PRECISION FARMING

Abdul-Azeez Olanlokun
Izuchukwu George Enekwa
Patrick Nonki
Pritish Sanjay Samant

Contents

List of Figures

Abstract: A novel idea called ´´Precision Farming" aims to increase the productivity and effectiveness of agriculture by utilizing cutting-edge information technology. Using the most recent developments in automation, artificial intelligence, and networking, farmers are better able to keep an eye on every step of the process and administer exact treatments selected by machines with phenomenal accuracy. Engineers, data scientists, and farmers are still developing methods to optimize the human labour needed in agriculture. Smart farming develops into a learning system that becomes smarter daily as vital information resources improve. In this study and practical implementation, we have made use of two smart farming approaches, a timed Petri Net, which allows for real-time modelling of our use cases in solving agricultural problems, and also a deep learning approach which is a machine learning technique that employs the fundamentals of an artificial neural network. We have used the leNet Model, a CNN model, as our main model to train and evaluate our datasets, which gave us an accuracy of 93% with 60 epochs. With our approach, we were able to improve weed detection, plant health monitoring, soil moisture detection, and parasite detection. Studies can still be made in the future to improve and maintain our approach.

1 Motivation

Throughout the history of human agriculture, one of the key goals has always been to increase the economic efficiency of agricultural operations. However, due to the challenges in achieving quality/cost balance, this target has not been met to the intended degree. It may be able to implement all essential safeguards during crop production by visiting agricultural production sites regularly, which is important to obtain excellent goods. Farmers raise the price of the harvest by investing more time and money into each visit. Given that farmers spend a lot of time observing and assessing their crops, precision farming has become vital. Technologies based on the ´´Internet of things"(IoT) provide remote and accurate monitoring, making crop management not only wise but also economical [Ay19].

Real-time monitoring of agricultural activities is necessary, but it is insufficient to make agriculture intelligent. The cycle of observation, diagnosis, decision, and action should guide smart agriculture. Data should be gathered and used as soon as possible to make adjustments that maximize the agricultural process in this perpetual cycle. Sensors that capture characteristics from natural resources, including crops, animals, atmosphere, soils, water, and biodiversity, can be used to collect data during the observation phase. The sensor readings are sent to an IoT platform located in the cloud during the diagnostic phase, where predefined decision models are used to ascertain the condition of the object under study. The components based on deep learning approaches decide if an action is necessary during the decision phase. The end-user assesses the circumstance and executes the action during the action phase. And the process repeats itself[Sc20].

Being a farmer in this century requires more than just a passion of the land. To practice sustainable agriculture, farmers need to possess specialized expertise in agriculture, law, economics, accounting, and data analysis[Bl]. Farmers continued to use pesticides and fertilisers in most areas over the 20th century, which had an irreparable impact on the ecosystem[Ay19]. As awareness increased, it became clear that each plant should be handled according to its unique needs rather than treating every farm and produce in the same manner.

Farmers have gotten more and more professional counsel in recent years, but it is often expensive. Farmers may obtain such guidance at a reasonable cost thanks to the intelligent agriculture system made up of IoT and Deep learning technology. These technologies automate crop monitoring using the most cutting-edge techniques, necessitating little human involvement[VS18].

Our approach to precision farming has incorporated a real-time modelling tool known as the Timed Petri Net, which is essential for the planning, design and evaluation of the performance of discrete event dynamic systems. And also, a deep learning approach allows us to train our datasets to identify deficiencies, weeds, soil moisture, and parasites in the farmland to maximise crop yields.

2 Precision Farming

In precision agriculture, technologies and principles are applied to manage spatial and temporal variability associated with all aspects of agricultural production in order to improve production and environmental quality. Precision agriculture depends on the accurate assessment of variability, its management, and its evaluation in a time-space continuum. In practice, precision agriculture has shown to be agronomically feasible largely because traditional arrangement recommendations can be applied at finer scales to achieve agronomic efficiency. Precision agriculture has been quite successful in crops like sugar beet, sugarcane, tea, and coffee in terms of agronomic performance. Precision agriculture holds great potential for economic, environmental, and social benefits, yet it remains largely unrealized due to the lack of adequate consideration of the space-time continuum of crop production.

3 The Need for Precision Farming

The goal of farm management is to increase productivity and reduce environmental risks by accounting for variability. It is common for farms in developed countries to consist of several fields and to be large. However, precision farming is an integration of technologies that permits the collection of data on an appropriate scale and at a suitable time, ageing decisions, and appropriate time. Supporting management decisions by interpreting and analyzing data.

3.1 Overall yield increase

The precise selection of crop varieties, the application of exact types and doses of fertilizers, pesticides and herbicides, and appropriate irrigation meet the demands of crops for optimum growth and development. This leads to yield increase, especially in areas or fields where uniform crop management practices were traditionally practised.

3.2 Efficiency improvement

In agriculture, advanced technologies, such as machinery, tools, and information, help farmers improve the efficiency of labor and time.

3.3 Better decision-making in agricultural management.

A farmer's agricultural machinery, equipment, and tools provide accurate information that is then processed and analyzed for proper decision making during land preparation, seeding, fertilization, pesticide and herbicide application, irrigation, drainage, and post-production.

4 Emerging technologies

Developing a truly sustainable agriculture system requires the systematic implementation of best management practices into a site-specific system. In order to manage the right source at the right time, in the right place, and at the right rate, you need the right tools. A variety of technologies are available to assist with farm management decisions. The tools helps to fine-tune management decisions and create management plans for each field. Machinery such as;

4.1 Self-steering tractors

There have been self-steering tractors for a long time now. The tractor does most of the work, with the farmer providing assistance when needed. A driverless tractor programmed by GPS to spread fertilizer or plow land is becoming a reality. Among the innovations is a solar-powered machine that identifies weeds and precisely kills them with lasers or herbicides.

