Design and development of Nodel Deep Learning algorithm Hyperspectral Image Classification

An end-term report submitted towards the fulfilment of the degree of

Bachelor of Technology

In

Computer Science and Engineering

by

Priyanshu Gupta

Roll number: 112001033

Under the kind guidance of

Dr. Satyajit Das

IIT PALAKKAD

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

INDIAN INSTITUTE OF TECHNOLOGY, PALAKKAD

KERALA

May 2024

CERTIFICATE

This is to certify that the work contained in the project entitled "**Design and development of nodel Deep Learning algorithm Hyperspectral Image Classification**" is a bonafide work of **Priyanshu Gupta** (**Roll No. 112001033**), carried out in the Department of Computer Science and Engineering, Indian Institute of Technology Palakkad under my guidance and that it has not been submitted elsewhere for a degree.

Dr. Satyajit Das
Assistant Professor
Department of Data Science
Indian Institute of Technology Palakkad

Contents:

List of Figures:

❖ Objective

The objective of this project is to utilize hyperspectral imaging technology for the precise estimation of soil nutrient levels. Hyperspectral imagery obtained from satellites, which captures a multitude of narrow, contiguous wavelength intervals, offers a comprehensive spectral dataset that holds great promise for assessing the intricate variations in soil nutrient content. This all-around method includes collecting hyperspectral data in a planned way, carefully calibrating it, and using advanced modeling to make sense of it all. It then combines this data with measured soil nutrient values from the same geographical areas.

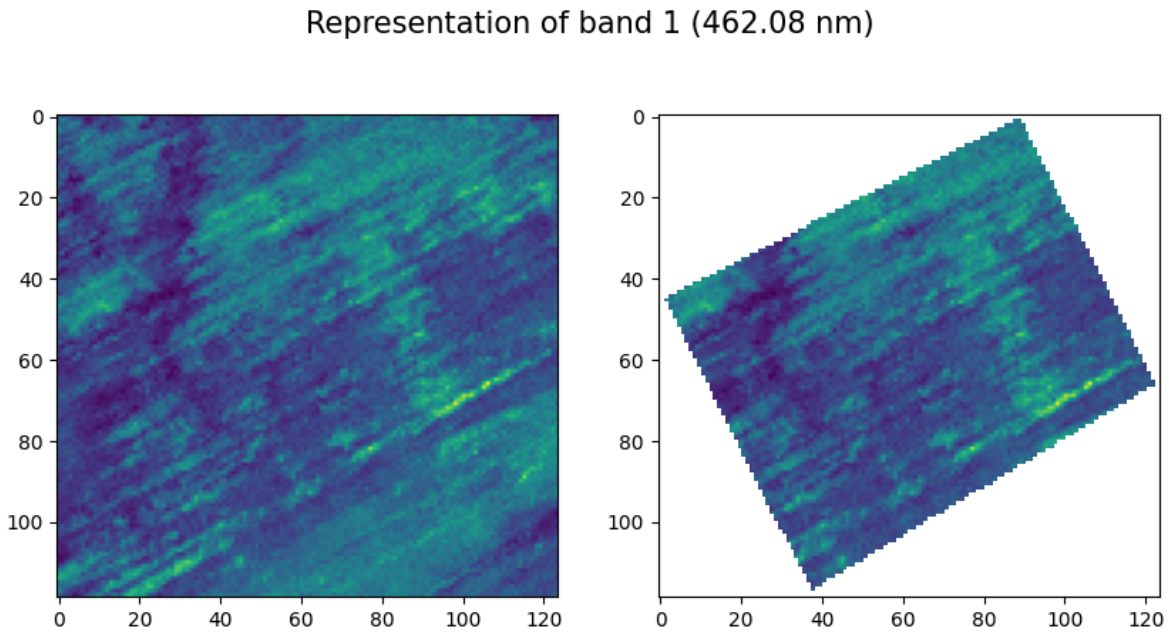
Through the establishment of accurate models and thorough validation processes, this study aspires to generate high-resolution nutrient maps. These maps are envisioned to empower precision agriculture and informed environmental decision-making. By creating a more detailed understanding of soil nutrient content and distribution, this project seeks to facilitate sustainable farming practices. It does so by enabling the precise application of fertilizers based on the specific nutrient requirements of localized soil areas, ultimately leading to optimized crop yields while minimizing the environmental impact.

The project's core focus lies in harnessing the potential of hyperspectral imagery to transform the way we assess and manage soil nutrient levels. This innovative approach holds the promise of revolutionizing agriculture by promoting resource-efficient and environmentally conscious farming practices.

❖ Datasets

1. **Previous Dataset :** The dataset comprises 1732 hyperspectral train images and 1154 test images, each encompassing 150 wavelength bands within the range of 462 to 938 nm. These images have a spectral resolution of 3.2 nm, facilitating precise spectral analysis for soil nutrient estimation.

To gain a closer insight into the dataset, let's examine an individual image that was plotted to represent a single spectral band. This focused examination allows us to observe the specific spectral characteristics and variations within the dataset, providing valuable information for further analysis and model development.



sample_index	P	K	Mg	pH
1731	29.5	146	133	6.3

Fig. 1: Representation of image 1731, corresponding to 462.08 nm band

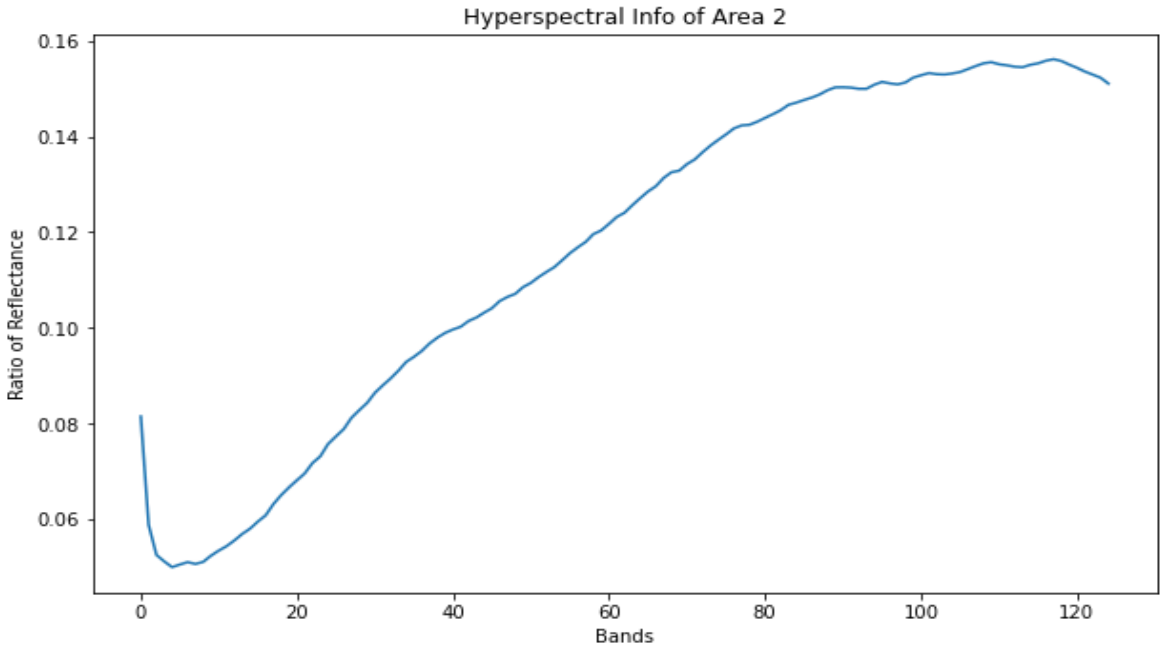
Upon closer examination of the dataset, a notable pattern emerged in the distribution of pixel sizes within the hyperspectral images. A comprehensive analysis revealed that a significant proportion of the images, specifically close to 40 percent, had dimensions of approximately 11x11 pixels. Interestingly, another segment, accounting for about 10 percent, featured dimensions near 50x50 pixels. This pattern was consistently observed in both the training and test image sets, reflecting a non-uniform distribution of image dimensions across the dataset. This observation is of critical importance as it may impact subsequent image processing and the development of accurate models.

The training dataset includes information on four essential soil nutrients, namely phosphorus (P), potassium (K), magnesium (Mg), and soil pH.

2. **New Dataset (Soil Moisture and Soil Temperature Data) :** The dataset comprises 679 rows of hyperspectral data, each encompassing 125 wavelength bands within the range of 454 to 950 nm with a spectral resolution of 4 nm.

This dataset was captured on the same patch of land in a five-day field campaign in May 2017 in Karlsruhe, Germany. An undisturbed soil sample is the centerpiece of the measurement setup. The soil sample consists of bare soil without any vegetation and was taken in the area near Waldbronn, Germany.

