

G5S3: Satellite Vision

EC 3714: Product Development Laboratory (Spring 2023-2024)

Team Members-

Pramod Kumar Bagh (121EI0383)

Vedant Asawa (121EI0382)

Amit Agrawal (121EI0381)

V Shashank (121EI0379)

Faculty Member Name: Jogesh Chandra Dash

Faculty Member Name: Situ Rani Patre



Department of Electronics and Communications Engineering
National Institute of Technology Rourkela, Odisha, India

Contribution

Pramod Kumar Bagh (121EI0383): Worked on gradio website to deploy Deep Learning model so that any other user can easily access that DL model to check pollution and deforestation level.

Vedant Asawa (121EI0382): Developed an DL Model such that when an image is given as an input it gives output data on deforestation and pollution level based the dataset it is trained.

Amit Kumar Agrawal (121EI0381): Worked on integrating DL model with web interface so that user can upload images

V Shashank (121EI0379): Worked on finding of various dataset of more than 40,000 images which includes habitation, slash and burn, industrialization and highways.

Introduction:

Pollution, deforestation, and their causes are interconnected issues that have been influenced by various government projects. Some of the main causes of deforestation due to government projects include:

1. Infrastructure development: Road projects, particularly those built to open up and "develop" forests, have been identified as significant contributors to deforestation. These roads provide access to forests for loggers, landless peasants, mining companies, and other actors, resulting in generalized deforestation
2. Macroeconomic policies: Economic policies that promote the incorporation of Southern agricultural exports, often at the expense of forests, can contribute to deforestation. Additionally, policies that concentrate wealth in the North and create unsustainable consumption patterns have a strong impact on Southern governments, leading to deforestation.
3. Land tenure policies: Policies over land tenure rights have resulted in the concentration of the best agricultural lands in a few hands, leading to large-scale felling of trees. In most cases, government policies are linked to external factors such as multilateral institutions, cooperation agencies, and transnational corporations who share the blame for deforestation.
4. Deforestation for infrastructure projects: A wave of infrastructure projects, including roads, waterways, railways, and dams, is expected to open up vast tracts of land to natural resource extraction and deforestation. For example, in the Amazon, five countries plan to spend \$27 billion over the next five years to build or update 7,500 miles of road that could result in the deforestation of 5.9 million acres.

In conclusion heavy industrialization, various government projects have contributed to pollution and unregulated tree cutting through infrastructure development, macroeconomic policies, and land tenure policies. These projects have opened up forests to logging, mining, and other activities, leading to widespread deforestation and environmental degradation. In recent times, we have seen various protest and incidents due to the irregularity of environmental board such as Ladakh Protest, Hasdeo forest cut, Sinking of Joshimath and many more.

Sources Used:

- [1] <https://www.ncbi.nlm.nih.gov/pmc/articles/PMC9408456/>
- [2] <https://www.epa.gov/clean-air-act-overview/government-partnerships-reduce-air-pollution>
- [3] <https://www.epa.gov/newsreleases/biden-harris-administration-announces-53-million-132-community-air-pollution>
- [4] <https://www.worldbank.org/en/topic/pollution>

Here's the brief of our project:

Satellite images can be used to reduce pollution in several ways:

