

University of Science and Technology of Hanoi



GROUP PROJECT REPORT

Group members: BI12-081 Nguyen Tuan Dat
BI12-026 Nguyen Xuan Anh
BI12-080 Do Tien Dat
BI12-075 Nguyen Quang Dang
BI12-064 Nguyen Dinh Chi

-----**Topic**-----

Rice Field Fertilizer Status Using Multispectral Image

Supervisor: Dr. Nghiem Thi Phuong
Labname: USTH ICTLab

1. Introduction	1
1.1. Context and motivation	1
1.2. Objectives	1
1.3. Report Structure	1
2. Theoretical background	2
2.1. Spectral Reflectances	2
2.2. Image segmentation	3
2.3. Potassium concentration (K)	3
2.4. Machine learning	4
2.4.1. Convolution Neural Network	4
2.4.3. Optimizers	5
2.4.4. Gradient Descent Algorithms	5
2.4.5. The learning rate	5
2.4.6. Dropout	6
2.4.7. U-net model:	6
3. Materials and Scientific Methods	7
3.1. Data Description	7
3.1.1. General Information	7
3.1.2. Materials	7
3.2. Scientific Methods	9
3.2.1. Data Preparation	9
3.2.2. Data Preprocessing	11
3.4. Model Configuration and Training	11
3.5. Model Evaluation	12
3.5.1. Confusion matrix	12
3.5.2. Intersection over Union(IoU)	13
3.5.3. Dice coefficients	14
3.5.4. Accuracy, Precision, Recall, and F1-score	14
3.6. Tools and Library	15
4. Experiment and result	17
4.1. Effectiveness during training	17
4.1.1. Loss and Validate Loss	17
4.1.2. Accuracy during Training	18
4.2. Predict Result	19
4.2.1. Intersection over Union (IoU) and Dice coefficients	20

4.2.2 Confusion Matrix	20
4.3. Conclusion about the model	21
5. Web-Application	22
5.1. Frontend	22
5.2. Backend	22
6. Conclusion	23
References	24

List of Figures

Figure 1 - Spectral reflectance signature of healthy vegetation, dry soil, gray grass litter, water, and snow.....	3
Figure 2 - Convolution neural network.....	4
Figure 3 - U-Net architecture.....	6
Figure 4 - Multispectral image visualization.....	8
Figure 5 - The workflow for developing the model.....	10
Figure 6 - Crop image visualization.....	10
Figure 7 - One-hot Encoded Mask (Ground Truth).....	10
Figure 8 - Illustration of our preprocess workflow.....	11
Figure 9 - Intersection over Union.....	13
Figure 10 - An example of calculating IoU for different bounding boxes.....	13
Figure 11 - Dice coefficients.....	14
Figure 12 - Training and Validation Loss over Epochs.....	17
Figure 13 - Training and Validation Accuracy over Epochs.....	18
Figure 14 - Comparison of Matched Channels - Ground Truth vs Predicted.....	20
Figure 15 - Confusion Matrix.....	21
Figure A.1 - Project homepage.....	25
Figure A.2 - New project page opened after click “new”.....	25
Figure A.3 - Model homepage.....	25
Figure A.4 - New model modal opened after click “new”.....	26
Figure A.5 - ricefield_fertilizer API.....	26

List of Tables

Table 1 - Potassium fertilizer in plot field	9
Table 2 - Hyperparameter value.....	12
Table 3 - Confusion matrix.....	13
Table 4 - IoU and Dice coefficients.....	21
Table 5 - Performance metrics of 3 class.....	22

Abstract

The agricultural industry has played a pivotal role in our economy for a long time. In the center of the traditional practices, rice was and always has been our trademark commodity sustaining inland and international consumers. The growing process, in truth, demands much work and time from the farmers so that the quality of the product can be secured, circulated, and exported to other countries. One of the most important agricultural labor is fertilizer management in the initial stage and later stages of growing rice field crops. Ensuring optimal fertilizer management in these fields is crucial for maximizing crop yield while minimizing environmental impact. However, it is a hardship for farmers to gain the exact amount of nutrients persisting in the soil even after ever-changing weather conditions. Most of them just follow their intuition to adjust the fertilizer level or at best use some old-fashioned, ineffective techniques.

Aware of this ongoing problem, we have applied advanced computer technology, specifically Machine Learning, to resolve this dilemma. Our study's purpose is to provide cultivators with a more effective and exceptional option to manage the fertilization process. To do this, we utilized the semantic segmentation technique through the U-Net model to differentiate the rice field status, more specifically, if the soil is adequate, excessive, or deficient of the examined nutrient.

The result of this study is that we have expectations for it to prove to be useful for anyone working in the agricultural industry as it simplifies one of the most tiresome tasks in farming practices. In our opinion, it has potential to develop a new way of applying fertilizers by supplying the farmers with a convenient tool to carry out their daily growing tasks.

1. Introduction

1.1. Context and motivation

For farmers in general, fertilizer management is one of the most crucial assignments to ensure the quality of growing crops. Efficient nutrient management is helpful when it comes down to increasing crop yields and lessening environmental impact. Still, in reality, it is a challenge for farmers to assess crop health and nutrient status as they usually spend an enormous amount of time and work to manually inspect the field. Even after exhausting labor surveys on a substantial large field, people are incapable of exactly evaluating the amount of nutrients present in the soil and as a result, this may drag down their product quality. In general, the options are limited and they are not particularly effective.

In this project, our task is to develop a Segmentation Model to identify the nutrient density in the crop field. Our datasets are provided by Dr. Tran Giang Son and his team were collected from a rice field in Phu Tho. Our target is to predict the nutrient levels in the rice field based on various factors. In this work, we focus solely on Potassium(K) in the second season. In detail, our model job is to disclose the soil status whether the nutrient present is deficient, adequate, or excessive to adjust the fertilization process accordingly.

1.2. Objectives

The target of this project is to develop a model that is capable of identifying the concentration Potassium (K) based on reflectance data collected from a multispectral image. Furthermore, we make use of semantic segmentation techniques with the intention of precise detection of nutrient concentration in the already mentioned field, more specifically, whether that nutrient in consideration is deficient, adequate, or excessive. Additionally, as our project focal point is model training and evaluation, the web application is just a proof of concept for the model.

1.3. Report Structure

Our report is structured as follows:

Section 1: Introduction	Section 4: Experiment and Result
Section 2: Theoretical background	Section 5: Web application
Section 3: Materials and Scientific Methods	Section 6: Conclusion

2. Theoretical background

2.1. Spectral Reflectances

Spectral reflectance, measuring light reflection across different wavelengths, is crucial for evaluating rice fields using multispectral imagery. It reveals vital information about plant health and nutrient status. Healthy vegetation exhibits distinct reflectance patterns, particularly in the visible and near-infrared regions. Chlorophyll absorption in the blue and red regions is a key factor. Metrics like the Normalized Difference Vegetation Index (NDVI) quantify these patterns, aiding in the assessment of fertilizer status. Remote sensing technologies, such as satellites or drones, leverage spectral reflectance data for efficient and comprehensive monitoring of rice fields.