The information contains data of soil moisture (in percentage) and soil temperature (in °Celcius). For instance, let's check one row.



sample_index	Soil Moisture	Soil Temperature
2	33.9	36.04

Fig. 2: Representation of data row 2

❖ Progress Beforehand (Phase 1)

Data Augmentation:

Case 1: Finding 11x11 Patch without Masked Values : In the first data augmentation scenario, the algorithm meticulously searches for 11x11 patches devoid of masked values within each original sample. Once a suitable patch is found, it's seamlessly integrated into the augmented dataset. To introduce variability, controlled noise is added to the associated y values, enriching the training data's diversity and bolstering the model's ability to generalize across various scenarios.

Case 2: Creating Patch with Masked Values : When identifying pristine 11x11 patches proves challenging (typically after 10 attempts), the algorithm adopts an alternative strategy. It generates square patches using the minimum pixel dimension of the original sample and fills masked positions with random pixel values from within the same sample. Despite containing masked values, these patches serve as valuable augmentations, enhancing the model's resilience. Similarly, y values associated with these patches are perturbed to further enrich the augmented dataset.

Additional Augmentation: To amplify diversity, the algorithm introduces an augmentation constant, often set to 3. This constant dictates the number of iterations for augmenting each original sample. During each iteration, the algorithm either discovers pristine 11x11 patches or generates square patches with masked values, adjusting the associated y values. The cumulative effect of these iterations results in a robust augmented dataset, laying the groundwork for training a machine learning model capable of capturing intricate patterns within the input data.

Data Preprocessing:

In the preprocessing stage, we extract high-level features from raw field data, transforming the original 150-band spectral information into a set of 16 features per sample.

The process involves computing the average reflectance across all wavelengths, analyzing reflectance variations through derivatives, and performing Singular Value Decomposition (SVD) to understand dominant spectral components. Additionally, wavelet transformations (cAw1, cAw2) are employed, offering a multi-scale perspective on spectral characteristics. These features collectively capture nuanced patterns and variations in the spectral domain, providing a condensed yet informative representation of the field data for downstream machine learning tasks.

Next, we appended all of these values as our features which gave us numpy array of 150 x 16 (dXdl, d2Xdl2, d3Xdl3, dXds1, s0, s1, s2, s3, s4, real, imag, reals, images, cAw1, cAw2) = 2400 length for each sample. Tests were also carried out by normalizing and standardizing the features and values independently.

Models:

I took random 80% samples to be my train data and rest to be the test data but to increase the training data we increased the augmentation layers to be taken into account, so for the training samples, their augmented data was also trained and we had 3 times of the training data, meaning augment constant set to be two.

1. **Random Forest :** Used random forest for predictions with 100 estimators which gave a RMSE of 46.83.

2. **XGBoost :** Used learning_rate=0.1, max_depth=5, alpha=10, n_estimators = 100 and got a RMSE of 46.14.

3. **Support Vector Regression :** Even bad results with RMSE of 54.18 with epsilon to be 0.2.

Layer (type)	Output Shape	Param #
input_11 (InputLayer)	[(None, 2400)]	0
dense_37 (Dense)	(None, 2048)	4917248
dropout_257 (Dropout)	(None, 2048)	0
dense_38 (Dense)	(None, 1024)	2098176
dropout_258 (Dropout)	(None, 1024)	0
dense_39 (Dense)	(None, 512)	524800
dropout_259 (Dropout)	(None, 512)	0
dense_40 (Dense)	(None, 256)	131328
dropout_260 (Dropout)	(None, 256)	0
dense_41 (Dense)	(None, 128)	32896
dropout_261 (Dropout)	(None, 128)	0
dense_42 (Dense)	(None, 64)	8256
dropout_262 (Dropout)	(None, 64)	0
dense_43 (Dense)	(None, 4)	260

Total params: 7,712,964
Trainable params: 7,712,964
Non-trainable params: 0

Fig. 3: Model Summary

4. **Deep Learning :** used sequential models to test RMSE which was 38.54 but it increased on normalizing the parameters to 39.97 and on standardisation to 41.15.

❖ Network Architecture

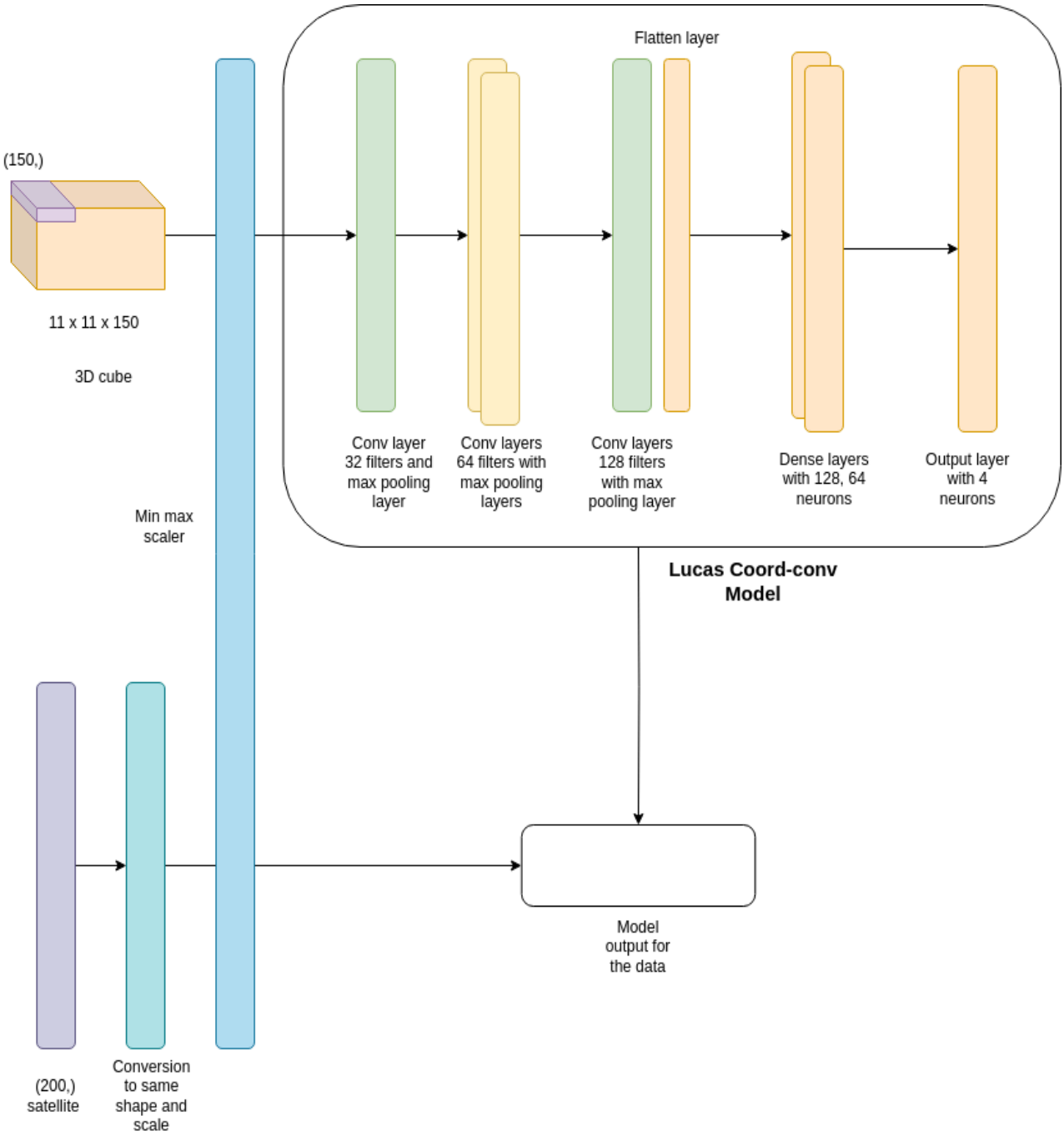


Fig. 4: Network Architecture

All the parts are explained in further detail in upcoming points.

❖ 2D and 3D CNN

Preprocessing 1:

The resizing process using OpenCV (cv2) involves the multiplication of the masked values and data matrix, applying inter area interpolation to adjust the image size to 11 x 11.

During this operation, the mask receives a weight of 0, effectively disregarding its influence on the final result, while the actual data is assigned a weight of 1, ensuring its preservation.

This approach maintains the integrity of the real data while accommodating the resizing requirements, effectively handling masked regions with minimal impact on the overall transformation outcome.

Preprocessing 2:

At first, the smallest patch was of size 11 x 11, which constituted around 650 images. So for the selection of a consistent image size, an 11 x 11 size was chosen.

Next, for the selection of the threshold number of pixels that needed to be unmasked for consideration of the patch for training, we again looked into the smallest images and found the minimum number of unmasked pixels out of 121.

Further, upon iteration over all images, we divided each patch into 11 x 11 multiple areas, and whichever area had more unmasked pixels as compared to the threshold was taken into consideration with the same labels as the bigger patch.