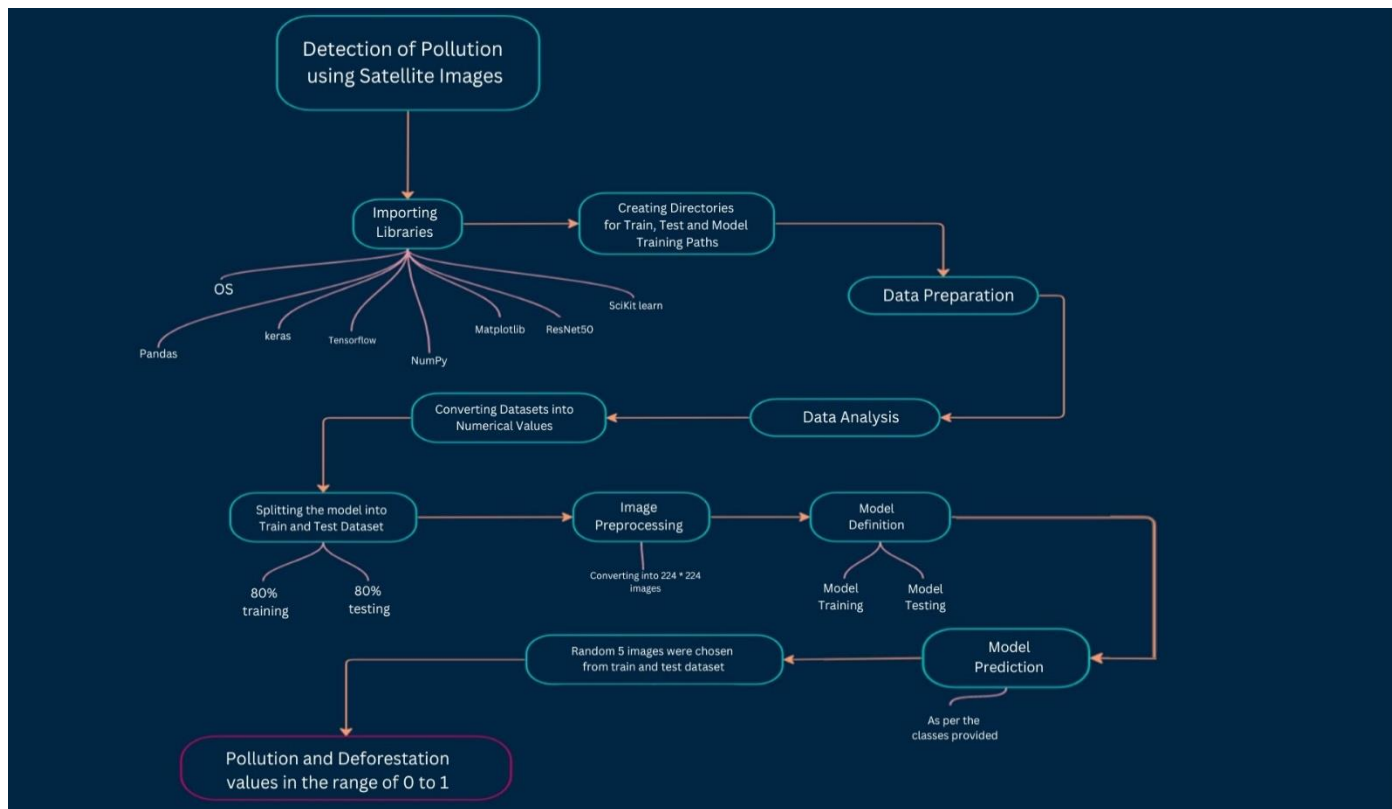
1. Monitoring air quality: Satellite remote sensing data can be used to monitor air quality by measuring trace gases and aerosol properties relevant to air quality. This information can be used to identify areas with high levels of pollution and to track changes in pollution levels over time. This data can help policymakers make informed decisions about reducing pollution and improving air quality.
2. Deforestation monitoring: Satellite images can be used to monitor deforestation and identify areas where trees are being cut down. This information can help policymakers take action to prevent further deforestation and promote reforestation efforts.
3. Environmental justice and equity applications: Satellite data can be used to identify areas with high levels of pollution that disproportionately affect marginalized communities. This information can help policymakers take action to address environmental inequalities and promote environmental justice.
4. In summary, satellite images can be used to monitor various types of pollution and help policymakers make informed decisions about reducing pollution and promoting environmental health.

5. This model can be used for various government projects to detect a pollution and deforestation level in the area where they are implementing it and set a bar according to an area that this project should not exceed this much of pollution levels.

6. The advantages of using this model in government projects is transparency, Anti- Corrupt, Low-Maintenance, Safety measures and reliable.

7. Various type of dataset can be used to train the model such as Industrial, Highway, Annual Crop, Sea Lake, Forest, Residential, River.

Project Block Diagram -



Deep Learning Model:

It is the most Important part of the product to detect the suspicious Activity.

language – We are using the Python Language as it already contains most of the libraries mostly used for Machine Learning and Deep Learning.

IDE To be used – We are using Jupyter Notebook and Google Collab to write and debug the code as here we can execute the line by line code of the python program.

In google Collab we already get all the Libraries pre installed and GPU and TPU server in free.



