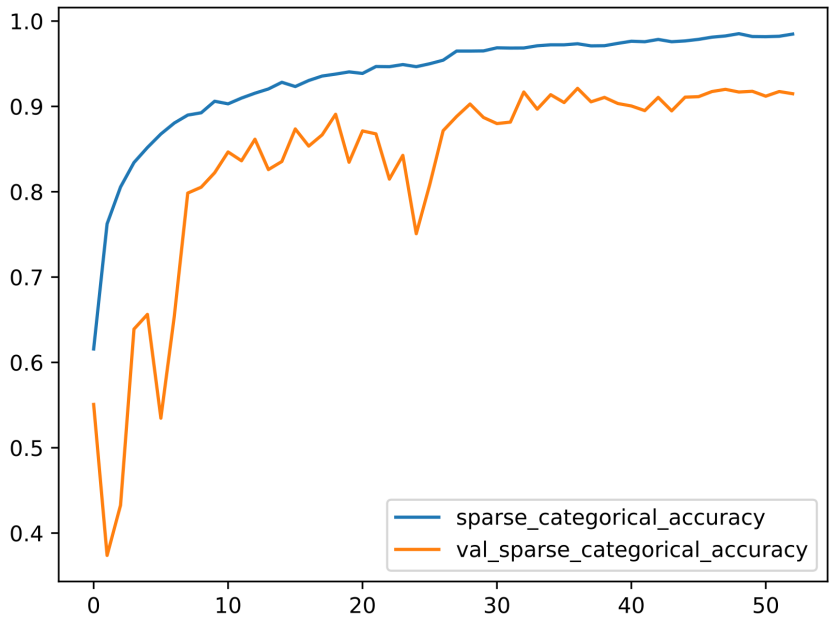


Figure IV.7: Validation accuracy, and Training accuracy for each epoch for MobileNet V2

Weed Species	Precision	Recall	F1-Score	Image Count
Chinee Apple	0.88	0.84	0.86	226
Lantana	0.89	0.91	0.90	213
Parkinsonia	0.91	0.98	0.95	207
Parthenium	0.94	0.89	0.91	205
Prickly Acacia	0.95	0.87	0.91	213
Rubber Vine	0.97	0.90	0.94	202
Siam Weed	0.87	0.99	0.93	215
Snake Weed	0.91	0.81	0.86	204
Negative	0.95	0.96	0.96	1822
Accuracy			0.93	3507
Macro Average	0.92	0.91	0.91	3507
Weighted Average	0.93	0.93	0.93	3507

Table IV.4: Precision, Recall, and F1-Score for MobileNetV2

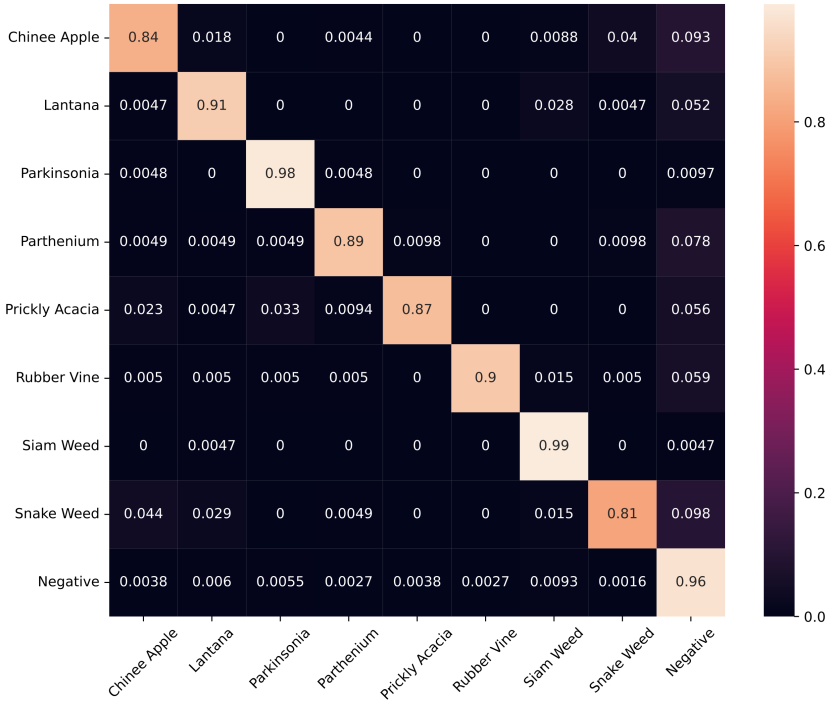


Figure IV.8: Normalized confusion matrix for MobileNetV2

MobileNet V2 did better than MobileNet V3 Large albeit slightly worse than ResNet50 V2. Despite being tailored to mobile platforms and thus optimized computational efficiency over performance it did achieve an accuracy of 93%.

