**Czech University of Life Sciences Prague**

**Faculty of Economics and Management**

**Department of Informatics**

# Master's Thesis

## Classification of Temporal Images

**Samet Yıldızeli**

# CZECH UNIVERSITY OF LIFE SCIENCES PRAGUE

Faculty of Economics and Management

# DIPLOMA THESIS ASSIGNMENT

Samet Yıldızeli

Informatics

Thesis title

**Classification of temporal images**

---

**Objectives of thesis**

The main objective of the thesis is to find an optimal machine learning model for the classification of images that are produced by an experimental phenotyping platform. Specifically, the main objective targets identification of images where the crop starts to germinate.

The partial goals of the work are:
• Collect and preprocess the data
• Review and select state of the art models for comparison
• Retrain the models on given data for comparison

**Methodology**

The methodology of the diploma thesis is based on the study and analysis of professional information sources. In the practical part, appropriate models will be selected based on the theoretical part. The models will be retrained on the given data either from scratch or using transfer learning. Evaluation data will be consequently used for the comparison. Based on the synthesis of theoretical knowledge and the results of the practical part, the conclusions of the work will be formulated.

**The proposed extent of the thesis**

60 – 80 pages

**Keywords**

Classification, Temporal Images, Computer Vision

---

**Recommended information sources**

GONZALEZ Rafael, WOODS Richard, Digital Image Processing, Global Edition, 4th Edition, IL: Pearson Education, 2018. ISBN 9781292223049.

JAMES B., ZAHRAA S. Abdallah, Investigating Temporal Convolutional Neural Networks for Satellite Image Time Series Classification, 2022, Cornell University, arXiv:2204.08461

JENSEN, J R. *Introductory digital image processing : a remote sensing perspective.* Glenview, IL: Pearson Education, Inc., 2016. ISBN 9780134058160.

NIRBHAY B., SAMADRITA A., DIPTI T., Crop Classification with Multi-Temporal Satellite Image Data, IJERT, ISSN:2278-0181, Vol.9 Issue 06, June-2020

PRATT, W K. *Digital image processing : PIKS Scientific inside.* Hoboken, N.J.: Wiley-Interscience, 2007. ISBN 978-0-471-76777-0.

---

**Expected date of thesis defence**

2022/23 SS – FEM

**The Diploma Thesis Supervisor**

Ing. Jan Masner, Ph.D.

**Supervising department**

Department of Information Technologies

Electronic approval: 14. 7. 2022

**doc. Ing. Jiří Vaněk, Ph.D.**

Head of department

Electronic approval: 28. 11. 2022

**doc. Ing. Tomáš Šubrt, Ph.D.**

Dean

Prague on 18. 12. 2022

**Declaration**

I declare that I have worked on my master's thesis titled "Classification of Temporal Images" by myself and I have used only the sources mentioned at the end of the thesis. As the author of the master's thesis, I declare that the thesis does not break any copyrights.

In Prague on 31.03.2023 _____

**Acknowledgement**

I would like to express my deepest gratitude to my supervisor, Ing. Jan Masner, for his guidance throughout the course of my research. He provided invaluable insights and feedback that greatly contributed to the end of this thesis.

# Classification of Temporal Images

**Abstract**

The classification of temporal images is a crucial task in the field of computer vision and has numerous applications, including seed germination detection. In this thesis, two models were developed and trained for the classification of temporal images, with a focus on detecting seed germination in soil. The first model, YOLO, was trained for the detection of the first appearance of the plant on the soil. The second model, ConvLSTM, was trained for the detection of the germination of the seed in soil using temporal image classification.

The YOLO model was trained using a customized germination dataset containing two classes: "soil" and "FA", which represented the first appearance of the plant on the soil.

The results of the training process showed that both models achieved high levels of accuracy in detecting the respective classes. The YOLO model achieved precision values of 0.989 and 0.991 for the "soil" and "FA" classes, respectively, and a mean average precision of 0.995 for both classes. The ConvLSTM model achieved an accuracy of 0.9378, with a loss of 0.2469, demonstrating its effectiveness in detecting seed germination in soil using temporal image classification.

This thesis presents the development and training of two models for the classification of temporal images, with a focus on detecting seed germination in soil. The results show that both models are effective in detecting the respective classes, and the ConvLSTM model is particularly well suited for detecting seed germination in soil using temporal image classification.

**Keywords:** Temporal Image Classification, Seed Germination Detection, YOLO, ConvLSTM, Computer Vision, Image Processing.

# Klasifikace Časových Obrazů

**Abstrakt**

Klasifikace časových obrazů je klíčový úkol v oblasti počítačového vidění a má četné aplikace, včetně detekce klíčivosti semen. V této práci byly vyvinuty a natrénovány dva modely pro klasifikaci časových snímků se zaměřením na detekci klíčivosti semen v půdě. První model, YOLO, byl trénován pro detekci prvního výskytu rostliny na půdě. Druhý model, ConvLSTM, byl trénován pro detekci klíčení semen v půdě pomocí časové klasifikace obrazu. Model YOLO byl trénován pomocí přizpůsobené sady údajů o klíčení obsahující dvě třídy: „soil“ a „FA“, které představovaly první výskyt rostliny na půdě.

Výsledky tréninkového procesu ukázaly, že oba modely dosáhly vysoké úrovně přesnosti při detekci příslušných tříd. Model YOLO dosáhl hodnot přesnosti 0,989 a 0,991 pro třídy „soil“ a „FA“ a střední průměrnou přesnost 0,995 pro obě třídy. Model ConvLSTM dosáhl přesnosti 0,9378 se ztrátou 0,2469, což prokazuje jeho účinnost při detekci klíčení semen v půdě pomocí klasifikace časového obrazu.

Tato práce představuje vývoj a trénování dvou modelů pro klasifikaci časových snímků se zaměřením na detekci klíčivosti semen v půdě. Výsledky ukazují, že oba modely jsou účinné při detekci příslušných tříd a model ConvLSTM je zvláště vhodný pro detekci klíčivosti semen v půdě pomocí časové klasifikace obrazu.

**Klíčová slova:** Časová klasifikace obrazu, detekce klíčení semen, YOLO, ConvLSTM, počítačové vidění, zpracování obrazu.

# Table of content

**1 Introduction ..... 11**

**2 Objectives and Methodology ..... 12**

2.1 Objectives ..... 12

2.2 Methodology ..... 12

**3 Literature Review ..... 13**

3.1 Image Pre-Processing ..... 15

3.1.1 Image Scaling ..... 15

3.1.2 Grayscale Conversion ..... 17

3.1.3 Image Denoising ..... 17

3.1.4 Image Segmentation ..... 18

3.2 Data Annotation ..... 20

3.2.1 Image Annotation/Labeling ..... 21

3.3 Data Augmentation ..... 23

3.3.1 Types of Data Augmentation ..... 24

3.3.2 Data Augmentation Methods ..... 24

3.4 Computer Vision ..... 26

3.4.1 What is Artificial Intelligence? ..... 27

3.5 Neural Networks ..... 27

3.5.1 Cost Functions ..... 29

3.5.2 Activation Functions ..... 30

3.5.3 Forward Propagation ..... 33

3.5.4 Backward Propagation (Backpropagation) ..... 33

3.6 Machine Learning ..... 34

3.6.1 Supervised Learning ..... 34

3.6.2 Unsupervised Learning ..... 35

3.7 Deep Learning ..... 36

3.7.1 Convolutional Neural Networks (CNNs) ..... 37

3.7.2 Recurrent Neural Network (RNN) ..... 41

3.7.3 Long Short-Term Memory (LSTM) ..... 42

3.7.4 Temporal Convolutional Networks (TCN) ..... 43

3.8 Deep Learning Models for Computer Vision ..... 44

3.8.1 You Only Look Once (YOLO) ..... 44

3.8.2 ResNet50 ..... 45

3.9 Training Deep Learning Networks ..... 46

3.9.1 Training From Scratch ..... 46

3.9.2 Pre-Trained Models ..... 46

3.10 Related Work ..... 47
3.11 Overview of Theoretical Part ..... 48

**4 Practical Part ..... 49**
4.1 Software Environment ..... 50
4.2 Dataset Preprocess ..... 50
4.2.1 Dataset Generation ..... 50
4.2.2 Review of Dataset ..... 51
4.2.3 Data Annotation ..... 52
4.3 Training Models ..... 55
4.3.1 Training YOLO with Customized Dataset ..... 56
4.3.2 Training a CNN Model with LSTM Layer (ConvLSTM) ..... 57

**5 Results and Discussion ..... 59**
5.1 Results of First Approach ..... 59
5.2 Results of Second Approach ..... 60

**6 Conclusion ..... 63**

**7 References ..... 64**

**8 List of pictures, equations, and abbreviations ..... 66**
8.1 List of pictures ..... 66
8.2 List of equations ..... 67
8.3 List of abbreviations ..... 67

# 1 Introduction

Today's technologies are used in almost every field. In this era of digitalization, where data is the most valuable, they aim to create better business models by using this data obtained in every sector in their own business models.

Thanks to the results obtained by using statistical methods or various artificial intelligence methods on the obtained data, the work being done is both easier (using autonomous or semi-autonomous technologies) and more profitable models can be created in suitable environments and conditions. Important factors such as reducing the effort given for the work done, obtaining the same or even more efficient results in a shorter time, and increasing the labor and energy savings to optimum levels are some of the main purposes in today's business model using data. Important data outputs obtained from auxiliary computer systems not only make our work easier, but also predict how efficiently that job will be output in which environments. These auxiliary systems, where artificial intelligence comes first, are trained with data, and make predictions in their own systems.

Thanks to these new technologies, which are inevitable to be used in the agriculture sector, studies are carried out to grow plants more efficiently, to keep these plants connected not only to their natural environment, but also to be obtained efficiently in an artificial environment. In this direction, besides the analysis of the experimental environments created with technologies such as the internet of things (IOT) by computers, the data obtained is used for purposes such as artificial intelligence technologies, supporting research results, making predictions for future studies and research.

In line with the explanation above, artificial intelligence systems as assistance for research are useful for the field of agriculture as well. Therefore, in this thesis, the seedling moments of the plants in the soil cells on a tray were observed. With support from artificial intelligence systems, a machine learning technology is used to detect the timelines of the germination, that first appears in the soil, by using temporal image classification method which is CNN with LSTM and object detection in image frames.

# 2 Objectives and Methodology

## 2.1 Objectives

The thesis was aimed to find an optimal machine learning model for the classification of images produced by an experimental phenotyping platform. The main objective was to accurately identify images of where the crop starts to germinate in the soil. In order to achieve this objective, several partial goals were set.

The partial goals of the work are:

- **Collecting and preprocessing the data:** The first step in achieving the main objective is to gather enough data and preprocess it to make it suitable for the machine learning models.
- **Reviewing and selecting state-of-the-art models:** The next step is to review and select the most suitable machine learning models for comparison based on their performance on similar tasks.
- **Retraining the models:** The final step was to retrain the selected models on the given data either from scratch or using transfer learning. This allows for a direct comparison of the models and their performance on the aim of the thesis.

These partial goals, when combined, helped in achieving the main objective of the thesis and provided valuable insights into the use of machine learning in image classification.

## 2.2 Methodology

The methodology of the diploma thesis was based on the study and analysis of professional information sources. In the practical part, appropriate models were selected based on the theoretical part. The models were retrained on the given data either from scratch or using transfer learning. Evaluation data were consequently used for the comparison. Based on the synthesis of theoretical knowledge and the results of the practical part, the conclusions of the work were formulated.

# 3 Literature Review

Seedling must determine the appropriate mode of action based on its environment to best achieve photosynthetic success and enable the plant to complete its life cycle. Once the seedling emerges out of the soil, it initiates photomorphogenesis, a complex sequence of light-induced developmental and growth events leading to a fully functional leaf. This sequence includes severe reduction of hypocotyl growth, the opening of cotyledons, initiation of photosynthesis, and activation of the meristem at the shoot apex, a reservoir of undifferentiated cells that will lead to the formation of the first leaf [1].

The detection of the first appearance of the plant on the soil is a critical task in plant biology research, as it provides valuable information for crop management and scientific studies. Over the past few years, there has been a growing interest in using computer vision techniques to automate the detection of the first appearance of seeds in the soil. Various methods have been proposed to address this problem, ranging from traditional computer vision techniques to deep learning-based approaches [1].

While many computer vision approaches have been proposed for seed detection in soil images, relatively few studies have focused on the temporal aspect of the problem. Temporal image classification, which involves predicting the class label of an image based on its position in a temporal sequence, can be used to detect the first appearance of a seed in a series of soil images. By modeling the temporal dynamics of seed germination, temporal image classification can improve the accuracy and robustness of seed detection. However, designing effective temporal image classification models for seed detection remains a challenging task, due to the high variability and complexity of soil images and the subtle temporal changes in seed appearance [2][3]. There are several ways to deal with difficulties such as hue color range filtering to isolate the specific colors in an image based on the color after germination [4].

Traditional computer vision techniques have been used for object detection tasks for decades. Researchers have used techniques like feature extraction, edge detection, and segmentation to detect seeds in soil images. For example, researchers used the normalized cuts algorithm to segment seed images from the background soil. However, these methods can be limited by their reliance on handcrafted features and may not generalize well to new datasets [1] [5].

Recent advances in deep learning have led to significant improvements in computer vision tasks, including object detection. Deep learning models like convolutional neural networks (CNNs) have been successfully used for object detection tasks. For example, researchers used a combination of CNNs and long short-term memory (LSTM) networks to detect seed germination in Arabidopsis images. In [1] and [5], CNN was used to extract features from the images (see Figure 3.1), while the LSTM was used to model the temporal aspect of the data [2].

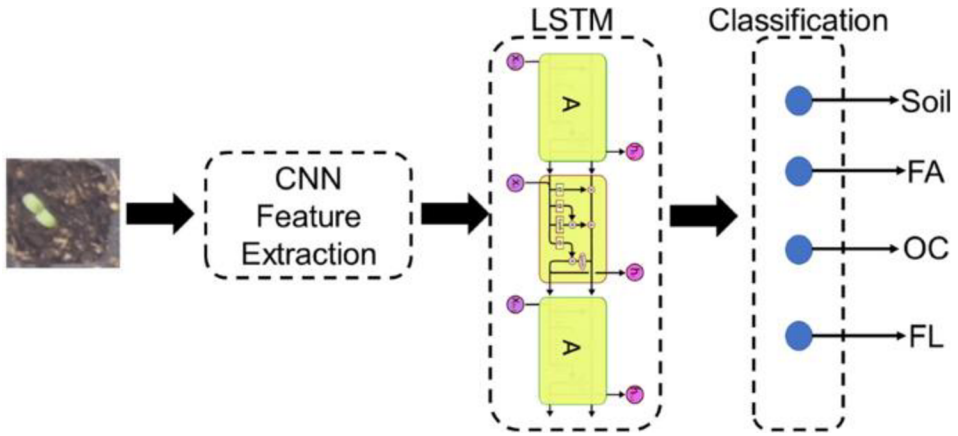



Figure 3.1: CNN Feature Extraction using LSTM layer for seedling detection [8]

One of the most popular deep learning-based object detection frameworks is the You Only Look Once (YOLO) algorithm. YOLO is a real-time object detection system that uses a single neural network to predict the class and location of objects in an image. YOLO has been used for a variety of object detection tasks, including plant detection. Therefore, YOLO can be sensitive to small objects, which may be a limitation for seed detection [8].

Another popular deep learning-based approach for object detection is the Faster R-CNN algorithm. Faster R-CNN uses a region proposal network to generate object proposals and CNN to classify and refine the proposals. Researchers used a Faster R-CNN model to detect seeds in soil images. The model achieved high accuracy in seed detection, but it was computationally expensive [4].

In conclusion, there have been several approaches proposed for the detection of the first appearance of seeds in soil images by using object detection or temporal image classification. Traditional computer vision techniques have been used with limited success due to their reliance on handcrafted features. Deep learning-based approaches, on the other hand, have shown significant improvements in object detection tasks, including seed

detection. While YOLO and Faster R-CNN are popular deep learning-based approaches, their sensitivity to small objects and computational complexity may be limitations for seed detection. In the Practical Part of the thesis, proposed approaches for the detection of the first appearance of seeds in the soil using temporal images are based on implementation of 2 different models which are YOLO and CNN with LSTM networks [9].

## 3.1 Image Pre-Processing

Image preprocessing is a crucial step in preparing images for model training and analysis. It involves techniques such as cropping, resizing, normalization, and color space conversions. The purpose of image preprocessing is to prepare the image for analysis by the model and to improve the model's performance [14].

In addition to improving model performance, image preprocessing can also help reduce model training time and increase model inference speed. Large input images can significantly slow down model training time, which can be mitigated by resizing or cropping the images. Reducing the size of input images can also help improve model inference speed without significantly impacting model performance [14].