The variability of results presents when sunlight makes contact with different sorts of surface. Depending on a number of variables, this interaction may involve either reflecting or absorbing sunlight. This behavior is determined heavily by the factors material. Distinct material with unlike properties can manifest in different ways on the direct contact under the sunlight. In addition, spectral reflectance is also influenced by the physical and chemical state of material - for instance, a rough-textured surface is inclined more to reflect less light than a smooth and polished surface.

The reflectance informed us how efficiently the surface bounces back the sun's energy. The angle at which the surface is illuminated by the sun, the direction where they come from, and the polarization of the light, all contribute to how the surface interacts with the sunlight. With reflectances across different wavelengths, there comes the creation of the reflectance spectrum as it gives us a rough idea of the reaction between different colors of lights and the surface. This can be particularly helpful in detecting the amount of N, P and K concentrations in rice leaves based on how they reflect light with different wavelengths for each concentration.

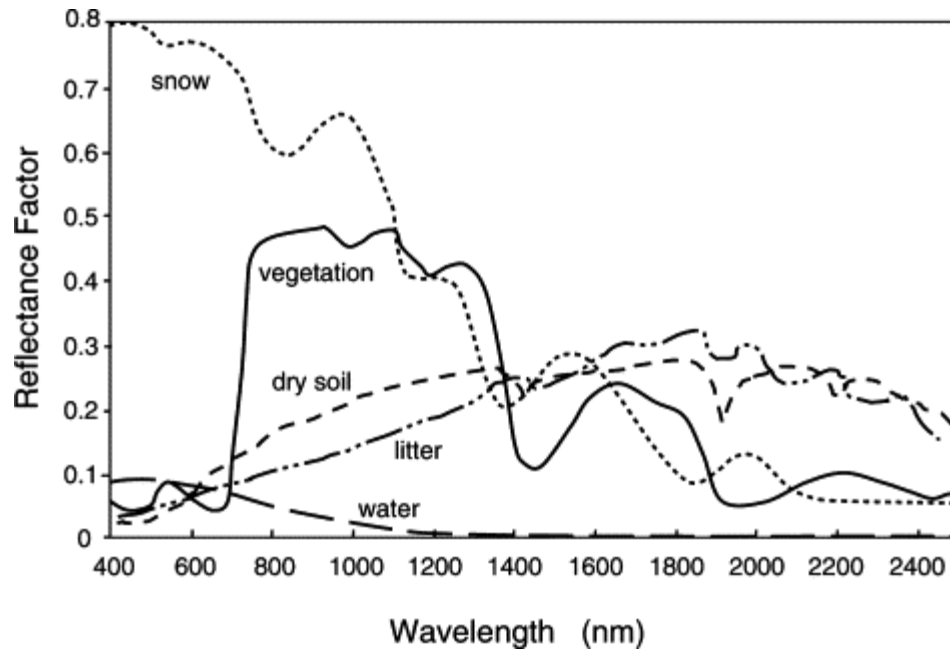


Figure 1 - Spectral reflectance signature of healthy vegetation, dry soil, gray grass litter, water, and snow

2.2. Image segmentation

Image segmentation is a computer vision task that segments an image into multiple areas by assigning a label to every pixel of the image. It provides much more information about an image than object detection, which draws a bounding box around the detected object, or image classification, which assigns a label to the object. Segmentation is useful and can be used in real-world applications such as medical imaging, clothes segmentation, flooding maps, self-driving cars, etc.

Semantic segmentation is a type of image segmentation that involves classifying each pixel in an image into distinct classes or categories. In the context of assessing rice field fertilizer status using multispectral images, semantic segmentation is employed to classify different regions within the images based on their spectral characteristics. The primary goal is to delineate areas with varying nutrient concentrations, providing a detailed and spatially explicit map of fertilizer distribution.

2.3. Potassium concentration (K)

Plants also need Potassium (K) to grow and develop. In physiological processes, Potassium can't be replaced as it is an essential nutrient for plant growth, development, and overall health. It plays a key role in maintaining enzyme activation, photosynthesis, and nutrient transport. Plants acquire potassium from the

soil, and its concentration in plant tissues is crucial for optimal functioning. Adequate potassium levels contribute to improved drought tolerance, disease resistance, and overall crop yield. Insufficient potassium can lead to stunted growth, poor fruit development, and increased susceptibility to stress and diseases.

2.4. Machine learning

2.4.1. Convolution Neural Network

A Convolutional Neural Network (CNN) is a specialized type of artificial neural network primarily utilized for processing visual data. Comprising an input layer, hidden layers, and an output layer, a CNN's intermediate layers are referred to as "hidden" due to the masking effect imposed by activation functions and final convolutions. Within a CNN, these hidden layers specifically involve operations related to convolutions. This commonly involves a layer executing a dot product between a convolutional kernel and the input matrix of that layer.

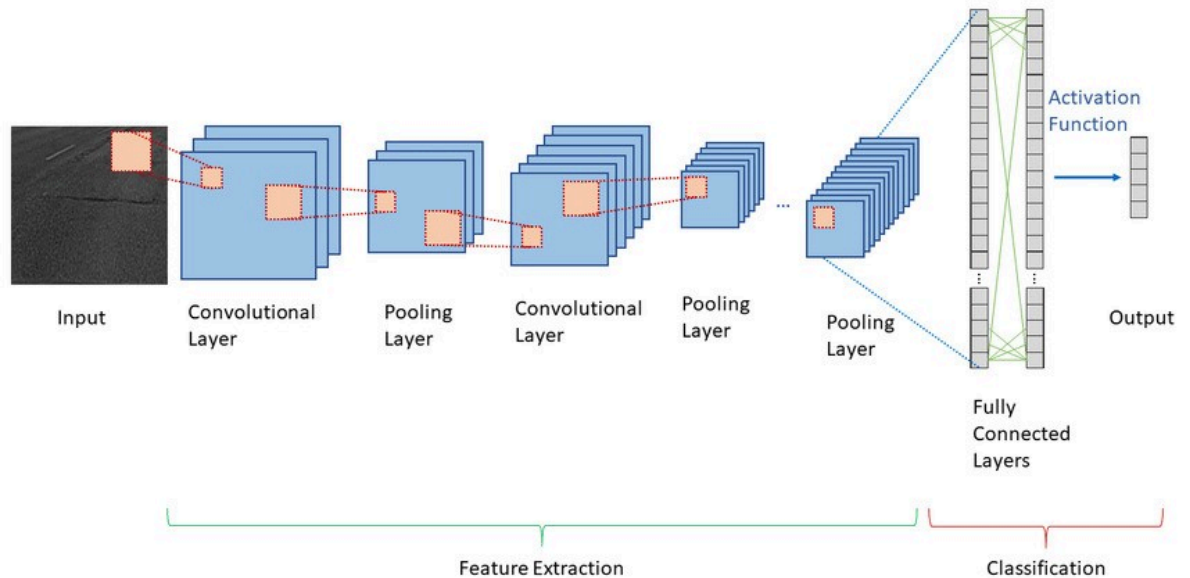


Figure 2 - Convolution neural network

A neural network is fundamentally composed of adjustable parameters, such as weights and biases, akin to those found in a single linear classifier. Consider a basic linear function represented by Equation below. In this setup, an input tensor (like a flattened image vector, particularly when working with images) undergoes multiplication with a suitably sized weight matrix and is augmented with a bias vector. Denoting the input vector as x_i , its corresponding weight matrix as W_i , and the bias vector as b_i , the linear function produces a score, Y_i , given by the equation:

$$Y_i = W_i x_i + b_i \quad (3.1)$$

This function essentially transforms the data x_i from image space into a score, Y_i , specific to that particular image. In the context of an image classification task, Y_i can be interpreted as encoding the confidence of the classifier that the image x_i belongs to a particular label. Conversely, for image segmentation, the problem is framed as per-pixel classification. Therefore, Y_i would signify the score assigned to a specific pixel in the image x_i , indicating its association with a particular class.