4.2 Drones and Satellite Imagery

Precision farming benefits from advances in drone and satellite technology because drones take high-quality images, while satellites capture a broad view. By combining aerial photography and satellite records, light aircraft pilots can predict future yields based on the current level of field biomass. With the images, we can create contour maps to track where water flows, determine variable-rate seeding, detect weeds, classify plant diseases, pests and create yield maps of areas that were more or less productive.

5 Use-cases

For this paper, we decided to move on with four use cases which represent the requirements of the system.
From the functions to the number of drones corresponding, we identified the following use cases.

5.1 Pest detection

This use case describes the function of detecting the pest or disease over one area by the drone and then spread medicine over the area by a vehicle. For the areas that could not be reached by the drone, an agricultural vehicle will be used to assure the function. The quantity of medicine spread should also be controlled on the area.

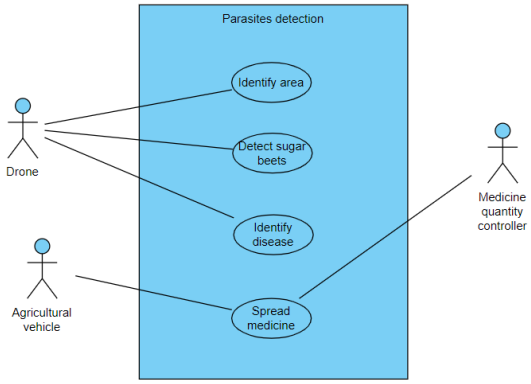


Fig. 1: Use case diagram Pest detection

5.2 Weed detection

This special use case is intended for the function of detecting the different weeds that could be present on an area regarding the species (for our case the drone should be able to identify the black grass, the Charlocks, the cleavers, the common chickweed, the common wheat, the fat hen, the loose silky-bent, the maize, the scentless mayweed, the shepherd's purse, the small-flowered cranesbills and the sugar beet).

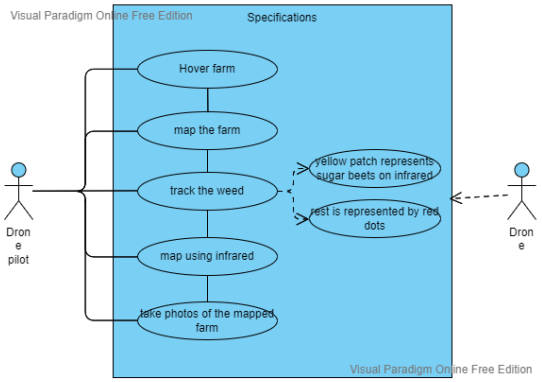
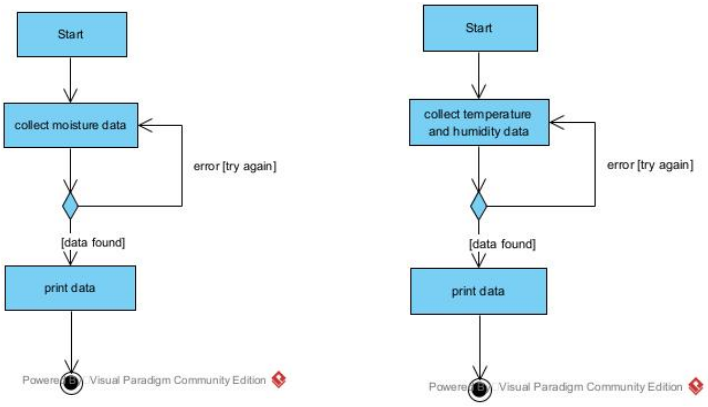


Fig. 2: Use case diagram Weed detection

5.3 Crop and soil monitoring

This use case is intended for the function of identifying both the soil moisture and the temperature/humidity of the soil and then send as the other drones the data to the server.



(a) Soil moisture (b) Temperature and humidity detection

Fig. 3: Activity diagram Crop and soil monitoring

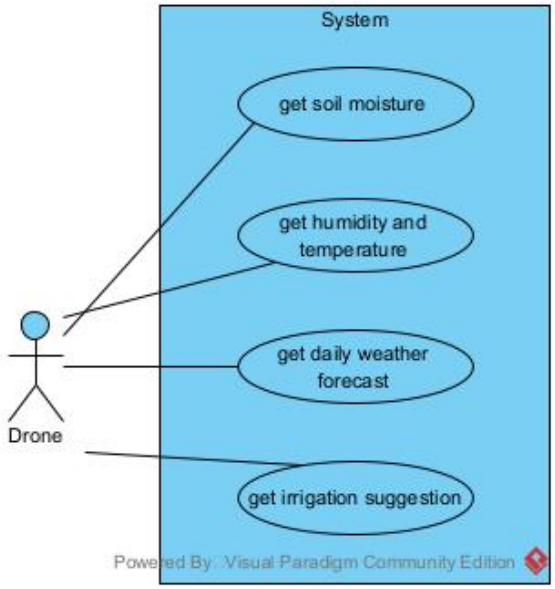


Fig. 4: Use case diagram Crop and soil monitoring

5.4 Plant health monitoring

This use case is for identifying and monitering the health of the plants present over the area. The drone identifies the plant deficiency and sends the diagnosis to the server. For this function, an autonomous vehicle could also be used for restricted areas where drones can't reach out to.