IV.6. Conclusion and comparison against state of the art results

Model	Accuracy
DenseNet121	**96.82%**
ResNet50 V2	92.90%
MobileNet V3 Large	91.93%
MobileNet V2	93.27%
ResNet50 ([43])	94.39%
DenseNet202 [46]	95.3%
GWN with DenseNet202 backbone ([46])	**98.1%**
GWN with ResNet50 backbone ([46])	97.4%

Table IV.5: Accuracy of different models with on *deepweeds*

DenseNet121 gave the best results with a top accuracy of 96.82%, followed by Mobilenet V2 with an accuracy of 93.27% then by ResNet50 V2 and the worst performing model was Mobilenet V3 with an accuracy of 92%.

While both GWNs with ResNet50 and DenseNet202 backbones by [46] surpassed our model by 0.7% and 1.28% respectively.

- Our main model (DenseNet121) surpassed [43], and the Heavier DenseNet202 model by [46]
- The GWN with both ResNet50 and DenseNet202 backbones by [46] surpassed our model by 0.7% and 1.28% respectively.
- Higher accuracy with these models comes at the cost of inference speed with DenseNet202 backbone, and probably with the ResNet50 backbone too.
- No conclusive results regarding inference time (except that DenseNet202 is heavier, thus slower than its 121 variant).

V

Future Prospects

"AI will propel us into a future where machines master complexity, learn from data, and autonomously solve complex problems, revolutionizing industries and transforming our understanding of what is possible."

—Andrew Moore, Computer Scientist and AI Expert

This work leaves a lot to be desired, it's not enough to train a CNN on a dataset, without any consideration for real applications, but alas, the work was hampered thanks to the lack of proper tooling and other obstacles.

V.1. Multi-Agent System

One fundamental distinction between multi-agent and single-agent systems lies in their coordination capabilities and coverage efficiency. Multi-agent systems comprise multiple robots that can work in parallel and cover larger areas simultaneously. With their ability to communicate and coordinate tasks, these systems exhibit higher overall efficiency, reducing the time required for weed control in big farms.

In contrast, single-agent systems operate with a solitary robot, limiting their coverage and potentially prolonging the eradication process.

It's interesting to explore such system, especially the networking aspect, a *MANET (mobile ad hoc network)* for example. Because this is the first step towards real technological robot-assisted agriculture.

For example a if a it detects a type of weed which can't be eradicated through herbicides but requires other tools/measures, it could signal that to the appropriate robot(s). If a robot becomes faulty it could signal that to others (similar to what happens in *Tesla, Inc.* Factories). Updates with newer better neural networks (or any software updates) could be rolled over-the-air.

There can Also be communications between a drone and the weeding robots, the purpose of which is it to delegate them to where weeds are concentrated hence reducing the time of eradication. [50] explored similar ideas in more depth.

Figure V.1: Bosch startup Deep Field Robotics robots deployed in a field, an example of an autonomous system with communications between the agents, source:Bosch

V.2. Experiment with Visual Transformers

Visual transformers (ViT), introduced by *Vaswani et al.* [15] have emerged as a groundbreaking approach for image classification, revolutionizing the field of computer vision. Originally focused on the application of transformers in natural language processing tasks, It was later in 2020 that *Dosovitskiy et al.* presented the groundbreaking work *the Image Transformer* [51].

Unlike traditional convolutional neural networks (CNNs), which rely on predefined hierarchical structures, visual transformers leverage the power of *self-attention* mechanisms to capture long-range dependencies within images. This enables the model to attend to relevant image regions and learn complex relationships between pixels. By employing self-attention layers, visual transformers excel in capturing global contextual information, leading to superior performance in tasks such as object recognition, semantic segmentation, and scene understanding.