2.4.3. Optimizers

In the training process, we adjust and modify the parameters (weights) of our model with the goal of minimizing the loss function, thereby enhancing the accuracy and optimization of our predictions.

Optimizers serve as the link between the loss function and the model parameters, influencing the model based on the feedback from the loss function. Put simply, optimizers refine and sculpt the model, fine-tuning the weights to achieve the most accurate representation possible. The loss function acts as a guide, indicating to the optimizer whether it is progressing in the correct or incorrect direction in the model optimization process.

2.4.4. Gradient Descent Algorithms

Gradient Descent is a widely used optimization algorithm in machine learning due to its speed, robustness, and flexibility. The process involves following steps:

1. Compute the impact of a small change in each weight on the loss function.
2. Adjust each weight based on its gradient.
3. Iterate through steps 1 and 2 until the loss function reaches its minimum.

Understanding gradients, which signify the effect of a slight change in a weight or parameter on the loss function, is crucial for this algorithm and optimizers in general.

2.4.5. The learning rate

Adjusting our weights too rapidly, either by increasing or decreasing them significantly, can impede our capacity to minimize the loss function. It's crucial not to take excessively large steps that might cause us to overshoot the optimal value for a given weight.

To prevent such issues, we introduce a parameter known as the "learning rate." This variable is typically a minute number, such as 0.001, which we multiply with the gradients. This multiplication ensures that any modifications made to our

weights remain relatively modest in scale, preventing drastic jumps and promoting a more stable convergence towards optimal values.

2.4.6. Dropout

Dropout is a regularization technique in machine learning where randomly selected neurons are ignored during training. This helps prevent overfitting by promoting the robustness of the model and improving generalization performance. During each training iteration, a random subset of neurons is dropped out, forcing the model to learn more robust features.

2.4.7. U-net model:

An encoder (for downsampling) and a decoder (for upsampling) with skip connections. As the figure below shows, it shapes like the letter U hence the name U-Net.

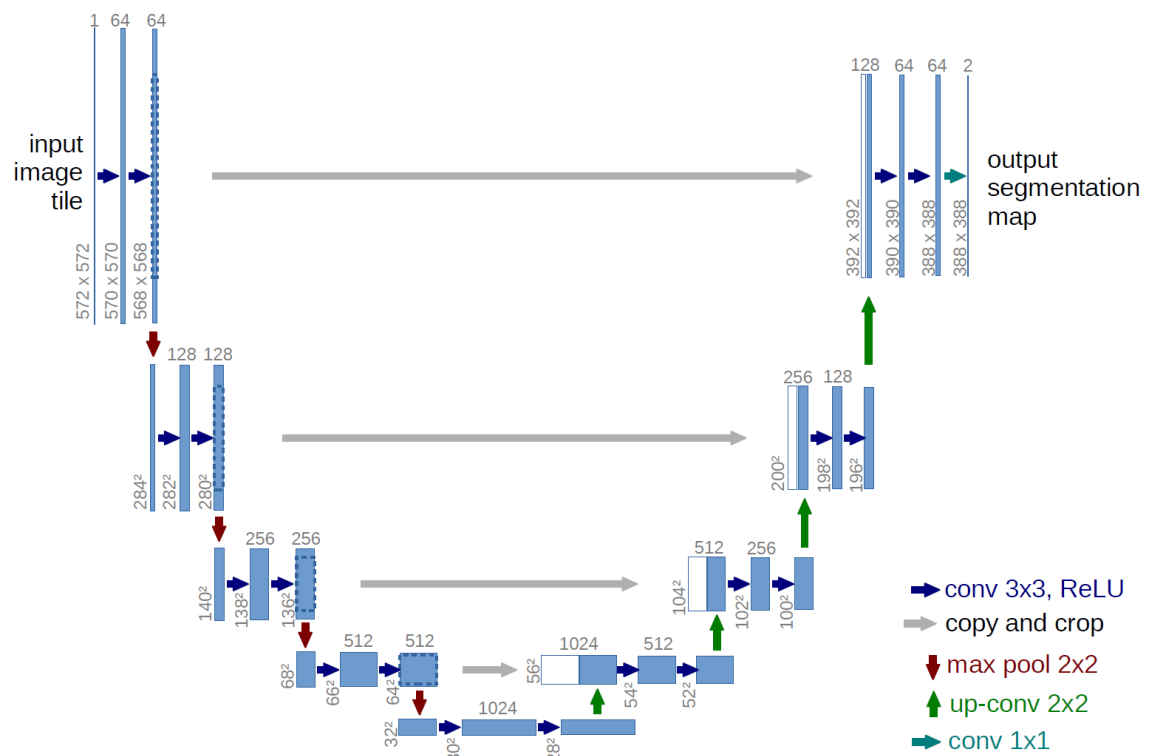


Figure 3 - U-Net architecture

The encoder network - also called a contracting network - plays a role in capturing contextual information and extracting high-level features. There are four encoder blocks in this encoder network. Every block has a Relu activation function after two convolutional layers with a 3*3 kernel size and appropriate padding. A max pooling layer with a 2*2 kernel size receives this as input. After cutting in half

the spatial dimensions learned with the max pooling layer, the computational cost of training the model is lowered as a result.

The decoder network - also called an expansive network - is responsible for transforming feature maps into the size of our input image. This network generates a segmentation mask with the help of skip connections by the feature map taken from the bottleneck layer. There are four decoder blocks in all. Every block begins with a transpose convolution with a kernel size of 2×2 , denoted in the diagram as up-conv. The relevant skip layer connection from the encoder block is concatenated with this output. After that, a Relu activation function is used after two convolutional layers with a kernel size of 3×3 .

The bottleneck layer is located between the encoder and decoder networks. As we can see in the model above, this is the lowest layer. It is composed of two convolutional layers, then Relu. The completed feature map representation is what comes out of the bottleneck.

In the model architecture, a gray arrow designates a skip connection. In order to create our segmentation map, skip connections enable us to use the contextual feature data gathered in the encoder blocks. The idea is to project our feature map (the output of the bottleneck layer) using our high-resolution features that we learned from the encoder blocks (via skip connections).

3. Materials and Scientific Methods

3.1. Data Description

3.1.1. General Information

In this project, we have received Multispectral images taken by UAV over rice fields. The rice fields are divided into 2 areas with 2 different types of rice, and there are 3 types of fertilizers for each field including N, P, K. We were then provided with maps and fertilizer application details for each field. Our job is to combine these resources to create a meaningful dataset for training purposes. With this report, due to limited time, we will only conduct research on the excess data from the second crop season, focusing on Potassium.