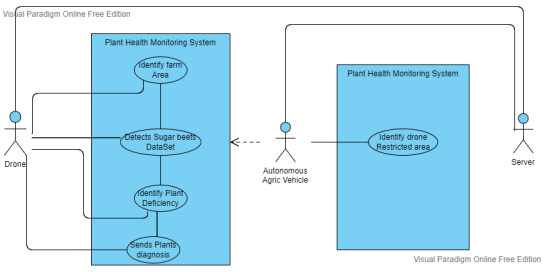


Fig. 5: Use case diagram Plant health monitoring

6 Timed Petri Net

It is essential to plan, design, and evaluate the performance of a discrete event dynamic system. A discrete event dynamic system involves decision-making throughout its life cycle, including planning, design, and operation. Performance modeling's function is to effectively support this decision-making. In planning and decision-making, for example, we consider the number and type of machines, the number of material-handling devices, the number of fixtures, the best layout, the storage capacity for tools, the selection of part types, the grouping of machines, batching and balancing decisions, and the scheduling of jobs.

It is also useful for answering basic design questions such as central storage versus local storage, push production versus pull production, shared resources versus distributed resources, and the effect of flexibility. In addition to enhancing customer confidence, faithful models can be used to make performance predictions. Additionally, modeling tools give system designers and operators improved system insight. Better control strategy design will benefit from this.

The timed Petri net can be used as a logical as well as a quantitative model. In these models, functional/logical properties (such as deadlocks absence) and performance properties (such as average waiting times) can all be specified and validated using the same modeling language.

7 Timed Petri Net Models

Our systems are made up of five (5) implemented models as seen from the use cases listed above, we therefore mapped the use cases into our timed petri net models to carefully verify and validate the tasks to be carried out.

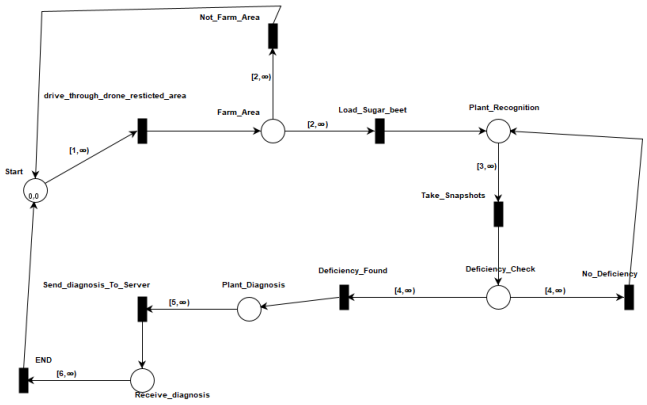


Fig. 6: Autonomous Vehicle

The autonomous vehicle system from figure 6 is responsible for driving and monitoring farm areas that can not be reached and tracked by the drones, this system is synchronised and shares data with the drone as seen in figure 16. The Autonomous vehicle drives through drone restricted area, loads sugar beet, recognises plant, takes snapshots to check for deficiencies in plant, if there are no deficiencies, the system will check again, but if there there are deficiencies, the system would send plant diagnosis to the server.

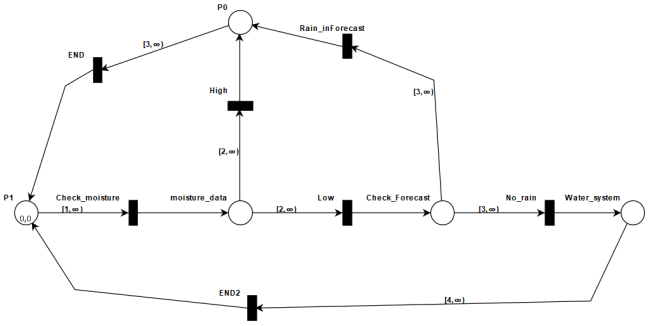


Fig. 7: Moisture Monitoring System

The system fig.7 monitors the soil moisture, gets humidity and temperature data and also gives irrigation suggestions. The system first checks for soil moisture data, if the soil moisture data is high, the system proceeds to restart or end, but if soil moisture is low, the system proceeds further to check rain in forecast. If there is rain in forecast, the system proceeds to end or restart, but if there is no rain in forecast, then the system proceeds to watering the farm area. The system must only water the farm area when soil moisture is low and no rain in forecast.

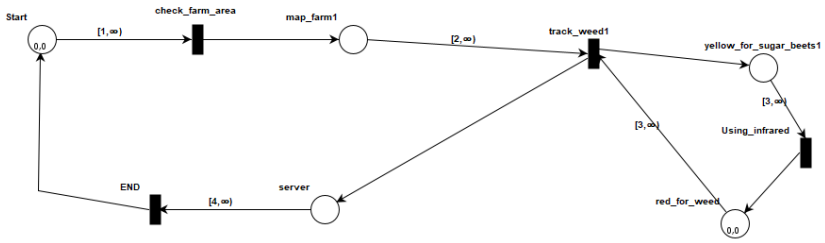


Fig. 8: Weed detection system

This system first checks and map farm area, then it would track or check for weed in the mapped area. Using infrared, the system identifies yellow for sugar beets, and red for weed.

After the weeds are detected, the datas are then sent to the server, where the necessary actions would follow there after.

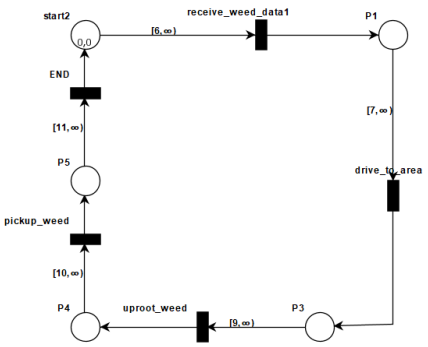


Fig. 9: Weeding system

This system has to do with the weeding of the farm area, the autonomous vehicle picks up the information of the weed that were stored in the server, drives to the weed area, and then proceeds to uproot weed. This system works in accordance to the assigned time constraints and also works based on the information received/stored in the server by the weed detection drone.