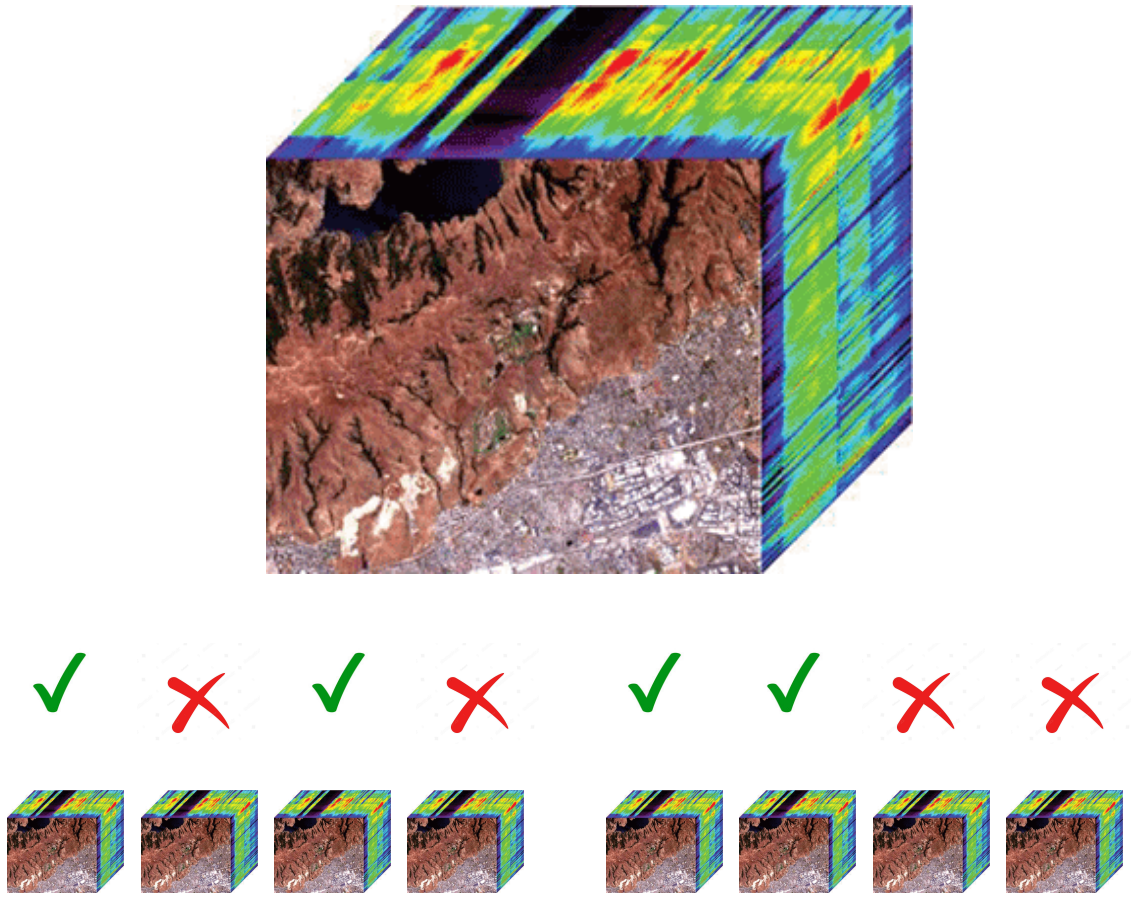


Fig. 5: Breaking big patch in small patches

Finally, for each patch, data was distributed in training and testing in an 80-20 ratio.

In total, we got a 29383 x 11 x 11 x 150 array as our data.

Models :

1. 2D CNN :

As the images were in 3D, the 150 wavelength bands were taken as channels in the case of 2D CNN, and image size was fixed to be 11 x 11.

- Sequential model defined using TensorFlow's Keras API for deep learning.
- Comprises convolutional layers followed by average pooling layers for downsampling and feature extraction from 2D input data.
- Includes two convolutional layers with 32 and 64 filters and a (3, 3) kernel size, employing ReLU activation.
- Utilizes average pooling layers with a (2, 2) pool size after each convolutional layer to reduce spatial dimensions while retaining features. Additionally, incorporates two dense layers with 64 and 4 neurons for further processing.

2. 3D CNN :

Moving to 3D, as we have only reflectance values for each pixel corresponding to a band and image size to be the same, the input was given a shape of 11 x 11 x 150.

- Utilizes TensorFlow's Keras API to construct a sequential model for 3D convolutional neural networks (CNNs).
- Incorporates 3D convolutional layers with 32 and 64 filters, and a (3, 3, 3) kernel size, employing ReLU activation functions for feature extraction.
- Applies average pooling layers with a (2, 2, 2) pool size after each convolutional layer to downsample and preserve relevant features.
- Includes two fully connected dense layers with 128 and 4 neurons respectively, activated using ReLU and linear functions, for regression tasks.

3. HybridSN 3D CNN:

- Sequentially integrates 3D convolutional layers with varying filter sizes (8, 16, 32, 64) and kernel sizes ((3, 3, 7), (3, 3, 5), (3, 3, 3), (3, 3, 3)) for hierarchical feature extraction.
- Incorporates fully connected dense layers with 256 and 128 neurons respectively, utilizing ReLU activation functions and dropout regularization (40% dropout rate) to mitigate overfitting.
- The output layer consists of four neurons with linear activation, making it suitable for regression tasks on volumetric data.

4. HybridSN 3D-2D CNN:

- Utilizes a combination of 3D convolutional layers followed by a reshaping operation to accommodate subsequent 2D convolutional layers.
- Implements a 2D convolutional layer with 64 filters and a (3, 3) kernel size to process the reshaped output of the 3D convolutional layers.
- Employs fully connected dense layers with 256 and 128 neurons, incorporating ReLU activation and dropout regularization (40% dropout rate) to prevent overfitting.
- Concludes with an output layer consisting of four neurons with linear activation, suitable for regression tasks on hybrid 2D and 3D data.

	2D CNN		3D CNN		HybridSN 3D CNN		HybridSN 3D-2D CNN	
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
Preprocessing 1	34.66	21.97	34.02	22.3	33.68	20.9	36.6	21.38
Preprocessing 2	35.88	22.78	41.43	27.24	56.4	30.16	38.24	22.67

Fig. 6: Results of model testing

❖ 1D CNN

Preprocessing :

Based on the results, it appears that the correlation among the pixels is not good enough. So, thought comes to finding correlation between bands. For this the following was done.

1. **Iterating over Patches:**

- We iterate over all the patches in the images.
- For each unmasked pixel within a patch, we extract a 1D vector of length 150, representing the spectral values across bands.

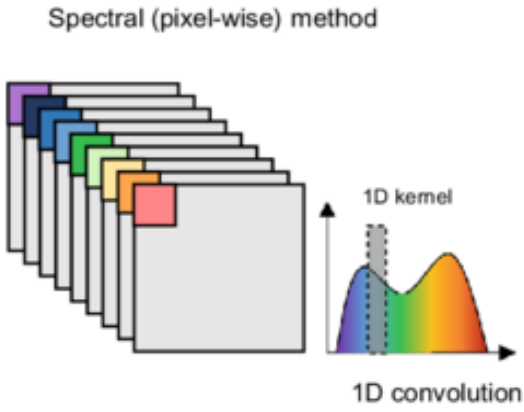


Fig 7: Taking 1d unmasked vectors out of big 3d patch

2. **Data Preparation and Train-Test Split:**

- To combine data preparation and the train-test split, we can iterate over all patches, extract the 1D vectors for each unmasked pixel, and split these vectors into training and testing sets using an 80-20 ratio.
- This approach streamlines the process by performing both data preparation and train-test split simultaneously, ensuring that the model is trained and tested on appropriately partitioned data.

3. **Another Trial With Differentiation:**

- In accordance with the prior work, we also checked the results by adding first order and second order derivatives along with the 150 vector.
- Total size of the vector becomes 450 and further 1D CNN models were trained on this data also.

Models :

1. **LucasCNN:**

- Sequential Keras model implementing a 1D CNN architecture.
- Features four convolutional layers with 32 and 64 filters, followed by max-pooling layers.
- Output is flattened and processed through two fully connected layers with 120 and 160 neurons.
- Final layer has four neurons with linear activation, suitable for regression tasks.

2. HuEtAl:

- 1D CNN architecture proposed by Wei Hu et al. (2014).
- Consists of a single convolutional layer with 20 filters and tanh activation.
- Utilizes specific parameter settings for kernel size and pooling to optimize feature extraction.
- Output is flattened and connected to a dense layer with 100 neurons, followed by a final output layer with four neurons for regression.

3. LiuEtAl:

- 1D CNN architecture proposed by Lanfa Liu et al. (2018).
- Consists of four convolutional layers with 32 and 64 filters, followed by max-pooling.
- Flattened output is connected to a dense layer with four neurons, suitable for regression.

4. LucasCoordConv:

- Variant of LucasCNN incorporating CoordConv layers.
- Employs four convolutional layers with 32, 64, and 128 filters.
- Uses CoordConv layers in the first convolutional layer to incorporate coordinate information.
- Flattened output processed through two fully connected layers before the final regression output layer.

5. SimpleCNN:

- A simple sequential model featuring 1D convolutional layers.
- Comprises two convolutional layers with 64 filters and a kernel size of 3, followed by max-pooling.
- Flattened output is passed through a dense layer with 64 neurons and ReLU activation.
- Final output layer has four neurons, suitable for regression.