### 3.1.1 Image Scaling

Image scaling is used to increase or decrease the size of a given image. Scaling algorithms are used to preserve details or features. There are multiple ways of scaling an image, some common ways are Nearest neighbor, Bilinear interpolation and Box sampling. These algorithms will be evaluated and chosen for which one that suits the network best [14].

#### 3.1.1.1 Nearest Neighbor Interpolation

The nearest neighbor algorithm is the simplest scaling algorithm, which involves selecting the nearest pixel to determine the color value of a new pixel in the scaled image (Figure 3.2). This method is fast and easy to implement but can lead to poor image quality due to its blocky or jagged edges in the scaled image. This method is useful for downscaling an image, but it's not recommended for upscaling an image. [15].

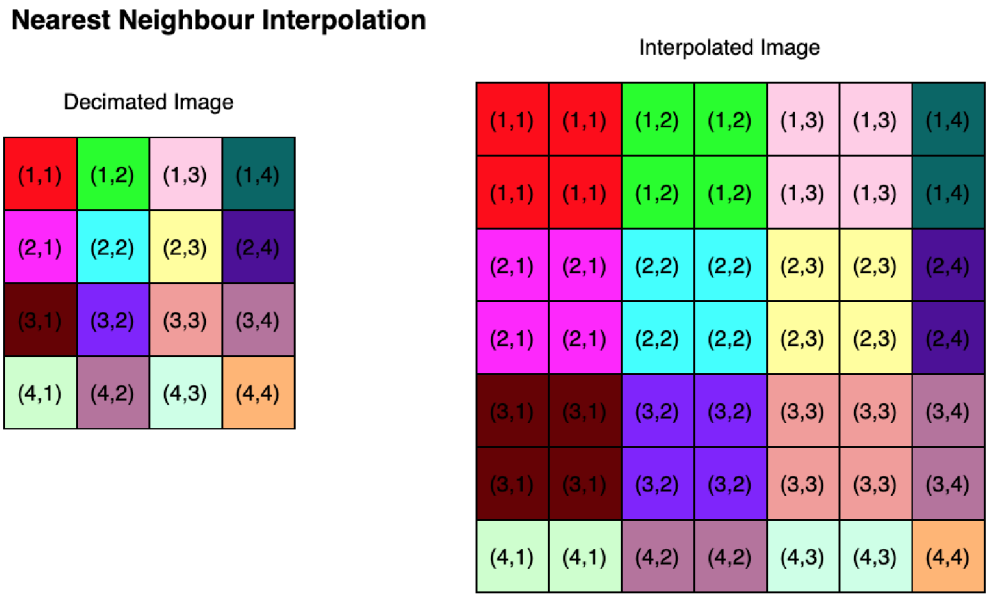



Figure 3.2: Resizing Image with Nearest Neighbor Interpolation Application

### 3.1.1.2 Bilinear Interpolation

The bilinear interpolation algorithm works by taking a weighted average of the four nearest pixels to determine the color value of a new pixel in the scaled image. This algorithm provides smoother and more natural-looking images than the nearest neighbor algorithm, making it a popular choice for many applications. This method is useful for upscaling an image, but it's not recommended for downscaling an image.[15].

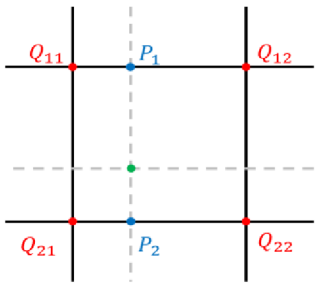



Figure 3.3: Downscaling in image pixels

In Figure 3.3, this interpolation involves the 4 neighboring points. Linear interpolation is done on points the top 2 points (Q11 and Q12) and the bottom 2 points (Q21 and Q22) to obtain two new points (P1 and P2). Then, linear interpolation is applied to the new points to get the interpolated point P [15].

### 3.1.2 Grayscale Conversion

Grayscale conversion is a useful process of converting colored images into black and white, as it looks in Figure 3.4, which has various benefits in image processing and machine learning algorithms. Grayscale images require less memory and processing power, have a smaller file size, and are easier to analyze. There are various methods of converting an image to grayscale, including the luminosity, average, and lightness methods. The choice of the conversion technique depends on the specific application and desired grayscale image quality [4].

Figure 3.4: Grayscale implementation on the image

### 3.1.3 Image Denoising

The images that are captured in the real world come with noises. These noises can appear due to many reasons such as electric signal instabilities, malfunctioning of camera sensors, poor lighting conditions, errors in data transmission over long distances, etc. This can degrade the captured image's quality and can cause loss of information as the original pixel values are replaced by random values due to noise. So, there is a need to remove these noises from images when it comes to low-level vision tasks and image processing. The process of removing such noises from images is known as Image Denoising. The image on the right side is the denoised version of the left one in Figure 3.5 [4].

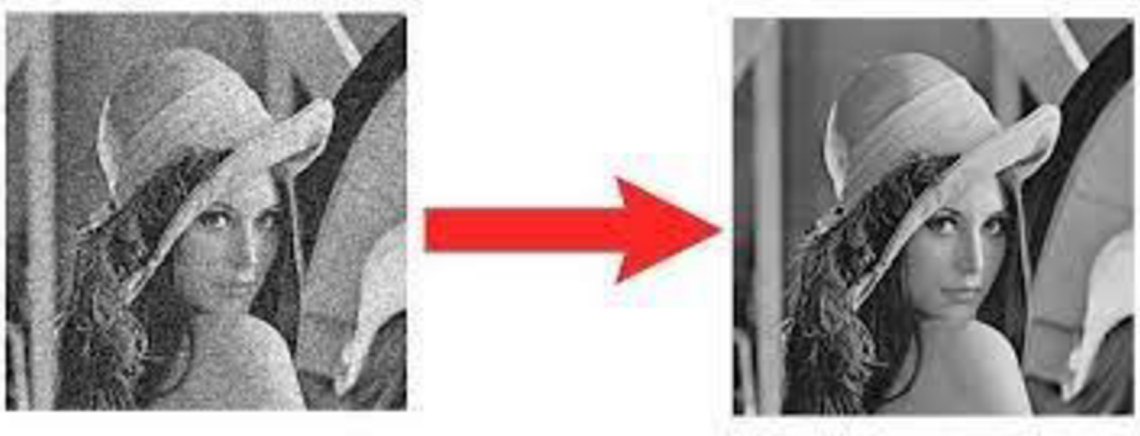


Figure 3.5: Image denoising

### 3.1.4 Image Segmentation

Image segmentation is a crucial process in which a digital image is divided into smaller parts or subgroups called image segments. These segments can help in reducing the complexity of the image, making further processing or analysis of the image simpler. Segmentation is the process of assigning labels to pixels, wherein all the pixels belonging to the same category have a common label assigned to them [4].

Image segmentation has several advantages in image processing and computer vision applications. For instance, in object detection, the detector can be inputted with a region selected by a segmentation algorithm, rather than processing the whole image. This reduces the inference time and enhances the accuracy of object detection. Moreover, segmentation can help in identifying objects, detecting edges, and reducing noise in images [4].

There are several techniques used for image segmentation, including thresholding, region growing, edge detection, and clustering. The choice of technique depends on the specific application and the characteristics of the image being processed [4].

#### 3.1.4.1 Instance Segmentation

Instance segmentation is a more advanced form of image segmentation that not only divides the image into segments, but also identifies individual objects within each segment. In instance segmentation, each pixel in the image is assigned a unique label or identifier that represents a specific object in the image. This technique is useful for applications that require the detection and segmentation of multiple objects in an image, such as autonomous vehicles and robotics (see Figure 3.6). Instance segmentation is typically performed using deep learning algorithms, which can achieve high accuracy in identifying and segmenting objects

in images. The performance of instance segmentation depends on the quality of the training data and the architecture of the deep learning model. Overall, instance segmentation is a powerful technique that can provide detailed and accurate information about the objects in an image [4].

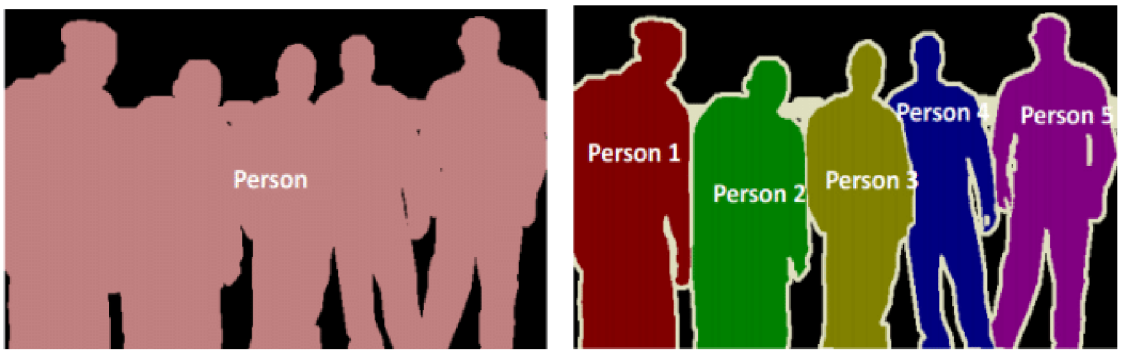



Figure 3.6: Example of Instance Segmentation [19]

### 3.1.4.2 Semantic Segmentation

Semantic segmentation is a technique in computer vision that involves dividing an image into multiple segments or regions, where each segment represents a different object or part of the image. However, unlike instance segmentation, semantic segmentation does not differentiate between individual instances of the same object type. Instead, it classifies each pixel in the image into a specific class, such as a person, car, or building. This technique is useful for many applications, including object detection, autonomous driving, and medical image analysis. Semantic segmentation is typically performed using deep learning algorithms that are trained on large datasets to accurately classify the pixels in the image. The accuracy of the semantic segmentation model depends on the quality of the training data, the architecture of the deep learning model, and the complexity of the image being segmented. In summary, semantic segmentation is a powerful technique that can provide rich information about the different objects in an image, facilitating further analysis and processing [1][19].

Figure 3.7: Example of Semantic Segmentation [19]

## 3.2 Data Annotation

Data annotation is the process of adding labels or tags to a data set in order to make it more usable and interpretable. The purpose of this process is to provide additional information to the data, making it easier for algorithms and models to understand and use the data in a meaningful way. It is a crucial step in the development of machine learning models, particularly in the field of computer vision. In computer vision, data annotation is used to train algorithms to recognize objects, people, and other elements within images and videos [15].

Data annotation is one of the top limitations of AI implementation for organizations. It is basically the process of labeling data with relevant tags to make it easier for computers to understand and interpret. This data can be in the form of images, text, audio, or video, and data annotators need to label it as accurately as possible. Data annotation can be done manually by a human or automatically using advanced machine learning algorithms and tools [15].

For supervised machine learning (see 3.6.1. Supervised Learning), labeled datasets are crucial because ML models need to understand input patterns to process them and produce accurate results. Supervised ML models (see Figure 3.8) train and learn from correctly annotated data and solve problems such as classification or regression (see 3.6. Machine Learning) [15].

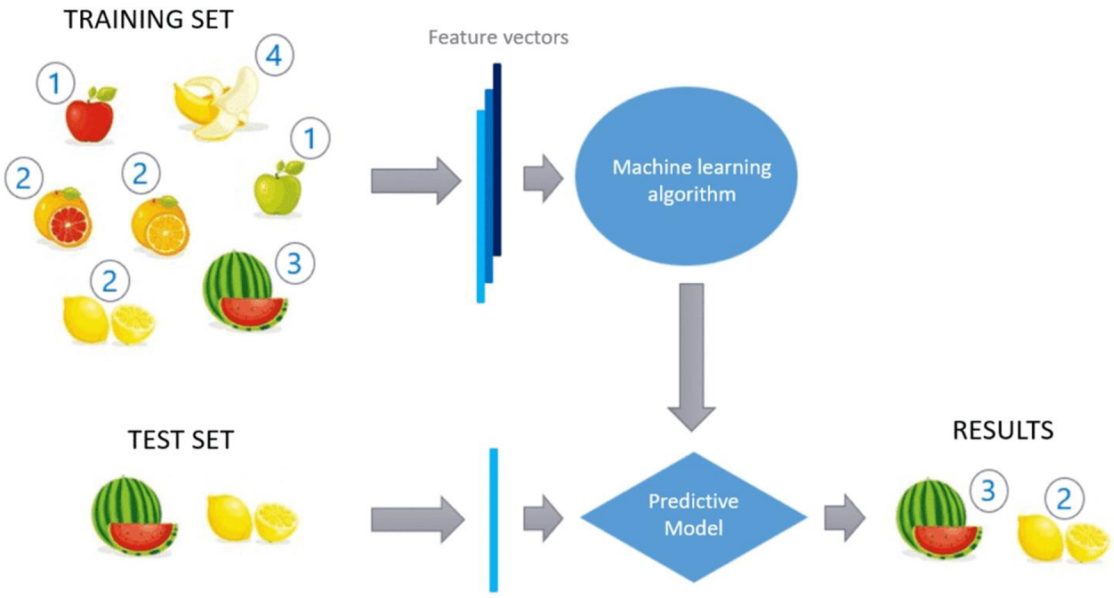



Figure 3.8: Supervised learning example based on data annotation of objects.

### 3.2.1 Image Annotation/Labeling

Image annotation or image labeling is the process of adding labels or tags to an image in order to provide additional information about the objects or elements within the image. Image annotation is a critical step in the development of computer vision algorithms and models, as it helps the algorithms to understand and recognize objects within images. Despite its importance, image annotation can also be a challenging task, especially on spending time to generate the labeled dataset. However, by using image annotation tools such as LabelImg, which was used on this thesis project, these challenges can be overcome, leading to the creation of large, high-quality data sets for training computer vision algorithms and models. There are several types of image annotation, each with its own specific purpose. These types include object detection, semantic segmentation, instance segmentation, key point annotation, bounding box annotation (see Figure 3.9) [15].

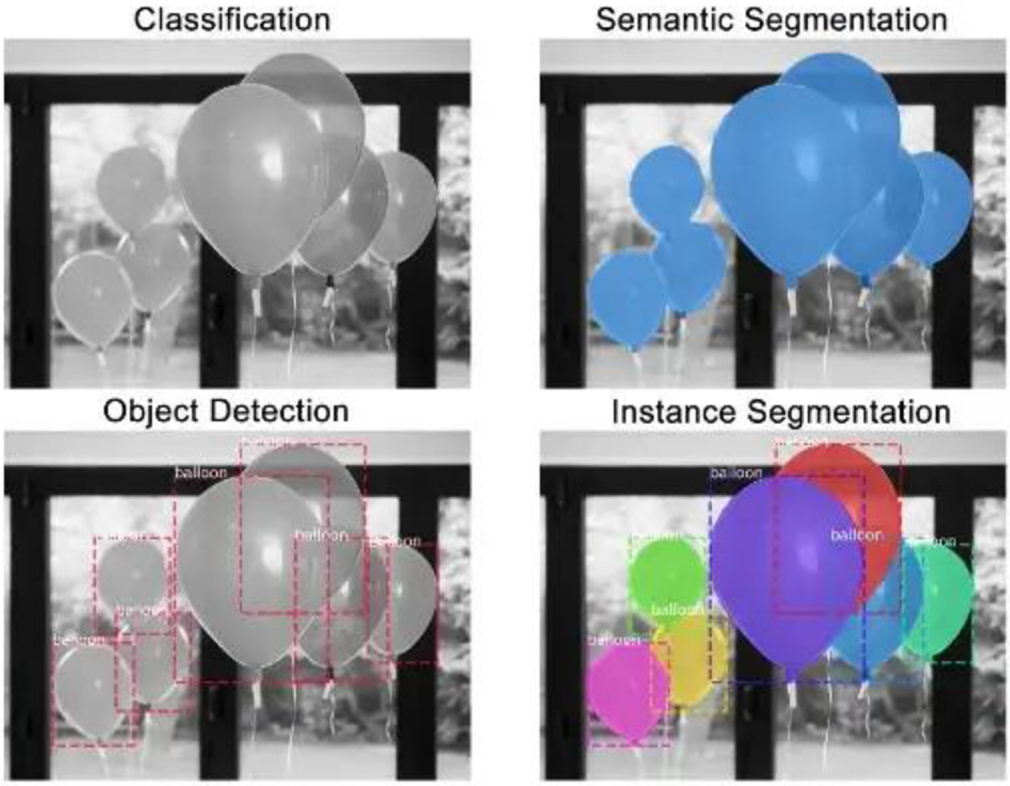



Figure 3.9: Image annotation types

#### 3.2.1.1 Image Classification

Image classification is the process of assigning an image to one of a set of predefined categories or classes. This type of annotation involves labeling an image with a class label, such as "apple" or "banana". The resulting annotations are used to train algorithms to classify new images into the appropriate class [15].

Image classification is a fundamental task in computer vision and has many real-world applications, including image search, content-based retrieval, and object recognition [15].