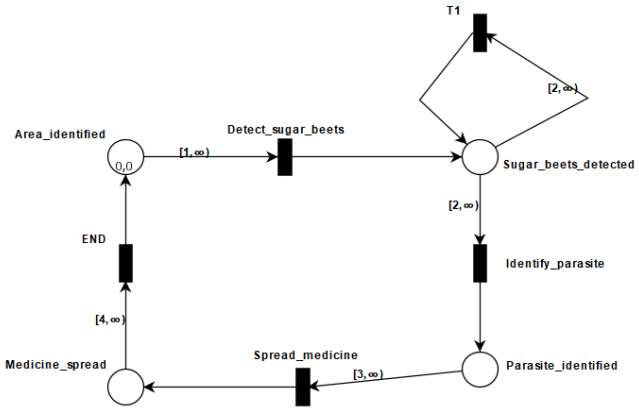


Fig. 10: Parasite Detection System

For this system, the drone would detect sugar beets, if sugar beets are detected, then the

system would proceed to check for parasites. If parasites are found, the system will spread medicines in the area where these parasites were detected.

8 Verification and validation of Timed Petri Net models

For determining the performance of the system in this project we focused on the reachability, the deadlock free performance and the synchronization over petri nets models since they are mainly used to determine the functionality of timed petri nets.

8.1 Reachability

According to [WS14], the reachability graph of a Petri net is a directed graph, G = (V, E), where each node, $\sigma \in$ V, represents a reachable marking and each edge, e $\in$ E, represents a transition between two reachable markings [WS14].
In order to verify the reachability of the system, all the different use cases have been first modelled using the tapaal software as we can see in figure 11 (The figure shows the use case Weed detection)

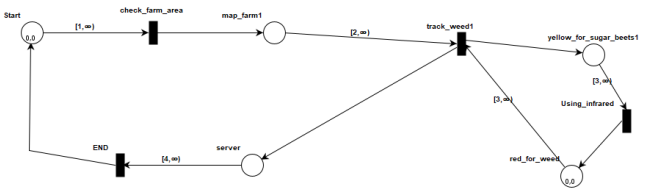


Fig. 11: Petri net example of Use case diagram weed detection

In order to verify the reachability of the use case, as it can be seen in figure 12 in the left panel named "Queries", the reachability "Query"needs to be created by clicking on "new"which redirects to the figure 12.
The "predicate"referring to the entity that has to be verified must be selected and the "quantificationËF has to be selected. When the **EF True** Query is as shown in 11, the next step is to click on SSave and Verify". The figure 13 indicates that the reachability performance has been verified.

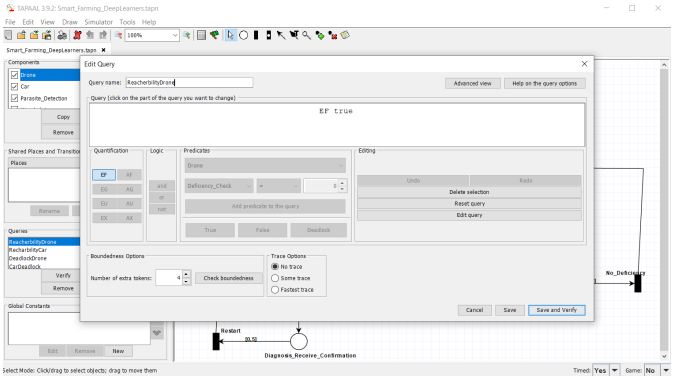


Fig. 12: Reachability verification

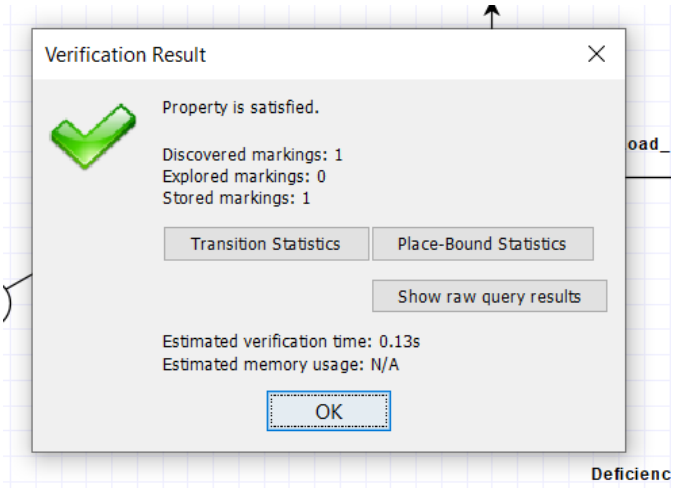


Fig. 13: Reachability result

8.2 Deadlock free

In order to verify the deadlock free of the use case (Drone in this case), as it can be seen in figure 11 in the left panel named "Queries", the reachability "Query"needs to be created by clicking on "new"which redirects to the figure 14.
The "predicate"referring to the entity that has to be verified must be selected and the "quantificationÄG has to be selected and then negated using the logic "not". When the **AG !(deadlock)** Query is as shown in 14, the next step is to click on SSave and Verify". The figure 15 indicates that the deadlock performance has been verified.