Model To Be Used –

we are using **ResNet50** model for classification of images. As in our model simultaneously detects images so this model will give the better accuracy.

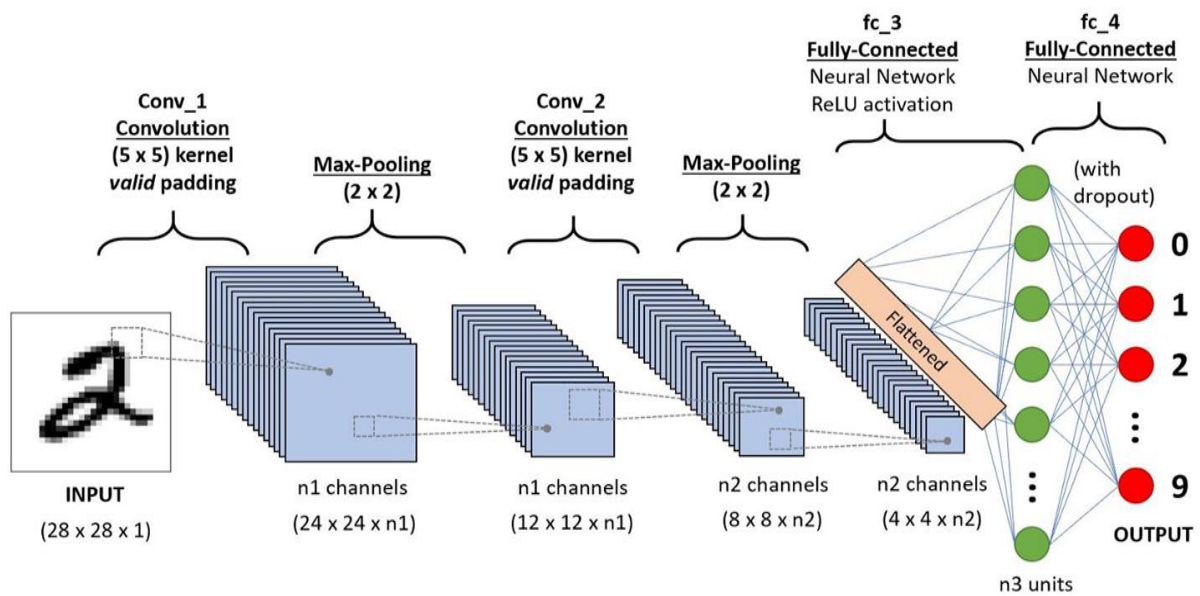
ResNet50 –

ResNet 50 is a powerful model for image classification tasks, and its transfer learning capabilities makes it suitable for a wide range of problems. By fine-tuning the pre-trained model with your specific dataset, you can achieve high accuracy in your classification tasks. The architecture includes 48 convolutional layers, one max pool layer, and one average pool layer. ResNet50 is known for its efficiency and accuracy in image classification tasks.

|

CNN –

A convolutional neural network (CNN or convnet) is a subset of [machine learning](#). It is one of the various types of artificial [neural networks](#) which are used for different applications and data types. A CNN is a kind of network architecture for [deep learning](#) algorithms and is specifically used for [image recognition](#) and tasks that involve the processing of [pixel](#) data.



TensorFlow, Keras, Numpy , Pandas, Matplotlib, SKLearn –

Tensorflow and Keras – TensorFlow is an open-sourced end-to-end platform, a library for multiple machine learning tasks, while Keras is a high-level neural network library that runs on top of TensorFlow. Both provide high-level APIs used for easily building and training models, but Keras is more user-friendly because it's built-in Python.



MACHINE LEARNING LIBRARIES!