3.1.2. Materials

- **Multi_2Cm_Ortho_20220923.tif**

Image Shape: (8231, 7678, 6)

Image Data Type: uint16

Number of Pages (Frames): 7

Dimensions: 7678 x 8231

Resolution (dpi): N/A x N/A

Photometric Interpretation: PHOTOMETRIC.MINISBLACK

Compression: COMPRESSION.LZW

Description: This is the Multispectral Image of the Crop Field, it contain 6 bands

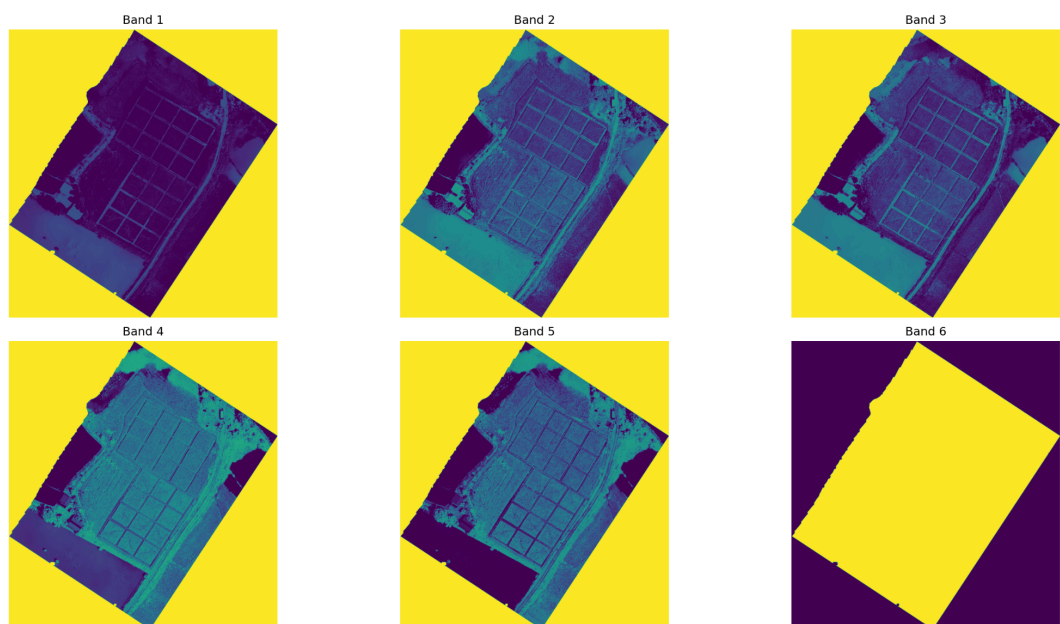


Figure 4 - Multispectral image visualization

- **Fertilizer application details**

In the table below, under the column 'K', the values 0 and 0.5 will be identified as deficient, 1 as adequate, and 1.5 as an excess of fertilizer

Block	K		Block	K
T1	0		J1	0
T2	0.5		J2	0.5
T3	1		J3	1
T4	1.5		J4	1.5
T5	0		J5	0
T6	0.5		J6	0.5
T7	1		J7	1
T8	1.5		J8	1.5
T9	0		J9	0
T10	0.5		J10	0.5
T11	1		J11	1
T12	1.5		J12	1.5

Table 1 - Potassium fertilizer in plot field

3.2. Scientific Methods

3.2.1. Data Preparation

After downloading the Multispectral Image, we prepare the dataset using below workflow

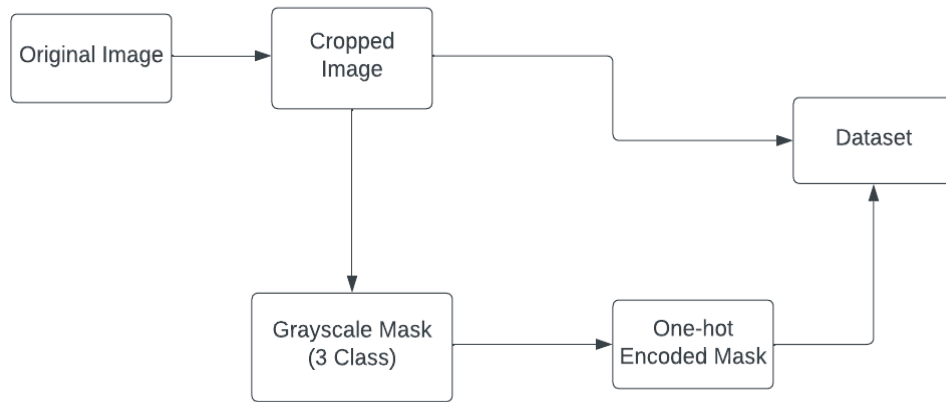


Figure 5 - The workflow for developing the model

Because the size of the original image is too large and to avoid having too much redundant data when training the model, we have cropped the image to remove irrelevant parts of the field, keeping only the main field area for the dataset. This will optimize the training speed and prevent an imbalance between the Background and the main classes.

Note: Band 6 of the Original Image was removed during training as it contained no relevant data.

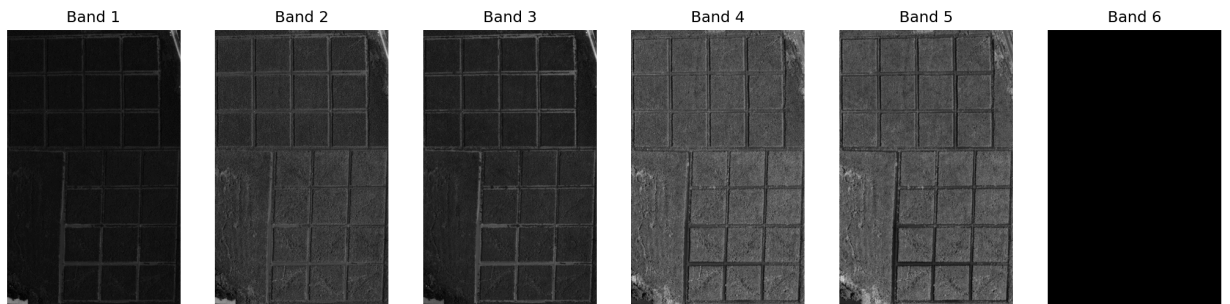


Figure 6 - Crop image visualization

Afterward, we used Photoshop to create a Label Mask by coloring the different classes separately. The Label Mask image was then converted to Grayscale with 4 Unique Labels (0, 1, 2, 3) corresponding to the Background and 3 respective Classes. To serve the Model, the Grayscale image was one-hot encoded into a 4 bands image, each band representing a class (including the background). The band containing the background was then removed to avoid affecting the other Classes during training.