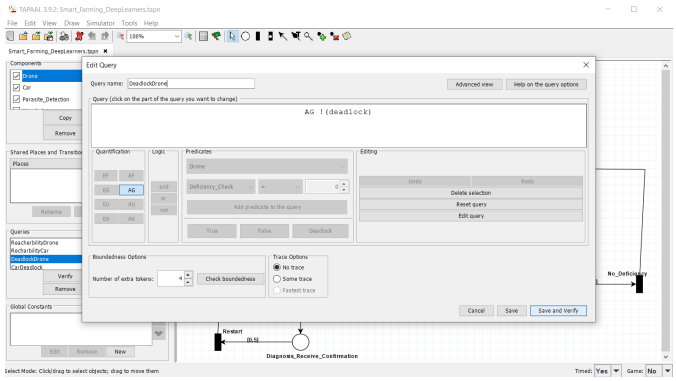


Fig. 14: Deadlock verification

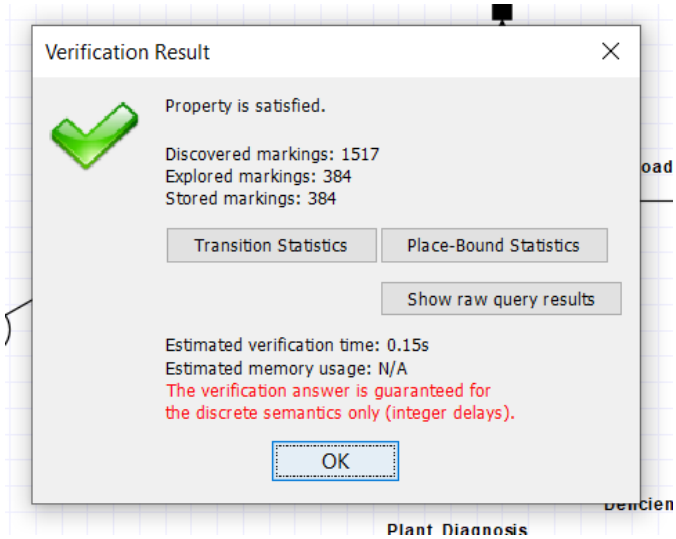
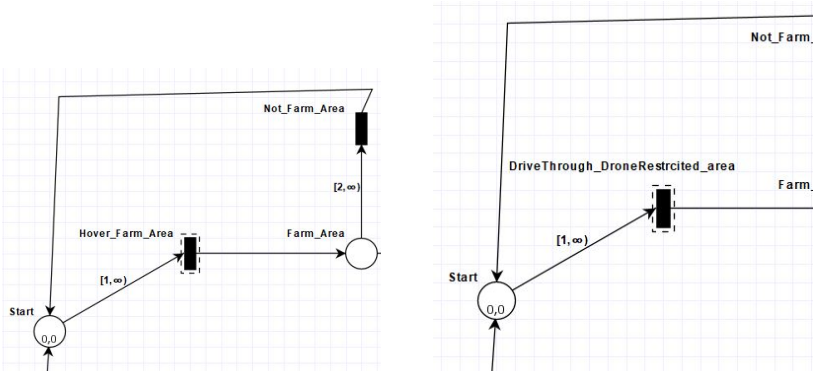


Fig. 15: Deadlock result

8.3 Synchronization

The synchronization of the timed petri nets, the use case of the plant health monitoring has been chosen to explain the synchronization over the petri nets. Knowing from the system that a car is supposed to access restricted areas where the drone cannot access, the switching from the twopetri nets is done at this exact point.
In this situation, switching from one component (petri net) to another will be conditioned by the possibility or not of the drone to reach a specific area.
In the petri net, this change is materialized by a shared transition. As it can be seen in

figure... , the transitions "Hover_farm_areaänd "Drivethrough_DroneRestricted_Areaäre shared as they are surrounded by dotted lines.



(a) Shared transition on Drone component (b) Shared transition on Car component

Fig. 16: Shared transitions example

9 Deep Learning

Deep learning is the sub-field of artificial intelligence that focuses on creating large neural network models that are capable of making accurate *data-riven decisions*. It is basically a neural network using layers and they try to simulate the human brain behavior by its ability to "learn"from large amount of data. Deep learning is particularly suited to contexts where the data is complex and where there is large data-sets available. Today most online companies and high end consumer technologies use deep learning [Ke19].
In this paper, in order to determine the different types of species that could be identified in an area in the context of precision farming, and focusing on our use cases, different models of deep learning have been studied and one has been chosen based on computational performances.

9.1 Convolutional neural network

A convolutional neural network, or CNN, is a deep learning neural network designed for processing structured arrays of data such as images. Convolutional neural networks are very good at picking up on patterns in the input image, such as lines, gradients, circles, or even eyes and faces. It is this property that makes convolutional neural networks so powerful for computer vision. A convolutional neural network is a feed-forward neural network, often with up to 20 or 30 layers. The power of a convolutional neural network comes from a special kind of layer called the convolutional layer. Convolutional neural networks contain

many convolutional layers stacked on top of each other, each one capable of recognizing more sophisticated shapes[Wo19].
The architecture of a convolutional neural network is a multi-layered feed-forward neural network, made by stacking many hidden layers on top of each other in sequence. It is this sequential design that allows convolutional neural networks to learn hierarchical features. The hidden layers are typically convolutional layers followed by activation layers, some of them followed by pooling layers [Wo19].