Code For the Project –

First we are importing the Important Libraries –

- Numpy, Pandas, Matplotlib, Sklearn, TensorFlow, Keras, Collection, OS, ResNet50, Matplotlib, Random


```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
```

```
import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

```
import numpy as np
import tensorflow as tf
import pandas as pd
from sklearn.model_selection import train_test_split
from tensorflow.keras.metrics import AUC
from tensorflow.keras.metrics import Metric, Precision, Recall
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.layers import Dense, GlobalAveragePooling2D
from tensorflow.keras.models import Model
```

```
# Load pre-trained ResNet50 model
base_model = ResNet50(weights='imagenet', include_top=False)

# Load labels
labels_df = pd.read_csv('/kaggle/input/amazonsatelliteimages/train_v2.csv/train_v2.csv')
```

```
# Create binary labels for 'habitation' and 'slash_burn'
labels_df['deforestation'] = labels_df['tags'].apply(lambda x: 1 if 'habitation' in x else 0)
labels_df['pollution'] = labels_df['tags'].apply(lambda x: 1 if 'slash_burn' in x else 0)
```

➤ Splitting a dataset in 80:20 ratio to train and test the model

```
# Split dataset into training and validation sets
train_df, val_df = train_test_split(labels_df, test_size=0.2, random_state=42)
```

```
# Add file extensions to image filenames
train_df['image_name'] = train_df['image_name'].apply(lambda x: f"{x}.jpg")
val_df['image_name'] = val_df['image_name'].apply(lambda x: f"{x}.jpg")
```

- Augmentation and preprocessing of images before training it :

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Function to create generator for the dataset
def create_dataset_generator(df, data_dir, batch_size=32, target_size=(224, 224)):
    datagen = ImageDataGenerator(rescale=1./255)
    generator = datagen.flow_from_dataframe(
        df, directory=data_dir, x_col='image_name', y_col=['deforestation', 'pollution'],
        target_size=target_size, batch_size=batch_size, class_mode='raw')
    return generator

# Set the path to dataset directory
data_dir = '/kaggle/input/amazonsatelliteimages/train-jpg/train-jpg'

# Create dataset generators for training and validation sets
train_generator = create_dataset_generator(train_df, data_dir)
val_generator = create_dataset_generator(val_df, data_dir)
```

- Now creating the Deep Learning Model with sigmoid as an activation function for an output –

```
from tensorflow.keras.layers import Dropout

# Add dropout layers to the model
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(2048, activation='relu')(x)
x = Dense(1024, activation='relu')(x)
predictions = Dense(2, activation='sigmoid')(x)

# Create final model
model = Model(inputs=base_model.input, outputs=predictions)
```

- To reduce the Loss, optimize the model and calculate the accuracy:
If $AUC < 0.5$, model is behaving randomly

```
# Freeze the base_model layers
for layer in base_model.layers:
    layer.trainable = False

# Compile model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=[AUC(name='auc')])
```

```
# Train the model
history = model.fit(train_generator, epochs=3, validation_data=val_generator)
```

- Verifying test loss and test accuracy: We used here only 3 epochs so that model doesn't overfit with the given test cases and hence we got the accuracy of 88.48% .

```
Epoch 1/3
1012/1012 [=====] - 379s 363ms/step - loss: 0.1736 - auc: 0.7596 - val_loss: 0.1596 - val_auc: 0.8785
Epoch 2/3
1012/1012 [=====] - 141s 139ms/step - loss: 0.1582 - auc: 0.8095 - val_loss: 0.1536 - val_auc: 0.8802
Epoch 3/3
1012/1012 [=====] - 142s 140ms/step - loss: 0.1529 - auc: 0.8317 - val_loss: 0.1487 - val_auc: 0.8848
```

- Importing the images from the provided test and train dataset to test the model and to lowers its spatial resolution so that model computation takes much time to compute.

```
from tensorflow.keras.preprocessing import image

def predict_deforestation_pollution(model, img_path, img_size=(224, 224)):
    # Load and preprocess image
    img = image.load_img(img_path, target_size=img_size)
    x = image.img_to_array(img)
    x = np.expand_dims(x, axis=0)
    x = x / 255.0

    # Make a prediction using model
    preds = model.predict(x)

    return preds
```