Class 0 corresponds to an excess of fertilizer, Class 1 to an adequate amount, and Class 2 to a deficiency

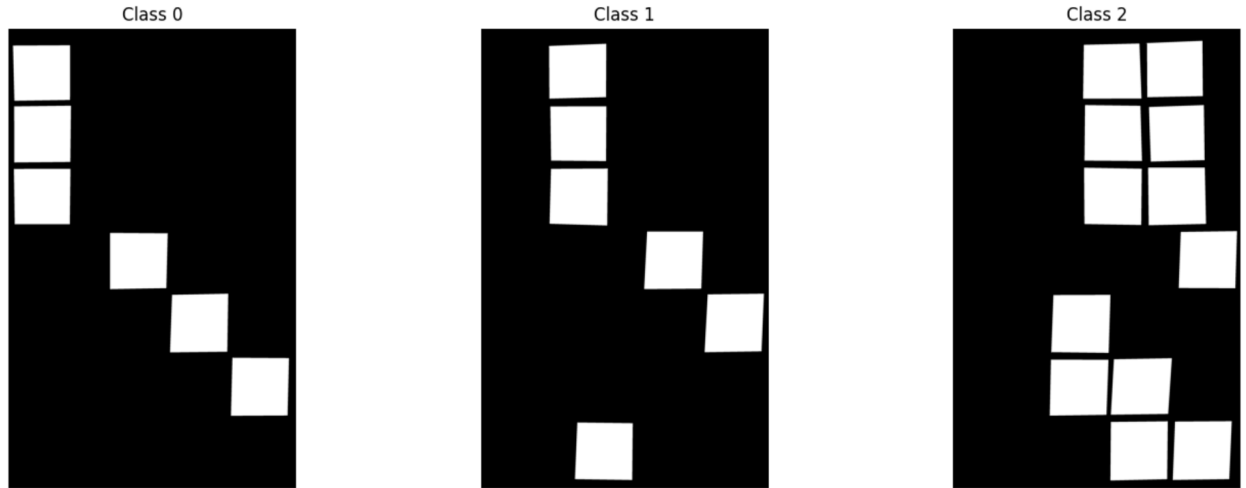


Figure 7 - One-hot Encoded Mask (Ground Truth)

3.2.2. Data Preprocessing

Due to the dataset having a very limited quantity (1 image and 1 mask), we created various patches and transformed them to make the dataset more diverse and meaningful for training. And partly because the system resources are not sufficient to train an overly large image, this could lead to RAM overflow. This process can be understood according to the workflow outlined below

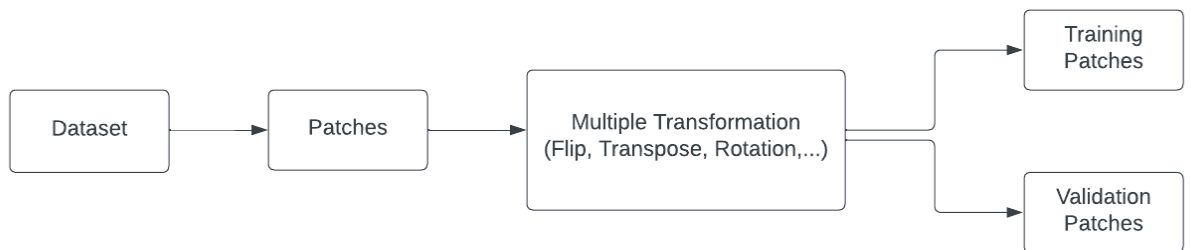


Figure 8 - Illustration of our preprocess workflow

Dividing the image into multiple patches will increase the quantity of training data, which is crucial for deep learning models that often require large datasets to perform effectively. This approach helps in overcoming the limitation of having a small dataset. Each patch can be considered as a separate training example, allowing the model to learn from a more extensive set of variations within the data.

The patches method also allows for easier handling and processing of large images, which can be computationally intensive if used in their original size.

Based on the system resources provided by ICTLab and after several trials, we decided that each patch would be **320x320 pixels**. This size ensures that each patch contains enough data for differentiation and training, while also not overloading the system.

3.4. Model Configuration and Training

As introduced earlier, we will employ the U-net model for our training purposes. The architecture of this model is outlined below, demonstrating its suitability for our specific requirements in image segmentation.

We train our model on a total of 60 epochs. Due to memory limitations of our hardware infrastructure for training, we had to set the training batch size to 8. Any value greater than that causes out-of-memory error.

Due to the instability of the Validation Loss, we had to implement a condition that Checkpoints are only saved after the 40th Epoch. This ensures that the model has learned something substantial during training.

Parameter	Values
Epoch	60
Batch Size	8
Minimum Checkpoint	40
Optimizer	Adam
Loss Function	Binary Cross-Entropy
Learning Rate	1e-4

Table 2 - Hyperparameter value

3.5. Model Evaluation

3.5.1. Confusion matrix

The confusion matrix serves as a tool to evaluate the performance of classification models on a specific set of test data. Its utility relies on knowing the true values of the test data. While the matrix itself is straightforward to interpret, the associated terminology might pose challenges. Often referred to as an error matrix,

it visually represents the model's performance errors. Key features of the confusion matrix include:

- For binary classification with two prediction classes, the matrix is a 2x2 table.
- For scenarios involving three classes, it becomes a 3x3 table, and so forth.
- The matrix is structured into two dimensions: predicted values and actual values, along with a summary of the total number of predictions.

n = total predictions	Actual: No	Actual: Yes
Predicted: No	True Negative	False Negative
Predicted: Yes	False Positive	True Positive

Table 3 - Confusion matrix

3.5.2. Intersection over Union(IoU)

The IoU, also known as the Jaccard index, is an evaluation metric allowing to evaluate the similarity of the predicted mask to the ground-truth mask. In other words, the IoU is used to measure the accuracy of an object detector on a particular dataset by assessing the overlap between the prediction result and the target mask.


$$\text{IoU} = \frac{\text{Overlap}}{\text{Union}}$$


Figure 9 - Intersection over Union

Figure above is a visual explanation of the IoU. The bounding boxes of ground truth and predictions are represented by the two squares. The IoU is calculated as a ratio between the area of overlap and the union area of the predicted bounding box and the ground-truth bounding box. It is an important metric to evaluate the model's effectiveness and is closely related to the Dice coefficient which is often used as a loss function during training.

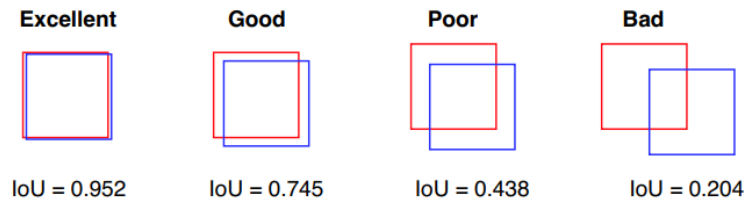


Figure 10 - An example of calculating IoU for different bounding boxes

Figure shows examples of good and bad IoU scores, in which the predicted The bounding box is drawn in blue while the ground-truth bounding box is drawn in red. Provided with a IoU value i and a defined threshold τ , a lung nodule segmentation result can be evaluated as follows:

- True positive: the predicted instance overlaps with the groundtruth with $i > \tau$.
- True negative: the model does not detect any nodule for scans without nodules.
- False positive: the predicted mask has no associated ground truth mask in the input image.
- False negative: the groundtruth mask has no associated predicted mask (prediction missed)

3.5.3. Dice coefficients

Dice coefficient (DC) for each prediction mask over a ground truth mask is defined as the ratio between 2 times overlapped area divided by the total area of both masks

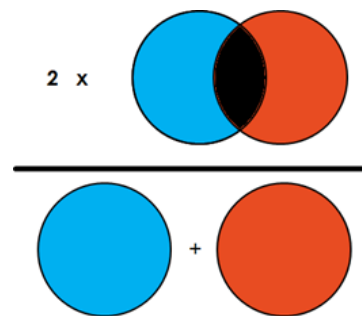


Figure 11 - Dice coefficients

DC is very similar to IoU. However, IoU generally penalizes individual bad classification of a single instance rather than DC. DC measures segmentation accuracy closer to average performance.