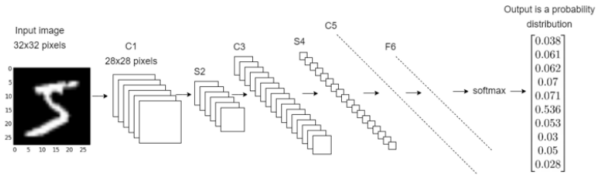


Fig. 17: Convolutional neural network example [Wo19]

9.2 LeNet Model(LeNet-5)

LeNet, one of the first published CNNs to get a widespread notice for its performance on computer vision tasks, will be introduced in this section. The model was developed by Yann LeCun, an ATT Bell Labs researcher at the time, to identify handwritten digits in photographs[Le98]. This effort was the result of ten years of study and technological development. The first work to effectively train CNNs via backpropagation was reported by LeCun's team in 1989[Le89].
LeNet produced exceptional results at the time, matching the performance of support vector machines, which were at the time the leading supervised learning method, and obtaining an error rate of less than 1% per digit. LeNet was ultimately modified to handle deposits in ATMs by recognizing digits. The code that Yann LeCun and his colleague Leon Bottou created in the 1990s is still used by certain ATMs today!

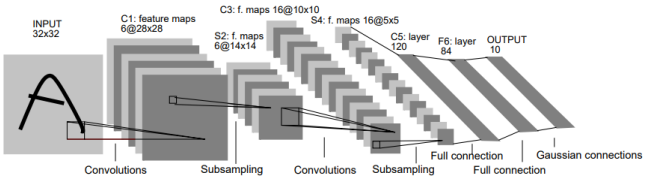


Fig. 18: A LeNet Architecture [Le98]

9.3 Inception v3 Model

Inception Modules are used in Convolutional Neural Networks to allow for more efficient computation and deeper Networks through a dimensionality reduction with stacked 1×1

convolutions. The modules were designed to solve the problem of computational expense, as well as overfitting, among other issues. The solution, in short, is to take multiple kernel filter sizes within the CNN, and rather than stacking them sequentially, ordering them to operate on the same level [De19].
Inception Modules are incorporated into convolutional neural networks (CNNs) as a way of reducing computational expense. As a neural net deals with a vast array of images, with wide variation in the featured image content, also known as the salient parts, they need to be designed appropriately. The most simplified version of an inception module works by performing a convolution on an input with not one, but three different sizes of filters (1x1, 3x3, 5x5). Also, max pooling is performed. Then, the resulting outputs are concatenated and sent to the next layer. By structuring the CNN to perform its convolutions on the same level, the network gets progressively wider, not deeper [De19].

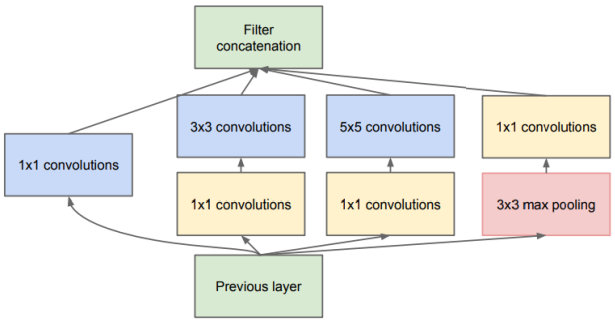


Fig. 19: Inception module [Sz14]

9.4 Resnet50 Model

In their 2015 publication "Deep Residual Learning for Image Recognition,"He Kaiming, Zhang Xiangyu, Ren Shaoqing, and Sun Jian created a particular kind of convolutional neural network (CNN) known as ResNet. Computer vision applications frequently make use of CNNs[da].
Convolutional neural network ResNet-50 has 50 layers (48 convolutional layers, one MaxPool layer, and one average pool layer). Artificial neural networks (ANNs) of the residual kind build networks by stacking blocks of residual information[da].

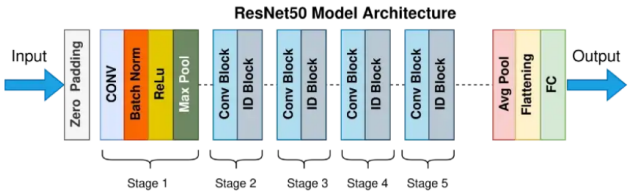


Fig. 20: Resnet50 architecture [Mu22]

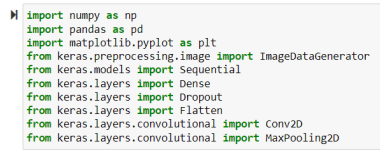
9.5 Implementations

For our project, we have chosen three architectures for our models; LeNet, Inception v3 Model and Resnet50 Model. We have chosen LeNet to be our preferred model over the other two, even though we have been able to implement both models. More details will be given in the comparison section on why LeNet was our preferred model.

9.5.1 LeNet Implementation and Result

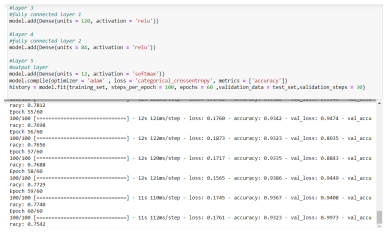
The LeNet Model was chosen as a considerable model for our project based on its well-known performance. The Implementation goes as follows:
We made use of the Jupyta Notebook to develop our python code. Firstly, all the required libraries were imported on the first line; secondly, using the split folder function, we split the main folder into test, validation and test data in 70,20,10(%) ratio on the second and third lines; On the fourth line, using image data generator, we re-scaled the images and changed them into a random changed data. On line five and six, using flow_from_directory, we imported the data from the source and divided it into training data and validation data. On line seven, we used the leNet model; It includes two convolutional layers, two pooling layers, two fully connected layers and one output layer. After that, we compiled the model. We then used Model.fit function to run the model. We Used “adam” as the optimizer and the loss function as categorical cross-entropy. We used 100 as steps_per_epoch and 60 epochs.

The Steps per epoch are calculated as the number of training samples/ batch size. 1 Epochs = 1 cycle (from 1st sample to last sample). After the completed epochs. Accuracy was around 93%. on line eight, model.summary shows all the statistics of the model, including trainable and non-trainable parameters. On line Nine- we saved the created model using model.save function.
Figure 21 below, shows the processes of implementing the LeNet Model.