	LucasCNN		HuEtAl		LiuEtAl		Lucas Coordco		Simple CNN	

							nv			
	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE	RMSE	MAE
1d normalised (150)	32.3	18.8	37.2	22.6	34.14	20.35	32.5	19.01	33	19
1d normalised (450)	32.09	18.9	33.88	20.31	33.16	19.69	31.55	18.27	32.8	19.44

Fig 8: Model Results

FOR DATASET 2 (SOIL MOISTURE AND SOIL TEMPERATURE):

Having the data to be in the format of 125 reflectance value vectors for each reading, our work of preprocessing for applying 1D CNN was relieved.

After normalization, applying all 5 models to the dataset yielded the results as follows:

Here blue ones are true values and orange is the predicted output for temperature.

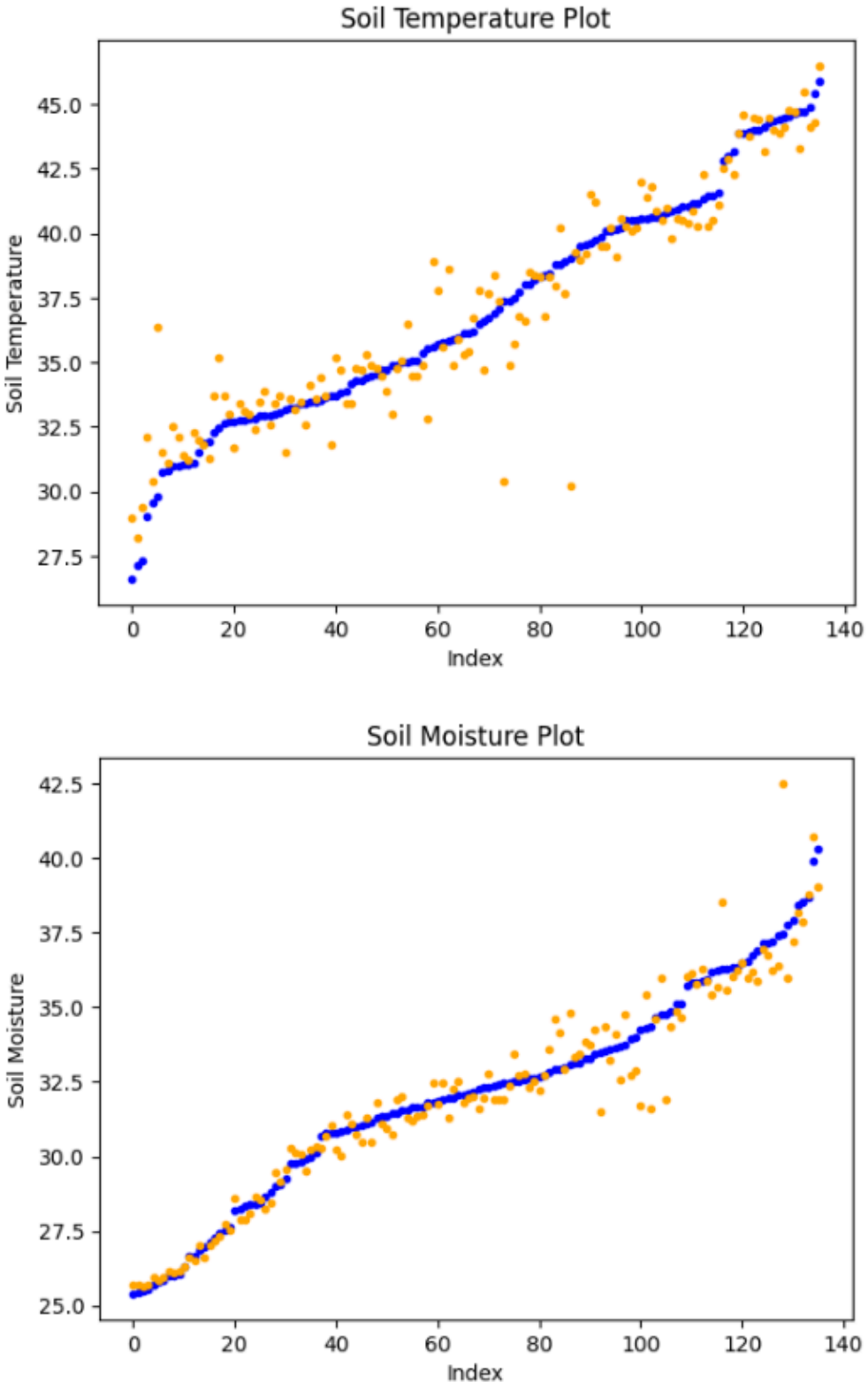


Fig 9: Predictions Plot

LucasCNN		HuEtAl		LiuEtAl		Lucas coord conv		Simple cnn		PCA number of components
MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	
3.4	1.31	0.98	0.68	2.1	1.1	1.49	0.85	0.85	0.82	normal
7.13	1.91	2.82	1.18	4.75	1.75	6.72	2.1	2.79	1.29	pca 2
2.43	1.15	2.11	1.03	6.88	2.24	2.19	1.1	1.66	0.89	pca4
3.11	1.48	1.13	0.7	3.9	1.5	1.16	0.78	1.63	0.96	pca 6
1.37	0.82	1.05	0.67	3.04	1.27	1.08	0.66	1.05	0.65	pca 8

Fig 10: Model Results

❖ Random Forest and HIST

Random Forest : Using the random forest regressor for training on the data using 200 estimators gave us mse 1.06 and mean absolute error to be 0.66 and moreover the plots are as follows.

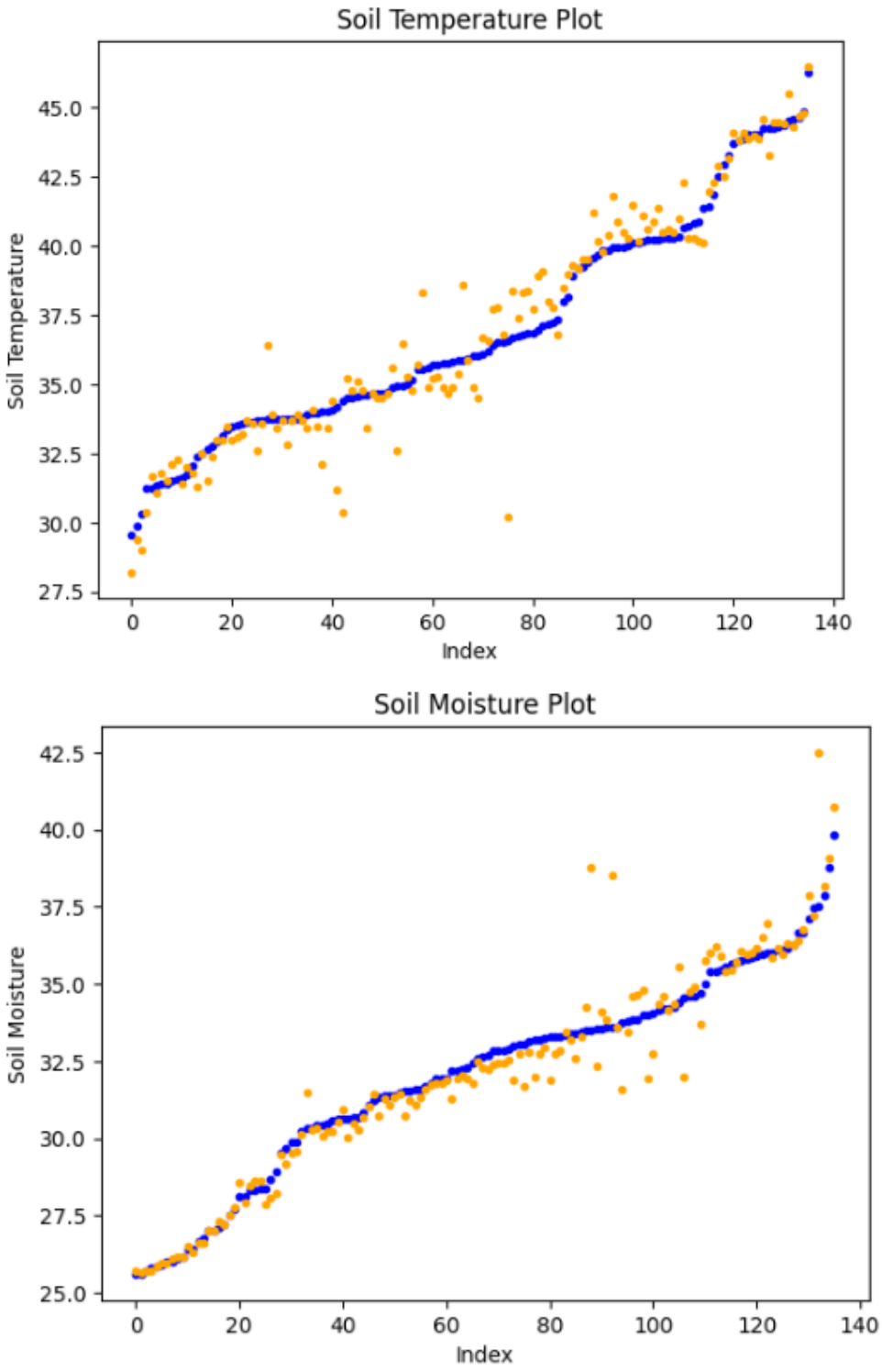
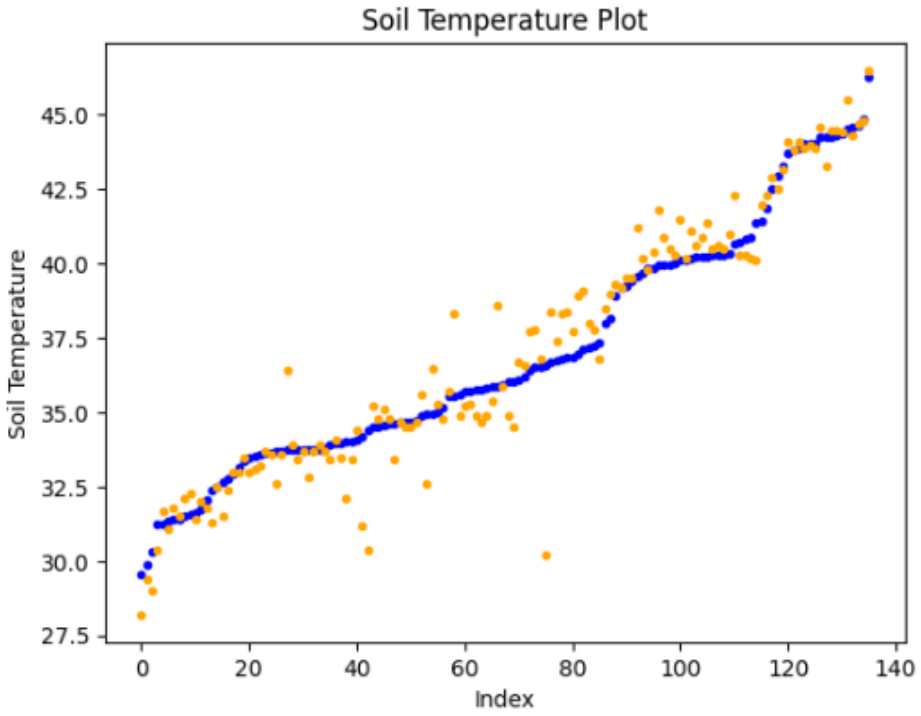