3.5.4. Accuracy, Precision, Recall, and F1-score

Accuracy. Accuracy is an aggregation metric, representing the ability to predict true cases in the test set. This metric can be calculated as follows:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision. Precision is an evaluation metric that gauges the accuracy of positive predictions, measuring the ratio of correctly predicted positive cases to the total predicted positives. It can be calculated using the formula:

$$Precision = \frac{TP}{TP + FP}$$

Recall. Recall, also known as sensitivity or true positive rate, assesses the ability of a model to capture all the actual positive cases. It is calculated as below:

$$Recall = \frac{TP}{TP + FN}$$

F1-score. F1-score is a composite metric that balances both precision and recall, providing a single score that considers both false positives and false negatives. It is computed using the formula

$$F1-score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

3.6. Tools and Library

The advancement of machine learning is significantly propelled by the evolution of hardware, particularly the rapid development of GPU processing speed. The group project leverages the ICT Lab's high-performance computing infrastructure, with a critical focus on the GPU. Our hardware setup includes:

- CPU: Intel(R) Xeon(R) CPU E5-2620 v3 @ 2.40GHz, featuring 6 cores and 12 threads.
- GPU: GeForce GTX 1080, equipped with 8GB of VRAM.
- RAM: 32GB DDR.

Complementing our robust hardware, we utilize Python, a high-level programming language known for its efficiency in system integration and emphasis on code readability. Python enables programmers to express complex ideas with fewer lines of code, and it stands out for its extensive collection of standard libraries. These libraries cater to a wide array of application domains, such as machine learning, image processing, web frameworks, and scientific computing. Python's popularity is evident as it is used by virtually all tech giants like Google, Facebook, Amazon, Dropbox, etc. For this project, we are utilizing Python version 3.10.

Our project extensively employs various noteworthy libraries and tools, including:

TensorFlow: An open-source library developed by Google Brain, TensorFlow is pivotal in machine learning and deep learning. It provides a comprehensive ecosystem of tools, libraries, and community resources for cutting-edge ML research and ML-powered application development.

NumPy: This is a fundamental package for scientific computing in Python, supporting large, multi-dimensional arrays and matrices. NumPy is indispensable for numerical computations, offering a broad spectrum of high-level mathematical functions.

TiffFile: A Python library for reading and writing TIFF files, TiffFile is particularly useful for handling large, complex image files in scientific imaging. It supports various TIFF formats and metadata.

Imagecodecs: This library provides algorithms for block-oriented, lossless, and lossy image compression and decompression. It efficiently handles different image formats used in scientific imaging, like TIFF and JPEG.

Matplotlib: A plotting library for Python and NumPy, Matplotlib offers an object-oriented API for integrating plots into various general-purpose GUI toolkits.

It's extensively used for creating a wide range of visualizations, including static, animated, and interactive plots.

Keras: An open-source library providing a Python interface for artificial neural networks, Keras serves as an interface for TensorFlow. Renowned for its user-friendliness, modularity, and extensibility, Keras is a go-to for rapid prototyping of deep learning models, supporting both convolutional and recurrent networks.

For data analysis, exploration, and visualization, we also incorporate **Jupyter Notebook**. Jupyter Notebooks are renowned in the data science and machine learning communities for their flexibility and efficiency in handling diverse programming tasks and visualizations.

4. Experiment and result

4.1. Effectiveness during training

4.1.1. Loss and Validate Loss

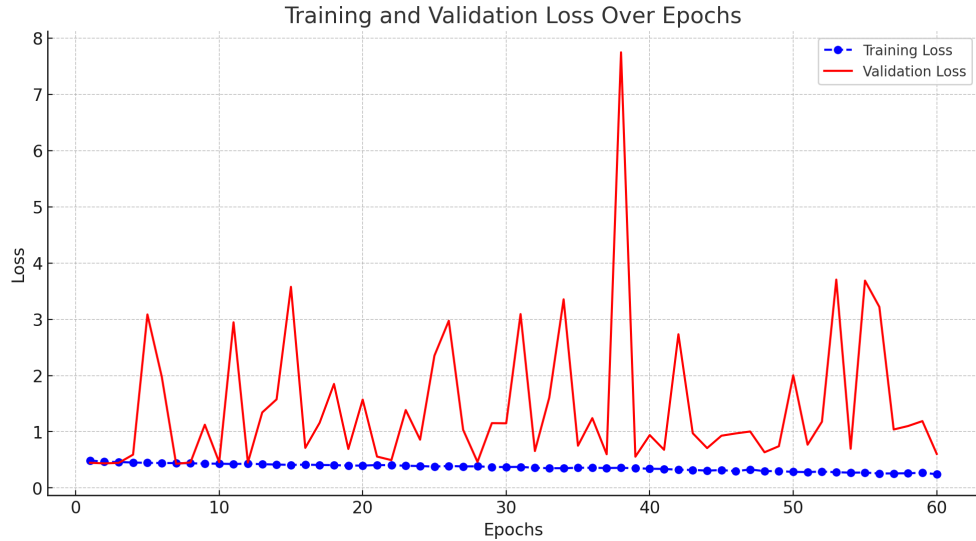


Figure 12 - Training and Validation Loss over Epochs

Overfitting Indication: The training loss is consistently low and does not vary much after the initial epochs. In contrast, the validation loss is much higher and shows significant variation, including spikes at certain epochs. This is a classic indication of overfitting, where the model is learning the training data very well but is not generalizing effectively to new, unseen data.

Data Limitation Effects: The model is trained on multispectral imagery of crops with a very limited dataset (only 1 image), the high variance in the validation loss suggests that the model may be struggling due to the lack of diverse examples to learn from. This can hamper the model's ability to capture the underlying patterns that generalize well.

This could perhaps be improved if we had a more diverse dataset and higher quality image data, which might make the model's ability to generalize to new, unseen data more effective and potentially overcome the current model's issues.

4.1.2. Accuracy during Training



Figure 13 - Training and Validation Accuracy over Epochs

Training Accuracy: The training accuracy, shown in green, generally trends upward throughout the epochs, suggesting that the model is improving its performance on the training dataset as it learns.

Validation Accuracy: The validation accuracy, depicted in purple, is quite volatile. While it does show some improvement over time, the fluctuations are severe. Unlike the training accuracy, it does not exhibit a clear upward trend, indicating that the model's performance on the validation set is not consistently improving.

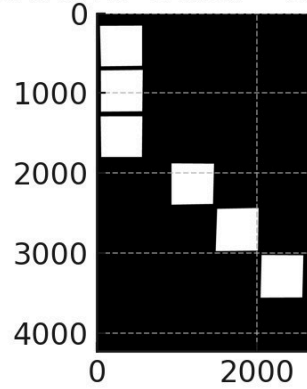
Generalization Gap: There is a noticeable gap between training and validation accuracy, with training accuracy being higher. This gap, especially where the training accuracy is steadily increasing but the validation accuracy is not, could be indicative of the model beginning to overfit the training data.