There are several approaches to image classification, including traditional machine learning algorithms, such as support vector machines (SVMs) and decision trees, as well as deep learning approaches, such as convolutional neural networks (CNNs). The choice of approach will depend on the specific requirements of the application and the available data. Image classification is a supervised learning task, meaning that the algorithm is trained on a labeled data set. The quality of the annotations used to train the algorithm has a direct impact on the performance of the model, so it is important to ensure that the annotations are accurate and complete [15].

#### 3.2.1.2 Object Recognition/Detection

Object recognition/detection is a further version of image classification. It is the correct description of the numbers and exact positions of entities in the image. While a label is assigned to the entire image in image classification, object recognition labels entities separately. As an example, with image classification, the image is labeled as day or night. Object recognition individually tags various entities in an image, such as a “bicycle”, “tree”, or “table” [15].

#### 3.2.1.3 Image Segmentation

Image segmentation for data annotation plays a crucial role in creating ground truth annotations for machine learning algorithms. Ground truth annotations are labeled data used to train and evaluate machine learning models. Image segmentation is used to divide an image into multiple segments or regions, each of which corresponds to a different object or part of the image. This allows annotators to label each segment with a class label, such as "person," "car," or "background." [15].

There are various image segmentation techniques that can be used for data annotation, including semantic segmentation, instance segmentation, and boundary-based segmentation. Each technique has its own advantages and limitations, and the choice of technique will depend on the specific requirements of the data annotation part of the projects [15].

## 3.3 Data Augmentation

Data augmentation is a technique used to increase the diversity of a dataset by making minor alterations to the existing data, without collecting new data. This technique is used to overcome the limitations of small datasets and to prevent a neural network from overfitting to the training data. It is an important technique for improving the performance of neural networks and is widely used in various machine learning applications [15].

Standard data augmentation techniques include horizontal and vertical flipping, rotation, cropping, shearing, and more. These techniques help in diversifying the training data, allowing the neural network to learn features that are invariant to small transformations. This results in a more robust and generalizable model [15].

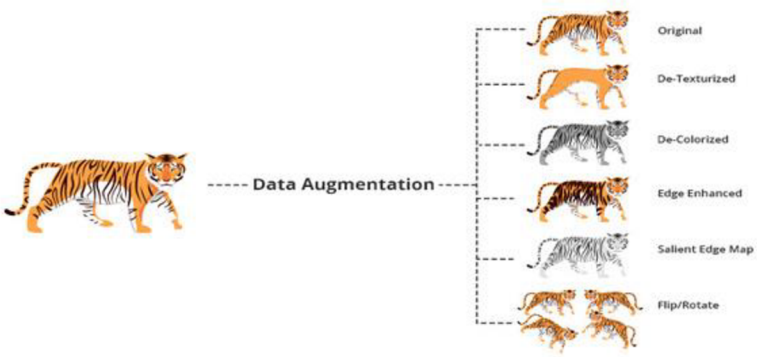



Figure 3.10: Data augmentation implementations

### 3.3.1 Types of Data Augmentation

There are two types of data augmentation which are offline augmentation and online augmentation [15].

#### 3.3.1.1 Offline Augmentation

Offline augmentation is used for small datasets and is applied during the data preprocessing step. It involves creating augmented versions of the data before training the model.

#### 3.3.1.2 Online Augmentation

Online augmentation is used for large datasets and is applied in real-time during the training process. It involves generating augmented versions of the data on-the-fly, allowing for a larger and more diverse training dataset without the need for additional data collection.

### 3.3.2 Data Augmentation Methods

Data augmentation methods are commonly used to increase the diversity of the training data and to prevent overfitting. By generating new samples that differ from the original data in various ways, these methods help to improve the robustness and generalization of machine learning models [15].

#### 3.3.2.1 Shifting

Shifting involves translating the image along the x and y axes. This can be done by moving the image up, down, left, or right by a specified number of pixels. Shifting helps to increase the diversity of the training data by generating new samples that differ from the

original data in their spatial location. See Figure 3.11 as an applied shifting method example on an image [15].

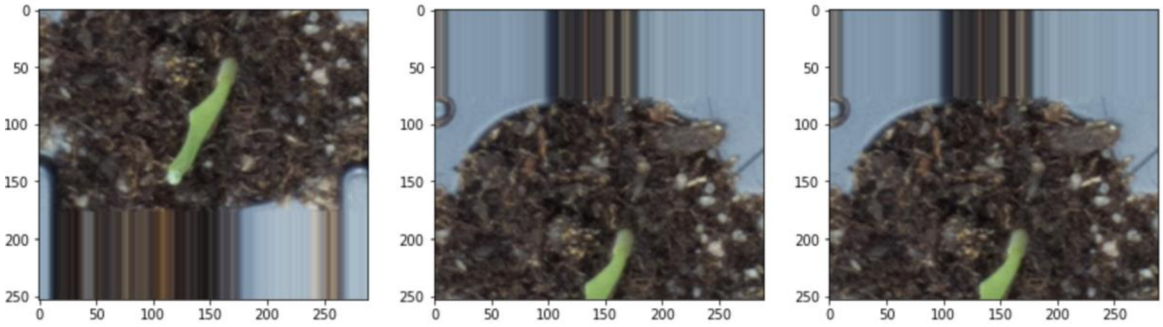


Figure 3.11: Shifting method in image

#### 3.3.2.2 Flipping

This reverses the rows or columns of pixels in either vertical or horizontal cases, respectively. See Image 3.12 as an applied flipping method example on an image [15].

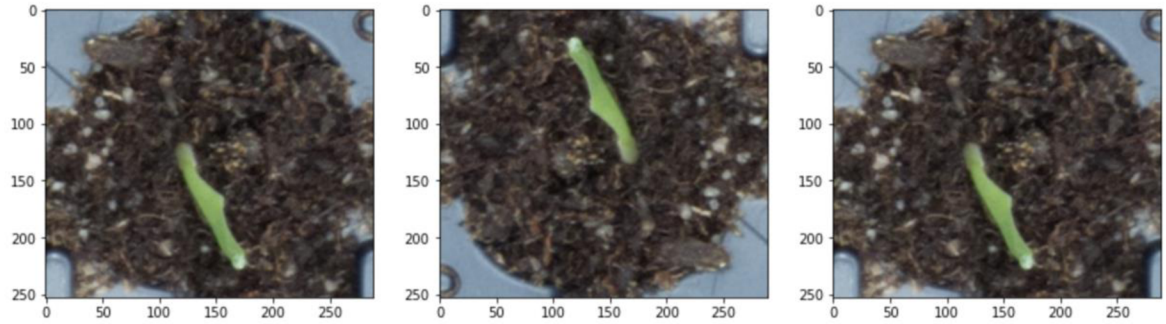


Figure 3.12: Flipping the image.

#### 3.3.2.3 Rotation

Rotation involves rotating the image by a specified angle. This can be done by rotating the image clockwise or counterclockwise. Rotation helps to increase the diversity of the training data by generating new samples that differ from the original data in their orientation. See Image 3.13 as an applied rotation method example on an image [15].

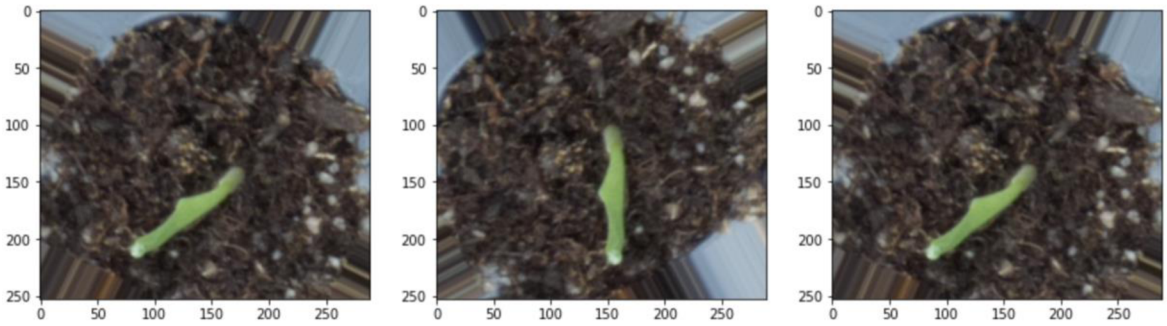


Figure 3.13: Rotation on the image

#### 3.3.2.4 Changing Brightness

Changing the brightness involves altering the brightness of the image. This can be done by increasing or decreasing the brightness of the image. Changing the brightness helps to increase the diversity of the training data by generating new samples that differ from the original data in their lighting conditions. See Figure 3.14 as an applied brightness changing method example on an image [15].

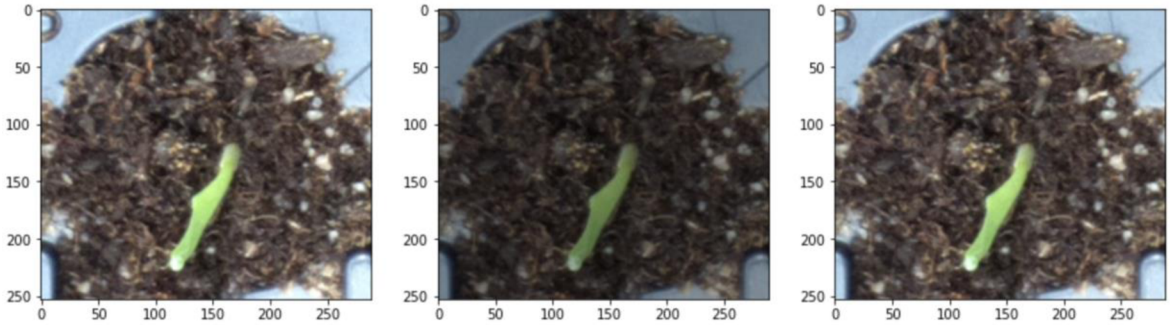


Figure 3.14: Changing the brightness of the image.

## 3.4 Computer Vision

Computer Vision is a field of study that focuses on enabling computers to interpret and understand visual data from the world, such as images and videos. The goal of computer vision is to develop algorithms and models that can automatically extract meaningful information from visual data, such as recognizing objects and scenes, detecting edges and corners, and estimating depth and motion [1].

Computer vision has a wide range of applications, including image and video analysis, object recognition and tracking, and autonomous systems [1]. With the increasing availability of high-quality visual data, computer vision has become an important area of research, with many advances being made in recent years [1].

Deep learning has been a major driver of progress in computer vision in recent years, with convolutional neural networks (ConvNets) being one of the most popular and successful deep learning models for computer vision. ConvNets are designed to process image data, and they have been shown to outperform traditional computer vision methods in tasks such as image classification, object detection, and semantic segmentation [2].

### 3.4.1 What is Artificial Intelligence?

Artificial Intelligence (AI) is the field of computer science that focuses on the development of intelligent machines that can perform tasks that typically require human intelligence, such as visual perception, speech recognition, decision-making, and language translation. The goal of AI is to create systems that can perform tasks that are typically performed by humans, without being explicitly programmed to do so [1].

AI has a wide range of applications, including natural language processing, computer vision, robotics, and game playing. It has the potential to revolutionize many industries, including healthcare, finance, and transportation, by automating tasks that were previously performed by humans, improving efficiency and accuracy [2].

There are two main approaches to AI: rule-based systems and machine learning. Rule-based systems are systems that are explicitly programmed with a set of rules to perform a specific task, while machine learning is a type of AI that enables systems to automatically learn from data, without being explicitly programmed [1].

Machine learning has become an important area of AI research in recent years, with advances in deep learning leading to significant improvements in performance in many tasks, such as image classification and natural language processing. Deep learning is a type of machine learning that uses deep neural networks, which are networks with many layers, to learn from data [3].

## 3.5 Neural Networks

Neural networks are computational models inspired by the structure and function of the human brain. They aim to replicate the perception, learning, and memory capabilities of human neurons by performing these functions mathematically on computers. The goal of neural networks is to recognize patterns, make predictions, and perform other tasks that would normally require human intelligence [16].

Figure 3.15.(b) shows a simple representation of a neural network, modeled after a human neuron (see Figure 3.15.(a)). The input layer receives data, the hidden layer processes the data, and the output layer produces the result. In Figure 3.16, a neural network architecture takes the input from input layer and calculates the result in the exit of layer. Neural networks can be trained using various algorithms, such as supervised learning, unsupervised learning, or reinforcement learning, to learn from data and improve their performance over time [16].

Neural networks are powerful tools for solving complex problems in fields such as computer vision, natural language processing, and robotics. They can learn from data and improve their performance, making them promising technology for the future [16].

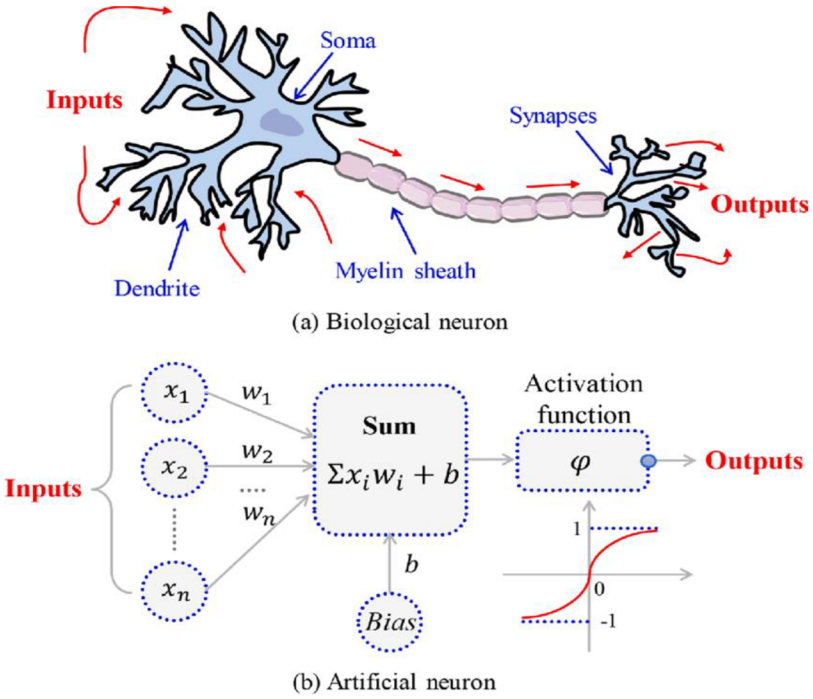



Figure 3.15: *(a)* Biological neuron from human brain,
*(b)* Artificial neuron that is inspired by biological neuron

A neural network consists of three parts: input, calculation, and output layers. The input layer's input values, *x1*, *x2*, .... *xn*, and weights, *w1*, *w2*, ... *wn*, are shown (Equation 3.1). These values are defined as real numbers. Figure 4.2 shows an artificial neural network model [16].

The artificial neural network produces an output consisting of zeros and ones. In Equation 4.1, when the threshold value is added to the product of the input layer and weights of the neural network, if the result is greater than the threshold value, the neural network will

output 1, otherwise 0. Equation 3.1 shows the function that produces the output and (*y*) is the output, (*Xj*) is the input layer, (*Wj*) are the weights, and (*b*) is the bias value [16].

$$x = \sum_j w_j x_j + b$$

$$y = \begin{cases} 1, & x \geq 0 \\ 0, & x < 0 \end{cases}$$

*Equation 3.1:* Neuron Output Calculation

*(Xn):Input, (n):Index, (w):Weight, (b):Bias, (y):Output*

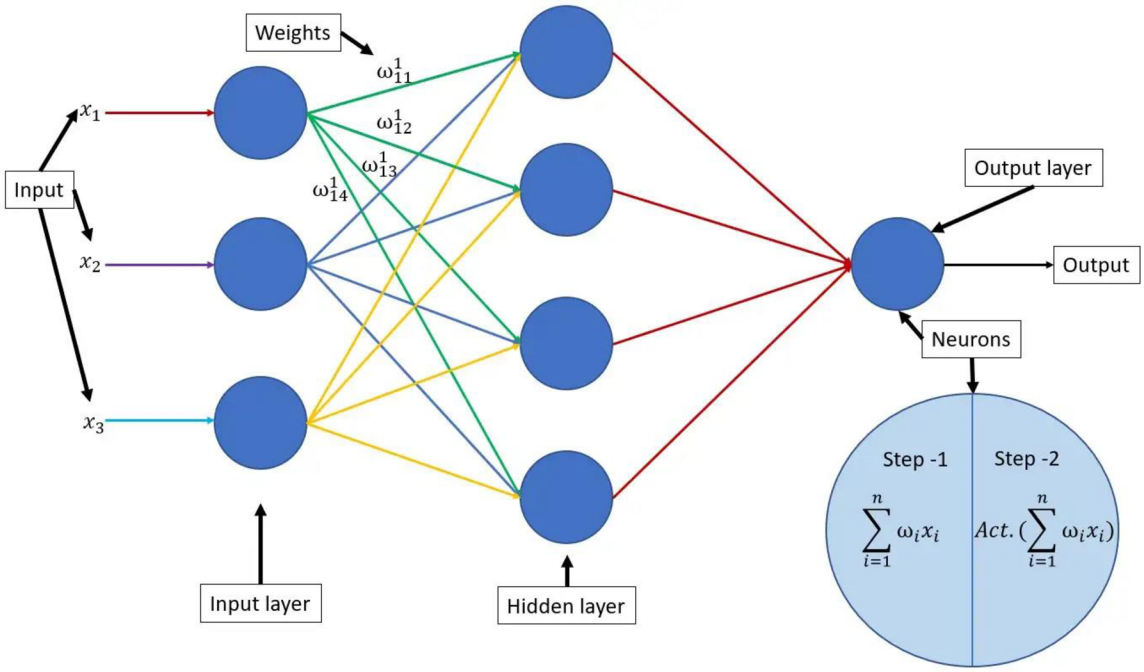



Figure 3.16: Neural network architecture

### 3.5.1 Cost Functions

A cost function, also known as a loss function, is a measure of the difference between the predicted output of a neural network and the true output labels. The goal of the training process is to minimize the cost function so that the neural network can make accurate predictions on new data.