Fig 11: Predictions Plot

HIST Gradient boosting regressor : We got mse of 1.38 and mae of 0.72 while testing for soil temperature.



Moreover, while testing for soil moisture we got 0.87 and 0.45 mse and mae respectively.

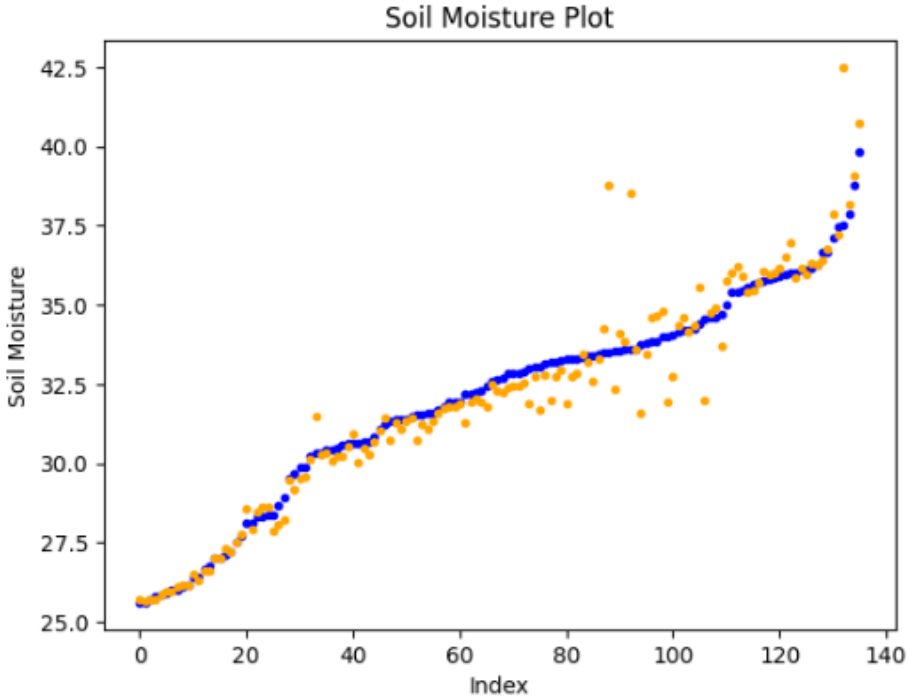


Fig 12: Predictions Plot

❖ Satellite Testing

In our project, we embarked on testing the efficacy of our models using real-time data obtained from the ARSET Satellite. Our primary dataset of interest was the EO-1 Hyperion dataset, renowned for its comprehensive coverage of patches from various geographic regions across the globe spanning the years 2000 to 2017. However, despite its extensive coverage, we encountered a significant challenge: the difficulty in accurately identifying patches containing agricultural soil.

Fig 13: EO-1 Hyperion (Credits-google)

Given that each pixel in the EO-1 Hyperion dataset represents an area of 30m in width, pinpointing agricultural soil patches proved to be a daunting task. Agricultural soil, characterized by its diverse spectral signatures and subtle variations, presented a formidable obstacle in our analysis.

To address this challenge, we embarked on a meticulous exploration of multiple areas within the dataset. Leveraging advanced scaling techniques, we meticulously prepared our data for analysis, ensuring that it met the stringent requirements of our models.

Despite our best efforts, the elusive nature of agricultural soil continued to evade easy detection.

The models we used were 1D CNN in both cases, they were close to the best results after standardization. Both models and scalers were saved and used for satellite testing.

Demonstration 1:

For seeing the performance of our models on the ARSET Dataset, we made a colab ipynb which plots the map and takes the coordinates as input and give a patch captured by satellite on top of the area which is captured.

The place where we click, we obtain the band values of that place 30 m in a dimension and of band values ranging from 400-2000 nm with 10 nm spectral resolution.

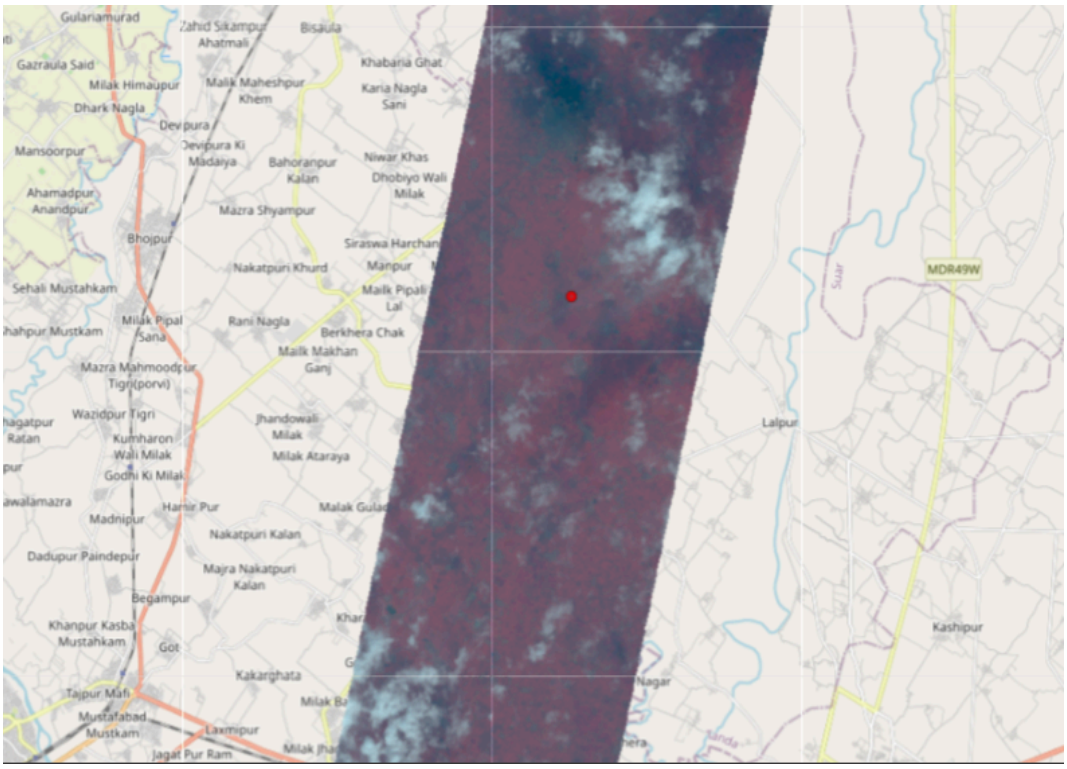
For these to coincide with the values of our trained dataset we took 25 bands from 460 nm to 700 nm with 10 nm spectral resolution.