The attention mechanism allows the model to focus on fine-grained details while considering the image as a whole, facilitating robust feature extraction and reducing the impact of spatial transformations. Moreover, visual transformers exhibit impressive scalability, allowing for the processing of high-resolution images without significant loss in accuracy.

These models have showcased remarkable generalization abilities, outperforming CNNs on several benchmark datasets. With their ability to model both local and global context, visual transformers have opened up new avenues for advancing image classification and paving the way for further breakthroughs in computer vision research.

However, they have some drawbacks, such as requiring more computational power to train compared to CNNs, *Feature collapsing* (also known as *attention collapse* or *attention deficiency*, refers to a phenomenon observed in transformers where the self-attention mechanism fails to effectively capture meaningful dependencies across different positions or tokens in the input sequence), and requiring a fixed size input [52].

V.3. Generate More Data With GANs

A *generative adversarial network* (GAN) is a class of machine learning frameworks designed by *Goodfellow et al.* in 2014 [13]. Generating data using *Generative Adversarial Networks* has revolutionized the field of machine learning by enabling the creation of synthetic data that closely resembles real-world examples.

GANs consist of two components: a generator and a discriminator, which engage in a game-like competition. The generator learns to generate increasingly realistic samples, while the discriminator strives to distinguish between real and fake data. As the training progresses, the generator becomes adept at producing synthetic data that exhibits the same statistical properties as the real

data it was trained on.

This capability opens up exciting possibilities for training machine learning models. Synthetic data generated by GANs can be used to augment existing datasets, addressing the common problem of limited labelled data availability. It allows for the creation of diverse and large-scale datasets, overcoming data scarcity issues and facilitating more robust and generalized model training.

Additionally, GAN-generated data can be employed to improve model performance in scenarios where collecting real data is time-consuming, expensive, or impractical.

By leveraging the power of GANs to generate realistic and representative data, machine learning practitioners can enhance the quality and efficiency of their models' training process, leading to better overall performance and more accurate predictions.[53] and [54] discuss this approach in more depth, Nvidia's Perfusion [16] could theoretically generate much more precise and accurate data, which is definitely something to explore further.

V.4. Explore The Space of Custom Hardware

The state of the art in custom hardware for deep learning training and inference is dominated by specialized chips known as *application-specific integrated circuits* (ASICs) and graphics processing units (GPUs).

ASICs designed specifically for deep learning, such as Google's Tensor Processing Units (TPUs) and NVIDIA's Deep Learning Accelerators (DLAs), offer high performance and energy efficiency. These chips are optimized for matrix multiplication, a fundamental operation in deep learning, and often incorporate specialized circuitry for neural network computations.

GPUs, originally developed for graphics rendering, have become a popular choice for deep learning due to their parallel processing capabilities. NVIDIA's GPUs, in particular, are widely used in the deep learning community and are supported by various software frameworks.

In recent years, there has been a growing interest in field-programmable gate arrays (FPGAs) for deep learning acceleration. FPGAs provide flexibility as they can be reprogrammed to adapt to different neural network architectures. Companies like Xilinx and Intel offer FPGA-based solutions for deep learning. A recent example is [55], where *Khoda, Elham E., et al.* used an FPGA for physics applications.

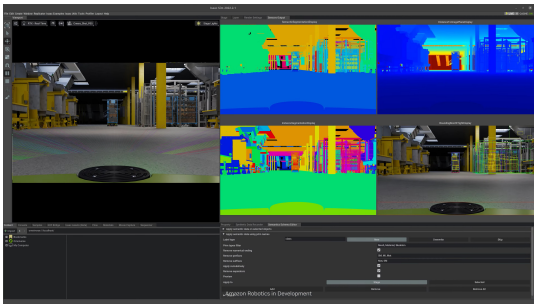
Additionally, there is ongoing research into other custom hardware architectures, including neuromorphic chips [56] and optical computing, aiming to further enhance the efficiency and performance of deep learning systems.

Overall, the state of the art in custom hardware for deep learning continues to evolve rapidly, driven by the demand for faster and more energy-efficient computations [57] [58].