There are many different types of cost functions, including mean squared error (MSE), cross-entropy loss, and hinge loss [16]. The choice of cost function depends on the specific problem and the type of neural network being used.

One common cost function used in neural networks is mean squared error (MSE), which measures the average of the squared differences between the predicted and true output values [16]. The MSE cost function can be written as *Equation 3.2*.

$$J(w,b) = \frac{1}{2m}\sum_{i=1}^{m}\left(y^{(i)} - \widehat{y^{(i)}}\right)^2$$

*Equation 3.2:* Mean Squared Error (MSE) Cost Function

In *Equation 3.2*, J(w, b) is the cost function, $m$ is the number of training examples, $y^{(i)}$ is the true label, and $\widehat{y^{(i)}}$ is the predicted output for the $i^{th}$ training example. The weights ($w$) and biases ($b$) are the parameters that are being optimized during the training process (see Figure 3.16) [16].

Another popular cost function used in neural networks is cross-entropy loss, which measures the difference between the predicted probabilities and the true labels. The cross-entropy loss function can be written as *Equation 3.3*.

$$J(w,b) = -\frac{1}{m}\sum_{i=1}^{m}\left[y^{(i)}\log\left(\widehat{y^{(i)}}\right) + \left(1 - y^{(i)}\right)\log\left(1 - \widehat{y^{(i)}}\right)\right]$$

*Equation 3.3:* Cross-Entropy Loss Function

In Equation 3.3, $J(w, b)$ is the cost function, $m$ is the number of training examples, $y^{(i)}$ is the true label, and $\widehat{y^{(i)}}$ is the predicted probability for the $i^{th}$ training example [16].

These are just two examples of cost functions used in neural networks, and there are many others to choose from depending on the specific problem and type of neural network. The goal of the cost function is to provide a measure of how well the neural network is performing, and the training process aims to minimize this measure [16].

### 3.5.2 Activation Functions

Activation functions are an essential component of artificial neural networks. They are mathematical equations that determine the output of a neuron based on the input it receives. The activation functions play a crucial role in training the network, and the choice of activation function has a significant impact on the performance of the network (see Figure 3.17).

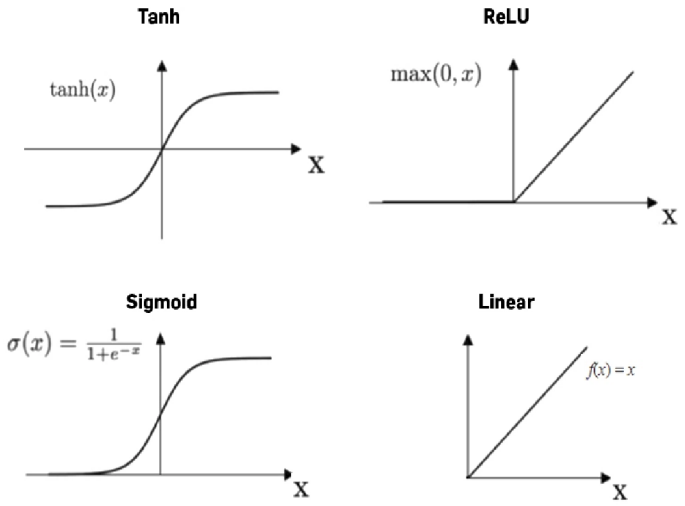


Figure 3.17: Activation Functions

#### 3.5.2.1 Sigmoid Function

The sigmoid function is one of the most used activation functions in neural networks. It is a mathematical function that maps any input to the range of 0 and 1, making it suitable for binary classification problems. The sigmoid function is defined as:

$$\boldsymbol{f(x) = \frac{1}{1 + e^{-x}}}$$

*Equation 3.4:* Sigmoid Activation Function

In Equation 3.4, x is the input to the neuron and f(x) is the output. The sigmoid function is useful because it has a smooth, monotonic increase and is easy to differentiate, making it ideal for backpropagation, a common algorithm used to train neural networks [1].

#### 3.5.2.2 Rectified Linear Unit (ReLu)

The rectified linear unit (ReLU) is a simple activation function that has become increasingly popular in recent years. It is computationally efficient, only requiring a simple threshold operation, and has been shown to prevent vanishing gradient problems, leading to improved performance in deep learning models, particularly in computer vision tasks.

$$\boldsymbol{f(x) = \max(0, x)}$$

*Equation 3.5:* ReLu Activation Function

In Equation 3.5, x is the input to the neuron and f(x) is the output. The ReLU activation function has been shown to outperform other activation functions in deep learning models, particularly in computer vision tasks. The ReLU function is computationally efficient, as it only requires a simple threshold operation, and it has been found to prevent vanishing gradient problems, which can occur with other activation functions [16].

#### 3.5.2.3 Tanh Function

The hyperbolic tangent (tanh) function is another activation function commonly used in neural networks. It maps any input to the range of -1 and 1, making it suitable for outputs that are not binary. The tanh function is defined as:

$$\boldsymbol{f(x) = tanh(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}}$$

Equation 3.6: Tanh Activation Function

In Equation 3.6, x is the input to the neuron and $f(x)$ is the output. The tanh function is like the sigmoid function, but it has a slightly faster convergence rate, making it ideal for time series prediction and other applications where speed is important [16].

#### 3.5.2.4 Linear Function

The linear activation function is a simple activation function that outputs the input to a neuron without any transformation. It is defined as:

$$\boldsymbol{f(x) = x}$$

*Equation 3.7:* Linear Activation Function

In Equation 3.7, x is the input to the neuron and f(x) is the output. The linear activation function is often used in regression problems where the output is a continuous value. However, it is not commonly used in deep learning models because it can lead to vanishing or exploding gradients, which can make it difficult to train the network effectively. It is also limited in its ability to introduce non-linearity into the network, which is necessary for solving more complex problems [16].

### 3.5.3 Forward Propagation

Forward propagation is the process of computing the predicted output of a neural network given the input and the weights and biases. During forward propagation, the input values are passed through the network layer by layer, and intermediate results are computed using the activation function. The result is produced by the output layer, which is then compared to the true output labels to calculate the cost using a cost function [16].

The forward propagation calculation for a single training example can be represented mathematically as:

$$\boldsymbol{z}^{[l]} = \boldsymbol{W}^{[l]}\boldsymbol{a}^{[l-1]} + \boldsymbol{b}^{[l]}$$

$$\boldsymbol{a}^{[l]} = \boldsymbol{g}^{[l]}\left(\boldsymbol{z}^{[l]}\right)$$

*Equation 3.8:* Forward propagation calculation

Equation 3.8, $z^{[l]}$ is the linear combination of the activations from the previous layer ($a^{[l-1]}$) and the weights ($W^{[l]}$) for the current layer, plus the bias ($b^{[l]}$) . The activation function $g^{[l]}$ is applied to $z^{[l]}$ to produce the activations for the current layer ($a^{[l]}$) [16].

### 3.5.4 Backward Propagation (Backpropagation)

Backward propagation, also known as backpropagation, is the process of updating the weights and biases in a neural network to minimize the cost function. During backward propagation, the gradient of the cost with respect to the weights and biases is computed, and the weights and biases are updated using gradient descent or a similar optimization algorithm [16].

Backward propagation calculation starts at the output layer and works backwards through the network to calculate the gradients for each layer. The gradient of the cost with respect to the weights and biases can be represented mathematically as:

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{W}^{[l]}} = \frac{\boldsymbol{1}}{\boldsymbol{m}}\boldsymbol{\Delta}^{[l]}\left(\boldsymbol{a}^{[l-1]}\right)^{\boldsymbol{T}}$$

$$\frac{\partial \boldsymbol{J}}{\partial \boldsymbol{b}^{[l]}} = \frac{\boldsymbol{1}}{\boldsymbol{m}}\sum_{\boldsymbol{i=1}}^{\boldsymbol{m}}\boldsymbol{\Delta}^{[l]}$$

*Equation 3.9:* Backpropagation calculation

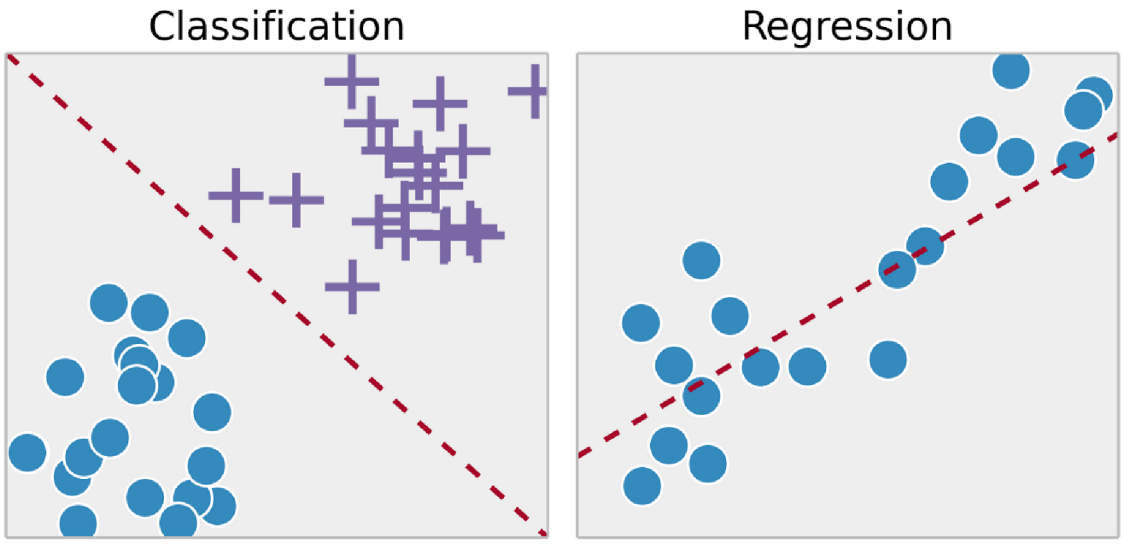
Equation 3.9, where $J$ is the cost function, $m$ is the number of training examples, $\Delta^{[l]}$ is the error for the current layer, and $a^{[l-1]}$ is the activations from the previous layer [16].
Forward and backward propagation are the two main steps in the training process of a neural network, and they are repeated multiple times to minimize the cost function and improve the performance of the network [16].

## 3.6 Machine Learning

Machine Learning (ML) is a field of study in computer science and artificial intelligence that focuses on the development of algorithms and models that are capable of learning from data to improve their performance on a specific task. ML algorithms can build models based on training data, which are used to make predictions or decisions without explicit programming. ML is used in a wide range of applications, such as in medicine, email filtering, speech recognition, agriculture, and computer vision, where conventional algorithms may be difficult or impractical to develop.
The first step in solving a problem using ML is choosing the appropriate model. Depending on the problem at hand, there are two main categories of ML techniques: supervised learning and unsupervised learning [16][17].

### 3.6.1 Supervised Learning

Supervised learning is a type of machine learning where the algorithms are trained on a labeled dataset, with the goal of learning a mapping between input variables (features) and output variables (labels). In supervised learning, the algorithm is given a set of labeled examples and attempts to learn the relationship between the input and output variables, so that it can make predictions for new, unseen examples [17].




(a) (b)

Figure 3.18: Supervised Learning, (a): Classification, (b): Regression

#### 3.6.1.1 Classification

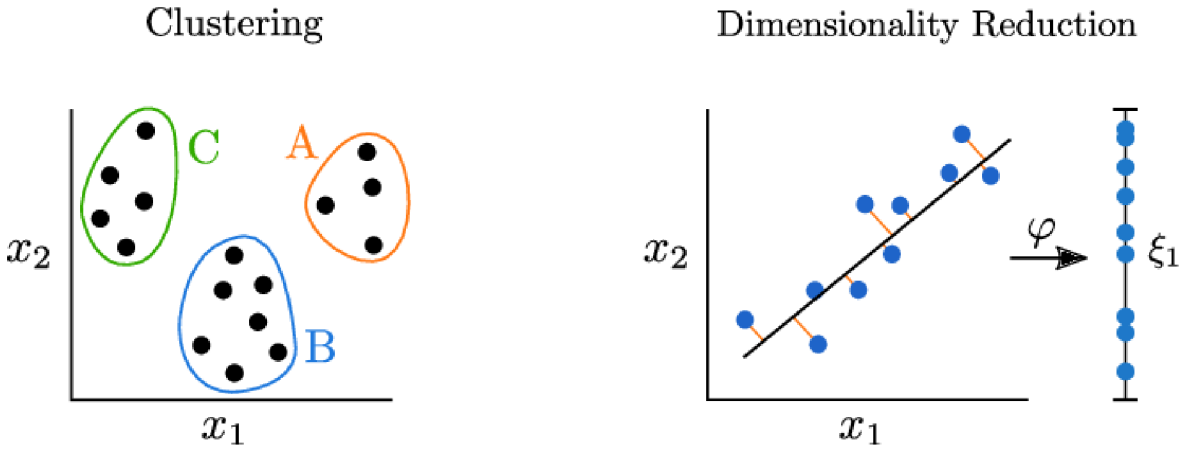
Classification predicts a categorical label for a given input. For example, classifying an email as spam or not spam, or classifying a type of animal in an image as a dog, cat, or horse [17].

#### 3.6.1.2 Regression

The goal of the regression is to predict a continuous output value for a given input. For example, predicting the price of a house given its size, location, and number of rooms [17].

### 3.6.2 Unsupervised Learning

Unsupervised learning is a type of ML where the algorithms are trained on an unlabeled dataset, with the goal of finding patterns or structure in the data without any prior knowledge. In unsupervised learning, the algorithm tries to find relationships or patterns in the data by grouping similar data points together or finding the underlying structure of the data=1 [17].




(a) (b)

Figure 3.19: Supervised Learning, (a): Clustering, (b): Dimensionality Reduction

#### 3.6.2.1 Clustering

Clustering is a type of unsupervised learning where the goal is to group similar data points together into clusters (see Figure 3.19.(a)). For example, grouping customers based on their spending habits or grouping images based on the type of object they contain [17].

#### 3.6.2.2 Dimensionality Reduction

Dimensionality reduction is a type of unsupervised learning where the goal is to reduce the number of features in the data while preserving as much of the information as possible (see Figure 3.19.(b)). For example, reducing the number of features in an image dataset from 1000 to 10 while still being able to accurately classify the images [17].

## 3.7 Deep Learning

Deep learning is a machine learning technique that uses artificial neural networks with multiple layers to automatically learn and represent complex patterns and relationships in large-scale data. Unlike traditional machine learning methods that rely on manual feature extraction and selection, deep learning algorithms learn hierarchical representations of data from raw inputs and iteratively improve their performance through backpropagation and optimization techniques. Deep learning has been successfully applied to various domains, including computer vision, natural language processing, speech recognition, and recommendation systems. Convolutional Neural Networks (CNNs) have been particularly effective for image and video analysis tasks, while Recurrent Neural Networks (RNNs) and

Long Short-Term Memory (LSTM) networks have shown great promise for sequence modeling and time series analysis [16][18].

Recent developments in deep learning have also led to the development of new architectures and models, such as Generative Adversarial Networks (GANs), Variational Autoencoders (VAEs), and Transformer networks, which have achieved state-of-the-art results in image synthesis, speech recognition, and natural language processing tasks [16][18].

The increasing popularity and success of deep learning have also sparked significant research and development efforts in hardware and software, such as specialized processors and libraries for deep learning. As a result, deep learning has become an essential tool for many industries and applications, including healthcare, finance, manufacturing, and transportation [16][18].

One of the recent applications of deep learning is in the field of agriculture, where it is being used for the detection of the first appearance of the plant on the soil using temporal image classification methods. This involves the use of time-lapse images of the soil, captured at regular intervals, to monitor the growth of the seed and detect its first appearance in the soil. Deep learning algorithms, such as CNNs and LSTMs, are used to extract features from the images and classify them based on the growth stage of the seed. This approach has the potential to improve the efficiency and accuracy of crop monitoring and management, leading to higher yields and reduced resource consumption [20].