Further few bands seem to have distorted values for taking into consideration and the rest go out of range.

Here after arranging the values accordingly we next scaled values from 400 to 1000 value in case of P, K, Mg, ph estimation and 0.1 to 0.2 in case of Temperature and Moisture conserving the observed trend.

So taking these band values we further get the values from the models and get an estimate of nutrients found in the area.

For example, we set the coordinates of Uttar Pradesh with latitude 28.934799 and longitude = 78.948349 in 2010. This is the patch we got.



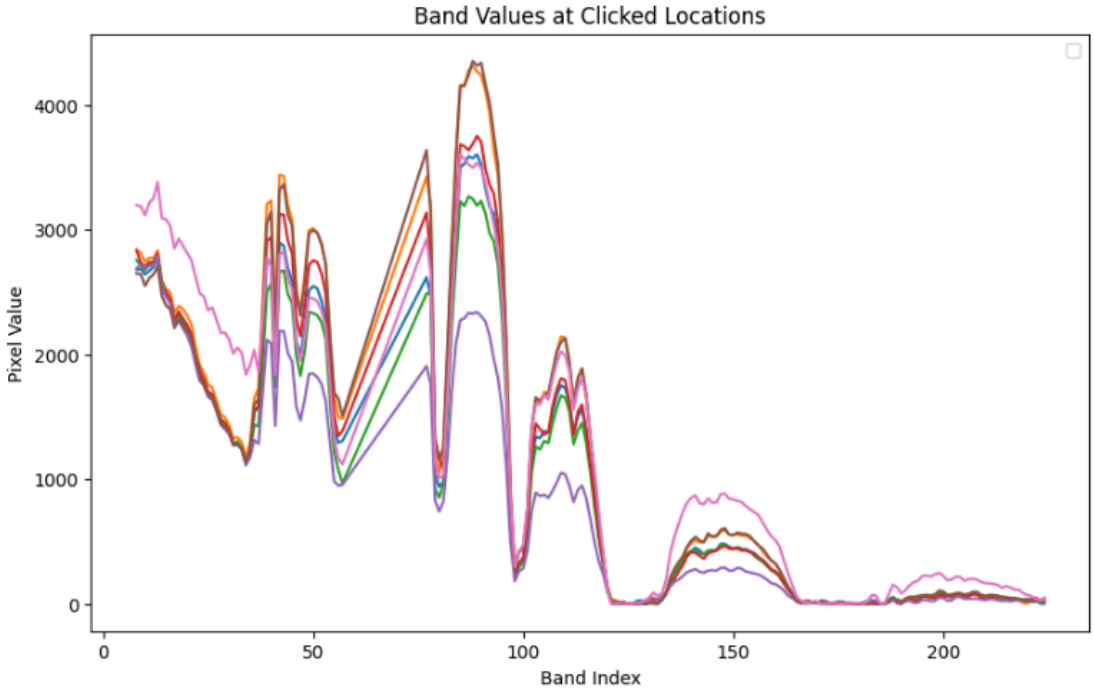


Fig 14: Demo 1 Patch 1

Most of the area which we got here was not of our interest as the trend we got in the band values got to be of vegetation.

Next we moved to Kerela with the coordinates latitude 11.09548 and longitude 76.51647 and clicked on various points to plot their band values and the best we could get was the light blue and rest seems to be the vegetation.

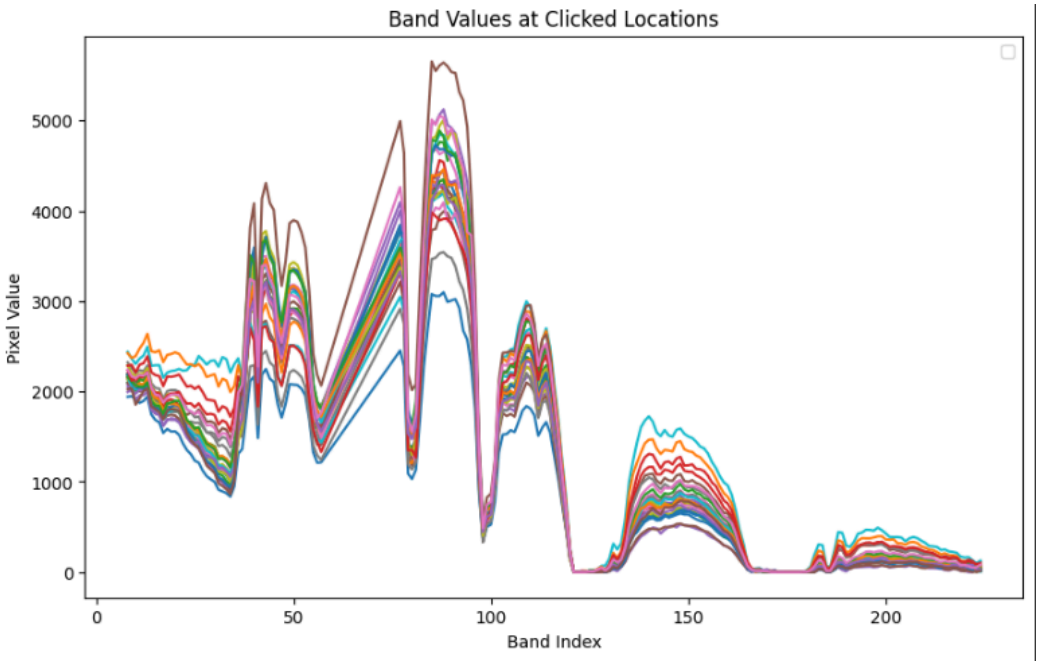
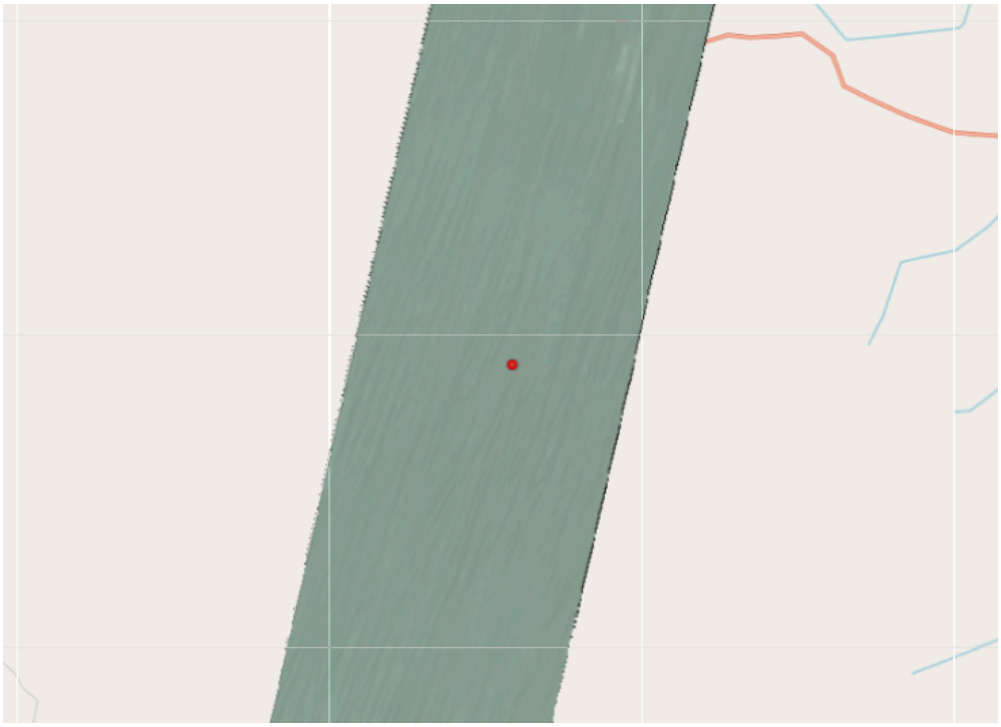


Fig 15: Demo 1 Patch 2

Another area with coordinates 27.13, 70.10 respectively.