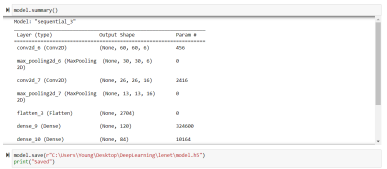


(a) Importing libraries

(b) Train and Test dataset import



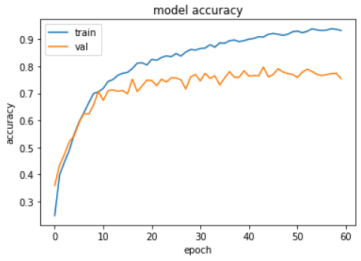
(c) Model implementation



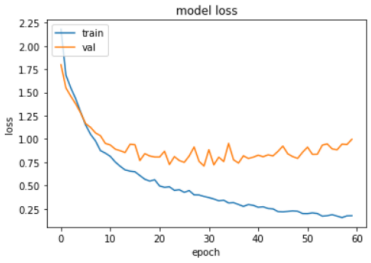
(d) Model summary

Fig. 21: LeNet model

Figure 22 below shows the graph plotting of our model accuracy and loss



(a) Model Accuracy



(b) Model Loss

Fig. 22: Model Accuracy and Loss

9.5.2 Inception v3 Model Implementation and Result

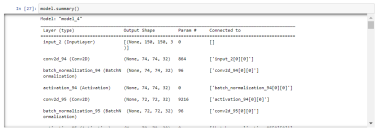
The steps we used as the implementation of the Inception v3 Model could be seen on the python code developed on the jupyter notebook as we can see in figure 23.
On the first line we imported all the required libraries. On the next line, using the "flow_from_directory"method, we imported the data from source divided into training data and validation data. On the next line we imported and implemented the Inception v3 model. We Imported a pre-trained Inceptionv3 loaded with weights pre-trained on ImageNet, used RMSprop as optimizer with learning rate of 0.001 and loss as sparse categorical entropy, used steps per epoch as 30 and epochs to 10 and after completion, accuracy was around

91.7%. In the next line, we presented the model implementation summary, the layers, shape, and parameters for each.

(a) Importing libraries

(b) Train and Test dataset import

(c) Model implementation



(d) Model summary

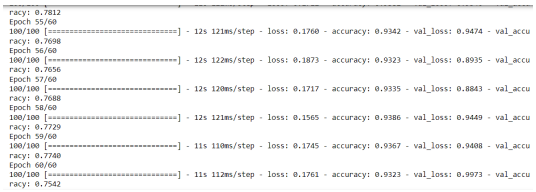
Fig. 23: Inception v3 model

9.5.3 Resnet50 Model Implementation

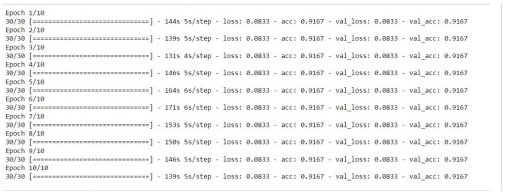
As for the Resnet50 models, for its implementation, it follows almost the same steps as: Importing the libraries and the datasets, importing the ResNet-50 model from the keras library, train the ResNet-50 model and evaluate it, and at the end use the model to classify the images. But due to computational limitations, we could not get a single epoch completed. Due to this limitation, we had to discontinue the implementation.

9.5.4 LeNet Vs Inception v3 Model Comparison

As mentioned above, at the beginning of our Implementation section, we mentioned how we chose the LeNet Model as our preferred model. During the Implementation of both models, as we know that the LeNet model is known to have a high-level performance compared to the Inception Model, it was visibly shown during the implementation of the models how the LeNet model was able to give more accurate testing and significantly reduced losses. The Inception model was static with its output, starting from 91.7% accuracy from the first epoch and ending at the same 91.7% accuracy at the last epoch, the same as the loss. While the LeNet model started from 24% accuracy and 2.17 loss from the first epoch and ended at 93% accuracy and 0.17 loss at the last epoch. These results have shown how efficient the LeNet Model is, which was why we chose it as our preferred model for our project.
Figure 24 below shows the epoch comparisons.



(a) LeNet Model Epochs



(b) Inception Model Epochs

Fig. 24: Epochs Comparisons for both Models

10 Conclusion

Agriculture is one of the significant areas in the world economy, and it is important to foresee its growth and development. This is why different technological approaches must be incorporated to have maximum crop yields for consumption and economic growth. To have precision and automated farming, our study has used different smart farming techniques and tools with deep learning approach. We have used a real-time modelling tool named Timed Petri Net to model and validate our use cases; this allows us to have an overview of our automated farming. We have also used CNN deep learning models, LeNet, Inception and Resnet50, and we finally based our main project on LeNet, which was used to train our datasets to identify and classify plants based on our use cases which are monitoring plant health, soil moisture detection, weed detection and pest detection. Our work can be deepened further in future work and provide more accuracy in training and testing our datasets.

11 Declaration of Originality

We, Abdul-Azeez Olanlokun, Izuchukwu George Enekwa, Patrick Nonki, and Pritish Sanjay Samant, herewith declare that I have composed the present paper and work by myself and without the use of any other than the cited sources and aids. Sentences or parts of sentences quoted literally are marked as such; other references with regard to the statement and scope are indicated by full details of the publications concerned. The paper and work in the same or similar form have not been submitted to any examination body and have not been

published. This paper was not yet, even in part, used in another examination or as a course performance. I agree that our work may be checked by a plagiarism checker.