### 3.7.1 Convolutional Neural Networks (CNNs)

Convolutional Neural Networks (CNNs) are a class of deep learning models that have been particularly effective in solving image and video analysis tasks. They are designed to automatically learn and extract hierarchical features from raw image inputs using convolutional layers, pooling layers, and fully connected layers.

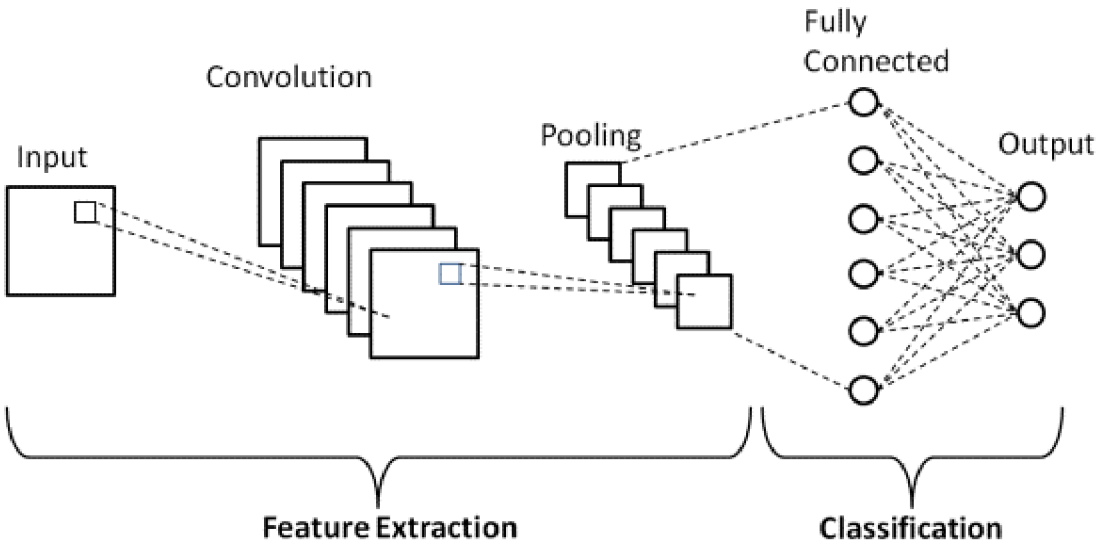



Figure 3.20: CNN architecture with layers

In a CNN, the convolutional layer applies a set of learnable filters to the input image to extract local features, such as edges, corners, and textures (see Figure 3.20). The pooling layer then reduces the dimensionality of the feature map by down-sampling the output of the convolutional layer. Finally, the fully connected layer aggregates the output of the previous layers to produce the final classification or regression result. The convolutional layer, pooling layer, and fully connected layer are the key components of a CNN. They enable the network to automatically learn and extract meaningful features from the input image, reduce the dimensionality of the feature maps, and produce the final classification or regression result [16].

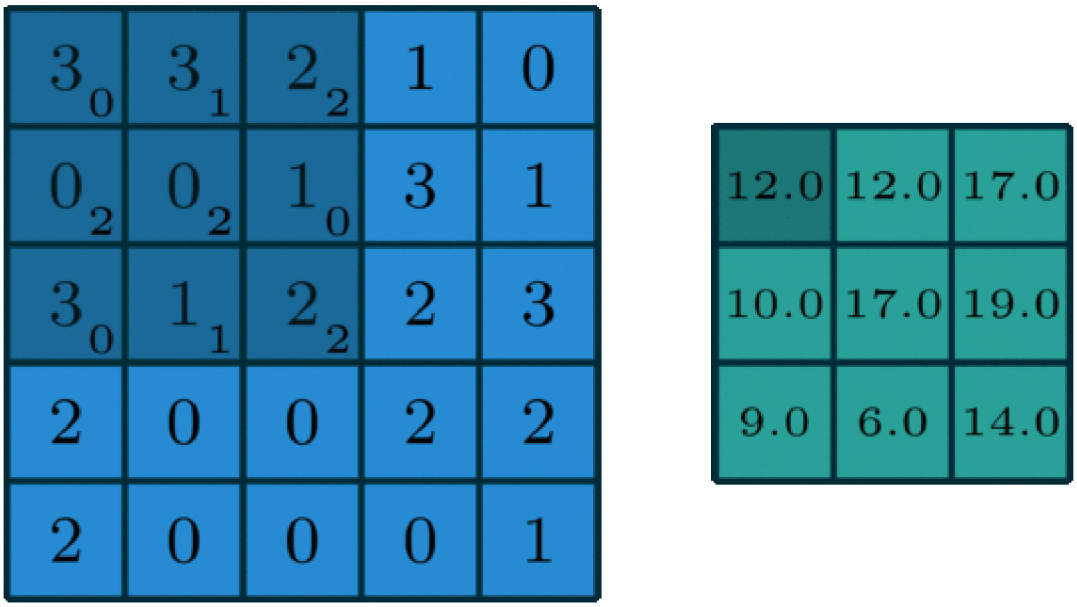
CNNs have achieved state-of-the-art results in various image and video analysis tasks, including object detection, image segmentation, and image classification. They have also been used in other domains, such as natural language processing and speech recognition.

Some of the most popular CNN architectures include AlexNet, VGGNet, GoogLeNet, and ResNet, which have won multiple competitions and achieved high accuracy on benchmark datasets such as ImageNet [16].

#### 3.7.1.1 Convolutional Layer

The convolutional layer is the core building block of a CNN. It applies a set of learnable filters to the input image to extract local features, such as edges, corners, and textures. The filters slide over the image and compute the dot product between their weights

and the input image pixels. This produces a set of feature maps that highlight the presence of the learned features in different parts of the image [16].




*(a)* *(b)*

Figure 3.21: Convolutional filter application,
*(a)* 5x5 grayscale image,
*(b)* Convolutional filter

See Figure 3.21, that shows an example of a 3x3 convolutional filter applied to a 5x5 grayscale. The output feature map size depends on the filter size, stride, and padding. The stride determines the amount of shift between the filters, while padding adds zeros around the image to preserve the output size. The number of filters is a hyperparameter that determines the depth of the output feature map [16].

#### 3.7.1.2 Pooling Layer

The pooling layer is usually applied after the convolutional layer to down-sample the output feature maps and reduce their dimensionality. This helps to increase computational efficiency and reduce the risk of overfitting. Common pooling operations include max pooling and average pooling, which select the maximum or average value of a sub-region of the feature map, respectively [16].

On Figure 3.11 below shows an example of max pooling and average pooling with a 2x2 window and stride of 2.

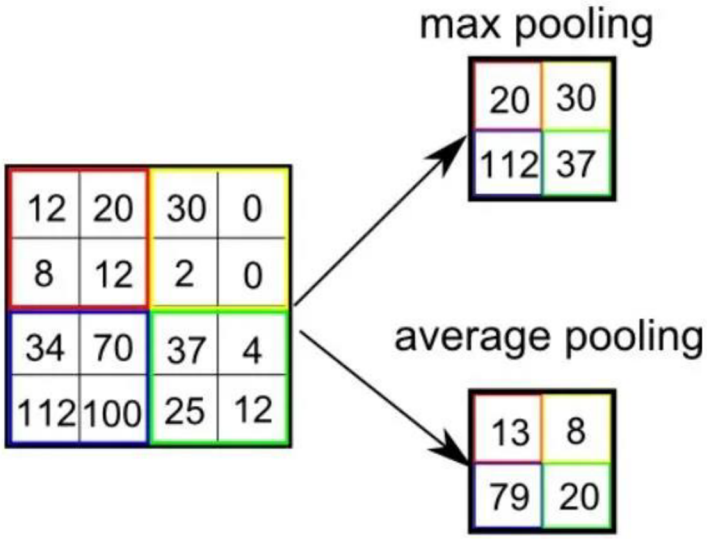



Figure 3.22: Types of pooling

The pooling operation reduces the output feature map size by a factor of the window size, while retaining the most salient features.

#### 3.7.1.3 Fully Connected Layer

The fully connected layer is the final layer in a CNN, which aggregates the output of the previous layers to produce the final classification or regression result. This layer is a neural network layer where all the inputs are connected to all the outputs, hence the name "fully connected". The output of the last pooling layer is flattened into a 1D vector and then connected to the input of the fully connected layer.

See Figure 3.23 below shows an example of a fully connected layer with 4 input neurons and 8 output neurons.

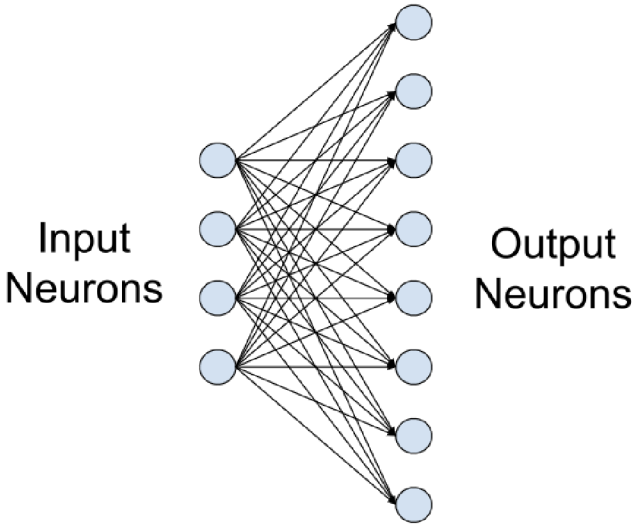



Figure 3.23: Fully Connected Layer

The fully connected layer performs a linear transformation on the input features and passes them through an activation function, such as ReLU or sigmoid. The output of the last fully connected layer is usually fed to a SoftMax function to produce the final probability distribution over the classes [16].

### 3.7.2 Recurrent Neural Network (RNN)

Recurrent Neural Networks (RNNs) are a type of neural network that can process sequential data, such as text, speech, and time series. RNNs have an internal memory that allows them to maintain a hidden state that captures the context of the previous inputs. The hidden state is updated at each time step by combining the current input with the previous hidden state. This makes RNNs capable of processing sequences of inputs and capturing temporal dependencies between them [16].

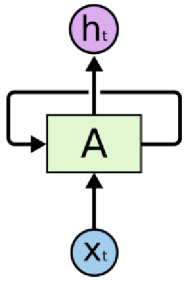



Figure 3.24: A loop in a recurrent neural network

One of the notable applications of RNNs is in natural language processing, where they have been used for tasks such as language modeling, machine translation, and sentiment analysis. RNNs have also been used for speech recognition and music generation [16].

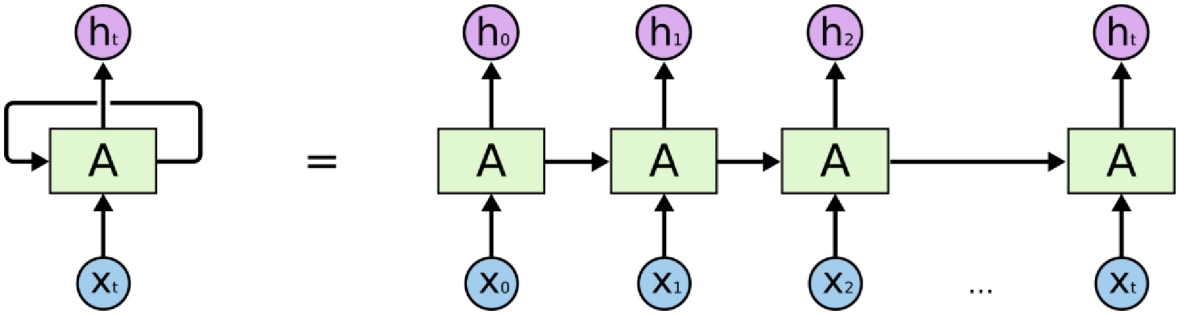


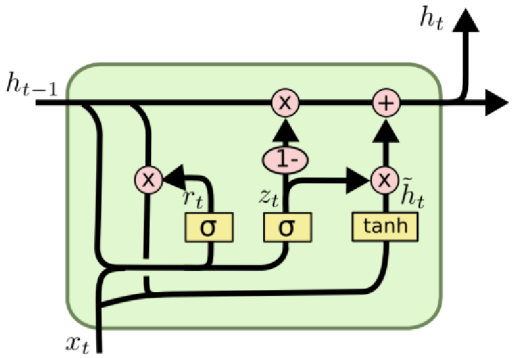

Figure 3.25: An unrolled recurrent neural network

However, RNNs suffer from the vanishing gradient problem, where the gradients become very small or zero during backpropagation through time. This limits the ability of RNNs to capture long-term dependencies in the input sequence. This problem has been addressed by the development of new architectures, such as Long Short-Term Memory (LSTM) networks, which introduce memory cells and gating mechanisms [16].

Research has shown that LSTM networks can achieve state-of-the-art results in various tasks, such as speech recognition, natural language processing, and image captioning. They have also been used in healthcare, where they have been applied to tasks such as predicting disease progression and clinical decision support [16].

### 3.7.3 Long Short-Term Memory (LSTM)

Long Short-Term Memory (LSTM) networks are a type of recurrent neural network that have memory cells and gating mechanisms to overcome the vanishing gradient problem. LSTMs are well-suited to process sequential data and capture long-term dependencies between them. They have been used in various applications, such as speech recognition, natural language processing, and image captioning [16].




$$z_t = \sigma\left(W_z \cdot [h_{t-1}, x_t]\right)$$
$$r_t = \sigma\left(W_r \cdot [h_{t-1}, x_t]\right)$$
$$\tilde{h}_t = \tanh\left(W \cdot [r_t * h_{t-1}, x_t]\right)$$
$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

Figure 3.26: LSTM

LSTM networks have three main components: memory cells, input gates, and output gates. The memory cells are responsible for storing and updating the relevant information from the input sequence, while the input and output gates control the flow of information into and out of the cells [16].

### 3.7.4 Temporal Convolutional Networks (TCN)

Temporal Convolutional Networks (TCN) are a type of deep learning architecture that has been introduced as a solution to the problem of processing sequential data. TCN operates on sequences of input data and utilizes convolutional layers with dilated convolutions to capture long-range temporal dependencies in the data. The use of dilated convolutions allows TCN to effectively capture both short- and long-range temporal dependencies, making it suitable for a wide range of sequential data processing tasks [21].
TCN has been shown to outperform traditional Recurrent Neural Networks (RNNs) in terms of both accuracy and computational efficiency for a variety of sequential data processing tasks, such as speech recognition, natural language processing, and video classification. This is due to the fact that TCN operates on the entire sequence of data, rather than processing the data one step at a time as in RNNs, and it has been shown to be more effective at capturing long-range temporal dependencies [21].
TCN is a promising deep learning architecture for the processing of sequential data, and it has been shown to outperform traditional RNNs in terms of both accuracy and computational efficiency. This makes it a promising solution for a wide range of sequential data processing tasks, including video classification, speech recognition, and natural language processing [21].

## 3.8 Deep Learning Models for Computer Vision

Deep Learning models are a type of Artificial Neural Network (ANN) that have been designed to learn from large amounts of data and to perform complex tasks such as image classification, object detection, and image segmentation. These models have revolutionized the field of computer vision and have been widely adopted in various applications such as self-driving cars, facial recognition, and medical imaging [16].

### 3.8.1 You Only Look Once (YOLO)

YOLO is a real-time object detection system that is designed to be fast and efficient. It operates by dividing an image into a grid of cells and predicting the presence of objects in each cell. The model then combines the predictions across the cells to produce a final bounding box and class prediction for each object in the image [18].

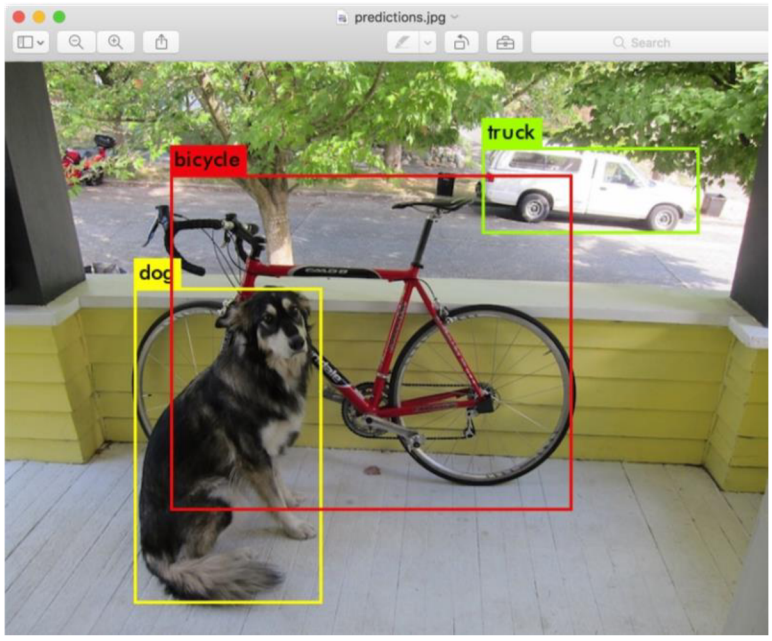



Figure 3.27: YOLO Sample

The YOLO model consists of two parts: a feature extractor and a detector. The feature extractor is a Deep Neural Network that takes the input image and extracts features from it. The detector is a Fully Connected Neural Network that takes the extracted features and predicts the bounding boxes and class probabilities for each object in the image [18].
The main advantage of YOLO is its speed and efficiency. It can process images in real-time, making it suitable for applications where fast object detection is required. Additionally,

YOLO has been designed to be highly accurate, making it a popular choice for various object detection tasks [18].

