**NITROGEN DEFICIENCY DITECTION AND CORRECTION DATASET**

INTRODUCTION

The aim of this project was to create a dataset for computer vision. The dataset will be used to train a model to be able to identify various signs of nitrogen deficiency in maize at various stages of maize growth. To begin with, Nitrogen is a crucial nutrient that helps in plant growth and most plants need it to produce effectively. From germination to plant maturity, it plays a crucial role which includes promoting the development of the root system, chlorophyll production, ensures proper kernel development and grain filling and also influences the protein content and other nutritional qualities of the maize grains. With its importances its really important to make sure that as the plants grow there is a balance of this nutrient. The dataset will identify multiple signs of nitrogen deficiency by placing bounding boxes around them, allowing a model to be trained that can help farmers recognize these signs. The signs include: stunted growth with shorter stems and smaller leaves, pale or yellowing of leaves which starts from the tip going to the base of the leaf, thin and narrow leaves, purple or reddish coloration, reduced tillering and ear development and reduced kernel size and quality there is also delayed and uneven maturity but that can not be visualized so I wont be showing that. This dataset will show images with these signs and some healthy maize plants. A model trained to identify nitrogen deficiency could greatly assist farmers in monitoring their expansive croplands, enabling them to address these nutrient imbalances in a timely manner and promote bountiful harvests, thereby enhancing overall food security.

DATASETS DESCRIPTIONS

The data was sourced from multiple sites. I have combined three datasets and then added a few more images which I downloaded through a python script. The datasets used include:

1. Nitrogen deficiency in maize: annotated images classification dataset

* This had 1200 images that were taken using a canon 80D camera with an18-50 mm lens at 18mm. aperture priority was used with aperture set to f/8. The focus was set manually to approx.
* Its category as put on the dataset’s description is : Agricultural Science, Computer vision, Nitrogen , Maize
* Published on 4 August 2023
* Contributors: Vlatko Galic, Ivan Podnar Zarko, Filip Novoselnik , Miroslav Salaic
* Licence: CC BY 4.0
* Version 1

1. Agricultural crops image classification: the dataset contained 30 different types of crop images in separate folders. It had 829 files altogether.

* The author: Md waquar azam (Mechanical engineer , ML and Data science Enthusiast)
* Temporal coverage: 08/25/2022 to 08/25/2024
* Provenance: Google chrome image
* Licence : [CC0: Public Domain](https://creativecommons.org/publicdomain/zero/1.0/)

1. agriculture crop images: this datasets: Dataset (Crop Images) contain 40+ images of each Agriculture crop(Maize, Wheat, jute, rice and sugarcane) Dataset (kag2) contains 159+ augmented images of Crop Images of each class. Augmentation contain Horizontal flip, rotation, horizontal shift, vertical shift.

* Collaborators: aman2000jaiswal (Owner)

I have combined these three datasets. The last two datasets contained 30 and 41 images of maize respectively, so I took all of them. Then I added 60 images that I downloaded using the big image downloader module in Python. I also included 479 images from the first dataset, because my machine could not load all of them, so I just took what I could manage. In total, this left me with 610 images to begin the preprocessing journey.

DATA TRANSFORMATIONS

1. resizing images: I changed the image size to (255,255) pixel dimensions where the numbers represent the heigh and width of the images. The idea behind resizing the images was to come up with a dataset of images with uniform sizes. The reasons for resizing are:

* most deep learning models require images to be fixed size so the images coming from four different sources contained images with different sizes. So I needed uniformity which is preferred by most models
* to improve computational efficiency and memory resources. This can be achieved by scaling the images down to a smaller dimension which I have done. The (255,255) size is a good on e for this.

To resize the images I used the opencv function called resize() which take two arguments the image and the sizes which are in 2 dimensions the height and width. And that was set to (255,255) for height and width respectively. I then saved them in a folder so that I can annotate them.

1. Adding bounding boxes:

I added bounding boxes using roboflow website which has all the tools needed for computer vision tasks. Putting bounding boxes was done manually by simply adding rectangles on the object needed. In this case the sign of nitrogen deficiency. Then I downloaded the final product in coco format. This format contains the images and a json file containing the coordinates of the bounding boxes. Then a code was needed top combine them with the images. Roboflow also divides the images into test, train and validate folders, each with its own annotations. the json file was first loaded using the json library function called load() and stored in a variable

* Then the images whose annotations were loaded were also loaded. The json file is put in the same folder as the images
* Then I used the matplotlib library to plot the boxes on the images and then saved the plots one by one as a graph sort of. So I used a loop to loop through all images in the folder which contain the images. Then save them in separate folders which I have submitted.

The addition of the bounding boxes was done because it’s the most important part of my project because it’s the one that tells the model where the object to identify is.(in this case a nitrogen deficiency symptom or sign).

DATA CLEANING

1. Correcting data types: the images were also converted to one format which is png. This also helps to maintain compatibility with the model’s input requirements that will avoid any delays if the image is being trained on images with different formats.

This was done when I was saving the final images with bounding boxes. I used the extensions png for all of them so when they were being saved they all had the png format.

EXPLORARTORY DATA ANALYSIS

1. I used histogram to visualize pixel distribution of the images before and after normalization. This was used the hist() function from matplotlib library.

* I did this for only first 5 images.

1. I have calculated mean, median and standard deviation for the pixel dimensions of the loaded images before they were preprocessed. This was achieved using the numpy library.

* Mean was calculated using the function called mean()
* Median was calculated using median() function
* Standard deviation was calculated using the std() function

CONCLUSION

It's been observed that when doing object detection, it's not a good idea to put multiple bounding boxes on a single image. I've encountered a problem where some images have bounding boxes that don't belong there. After closer inspection, I realized that this issue was caused because there were more than one box on other images. So I believe that just putting a single large box on the image that contains multiple signs can help improve the quality of the dataset.

I've also discovered that loading all the images at once can slow down the system. Sometimes the system can't even handle it, so loading them in smaller batches is actually more helpful and a bit faster.

I believe the next step is to normalize the dataset, which I haven't fully achieved in the final product. The problem is that after normalization, I was losing the color of the images, which made it difficult for me to identify the signs I was looking for. So instead, I just resized the images and then annotated them. The normalization code is there, and I have normalized the images, but they are not the ones I used for the annotations. The recommendation is that there's a need to learn more about the effects of certain preprocessing steps on the color of images, and see how they affect it, as well as how we can identify different colors after the resolution has been changed.