Bibliography

[Ay19] Ayaz, Muhammad; Ammad-Uddin, Mohammad; Sharif, Zubair; Mansour, Ali; Aggoune, El-Hadi M.: Internet-of-Things (IoT)-Based Smart Agriculture: Toward Making the Fields Talk. 7:129551–129583, 2019. Conference Name: IEEE Access.

[Bl] BLOG: Smart Farming is key for the future of agriculture - Schuttelaar & Partners.

[da] datagen: , ResNet-50: The Basics and a Quick Tutorial.

[De19] DeepAI: , Inception Module, May 2019.

[Ke19] Kelleher, John D.: Deep Learning. MIT Press, September 2019. Google-Books-ID: b06qDwAAQBAJ.

[Le89] LeCun, Y.; Boser, B.; Denker, J. S.; Henderson, D.; Howard, R. E.; Hubbard, W.; Jackel, L. D.: Backpropagation Applied to Handwritten Zip Code Recognition. 1(4):541–551, 1989.

[Le98] Lecun, Y.; Bottou, L.; Bengio, Y.; Haffner, P.: Gradient-based learning applied to document recognition. 86(11):2278–2324, 1998.

[Mu22] Mukherjee, Suvaditya: , The Annotated ResNet-50, August 2022.

[Sc20] Smart Farming: The Future of Agriculture.

[Sz14] Szegedy, Christian; Liu, Wei; Jia, Yangqing; Sermanet, Pierre; Reed, Scott; Anguelov, Dragomir; Erhan, Dumitru; Vanhoucke, Vincent; Rabinovich, Andrew: , Going Deeper with Convolutions, September 2014. Number: arXiv:1409.4842 arXiv:1409.4842 [cs] version: 1.

[VS18] Varghese, Reuben; Sharma, Smarita: Affordable Smart Farming Using IoT and Machine Learning. In: 2018 Second International Conference on Intelligent Computing and Control Systems (ICICCS). pp. 645–650, 2018.

[Wo19] Wood, Thomas: , Convolutional Neural Network, May 2019.

[WS14] White, Steven C.; Sedigh Sarvestani, Sahra: Chapter One - Comparison of Security Models: Attack Graphs Versus Petri Nets. volume 94 of Advances in Computers, pp. 1–24. Elsevier, 2014.

12 Contribution of each member in the paper

1. Abstract and Motivation - Abdul-Azeez Olanlokun
2. Precision Farming - Izuchukwu george Enekwa
3. The need for Precision Farming - Izuchukwu george Enekwa
4. Emerging Technologies - Izuchukwu george Enekwa
5. Use-cases - Patrick Nonki
6. Timed Petri Net - Izuchukwu george Enekwa
7. Timed Petri Net Models - Izuchukwu george Enekwa
8. Verification and validation of the Timed Petri Net Models - Patrick Nonki
9. Deep Learning - Abdul-Azeez Olanlokun, Patrick Nonki, Pritish Sanjay Samant
10. Implementations - Abdul-Azeez Olanlokun, Patrick Nonki, Pritish Sanjay Samant
11. Conclusion - Abdul-Azeez Olanlokun

Github Repository of our Project:

[GitHub Deep Learners]

	#Week	
	Task	Verification and Validation Env.
		• Each group member • Model your own example (should have nothing to do with ‘Precision Farming’) to familiarize with TAPAAL • Verify: There is no deadlock in the model
Izuchukwu George Enekw	To-do	
	Status	Done
Abdul-Azeez Olanlokun	To-do	
	status	Done
Patrick Nonki	To-do	Installed Tapaal, implemented a simple ATM Withdraw task using timed petri
	status	Done
Pritish Sanjay Samant	To-do	
	status	Done

14.10.2022
Milestone 1
Explain example/TAPPAAL • Explain all features of Timed Petri Nets
Installed Tapaal, implemented a simple traffic light task using timed petri net
Done
Installed Tapaal, implemented a simple Coffee machine task using timed petri net
Done
Installed Tapaal, implemented a simple ATM Withdraw task using timed petri net
Done
Installed Tapaal, implemented a simple barber shop task using timed petri net
Done

21.10.2022
Milestone 2
• Get familiar with • Jupyter notebook and JupyterLab
Implemented a test file of a university student, which included grades, and student data.
Done
Implemented a program to read text files and gets the information inside. Also created a program to read xlx/excel file
Done
Created a text file of a university student, which included grades, and student data. (Implementation of reading and writing in a file)
Done
Implemented a test file of a university student, which included grades, and student data.
Done

28.10.2022	25.11.2022
Milestone 3	Milestone 4
Use Case specification • Identify and specify appropriate Use Cases in detail for agricultural tasks, get familiar with • plant data sets	• Model and verify the Use Cases
Modelled a use-case diagram for Soil moisture, weeding and irrigation suggestion	Timed Petri Net model of soil moisture
Done	Done
Modelled a use-case diagram Plant monitoring for all drones and autonomous vehicles.	Timed Petri Net model of plant monitoring
Done	Done
Modelled a use-case diagram for Parasite detection	Timed Petri Net model of parasite detection
Done	Done
Modelled a use-case diagram for weed detection	Timed Petri Net model of weed detection
Done	Done

02.12.2022	
Milestone 5	
• Learn models to be able to identify by using appropriate dl-algorithms • Weed, parasites, diseases etc	
live presentation of the models	
Done	
Created Different DL models using CNN with architectures like Lenet, InceptionV3 along with live presentation	
Done	
Presentation of the deep learning models and architectures	
Done	
Created Different DL models using CNN with architectures like Lenet, InceptionV3 along with presentation	
Done	