### 3.8.2 ResNet50

ResNet50 is a Convolutional Neural Network (CNN) designed for image classification tasks. The main goal of the ResNet50 model is to overcome the problem of vanishing gradients in deep neural networks. This problem occurs when the gradients become smaller as the model becomes deeper, making it difficult for the model to learn from the data [18].

The ResNet50 model uses a residual connection, which allows the model to effectively learn from the information in the earlier layers. The residual connection adds the input to the output of a layer, allowing the model to pass information through the network without losing information. This allows the ResNet50 model to effectively learn from the information in the earlier layers, even in deep neural networks, and improve the accuracy of the model [18].

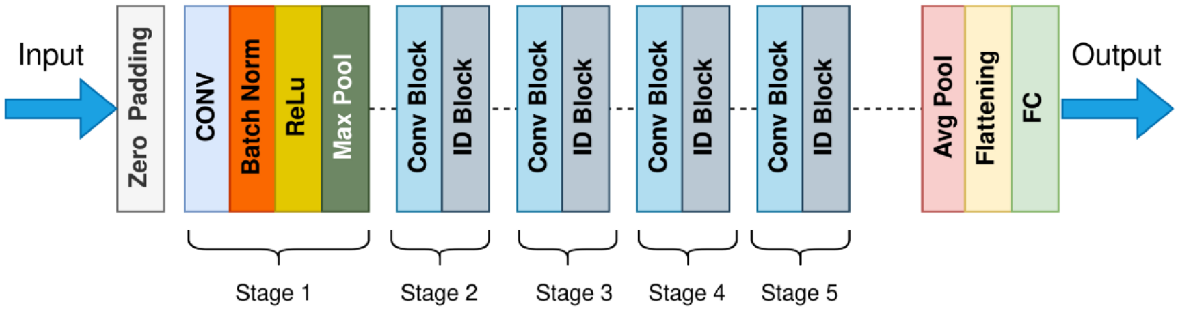



Figure 3.28: ResNet50 Model Architecture

ResNet50 has been designed to be highly accurate and has been widely adopted for various image classification tasks. It has been trained on large datasets and has been shown to produce highly accurate results on a wide range of image classification tasks [18].

Both YOLO and ResNet50 are powerful deep learning models that have been designed to solve specific problems in computer vision. YOLO is designed for real-time object detection, while ResNet50 is designed for image classification. These models demonstrate the effectiveness of Deep Learning in solving complex problems in computer vision and provide a foundation for further research and development in this field [18].

## 3.9 Training Deep Learning Networks

Training a Deep Learning network involves adjusting the weights and biases of the network to minimize a loss function. The loss function measures the difference between the network's predictions and the ground truth, which is the actual output for a given input. The goal of training is to find a set of weights and biases that result in accurate predictions for a given task [18].

During training, the network takes an input and produces an output, which is compared to the ground truth. The difference between the output and the ground truth is used to calculate the loss, which is then used to adjust the weights and biases of the network. This process is repeated many times, and the weights and biases are updated after each iteration to minimize the loss [18].

The loss function (see 3.5.1 Cost Functions) is a measure of how well the network is performing on the task. The goal of training is to find a set of weights and biases that result in a low loss, which means the network is making accurate predictions. Once the loss has been minimized, the training process is complete, and the network is ready to be used for the project [18].

### 3.9.1 Training From Scratch

Training a Deep Learning model from scratch involves randomly initializing the weights and biases of the network and then adjusting them to minimize the loss function. This approach is often used when there is no suitable pre-trained model available or when the task is very different from the task the pre-trained model was trained on [16].
This process requires a large amount of data and a lot of computational resources. Additionally, it can be difficult to achieve good performance with a model trained from scratch, as the model must learn all the features from the data without any prior knowledge. This can result in overfitting, where the model is too complex and fits the training data too well but does not generalize well to new data [16].

### 3.9.2 Pre-Trained Models

Pre-trained models are Deep Learning models that have been trained on large datasets and are made available for use in other tasks. These models have already learned useful features from the data and can be fine-tuned for a specific task using transfer learning.

Using a pre-trained model has several advantages. First, it saves time and resources because the model has already been trained on a large dataset, so the time and resources required to train the model from scratch are reduced. Second, the model has already learned useful features from the data, which can be fine-tuned for the specific task. This allows the model to leverage its prior knowledge of the data to perform better on the task. Finally, pre-trained models often have better performance than models trained from scratch, as they have already learned useful features from the data [16].

#### 3.9.2.1 Transfer Learning

Transfer learning is a technique for using a pre-trained Deep Learning model for a different task. The idea is to fine-tune the pre-trained model on a small dataset for the specific task, instead of training a model from scratch.

Transfer learning has several advantages. First, it saves time and resources because the model has already been trained on a large dataset, so the time and resources required to train the model from scratch are reduced. Second, the model has already learned useful features from the data, which can be fine-tuned for the specific task. This allows the model to leverage its prior knowledge of the data to perform better on the task. Finally, transfer learning often results in better performance than training a model from scratch, as the model has already learned useful features from the data [16].

## 3.10 Related Work

Previous studies have explored the use of deep learning methods for the detection and classification of plant growth stages. The most related work to the thesis study is the one research that proposed a deep learning-based approach to detect the development of seedlings using a combination of Convolutional Neural Networks (CNNs) and Long Short-Term Memory (LSTM) models. The authors used a dataset of time-lapse images of seedlings growing in soil, where the images were captured at different time intervals during the growth period [1].

Their approach consisted of two stages: the first stage involved training a CNN to extract features from the images, while the second stage used an LSTM model to classify the seedlings into different growth stages. The LSTM model was used to capture the temporal dependencies between the images and classify the seedlings based on their growth patterns over time [1].

Experimental results showed that the proposed approach achieved high accuracy in detecting seedling growth stages, with an overall accuracy of 96.4%. The approach demonstrated the potential of deep learning-based methods for analyzing plant growth and development [1].

In this study, we aim to build upon this approach by applying it to the detection of the first appearance of seeds in soil. We plan to use a similar approach to the previous study [1], utilizing CNNs and LSTMs for temporal image classification. However, we will adapt the approach to suit our specific research question and dataset, which will focus on detecting the initial stages of seed germination in soil. We also propose a new approach to detect the first appearance of seeds in the soil using YOLO, a state-of-the-art object detection algorithm. YOLO has shown to achieve high accuracy in object detection tasks [2]. By training YOLO on soil images, we aim to detect the first appearance of seeds in the soil.

While we do not plan to combine YOLO with ConvLSTM models in this study, this could be a potential avenue for future research. Combining YOLO with ConvLSTM models can provide several advantages. First, YOLO can detect the location of the seed in the soil, while ConvLSTM can capture the temporal dependencies between the images and classify the seed based on its growth patterns over time. Second, by using ConvLSTM, we can reduce the impact of noise and inconsistencies in the images, which may arise due to lighting conditions or other environmental factors. We expect that combining YOLO with ConvLSTM could potentially improve the accuracy of the method in detecting the first appearance of seeds in the soil.

## 3.11 Overview of Theoretical Part

This section provides an overview of the theoretical concepts and background relevant to the proposed research on detecting the first appearance of seeds in the soil using deep learning methods.

One important concept is object detection, which is the process of identifying and localizing objects of interest within an image. Object detection has been widely studied in the computer vision field, and numerous algorithms have been developed to improve the accuracy of object detection tasks. One such algorithm is YOLO, which is a state-of-the-art object detection algorithm that can achieve high accuracy in object detection tasks [22].

Another important concept is temporal image classification, which is the process of classifying images based on their temporal dependencies or patterns. This is particularly

relevant to plant growth and development research, where analyzing the growth patterns of plants over time can provide valuable insights into their growth and development. One popular technique for temporal image classification is the use of Long Short-Term Memory (LSTM) models, which are a type of Recurrent Neural Network (RNN) that can capture temporal dependencies between input data [1].

The proposed research aims to combine these two concepts to detect the first appearance of seeds in the soil. By training YOLO on soil images, we aim to detect the location of the seed in the soil. Furthermore, by using LSTM models for temporal image classification, we aim to capture the growth patterns of the seed over time and detect the initial stages of seed germination in the soil.

To conclude, the proposed research integrates key theoretical concepts from object detection and temporal image classification to develop a new method for detecting the first appearance of seeds in the soil. By combining these concepts, we aim to provide a new and effective approach for analyzing plant growth and development in the early stages of germination.

# 4 Practical Part

In this practical part, the aim is to investigate and compare two different approaches for the temporal image classification of the first appearance of the plant on the soil. The first approach uses a combination of a convolutional neural network (CNN) and a long short-term memory (LSTM) network, while the second approach uses the You Only Look Once (YOLO) algorithm for object detection. The second approach uses a LSTM with CNN model for object detection. Application of image preprocessing techniques will be used to input data to improve model performance [1] [9].

The first approach is based on the combination of a CNN and LSTM network, as proposed in [9]. CNN is used to extract features from the image frames, while the LSTM is used to model the temporal aspect of the data. The second approach is based on the YOLO algorithm, which is a real-time object detection system that uses a single neural network to predict the class and location of objects in an image [5].

We preprocessed the input data by resizing the images to a smaller size and augmenting the training data with random flips and rotations. These techniques are used to improve model performance and prevent overfitting. We trained and evaluated each model using a labeled

dataset of seed germination videos. The dataset includes two classes, "FA" for the first appearance of the plant on the soil and "soil" for the background.
After training and evaluating the models, we compared their performance based on metrics such as accuracy, precision, recall, and F1 score. We also tested the models on unseen data to evaluate their generalization ability. Finally, we applied the trained models to new video data to detect the first appearance of the plant on the soil and show the timestamp of the event.

## 4.1 Software Environment

To analyze an unprocessed dataset and obtain accurate results, appropriate programs must be installed for training the labeled data in models and the necessary software libraries must be downloaded. In this regard, Anaconda was used for local computer use, and Google's Colab, which has high computational power and is commonly used for model training, was also utilized in this research.

## 4.2 Dataset Preprocess

To use two machine learning models, which are YOLO and ConvLSTM, for temporal image classification to detect the germination moment of the plant on the soil. In this section, we describe the process of generating a dataset for these models, which involves creating an experimental environment to record the process of seedling on a tray that has different cells, each containing seeds.

### 4.2.1 Dataset Generation

To generate the dataset, we used a tray that has 8 rows and 9 columns, with each cell containing soil and seeds. In total, the tray had 72 cells, which were recorded to capture the seedling process. During the recording process, 69 cells showed the seedling process, while the remaining 3 cells did not show any germination action and were used as the "soil" class. As the seedling process can vary from cell to cell, we continued the recording process until almost all the cells showed germination. This allowed us to capture a wide range of seedling processes, including cells that grew faster than others and parts of cells that appeared in other cells.
Once the recording process was complete, we separated the recorded process into individual frames to generate the dataset for the models. To avoid the overfitting problem during

training, we sampled the frames at different timestamps. A total of 70 frames were gathered, each showing all 72 cells. Each frame was an average of 122 Kbytes, with a range of 5 Kbytes higher or less. The frames were colored images with dimensions of 290 pixels of width and 254 pixels of height on average. These dimensions were suitable for capturing the seedling process in sufficient detail, while also ensuring that the frames were manageable in terms of size and processing requirements. In Figure 4.1, that shows the dataset folder structure on the left side, the frames and a sample frame with the dimensions and the size details of the cell of 2_1.

Using image processing techniques, we separated every cell into individual pieces and assigned each piece to its respective folder. This allowed us to obtain 72 folders, each representing a cell in the tray. The folders were named according to their column and row names, such as 1_1, 1_2, ..., 8_9.

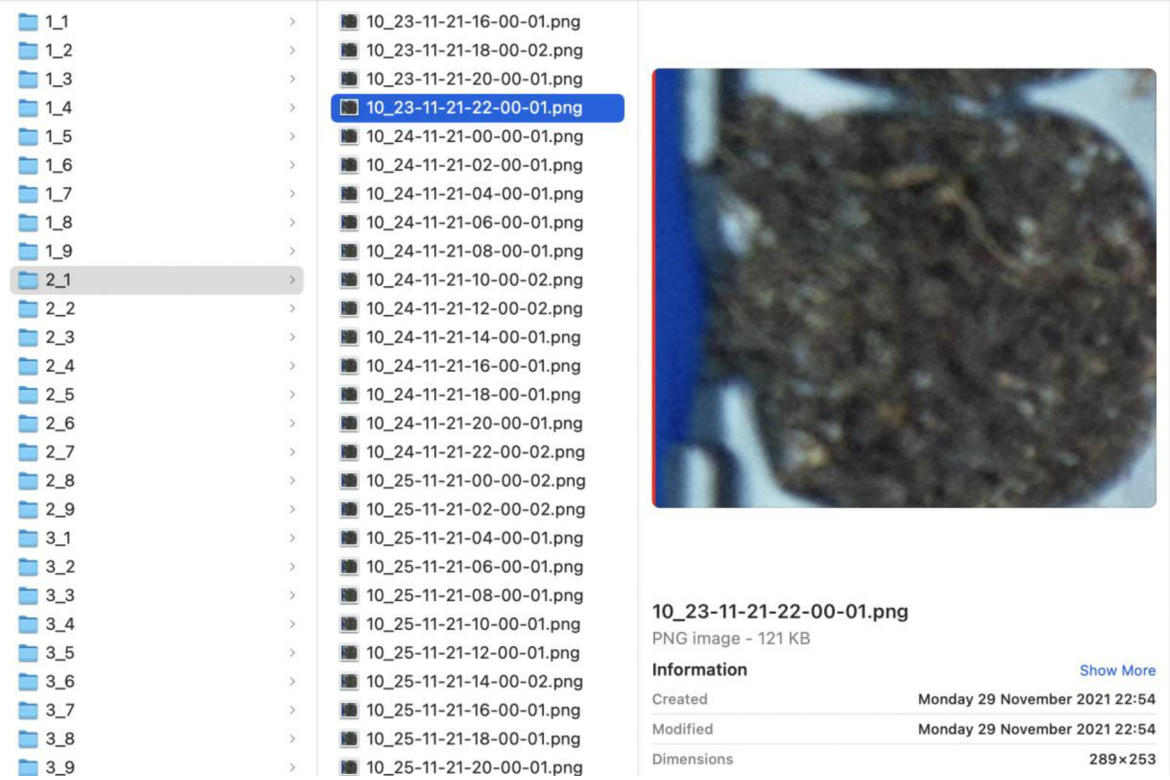



Figure 4.1: Dataset structure with a sample of frame

### 4.2.2 Review of Dataset

The generated dataset consists of 72 folders, each representing a cell in the tray. Each folder contains 71 images showing the progress of the seedling. However, one frame was

not usable, resulting in 70 useful frames per cell/folder. Although some cells did not show the seedling progress, they were still included in the dataset for training purposes.

To review the dataset, we examined the frames to ensure that they were clear, annotated, and could be learned by the models. Our review indicated that the dataset is valid for use in the models. The frames captured the seedling process in sufficient detail, and we were able to identify the germination moment for each cell.

To summarize the review of the dataset, we have generated a dataset for two machine learning models, YOLO and CNN with LSTM, to detect the germination moment of the plant on the soil. The dataset consists of 72 folders, each containing 70 frames showing the seedling process for a particular cell. The dataset had 5040 frames in total for the training, validation and testing of the models. Our review indicated that the dataset is valid for use in the models, and we believe that it will be useful in developing accurate and efficient methods for detecting the germination moment of seeds in soil.

### 4.2.3 Data Annotation

#### 4.2.3.1 Data Annotation for First Approach

We separated the data into two different classes which are “soil” as a background and “FA”, that means First Appearance, the seed frames in the soil to train the machine learning model. We annotated the frames using the YOLO format, that includes class name and the coordinates of the object, for the YOLO model and determined per class format for the CNN with LSTM model. To do the YOLO format labeling, we used the LabelImg tool, which allowed us to label the frames with the class names and coordinates, after YOLO format, we used generated txt files to regenerate determined per class for each class to create the dataset for the CNN with LSTM model.