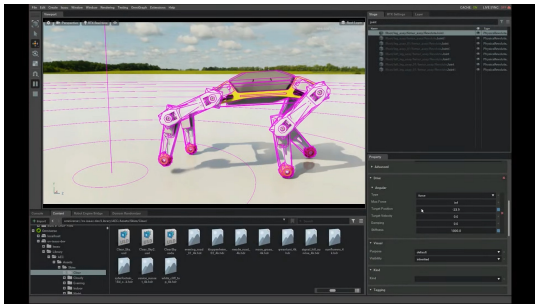
V.5. Run Simulations With Nvidia Isaac Sim

NVIDIA Isaac Sim is a high-fidelity simulator. It is specifically designed for training and testing autonomous robots and systems. It provides a realistic virtual environment where to simulate and evaluate models and applications before deploying them on physical robots.

Isaac Sim leverages advanced physics-based simulation, realistic rendering, and AI capabilities to create immersive and accurate virtual environments. It supports various sensors, such as cameras, lidar, and depth sensors, allowing developers to generate synthetic data for training deep learning models. The simulator also includes tools for creating and editing virtual scenes, defining robot behaviors, and conducting interactive simulations.



(a) Amazon simulated their warehouse robots in Isaac Sim, Source: Nvidia



(b) A robot being created in an environment in Isaac Sim, Source: Nvidia

Developing a DL model to be deployed on a robot without running simulations may lead to significant ineffectiveness and limitations. Simulations play a crucial role in identifying and rectifying potential flaws in the robot's design, programming, and operational strategies before real-world implementation.

Without simulations, it becomes challenging to anticipate and address various environmental factors, such as different soil types, weed species, and unexpected obstacles. This absence of comprehensive testing may result in suboptimal performance, and increased inefficiency of the robot when deployed. Additionally, the lack of simulations hampers the ability to optimize the robot's navigation, decision-making, and adaptability, diminishing its overall effectiveness in weed management.

V.6. Experiment With Graph Convolutional Networks

A graph convolutional neural network (GCN) is a type of neural network designed to operate on graph-structured data, introduced by *Thomas N. Kipf and Max Welling (2017)* [59]. Unlike traditional convolutional neural networks (CNNs) that are effective on grid-like data such as images, GCNs are specifically tailored to handle non-Euclidean and irregular graph structures.

GCNs employ a localized filtering operation that combines information from a node's neighbours to update its own representation. This operation is inspired by the convolution operation in CNNs but is adapted to graphs. It allows GCNs to capture both local and global dependencies in the graph structure.

Hu, Kun, et al. [46] reported an accuracy of 98.1% (previously unseen), this could hint at the potential of GCNs, which should be explored further.

Conclusion

"If we knew what we were doing, it would not be called research, would it?"
—Albert Einstein, Physicist, Developer of The Theory of Relativity

Our work demonstrates the potential of deep learning and smart farming technologies in addressing the challenge of weed eradication in agriculture. By training a Deep Convolutional Neural Network on a dataset of weed species, we have made a significant step towards automated detection and classification of weeds. This lays the foundation for developing robotic solutions that can effectively and efficiently combat weed infestations in agricultural fields.

Our work explored the effectiveness of 4 famous models through the technique of transfer learning and fine-tuning, namely *ResNet50 V2*, *DenseNet121*, *MobileNetV3Large* and *MobileNetV2*, for weed images detection and classification, on moderately sized dataset (comprised of 17,509 images of 8 species of weeds and various flora collected across a different regions in Australia). Our findings show that *DenseNet121* outperforms the other models, achieving an average classification accuracy of over 96% (thereby surpassing the original paper), *ResNet50 V2* came second, *MobileNet V2* came third and the worst was *MobileNetV3 Large*.

It is important to note however, that while DenseNet121 outperformed ResNet50v2, MobileNetV2 and MobileNetV3Large when trained in this particular way, to consider that different architectures may exhibit vastly different performance characteristics when trained under varying conditions, that is to say, the worse model might have fared better if trained differently, as the study by *Kornblith et al.* [60] shows.

However, further research and development are needed to refine and optimize our model for real-world deployment. This includes expanding the dataset to encompass a wider range of weed species and considering environmental factors that may affect weed detection accuracy. Additionally, the integration of our weed detection system with robotic platforms and field deployment experiments will be essential to validate its effectiveness and practicality.