Here is a desert area with band values are as follows

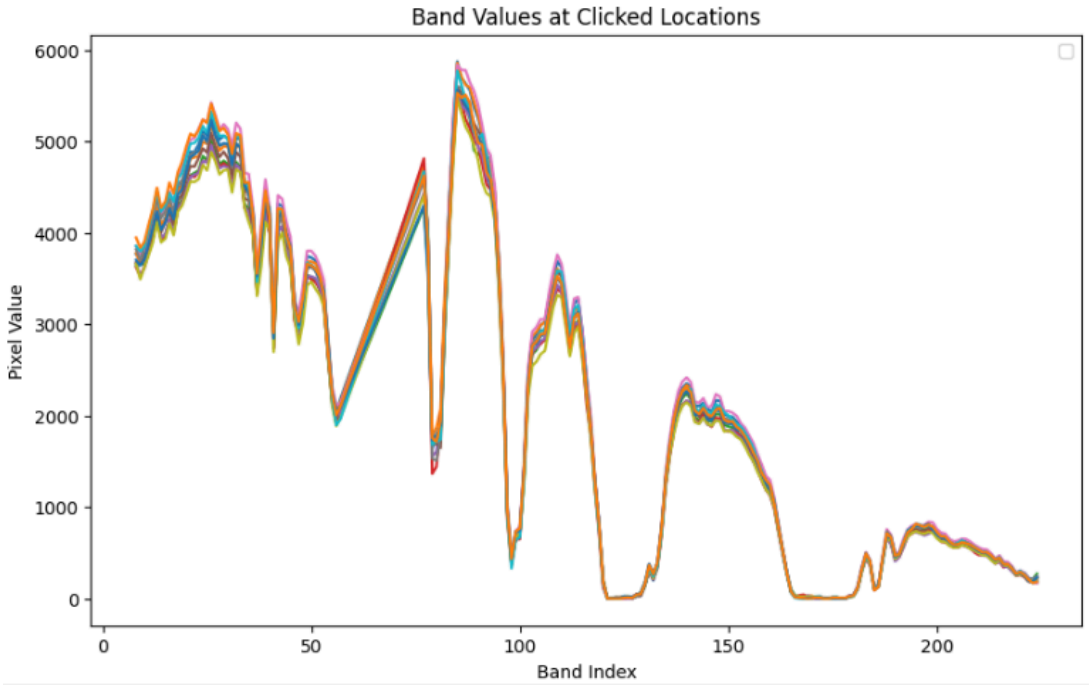


Fig 16: Demo 1 Patch 3

We are here seeing for coordinates 49.002 and 8.402 respectively.

Band values at the clicked locations are mixed of soil and vegetation.

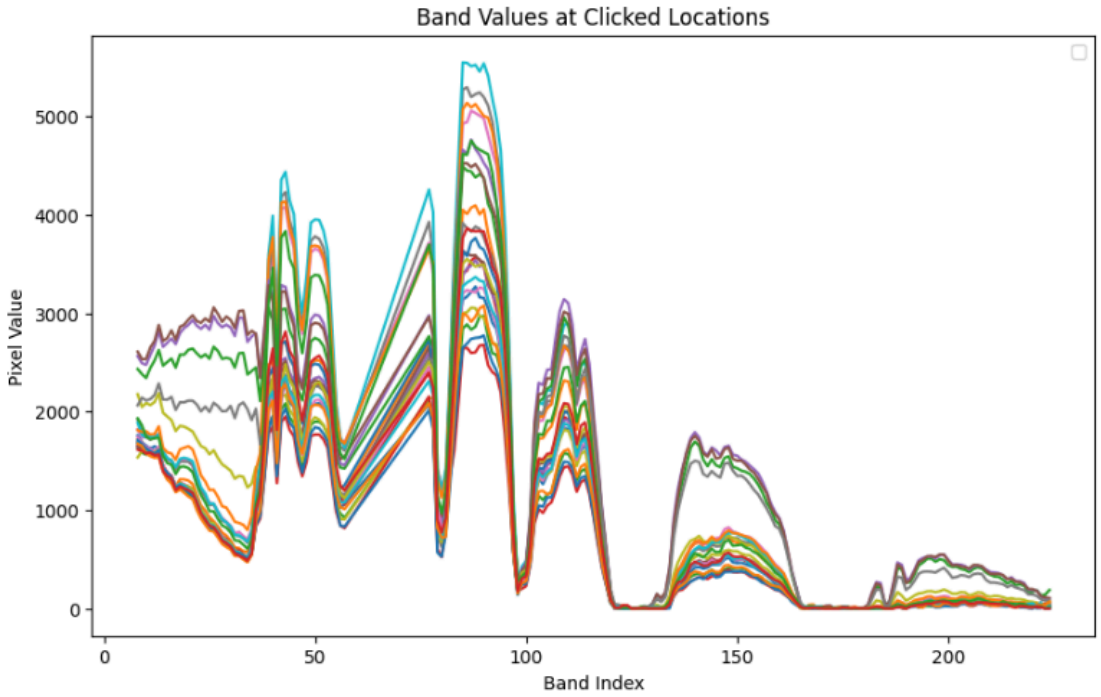


Fig 16: Demo 1 Patch 4

Demonstration 2:

In assistance with the web design of Nikhil Kumar (SATCARD), next is a web design, integrated with the Satcard website which goes on /satellite address.

Here we take the coordinates as input and showcase the coordinates in the map and signify the entered coordinates and plot the best trend captured in the square area. We move left to right and up and down and take the best values and find the corresponding nutrient values and return them as output.

An example of the corn belt in US.