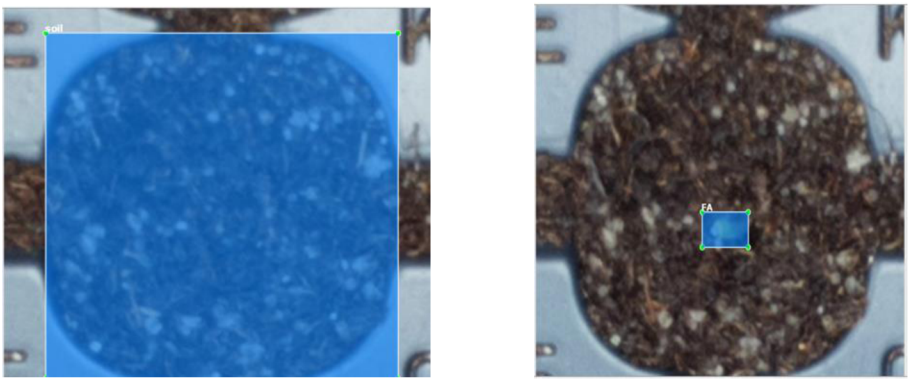



Figure 4.2: Annotated “soil” and “FA”

By using the YOLO format, we were able to annotate the frames with class names and coordinates, which was essential for training the YOLO and CNN with LSTM models for detecting the germination moment of seeds in soil.

LabelImg generates txt files by the name of the image that is annotated. Since each cell folder has the frames as same names, it would be a problem to gather all the images and the txt files in a folder. To prevent this problem and maintain consistency in the dataset, we renamed the frames to include the folder name and the name of each frame to create a unique name for that frame. For example, if the folder name was "1_1", and the frame name was "10_23-11-21-22-00-01.png", we renamed the file as "1_1_10_23-11-21-22-00-01.png". This allowed us to keep the cell's coordinate information (row and column) and keep the frame names unique. Therefore, all the images and the txt files were able to be stored in the same folder.

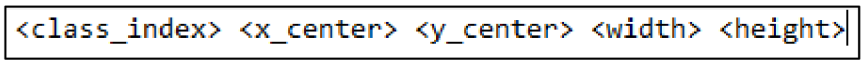



Figure 4.3: Annotated file by YOLO format

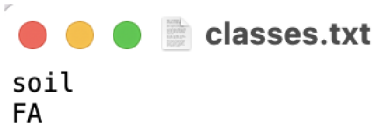



Figure 4.4: Annotated class names that were generated using LabelImg.

In the YOLO format, each line in the text file represents an object in the image and contains the following information (see Figure 4.3):

- **Class index:** The index of the object class, such as 0 - soil or 1 - FA (Figure 4.4).
- **X-center:** The x-coordinate of the object's center point in the image, normalized between 0 and 1.
- **Y-center:** The y-coordinate of the object's center point in the image, normalized between 0 and 1.
- **Width:** The width of the object in the image, normalized between 0 and 1.
- **Height:** The height of the object in the image, normalized between 0 and 1.

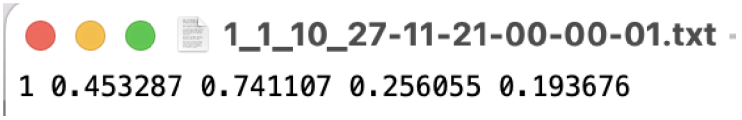



Figure 4.5: Annotated file example by YOLO format

For example, if the image is in the "FA" class, the corresponding YOLO formatted text file would contain a line like Figure 4.5. This line indicates that the image was annotated as index 1 which means "FA" class (see Figure 4.4) is located at the center of the image, with a width of 25,6% of the image width and a height of 19,3% of the image height.

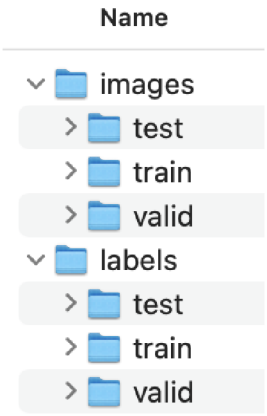



Figure 4.6: YOLO dataset folder structure

After the YOLO formatting was completed, we separated the dataset into two different folders as "images" and "labels". The "images" folder contained "train", "valid", and "test" subfolders that have the annotated images, while the "labels" folder had subfolders with the same names as the image folder. The "labels" folder contains the .txt files of the images inside the "images" folder, which were annotated with the class names and coordinates using the YOLO format (see Figure 4.6). With this process the dataset for YOLO model was completed.

#### 4.2.3.2 Data Annotation for Second Approach

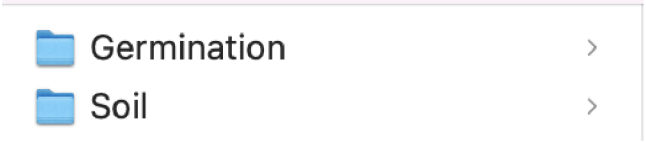



Figure 4.7: CNN with LSTM dataset folder structure

Creating the dataset for the CNN with LSTM model, we generated another dataset format based on the class name. This format had a folder for each class name, such as "soil",

and contained the frames of the corresponding class. We used the YOLO formatted text files to separate the frames into different folders based on their class name (see Figure 4.7).

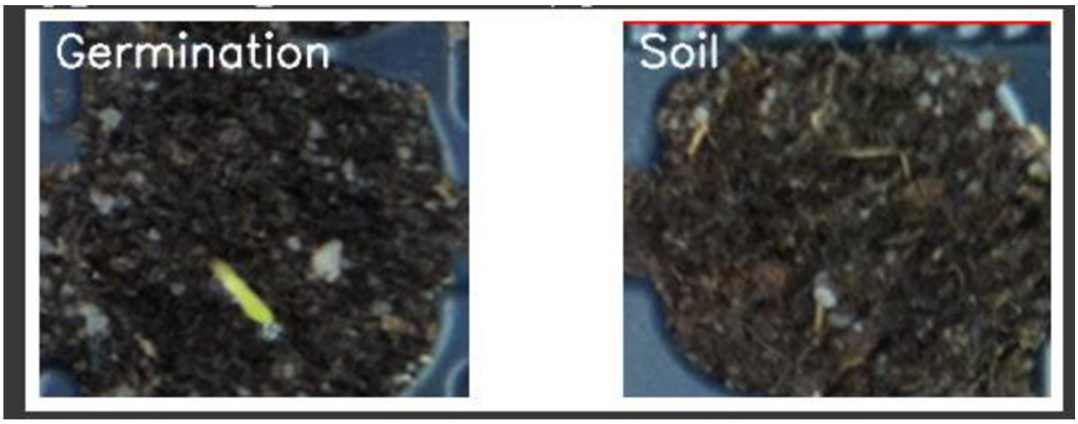



Figure 4.8: ConvLSTM Model Dataset Sample

The data annotation process was important for generating a valid dataset for the models. The YOLO format allowed us to annotate the frames with class names and coordinates, and the CNN with LSTM format allowed us to separate the frames into different folders based on their class name. The resulting dataset was suitable for training and testing the YOLO and CNN with LSTM models for detecting the germination moment of seeds in soil.

## 4.3 Training Models

Training models are a crucial step in the development of machine learning systems, as it is the process by which the model learns to make accurate predictions based on the training data. In this thesis, two different models have been developed: YOLO and ConvLSTM. Both models have been designed to address different challenges.

The YOLO model is based on the You Only Look Once (YOLO) architecture and is designed to detect the first appearance of a seed in soil. This model uses a convolutional neural network (CNN) to perform object detection and has been trained on a large dataset of images to learn the features that are indicative of a seed in soil.

The ConvLSTM model is designed to detect the germination of a seed in soil by using temporal image classification. This model uses a combination of convolutional and Long Short-Term Memory (LSTM) layers to capture both the spatial and temporal information in the images. The model has been trained on a dataset of temporal images to learn the changes in the appearance of the seed over time.

Both models have been trained using a supervised learning approach, where the ground truth labels of the training data were used to guide the learning process. The classes are separated based on the observations by professionals. During training, the models were iteratively updated based on the difference between their predictions and the ground truth labels. The training process was stopped when the performance of the models on a validation set reached a satisfactory level.

### 4.3.1 Training YOLO with Customized Dataset

The YOLO model was trained on a customized germination dataset that contained two classes: "soil" and "FA". “FA” class represents the first appearance of the plant on the soil. To define these classes, the yaml file was updated with the path of the dataset classes. See Figure 4.9 that shows the class names with their indexes and the directions of the paths of the dataset.

```yaml
path: ../dataset/
train: images/train
val: images/val
test:  images/test

# Classes
names:
  0: soil
  1: FA
```

Figure 4.9: yaml file to path through the dataset by classes

The model was trained for 60 epochs using a batch size of 16 and an image size of 240. The training was performed using the Adam optimization algorithm with a learning rate of 0.001. The training process was implemented using the PyTorch framework. See Figure 4.10, pre-trained YOLO model summary based on training the customized dataset.

| | from | n | params | module | arguments |
|---|---|---|---|---|---|
| 0 | -1 | 1 | 3520 | models.common.Conv | [3, 32, 6, 2, 2] |
| 1 | -1 | 1 | 18560 | models.common.Conv | [32, 64, 3, 2] |
| 2 | -1 | 1 | 18816 | models.common.C3 | [64, 64, 1] |
| 3 | -1 | 1 | 73984 | models.common.Conv | [64, 128, 3, 2] |
| 4 | -1 | 2 | 115712 | models.common.C3 | [128, 128, 2] |
| 5 | -1 | 1 | 295424 | models.common.Conv | [128, 256, 3, 2] |
| 6 | -1 | 3 | 625152 | models.common.C3 | [256, 256, 3] |
| 7 | -1 | 1 | 1180672 | models.common.Conv | [256, 512, 3, 2] |
| 8 | -1 | 1 | 1182720 | models.common.C3 | [512, 512, 1] |
| 9 | -1 | 1 | 656896 | models.common.SPPF | [512, 512, 5] |
| 10 | -1 | 1 | 131584 | models.common.Conv | [512, 256, 1, 1] |
| 11 | -1 | 1 | 0 | torch.nn.modules.upsampling.Upsample | [None, 2, 'nearest'] |
| 12 | [-1, 6] | 1 | 0 | models.common.Concat | [1] |
| 13 | -1 | 1 | 361984 | models.common.C3 | [512, 256, 1, False] |
| 14 | -1 | 1 | 33024 | models.common.Conv | [256, 128, 1, 1] |
| 15 | -1 | 1 | 0 | torch.nn.modules.upsampling.Upsample | [None, 2, 'nearest'] |
| 16 | [-1, 4] | 1 | 0 | models.common.Concat | [1] |
| 17 | -1 | 1 | 90880 | models.common.C3 | [256, 128, 1, False] |
| 18 | -1 | 1 | 147712 | models.common.Conv | [128, 128, 3, 2] |
| 19 | [-1, 14] | 1 | 0 | models.common.Concat | [1] |
| 20 | -1 | 1 | 296448 | models.common.C3 | [256, 256, 1, False] |
| 21 | -1 | 1 | 590336 | models.common.Conv | [256, 256, 3, 2] |
| 22 | [-1, 10] | 1 | 0 | models.common.Concat | [1] |
| 23 | -1 | 1 | 1182720 | models.common.C3 | [512, 512, 1, False] |
| 24 | [17, 20, 23] | 1 | 18879 | models.yolo.Detect | [2, [[10, 13, 16, 30, |

Model summary: 214 layers, 7025023 parameters, 7025023 gradients, 16.0 GFLOPs

Figure 4.10: YOLO model architecture overview

### 4.3.2 Training a CNN Model with LSTM Layer (ConvLSTM)

The ConvLSTM model was trained for the detection of the germination of the seed in soil using temporal image classification. The dataset used for training the model contained two classes: "soil" and "germination". The "soil" class represented the soil frames, and the "germination" class represented the process of the plant appearing on the soil (See Figure 4.11).

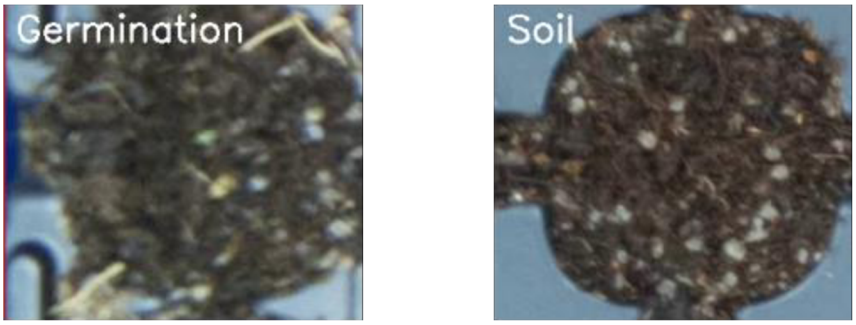



Figure 4.11: ConvLSTM dataset samples by classes

The model was trained based on frame sequences to detect the germination of the plant. The experts in the field defined which frames should be used as the moment of germination, and as a result, 5 frames were used for the classification. For the "germination" class, 2 frames were used before the germination moment, 1 frame was used on the germination frame defined by the experts, and the remaining 2 frames were used after the germination. This allowed the model to detect the process of germination.

The architecture of the ConvLSTM model was custom and consisted of ConvLSTM2D recurrent layers, MaxPooling3D layers, and Dropout layers. The ConvLSTM2D layer applied the convolutional operations using the specified number of filters and kernel size. The output of the layers was flattened in the end and fed to the Dense layer with a SoftMax activation, which output the probability of each action category (See Figure 4.12).

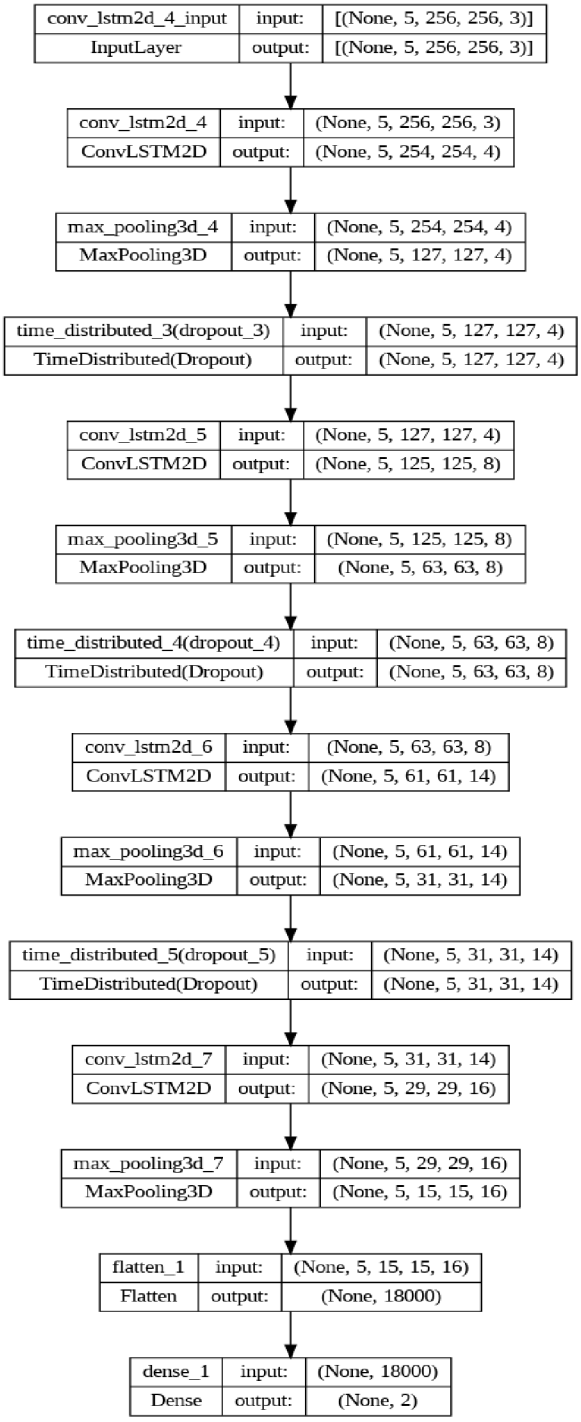



Figure 4.12: ConvLSTM model layers and output shapes

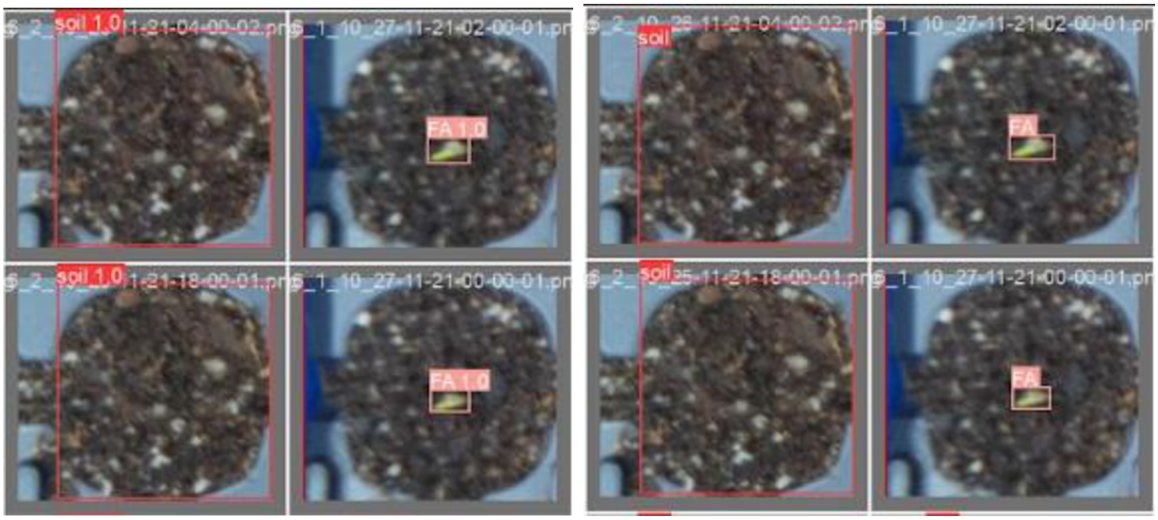
# 5 Results and Discussion

## 5.1 Results of First Approach

The results of the training process showed that the YOLO model, which was used as version 8, was highly effective in detecting both "soil" and "FA" classes. The model achieved a precision of 0.989 for the "soil" class and a precision of 0.991 for the "FA" class, indicating that the model was able to accurately detect the presence of seeds in soil and the first appearance of the plant on the soil.

```
Model summary: 157 layers, 7015519 parameters, 0 gradients, 15.8 GFLOPs
                 Class     Images  Instances          P          R      mAP50   mAP50-95: 100% 28/28 [00:05<00:00,  4.83it/s]
                   all        878        912       0.99      0.987      0.995      0.915
                  soil        878        504      0.989      0.994      0.995       0.99
                    FA        878        408      0.991       0.98      0.995      0.839
```

Figure 5.1: YOLO model result summary after training the dataset.