Overall, our work highlights the promising potential of deep learning and smart farming in addressing weed-related challenges in agriculture. By leveraging advanced technologies and interdisciplinary approaches, we can pave the way for a more sustainable and efficient agricultural sector that can contribute to global food security and mitigate the impact of weed infestations on crop production.

References

[1] FAO WFP. *Hunger Hotspots. FAO-WFP early warnings on acute food insecurity: October 2022 to January 2023 Outlook*. 2022.

[2] UNFPA. *8 Billion Strong*. 2023.

[3] Michael Haenlein and Andreas Kaplan. "A brief history of artificial intelligence: On the past, present, and future of artificial intelligence". In: *California management review* 61.4 (2019), pp. 5–14.

[4] Yann LeCun et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324.

[5] Sébastien Bubeck et al. "Sparks of artificial general intelligence: Early experiments with gpt-4". In: *arXiv preprint arXiv:2303.12712* (2023).

[6] Max Roser. *The brief history of artificial intelligence*. URL: https://ourworldindata.org/brief-history-of-ai (visited on June 1, 2023).

[7] Norvig. P Russell. S. *Artificial intelligence a modern approach*. Pearson Education, Inc., 2020.

[8] Jaime Sevilla et al. "Compute trends across three eras of machine learning". In: *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2022, pp. 1–8.

[9] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. http://www.deeplearningbook.org. MIT Press, 2016.

[10] Alon Halevy, Peter Norvig, and Fernando Pereira. "The unreasonable effectiveness of data". In: *IEEE intelligent systems* 24.2 (2009), pp. 8–12.

[11] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Communications of the ACM* 60.6 (2017), pp. 84–90.

[12] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[13] Ian Goodfellow et al. "Generative adversarial networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[14] David Silver et al. "Mastering the game of Go with deep neural networks and tree search". In: *nature* 529.7587 (2016), pp. 484–489.

[15] Ashish Vaswani et al. "Attention is all you need". In: *Advances in neural information processing systems* 30 (2017).

[16] Yoad Tewel et al. "Key-Locked Rank One Editing for Text-to-Image Personalization". In: *ACM SIGGRAPH 2023 Conference Proceedings*. SIGGRAPH '23. Los Angeles, CA, USA, 2023.

[17] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories". In: *2006 IEEE computer society conference on computer vision and pattern recognition (CVPR'06)*. Vol. 2. IEEE. 2006, pp. 2169–2178.

[18] David G Lowe. "Distinctive image features from scale-invariant keypoints". In: *International journal of computer vision* 60 (2004), pp. 91–110.

[19] Chih-Wei Hsu, Chih-Chung Chang, Chih-Jen Lin, et al. *A practical guide to support vector classification*. 2003.

[20] Leo Breiman. "Random forests". In: *Machine learning* 45 (2001), pp. 5–32.

[21] Hanife Kebapci, Berrin Yanikoglu, and Gozde Unal. "Plant image retrieval using color, shape and texture features". In: *The Computer Journal* 54.9 (2011), pp. 1475–1490.

[22] Fei-Fei Li. *CS231n: Convolutional Neural Networks for Visual Recognition*. URL: http://cs231n.stanford.edu/ (visited on June 5, 2023).

[23] Wei Yu et al. "Visualizing and comparing AlexNet and VGG using deconvolutional layers". In: *Proceedings of the 33 rd International Conference on Machine Learning*. 2016.

[24] Vihar Kurama. *A Review of Popular Deep Learning Architectures: AlexNet, VGG16, and GoogleNet*. URL: https://blog.paperspace.com/popular-deep-learning-architectures-alexnet-vgg-googlenet/ (visited on June 3, 2023).

[25] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[26] Sergey Ioffe and Christian Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. pmlr. 2015, pp. 448–456.

[27] Christian Szegedy et al. "Rethinking the inception architecture for computer vision". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 2818–2826.

[28] Karen Simonyan and Andrew Zisserman. "Very deep convolutional networks for large-scale image recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[29] Gao Huang et al. "Densely connected convolutional networks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 4700–4708.

[30] Andrew G Howard et al. "Mobilenets: Efficient convolutional neural networks for mobile vision applications". In: *arXiv preprint arXiv:1704.04861* (2017).

[31] Mark Sandler et al. "Mobilenetv2: Inverted residuals and linear bottlenecks". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018, pp. 4510–4520.