4.2. Predict Result

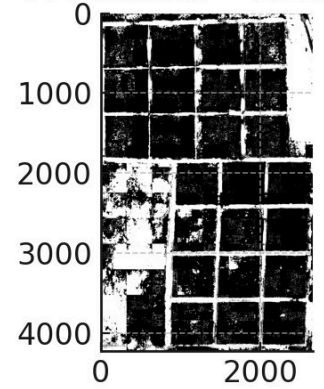
Below are the prediction results we obtained using the output weights after training with the original image at the outset. Class 1 corresponds to an excess, Class 2 to a deficiency, and Class 3 to an adequate amount

Comparison of Matched Channels - Ground Truth vs Predicted

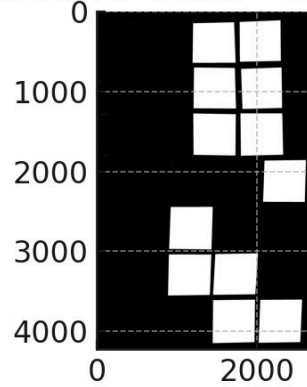
Ground Truth - Class 1



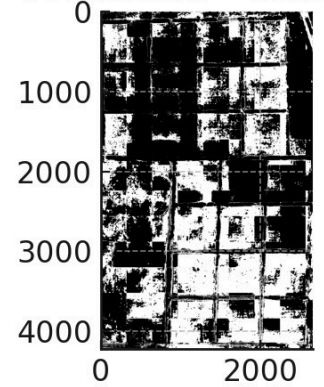
Predicted - Class 1



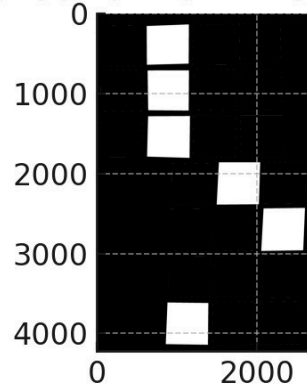
Ground Truth - Class 2



Predicted - Class 2



Ground Truth - Class 3



Predicted - Class 3

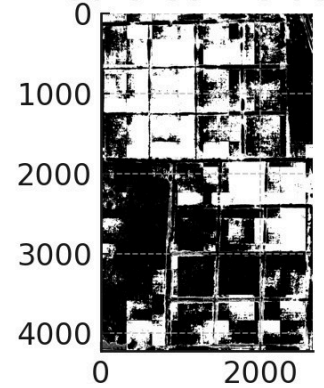


Figure 14 - Comparison of Matched Channels - Ground Truth vs Predicted

4.2.1. Intersection over Union (IoU) and Dice coefficients

	Class 1	Class 2	Class 3
IoU	0.0180	0.3432	0.2658
Dice coefficients	0.0001	0.0020	0.0016

Table 4 - IoU and Dice coefficients

Class 2 shows the highest IoU and Dice scores, indicating better model performance in segmenting this particular class.

Class 1 has the lowest scores, suggesting that the model struggles significantly with accurately identifying and segmenting features of this class.

The generally low Dice coefficients across all classes hint at a model that might have a high number of false positives and false negatives, leading to poor overlap between the predicted and actual segments.

In summary, the model evaluated by this table is performing best on Class 2, with Class 3 being the second-best, and Class 1 showing very poor performance. Both the IoU and Dice coefficients for Class 1 suggest that the model's predictions for this class are almost entirely incorrect. The low values for Dice coefficients across all classes suggest that the model might need significant improvements or retraining to be effective for this segmentation task.

4.2.2 Confusion Matrix

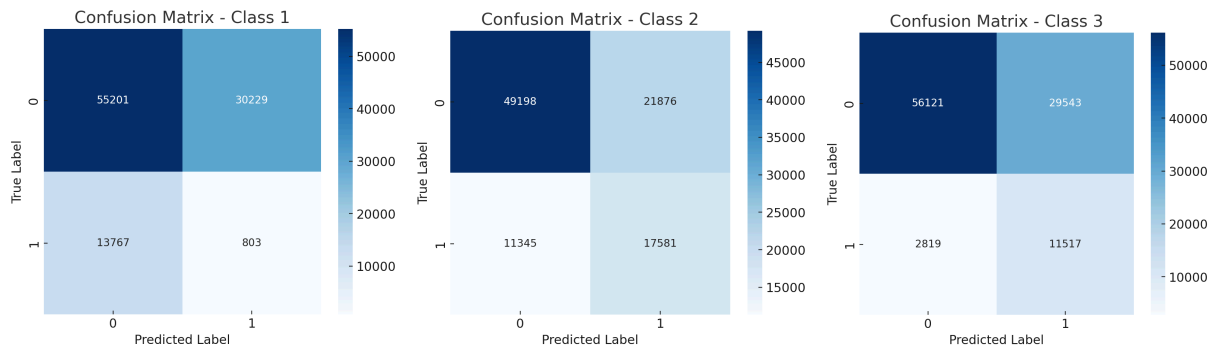


Figure 15 - Confusion Matrix

Class 1

	Precision	Recall	F1-Score	Accuracy
Positive	0.8029	0.6464	0.7162	0.5614
Negative	0.0258	0.0557	0.0352	

Class 2

	Precision	Recall	F1-Score	Accuracy
Positive	0.8103	0.6904	0.7456	0.6652
Negative	0.4427	0.6035	0.5107	

Class 3

	Precision	Recall	F1-Score	Accuracy
Positive	0.9522	0.6600	0.7797	0.6810
Negative	0.2861	0.8046	0.4221	

Table 5 - Performance metrics of 3 class

4.3. Conclusion about the model

After studying the outcomes of the model and its training process, we can observe certain anomalies. These irregularities can be attributed to the environmental conditions and the state of the rice fields at the time of image capture, including weather and the influence of external factors. Additionally, the limited size of the dataset may have contributed to the model's insufficient learning, as the lack of comprehensive data hindered effective training. This deficiency in learning could be a primary reason for the observed chaos in the model's prediction of various classes.

With a limited dataset and a self-developed model, this can be considered an acceptable outcome, given that using imagery to predict nutrient concentrations is still quite novel and there aren't many examples of this practice yet. To improve results, increase efficiency, and optimize the model in the future, we may need more data along with higher quality images. The results of this experimental model can serve as a foundation to demonstrate that the project is entirely feasible in the future.

5. Web-Application

5.1. Frontend

Before talking about Frontend, because we focus more on the Model part, we will briefly talk about the Frontend part.

Our website is designed with two main functionalities: "Model" and "Project".

In the "Model" section, users have the ability to add new models, as well as view and manage existing ones.

As for the "Project" aspect, users can create, view, edit, and delete projects. Importantly, within each project, users can submit data and select a corresponding model. This data is then sent to the server, where it is combined with the selected model to perform predictions and return the predictive results to the user.

5.2. Backend

Quarkus is a comprehensive Java framework optimized for Java virtual machines (JVMs) and native compilation, specifically tailored for containerized environments such as Kubernetes, cloud, serverless. Quarkus is designed to work with popular Java standards, frameworks, and libraries and in this project, we use these extensions: RESTEasy Reactive Jackson, Hibernate ORM with Panache, JDBC driver - MySQL, security JPA, SmallRye JWT build. We chose using Quarkus because it has live reload, which helps me a lot in debugging programs.