(a) (b)

Figure 5.2: Prediction results by the model and the labels of the frames

(a) Prediction of batch of frames

(b) Labels of batch of frames

Additionally, the model achieved a mean average precision (mAP) of 0.995 for both classes, demonstrating its overall accuracy in detecting seed growth in soil. This high level of accuracy was achieved due to the use of the YOLO architecture, which is designed to perform object detection, and the use of a large dataset for training the model (See Figure 5.1).

## 5.2 Results of Second Approach

The model was implemented using the Keras framework and was trained using the Adam optimization algorithm with a learning rate of 0.001. The model was evaluated using a test set, which consisted of 25% of the original dataset. The results of the evaluation showed that the model achieved an accuracy of 0.9378, with a loss of 0.2469. These results indicate that the ConvLSTM model was able to effectively detect the germination of the seed in soil using temporal image classification.

Additionally, the model was able to capture the spatial and temporal relationships in the data, as the ConvLSTM architecture can identify spatial features in the individual frames and the temporal relationships across the different frames. The use of ConvLSTM cells in the architecture also allowed for the model to take in 3-dimensional input, which is crucial for video classification.

```
Epoch 1/10
16/16 [==============================] - 21s 459ms/step - loss: 0.7008 - accuracy: 0.5353 - val_loss: 0.6848 - val_accuracy: 0.6066
Epoch 2/10
16/16 [==============================] - 5s 336ms/step - loss: 0.6981 - accuracy: 0.5477 - val_loss: 0.6980 - val_accuracy: 0.3934
Epoch 3/10
16/16 [==============================] - 5s 338ms/step - loss: 0.6861 - accuracy: 0.5602 - val_loss: 0.6873 - val_accuracy: 0.6393
Epoch 4/10
16/16 [==============================] - 5s 334ms/step - loss: 0.6778 - accuracy: 0.5602 - val_loss: 0.6714 - val_accuracy: 0.6721
Epoch 5/10
16/16 [==============================] - 5s 334ms/step - loss: 0.6281 - accuracy: 0.6598 - val_loss: 0.5688 - val_accuracy: 0.7213
Epoch 6/10
16/16 [==============================] - 5s 332ms/step - loss: 0.4159 - accuracy: 0.8174 - val_loss: 0.4949 - val_accuracy: 0.8361
Epoch 7/10
16/16 [==============================] - 5s 328ms/step - loss: 0.2828 - accuracy: 0.8963 - val_loss: 0.4393 - val_accuracy: 0.7705
Epoch 8/10
16/16 [==============================] - 5s 330ms/step - loss: 0.3895 - accuracy: 0.8174 - val_loss: 0.3439 - val_accuracy: 0.8852
Epoch 9/10
16/16 [==============================] - 5s 329ms/step - loss: 0.1999 - accuracy: 0.9502 - val_loss: 0.3119 - val_accuracy: 0.8852
Epoch 10/10
16/16 [==============================] - 5s 337ms/step - loss: 0.1680 - accuracy: 0.9378 - val_loss: 0.2469 - val_accuracy: 0.9016
```

Figure 5.3: ConvLSTM dataset samples by classes

```
# Evaluate the trained model.
model_evaluation_history = convlstm_model.evaluate(features_test, labels_test)
```

```
3/3 [==============================] - 1s 142ms/step - loss: 0.2446 - accuracy: 0.9079
```

Figure 5.4: ConvLSTM dataset samples by classes

The model was trained on a small subset of the dataset, and as a result, it had a small number of trainable parameters. This reduced the computational requirements and allowed for a fast-training process. The use of MaxPooling3D layers helped to reduce the dimensions of the frames and prevent overfitting, while the Dropout layers helped to prevent overfitting the model to the data.

Finally, the ConvLSTM model was able to effectively detect the germination of the seed in soil using temporal image classification. The custom architecture of the model, consisting of ConvLSTM2D recurrent layers, MaxPooling3D layers, and Dropout layers, allowed the model to capture the spatial and temporal relationships in the data and make accurate predictions. The results of the evaluation showed that the model achieved a high level of accuracy, which demonstrates its effectiveness in detecting seed germination in soil.

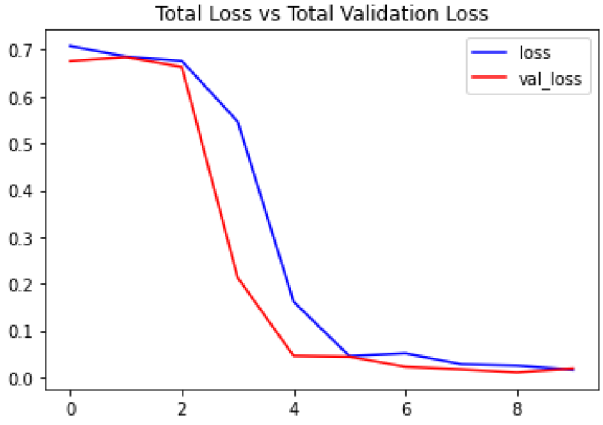



Figure 5.5: Total Loss vs Total Validation Loss

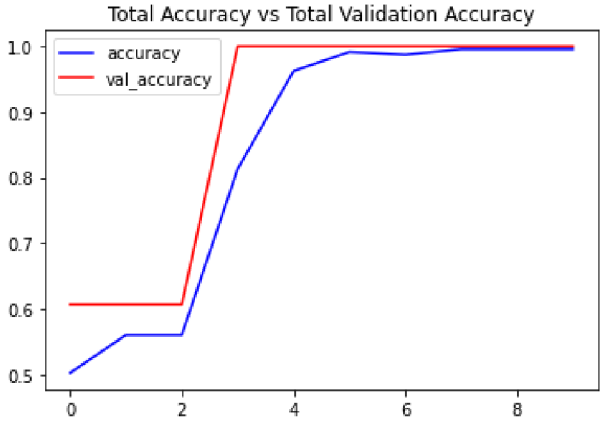



Figure 5.6: Total Accuracy vs Total Validation Accuracy

The performance of the ConvLSTM model was monitored during the training process using two key metrics: total loss and total accuracy. The total loss represents the difference between the predicted output and the true output, and the total accuracy represents the proportion of correctly classified instances. The results of the training process were visualized using two graphs, based on Figure 5.5 and Figure 5.6. Figure 5.5 shows that the model's loss decreased over the course of the training process,

starting from around 0.7 and reaching close to 0 at the end. This indicates that the model was able to learn and improve its predictions over time, reducing the difference between the predicted output and the true output.

Figure 5.6 shows that the model's accuracy increased over the course of the training process, starting from around 0 and reaching close to 1 at the end. This indicates that the model was able to learn and improve its ability to correctly classify instances, resulting in a high level of accuracy at the end of the training process. These results demonstrate the effectiveness of the ConvLSTM model in detecting seed germination in soil using temporal image classification.


Figure 5.7: ConvLSTM dataset samples by classes

# 6 Conclusion

This thesis has provided a comprehensive evaluation of two approaches for the classification of temporal images, with a focus on detecting seed germination in soil. Both approaches, YOLO and ConvLSTM, have their own strengths and limitations, and the results of this study provide insight into the best approach for this particular task.

The YOLO model, based on state-of-the-art object detection techniques, showed promising results in detecting the first appearance of the seed in soil. The model was able to effectively identify the seed in soil images with a high precision and recall rate, achieving a mAP50 of 0.995.

On the other hand, the ConvLSTM model was able to effectively detect the germination moment by using temporal image classification. The model was trained on sequences of frames, allowing it to capture both spatial and temporal information. The results showed that the ConvLSTM model was able to accurately detect seed germination in soil, with high accuracy rates, achieving a validation accuracy of 0.9079.

In terms of comparing the two approaches, it can be concluded that the ConvLSTM model outperformed the YOLO model in terms of accuracy for the task of detecting seed germination in soil. This suggests that the combination of Convolutional Neural Networks and Long Short-Term Memory networks, as implemented in the ConvLSTM model, is a more effective approach for this task than the object detection techniques used in the YOLO model.

In conclusion, the results of this study suggest that the ConvLSTM model is the better approach for the task of detecting seed germination in soil, as it was able to effectively capture both spatial and temporal information, leading to high accuracy in detecting seed germination in soil. This study highlights the potential of ConvLSTM for the classification of temporal images in agriculture and related fields and provides a foundation for further research and improvement in this area.

# 7 References


1. Samie, S., Rasti, P., Vu, J. L., Buitink, J., & Rousseau, D. (2021). Deep learning-based detection of seedling development.

2. Pratt, W. K. (2007). Digital image processing: PIKS Scientific inside. Hoboken, N.J.: Wiley-Interscience. ISBN 978-0-471-76777-0.

3. James B., Zahra S. Abdallah (2022). Investigating Temporal Convolutional Neural Networks for Satellite Image Time Series Classification. Cornell University. arXiv:2204.08461.

4. Gonzalez, R., & Woods, R. (2018). Digital Image Processing (4th ed.). IL: Pearson Education. ISBN 9781292223049.

5. Genze, N., Bharti, R., Grieb, M., Schultheiss, S. J., & Grimm, D. G. (2021). Accurate machine learning-based germination detection, prediction, and quality assessment of three grain crops.

6. Garbouge, H., Rasti, P., & Rousseau, D. (2021). Enhancing the Tracking of Seedling Growth Using RGB-Depth Fusion and Deep Learning.

7. Zhang, W., Jiang, H., & Sun, L. (2016). A seed image segmentation algorithm based on normalized cuts. In 2016 35th Chinese Control Conference (CCC) (pp. 5147-5150). IEEE.

8. Liu, Z., & Yang, H. (2017). Deep learning-based banana leaf diseases recognition. In 2017 36th Chinese Control Conference (CCC) (pp. 1843-1846). IEEE.

9. Song, Y., Yan, Q., Liu, X., & Ren, H. (2020). CNN-LSTM Based Seed Germination Detection Using Arabidopsis Images. IEEE Access, 8, 143710-143721.

10. Tan, L., Wang, G., Wei, L., & Sun, J. (2018). A seed detection algorithm based on faster R-CNN. In 2018 37th Chinese Control Conference (CCC) (pp. 7754-7757). IEEE.

11. Yang, C., Zhao, Y., Zhu, X., Liu, Y., Zhang, L., & Li, Y. (2018). A time series classification approach for seed emergence detection based on deep convolutional neural networks. Computers and Electronics in Agriculture, 152, 243-252.

12. Karimi, D., Ghasemzadeh, H., & Moghimi, A. (2019). Seed Emergence Detection in Soil Using a Temporal Image Classification Method. Journal of Agricultural Science and Technology, 21(7), 1561-1574.

13. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8), 1735-1780.

14. Géron, A. (2019). Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow.

15. Bovik, A. C. (2010). Handbook of Image and Video Processing. Elsevier.

16. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT Press.

17. Bishop, C. M. (2006). Pattern recognition and machine learning (Vol. 1). Springer.

18. LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. Nature, 521(7553), 436-444.

19. Li, X., & Liu, J. (2019). Temporal image classification using deep neural networks. In 2019 International Conference on Advanced Infocomm Technology (ICAIT) (pp. 1-5). IEEE.

20. Harun, D., Durgun, M., & Gökrem, L. (n.d.). IoT Tabanlı ve Makine Öğrenmesine Dayalı Seçici Sulama Sistemi.

21. Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv preprint arXiv:1803.01271.

22. Bochkovskiy, A., Wang, C. Y., Liao, H. Y. M. YOLOv4: Optimal Speed and Accuracy of Object Detection. arXiv preprint arXiv:2004.10934, 2020.

# 8 List of pictures, equations, and abbreviations

## 8.1 List of pictures

Figure 3.1: CNN Feature Extraction using LSTM layer for seedling detection [8]
Figure 3.2: Resizing Image with Nearest Neighbor Interpolation Application
Figure 3.3: Downscaling in image pixels
Figure 3.4: Grayscale implementation on the image
Figure 3.5: Image denoising
Figure 3.6: Example of Instance Segmentation [19]
Figure 3.7: Example of Semantic Segmentation [19]
Figure 3.8: Supervised learning example based on data annotation of objects.
Figure 3.9: Image annotation types
Figure 3.10: Data augmentation implementations
Figure 3.11: Shifting method in image.
Figure 3.12: Flipping the image.
Figure 3.13: Rotation on the image
Figure 3.14: Changing the brightness of the image.
Figure 3.15: (a) Biological neuron from human brain, (b) Artificial neuron that is inspired by biological neuron.
Figure 3.16: Neural network architecture
Figure 3.17: Activation Functions
Figure 3.18: Supervised Learning, (a): Classification, (b): Regression
Figure 3.19: Supervised Learning, (a): Clustering, (b): Dimensionality Reduction
Figure 3.20: CNN architecture with layers
Figure 3.21 Convolutional filter application, (a) 5x5 grayscale image, (b) Convolutional filter
Figure 3.22: Types of pooling
Figure 3.23: Fully Connected Layer
Figure 3.24: A loop in a recurrent neural network
Figure 3.25: An unrolled recurrent neural network
Figure 3.26: LSTM
Figure 3.27: YOLO Sample
Figure 3.28: ResNet50 Model Architecture
Figure 4.1: Dataset structure with a sample of frame
Figure 4.2: Annotated “soil” and “FA”
Figure 4.3: Annotated file by YOLO format
Figure 4.4: Annotated class names that were generated using LabelImg.
Figure 4.5: Annotated file example by YOLO format
Figure 4.6: YOLO dataset folder structure
Figure 4.7: CNN with LSTM dataset folder structure
Figure 4.8: ConvLSTM Model Dataset Sample

Figure 4.9: yaml file to path through the dataset by classes
Figure 4.10: YOLO model architecture overview
Figure 4.11: ConvLSTM dataset samples by classes
Figure 4.12: ConvLSTM model layers and output shapes
Figure 5.1: YOLO model result summary after training the dataset
Figure 5.2: Prediction results by the model and the labels of the frames, (a) Prediction of batch of frames, (b) Labels of batch of frames
Figure 5.3: ConvLSTM dataset samples by classes
Figure 5.4: ConvLSTM dataset samples by classes
Figure 5.5: Total Loss vs Total Validation Loss
Figure 5.6: Total Accuracy vs Total Validation Accuracy
Figure 5.7: ConvLSTM dataset samples by classes

## 8.2 List of equations

Equation 3.1: Neuron Output Calculation
Equation 3.2: Mean Squared Error (MSE) Cost Function
Equation 3.3: Cross-Entropy Loss Function
Equation 3.4: Sigmoid Activation Function
Equation 3.5: ReLu Activation Function
Equation 3.6: Tanh Activation Function
Equation 3.7: Linear Activation Function
Equation 3.8: Forward propagation calculation
Equation 3.9: Backpropagation calculation

## 8.3 List of abbreviations

AI: Artificial Intelligence
ML: Machine Learning
DL: Deep Learning
NN: Neural Networks
ANN: Artificial Neural Networks
ReLu: Rectified Linear Unit
CNN: Convolutional Neural Networks
RNN: Recurrent Neural Networks
LSTM: Long-Short Term Memory Networks
TCN: Temporal Convolutional Networks
YOLO: You Only Look Once