[32] Andrew Howard et al. "Searching for mobilenetv3". In: *Proceedings of the IEEE/CVF international conference on computer vision*. 2019, pp. 1314–1324.

[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. "Fully convolutional networks for semantic segmentation". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 3431–3440.

[34] Juan Terven and Diana Cordova-Esparza. "A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond". In: *arXiv preprint arXiv:2304.00501* (2023).

[35] ASM Mahmudul Hasan et al. "A survey of deep learning techniques for weed detection from images". In: *Computers and Electronics in Agriculture* 184 (2021), p. 106067.

[36] University of Sydeny. *Weed-AI weeds images datasets repository*. URL: https://weed-ai.sydney.edu.au/datasets (visited on June 5, 2023).

[37] I. Sa et al. "weedNet: Dense Semantic Weed Classification Using Multispectral Images and MAV for Smart Farming". In: *IEEE Robotics and Automation Letters* 3.1 (Jan. 2018), pp. 588–595. DOI: 10.1109/LRA.2017.2774979.

[38] Borja Espejo-Garcia et al. "Towards weeds identification assistance through transfer learning". In: *Computers and Electronics in Agriculture* 171 (2020), p. 105306.

[39] Agricultural University of Athens. *Early Crop Weed*. URL: https://github.com/AUAgroup/early-crop-weed/ (visited on June 1, 2023).

[40] S Umamaheswari and Ashvini V Jain. "Encoder–decoder architecture for crop-weed classification using pixel-wise labelling". In: *2020 International Conference on Artificial Intelligence and Signal Processing (AISP)*. IEEE. 2020, pp. 1–6.

[41] Xue Yan, Xiangwu Deng, and Jing Jin. "Classification of weed species in the paddy field with DCNN-Learned features". In: *2020 IEEE 5th Information Technology and Mechatronics Engineering Conference (ITOEC)*. IEEE. 2020, pp. 336–340.

[42] Tsampikos Kounalakis, Georgios A Triantafyllidis, and Lazaros Nalpantidis. "Deep learning-based visual recognition of rumex for robotic precision farming". In: *Computers and Electronics in Agriculture* 165 (2019), p. 104973.

[43] Alex Olsen et al. “DeepWeeds: A multiclass weed species image dataset for deep learning”. In: *Scientific reports* 9.1 (2019), p. 2058.

[44] Yayun Du et al. “Deep-cnn based robotic multi-class under-canopy weed control in precision farming”. In: *2022 International Conference on Robotics and Automation (ICRA)*. IEEE. 2022, pp. 2273–2279.

[45] Thomas Mosgaard Giselsson et al. “A Public Image Database for Benchmark of Plant Seedling Classification Algorithms”. In: *arXiv preprint* (2017).

[46] Kun Hu et al. “Graph weeds net: A graph-based deep learning method for weed recognition”. In: *Computers and electronics in agriculture* 174 (2020), p. 105520.

[47] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O’Reilly Media, Inc.", 2022.

[48] Michael Nielsen. *Neural Networks and Deep Learning*. URL: http://neuralnetworksanddeeplearning.com/ (visited on Mar. 5, 2023).

[49] Raúl Gómez. *Understanding Categorical Cross-Entropy Loss, Binary Cross-Entropy Loss, Softmax Loss, Logistic Loss, Focal Loss and all those confusing names*. URL: https://gombru.github.io/2018/05/23/cross_entropy_loss/#losses (visited on June 5, 2023).

[50] Wyatt McAllister et al. “Multi-agent planning for coordinated robotic weed killing”. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2018, pp. 7955–7960.

[51] Alexey Dosovitskiy et al. “An image is worth 16x16 words: Transformers for image recognition at scale”. In: *arXiv preprint arXiv:2010.11929* (2020).

[52] Bo-Kai Ruan, Hong-Han Shuai, and Wen-Huang Cheng. “Vision transformers: state of the art and research challenges”. In: *arXiv preprint arXiv:2207.03041* (2022).

[53] Gabriel Eilertsen et al. “Ensembles of GANs for synthetic training data generation”. In: *arXiv preprint arXiv:2104.11797* (2021).

[54] Claire Little et al. “Generative adversarial networks for synthetic data generation: a comparative study”. In: *arXiv preprint arXiv:2112.01925* (2021).