In the backend, we create three main packages, which are Login_Register, AdminPage and UserPage. Each package contains models and resources. The models have HTTP model and database model while the resources contain all the functions to create http endpoints. To secure the web, we have used user credentials to sign a JWT, which can allow the user to access the user page and admin access the admin page. In addition, MySQL is where we store all the data about the user, model and project from the frontend.

Note: The demo web application image is in the appendix section.

6. Conclusion

This is just the initial stage in the research and experimentation process of using imagery to analyze the nutrient concentrations of crops, and thus, there are many issues and uncertainties yet to be addressed. However, after this project, we can consider the analysis of nutrient concentrations using images as feasible and potentially applicable and improvable in the future.

In the modern day, the integration of advanced technologies into agriculture has gained a great amount of popularity. Furthermore, plant health is important in agriculture. Providing farmers with these technological advancement can help them optimize their operation, saving both time and money, which are huge support for farmers.

In the future, we want to use more data from the dataset that has been provided. Currently, we only teach the model with the dataset of one season, which only contains data to analyze the excess or deficiency of K in the soil. Using additional datasets from other seasons will help the model analyze the excess or deficiency of N and P as well. This will enable the model to perform better in predicting the nutrient surplus or deficiency in soil fertilizers. Also, studying and applying more models and different kinds of learning algorithms can get a better result than what we got here. After the improvements, we will be able to apply this to solve problems, helping farmers to provide enough nutrition to each field they have.

References

1. Adarsh Maindola, Arishtha Kirti, Milan Kandwal, Mukesh Bahuguna, Deep Learning Based Built-up Extraction Using Multispectral Satellite Imagery.
2. Angad Bajwa, Image Segmentation with U-Net.
<https://www.analyticsvidhya.com/blog/2022/10/image-segmentation-with-u-net/>
3. Vo Thuy Trang, Regression methods for spectral reflectance data
4. Red Hat, the developer of the Quarkus
<https://www.redhat.com/en/topics/cloud-native-apps/what-is-quarkus>
5. Loc Thi Thuy Linh, Image Segmentation Methods for Lung Nodule Detection in CT Scans

Appendix

Frontend

1. Project

GP6ModelProject

Search by Project or Model Name

SearchNew

Project name	Description	
Araxys	Model 1	<div>DetailUpdateDelete</div>
Kuronami	Model 2	<div>DetailUpdateDelete</div>
Prelude to Chaos	Model 3	<div>DetailUpdateDelete</div>
Sentinel of Light	Model 4	<div>DetailUpdateDelete</div>
Neo Frontier	Model 5	<div>DetailUpdateDelete</div>
Xenohunter	Model 6	<div>DetailUpdateDelete</div>

Figure A.1 - Project homepage

GP6

New Project

Project Name

Sentinel of Light

Description

Model 4

Submit

Undo

Figure A.2 - New project page opened after clicking “new”

2. Model

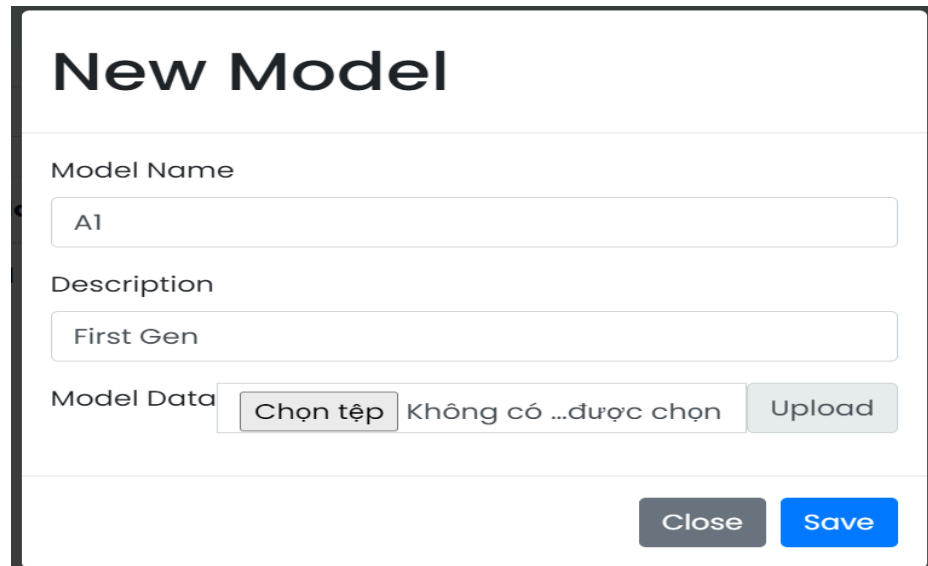
GP6ModelProject

Search

SearchNew

Model ID	Author	Model Name	Description	
1	Chind	AI	First Gen	<div>Delete</div>

Figure A.3 - Model homepage



New Model

Model Name

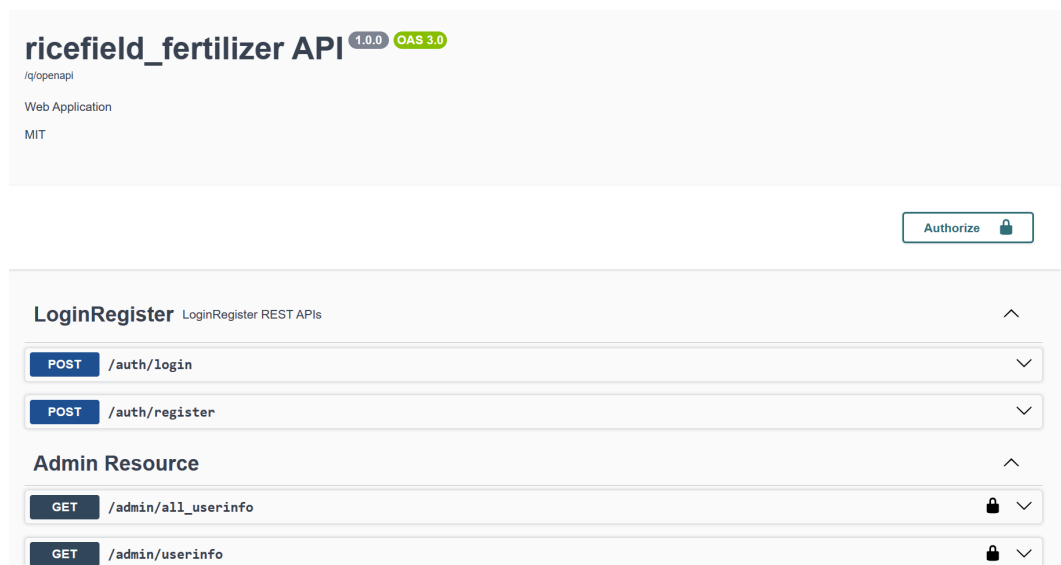
Description

Model Data
 Không có ...được chọn

Figure A.4 - New model modal opened after clicking “new”

Backend

APIs:



ricefield_fertilizer API 1.0.0 OAS 3.0
 /q/openapi
 Web Application
 MIT

[Authorize](#)

LoginRegister LoginRegister REST APIs ^

POST /auth/login v

POST /auth/register v

Admin Resource ^

GET /admin/all_userinfo v

GET /admin/userinfo v

Figure A.5- ricefield_fertilizer API