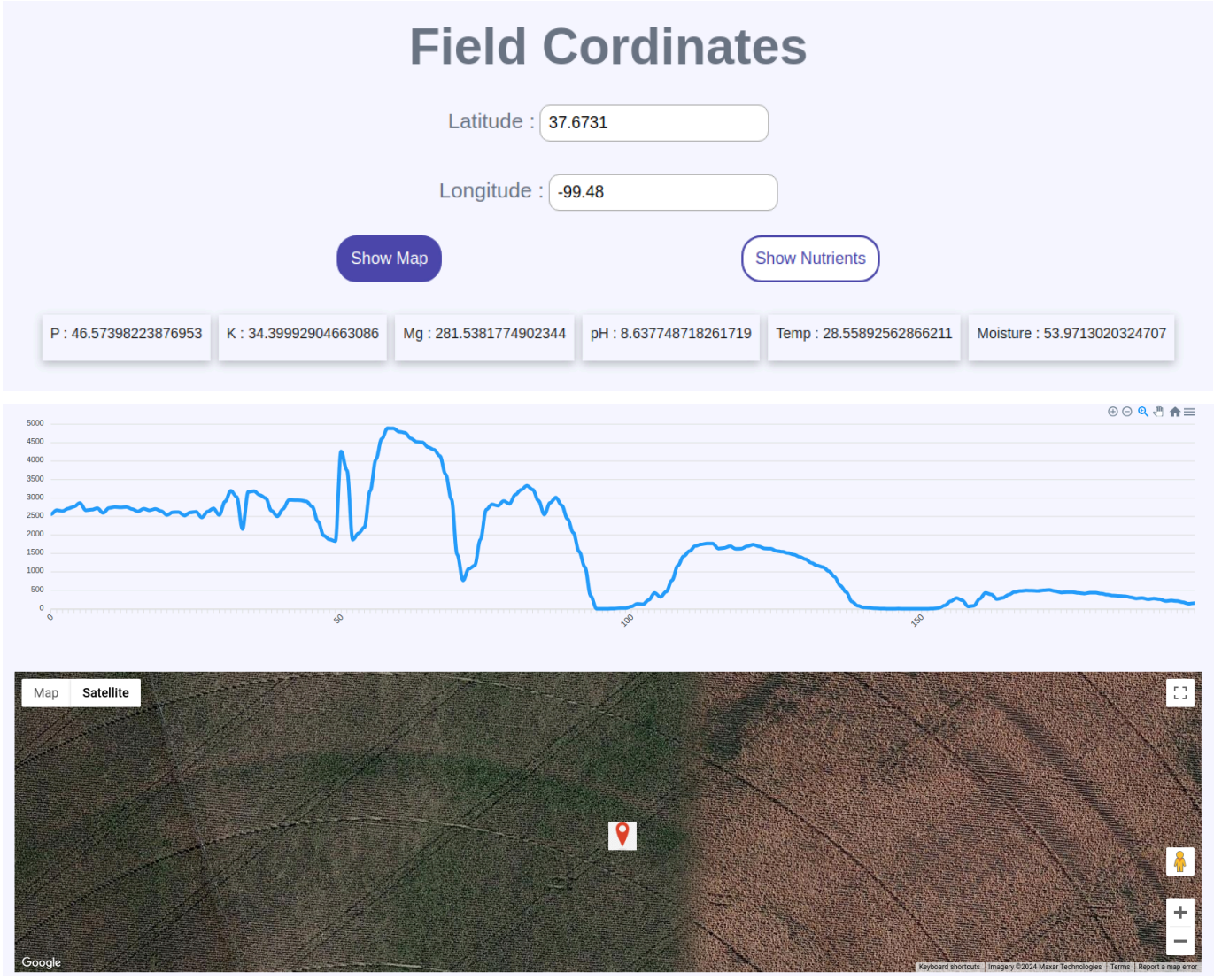


Fig 18: Demo 2 Patch 1

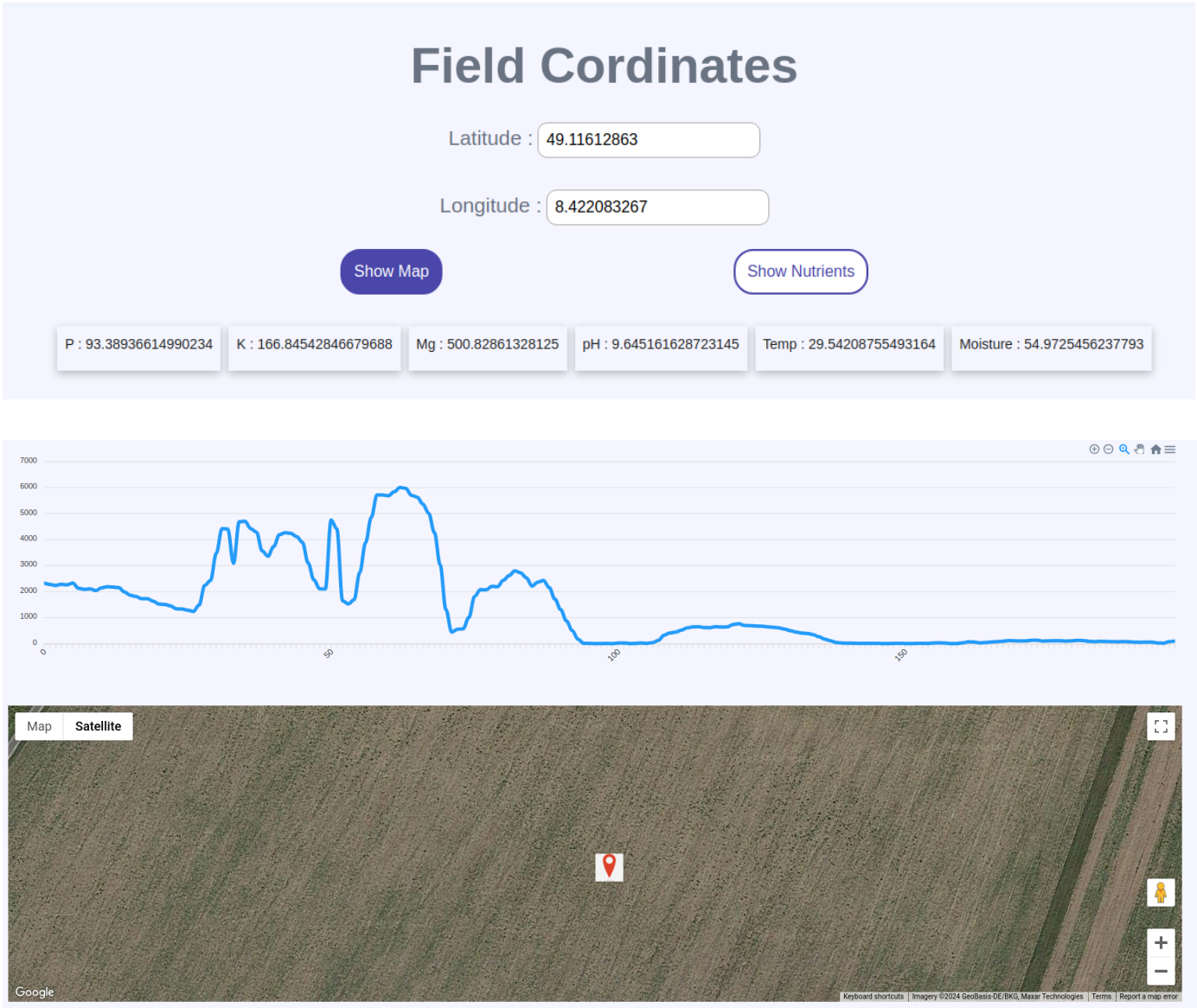


Fig 19: Demo 2 Patch 2

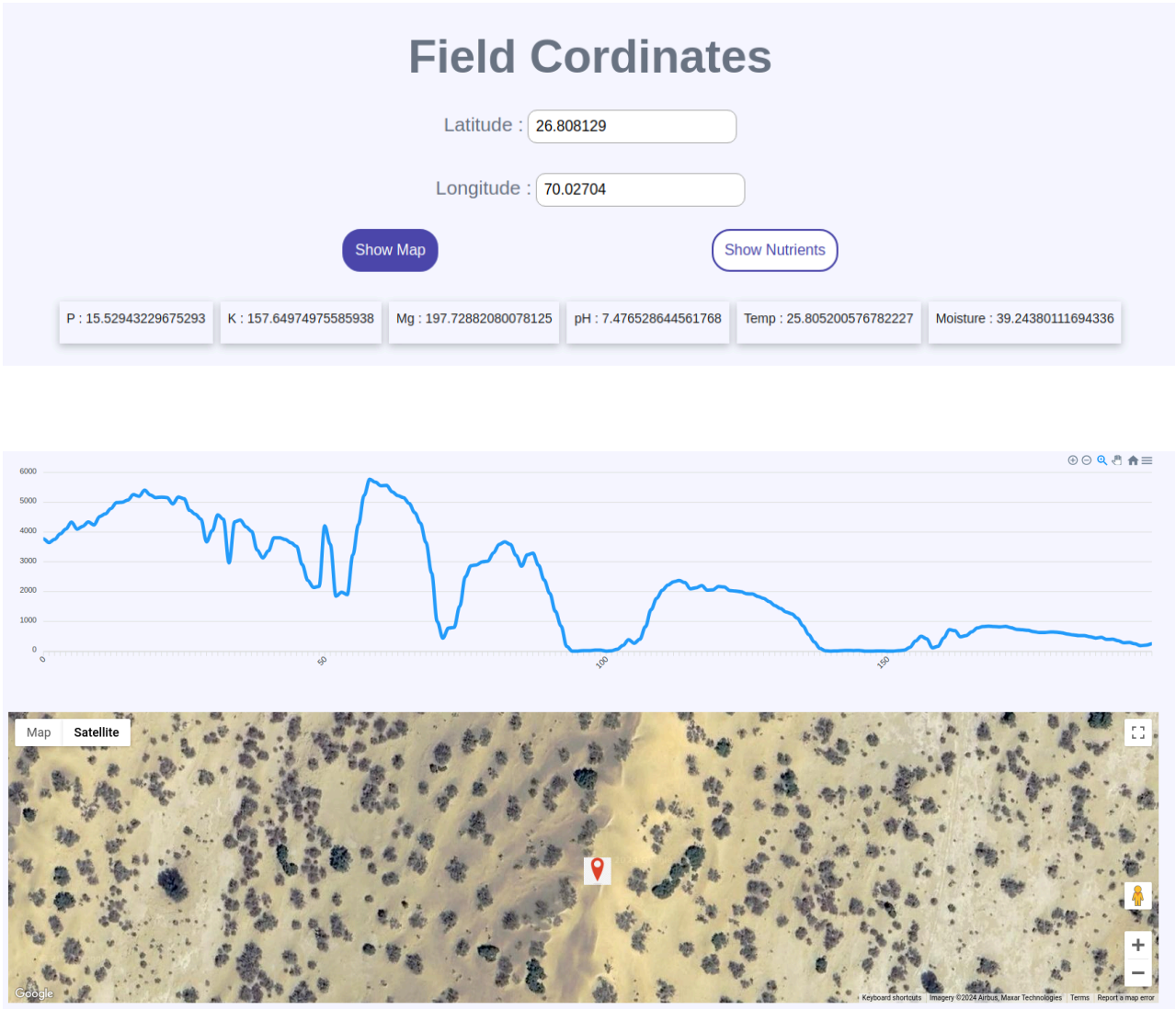


Fig 20: Demo 2 Patch 3

❖ Challenges Faced

As the models are trained on dataset from Poland and Germany only on few areas it is too much to expect them to give correct results for desert areas, non agricultural areas or barren lands. Moreover, for agricultural patches scaling needs to be reviewed on a global scale and a sophisticated algorithm could be found to find soil areas on its own.

❖ Future Work

In the pursuit of advancing hyperspectral soil nutrient estimation, we are presented with promising avenues for further exploration. These paths open up new possibilities for enhancing the accuracy and applicability of our models, driving us towards a deeper understanding of the complex relationships within environmental data. Model seems to be promising when trained on larger areas, and reviewed on global dataset.

Adequate finding technique of soil patch will solve the testing part to much extent and after training on bigger dataset scaling could be modified accordingly.

❖ Git Repository

https://github.com/priyanshu87511/BTP

❖ References

- SOIL TEXTURE CLASSIFICATION WITH 1D CONVOLUTIONAL NEURAL, NETWORKS BASED ON HYPERSPECTRAL DATA, F. M. Riese1, S. Keller11 Karlsruhe Institute of Technology (KIT), Institute of Photogrammetry and Remote Sensing,Englerstr. 7, D-76131 Karlsruhe, Germany, (felix.riese, sina.keller)@kit.edu (Link)

- HybridSN: Exploring 3D-2D CNN Feature Hierarchy for Hyperspectral Image Classification Swalpa Kumar Roy, Student Member, IEEE, Gopal Krishna, Shiv Ram Dubey, Member, IEEE, and Bidyut B. Chaudhuri, Life Fellow, IEEE (Link)

- "GeoAI Challenge Estimating Soil Parameters from Hyperspectral Images by ITU" by Zindi, Africa

- Hyperspectral benchmark dataset on soil moisture. Soil moisture data Karlsruhe (Germany), 2017 (kaggle)

- Hyperion Satellite under ARSET (Data)