[55] Elham E Khoda et al. “Ultra-low latency recurrent neural network inference on FPGAs for physics applications with hls4ml”. In: *Machine Learning: Science and Technology* 4.2 (2023), p. 025004.

[56] Christopher Wolters et al. “Biologically Plausible Learning on Neuromorphic Hardware Architectures”. In: *arXiv preprint arXiv:2212.14337* (2022).

[57] Maurizio Capra et al. “An updated survey of efficient hardware architectures for accelerating deep convolutional neural networks”. In: *Future Internet* 12.7 (2020), p. 113.

[58] Taiwo Samuel Ajani, Agbotiname Lucky Imoize, and Aderemi A Atayero. “An overview of machine learning within embedded and mobile devices–optimizations and applications”. In: *Sensors* 21.13 (2021), p. 4412.

[59] Thomas N Kipf and Max Welling. “Semi-supervised classification with graph convolutional networks”. In: *arXiv preprint arXiv:1609.02907* (2016).

[60] Simon Kornblith, Jonathon Shlens, and Quoc V Le. “Do better imagenet models transfer better?” In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2019, pp. 2661–2671.

A

The Code

Cell 01

```
import argparse
import os
import shutil
import pandas as pd
from time import time
from datetime import datetime
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.callbacks import ModelCheckpoint, EarlyStopping,
                                       ReduceLROnPlateau, TensorBoard, CSVLogger
from tensorflow.keras.optimizers import Adam
import csv
from tensorflow.keras.models import Model, load_model
import numpy as np
from sklearn.metrics import confusion_matrix, classification_report
from tensorflow.keras import backend as K
from skimage.io import imread
from skimage.transform import resize
from tensorflow.keras.applications import DenseNet121
from keras.layers import Dense, GlobalAveragePooling2D, Dropout
from tensorflow.keras import Input
import matplotlib.pyplot as plt
import seaborn as sns

# Global paths
OUTPUT_DIRECTORY = "./outputs/"
LABEL_DIRECTORY = "./labels/"
IMG_DIRECTORY = "./images/"

# Global variables
RAW_IMG_SIZE = (256, 256)
IMG_SIZE = (224, 224)
INPUT_SHAPE = (IMG_SIZE[0], IMG_SIZE[1], 3)
MAX_EPOCH = 70
BATCH_SIZE = 32
FOLDS = 1
STOPPING_PATIENCE = 8
LR_PATIENCE = 16
INITIAL_LR = 0.0001
CLASSES = [str(i) for i in range(9)]
CLASS_NAMES = ['Chinee Apple',
               'Lantana',
               'Parkinsonia',
               'Parthenium',
               'Prickly Acacia',
               'Rubber Vine',
               'Siam Weed',
               'Snake Weed',
               'Negative']
```

Cell 02

```
#read labels from csv files to pandas dataframe
train_label_file = "{}train_subset{}.csv".format(LABEL_DIRECTORY, 0)
val_label_file = "{}val_subset{}.csv".format(LABEL_DIRECTORY, 0)
test_label_file = "{}test_subset{}.csv".format(LABEL_DIRECTORY, 0)
train_dataframe = pd.read_csv(train_label_file, dtype=str)
val_dataframe = pd.read_csv(val_label_file, dtype=str)
test_dataframe = pd.read_csv(test_label_file, dtype=str)

# Training image augmentation
train_data_generator = ImageDataGenerator(
    #preprocessing_function = preprocess_input,
    rescale=1. / 255,
    fill_mode="constant",
    shear_range=0.2,
    zoom_range=(0.5, 1),
    horizontal_flip=True,
    rotation_range=360,
    channel_shift_range=25,
    brightness_range=(0.75, 1.25))

# No validation image augmentation (except for converting pixel values to floats)
val_data_generator = ImageDataGenerator(
    #preprocessing_function = preprocess_input
    rescale=1. / 255
)

# No testing image augmentation (except for converting pixel values to floats)
test_data_generator = ImageDataGenerator(
    #preprocessing_function = preprocess_input
    rescale=1. / 255
)

# Load train images in batches from directory and apply augmentations
train_data_generator = train_data_generator.flow_from_dataframe(
    train_dataframe,
    IMG_DIRECTORY,
    y_col='Label',
    x_col='Filename',
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    has_ext=True,
    classes=CLASSES,
    class_mode='sparse')

# Load validation images in batches from directory and apply rescaling
val_data_generator = val_data_generator.flow_from_dataframe(
    val_dataframe,
    IMG_DIRECTORY,
    y_col="Label",
    x_col="Filename",
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    has_ext=True,
    shuffle=False,
    classes=CLASSES,
    class_mode='sparse')

# Load test images in batches from directory and apply rescaling
test_data_generator = test_data_generator.flow_from_dataframe(
    test_dataframe,
    IMG_DIRECTORY,
    y_col="Label",
    x_col="Filename",
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    has_ext=True,
    shuffle=False,
    classes=CLASSES,
    class_mode='sparse')
```

Cell 03

```
base_model = MobileNetV2(weights="imagenet", include_top=False, pooling='avg',
                         input_shape=INPUT_SHAPE)

#Add dropout layer
x = Dropout(0.5)(base_model.output)
# Add fully connected output layer with softmax activation for multi class classification
outputs = Dense(len(CLASSES), activation='softmax', name='fc9')(x)

# Assemble the modified model
model = Model(inputs=base_model.inputs, outputs=outputs)

timestamp = datetime.fromtimestamp(time()).strftime('%Y%m%d-%H%M%S')

print('Fold {}/{} - {}'.format(0 + 1, FOLDS, timestamp))
output_directory = "{}{}/".format(OUTPUT_DIRECTORY, timestamp)

if not os.path.exists(output_directory):
    os.makedirs(output_directory)

# Checkpoints for training
model_checkpoint = ModelCheckpoint(output_directory + "lastbest-0.hdf5", verbose=1,
                                   save_best_only=True)
early_stopping = EarlyStopping(patience=16, restore_best_weights=True)
tensorboard = TensorBoard(log_dir=output_directory, histogram_freq=0, write_graph=True,
                          write_images=False)
reduce_lr = ReduceLROnPlateau('val_loss', factor=0.5, patience=8, min_lr=0.000003125)
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=Adam(learning_rate=INITIAL_LR), metrics=['sparse_categorical_accuracy'])
csv_logger = CSVLogger(output_directory + "training_metrics.csv")
```

Cell 04

```
history = model.fit(
        train_data_generator,
        steps_per_epoch=train_image_count // BATCH_SIZE,
        epochs=MAX_EPOCH,
        validation_data=val_data_generator,
        validation_steps=val_image_count // BATCH_SIZE,
        callbacks=[tensorboard, model_checkpoint, early_stopping,
                   reduce_lr, csv_logger], shuffle=False)

np.save(f'{output_directory}model_history.npy', history.history)
```

Cell 05

```
# Testing the model
#test_data_generator.reset()
results = model.evaluate(test_data_generator)
print(f'test_loss: {results[0]} - test_categorical_accuracy: {results[1]*100}')
```

Cell 06

```
# plot accuracy before fine tuning
pd.DataFrame(history.history)[['sparse_categorical_accuracy',
                               'val_sparse_categorical_accuracy']].plot()
plt.title("Accuracy")

plt.savefig(f"{output_directory}/accuracy.svg", format="svg")
```

Cell 07

```
# plot loss before
pd.DataFrame(history.history)[['loss', 'val_loss']].plot()
plt.title("Loss")
plt.savefig(f"{output_directory}/loss.svg", format="svg")
```

Cell 08

```
#Run predictions
test_data_generator.reset()
predictions = np.argmax(model.predict(test_data_generator), axis=1)
```

Cell 09

```
labels = dict((v, k) for k, v in test_data_generator.class_indices.items())
actual = list(test_dataframe.Label)
predictions = [labels[i] for i in predictions]
print(classification_report(actual, predictions))
```

Cell 10

```
#plot the normalized confusion matrix

cf = confusion_matrix(actual, predictions, normalize = "true")
plt.figure(figsize=(10, 8))
sns.heatmap(cf, annot=True,
            xticklabels = list(map(lambda x:CLASS_NAMES[int(x)], sorted(set(actual)))),
            yticklabels = list(map(lambda x:CLASS_NAMES[int(x)], sorted(set(actual)))))
plt.xticks(rotation=45)
plt.title('Confusion Matrix')
plt.savefig(f"{output_directory}/confusion_matrix.svg", format="svg")
```