- Considering any 5 images in test dataset to generate the output:

```
import os
import random
import matplotlib.pyplot as plt

def display_image_with_predictions(img_path, deforestation_prob, pollution_prob):
    img = image.load_img(img_path)

    plt.figure()
    plt.imshow(img)
    plt.title(f"Deforestation Probability: {deforestation_prob:.4f}\nPollution Probability: {pollution_prob:.4f}")
    plt.axis("off")
    plt.show()

# Set path to the test dataset directory
test_data_dir = '/kaggle/input/amazonsatelliteimages/test-jpg/test-jpg'

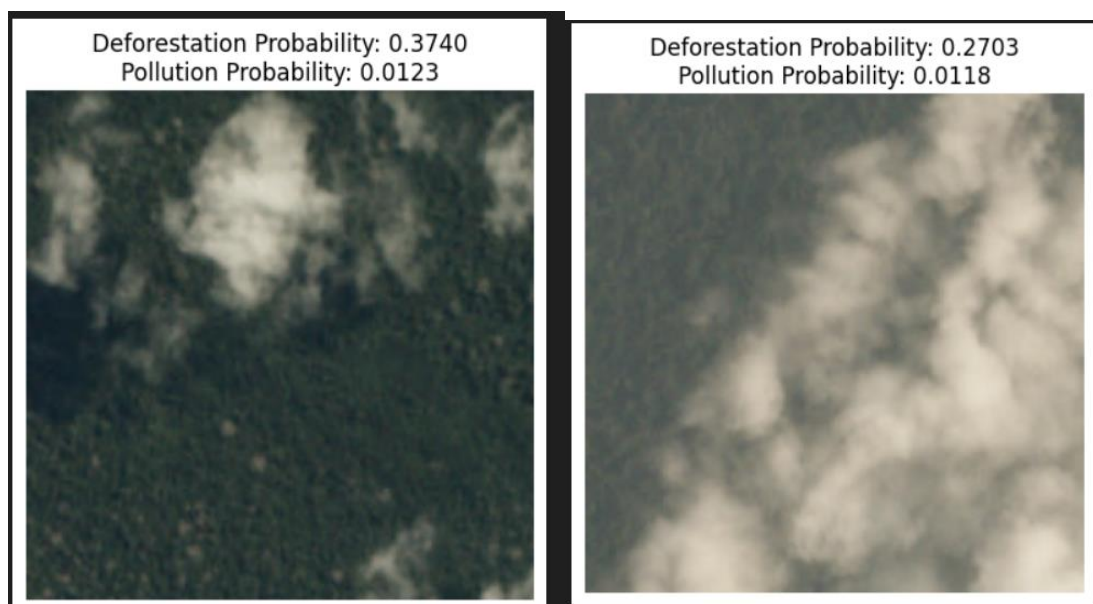
# Choose a few random images from test dataset
test_images = random.sample(os.listdir(test_data_dir), 5)

# Evaluate model's performance on selected test images
for img_name in test_images:
    img_path = os.path.join(test_data_dir, img_name)
    preds = predict_deforestation_pollution(model, img_path)

    deforestation_prob = preds[0][0]
    pollution_prob = preds[0][1]

    display_image_with_predictions(img_path, deforestation_prob, pollution_prob)
```

- Output after training the model with the dataset and testing the model with test and train data



Importing DL model from Kaggle to Google Collab to make the model work faster as the google collab has its own GPU power .

```
# IMPORTANT: RUN THIS CELL IN ORDER TO IMPORT YOUR KAGGLE DATA SOURCES
# TO THE CORRECT LOCATION (/kaggle/input) IN YOUR NOTEBOOK,
# THEN FEEL FREE TO DELETE THIS CELL.
# NOTE: THIS NOTEBOOK ENVIRONMENT DIFFERS FROM KAGGLE'S PYTHON
# ENVIRONMENT SO THERE MAY BE MISSING LIBRARIES USED BY YOUR
# NOTEBOOK.

import os
import sys
from tempfile import NamedTemporaryFile
from urllib.request import urlopen
from urllib.parse import unquote, urlparse
from urllib.error import HTTPError
from zipfile import ZipFile
import tarfile
import shutil

CHUNK_SIZE = 40960
DATA_SOURCE_MAPPING = 'amazonsatelliteimages:https%3A%2F%2Fstorage.googleapis.com%2Fkaggle-data-se

KAGGLE_INPUT_PATH='/kaggle/input'
KAGGLE_WORKING_PATH='/kaggle/working'
KAGGLE_SYMLINK='kaggle'

!umount /kaggle/input/ 2> /dev/null
shutil.rmtree('/kaggle/input', ignore_errors=True)
os.makedirs(KAGGLE_INPUT_PATH, 0o777, exist_ok=True)
os.makedirs(KAGGLE_WORKING_PATH, 0o777, exist_ok=True)

try:
    os.symlink(KAGGLE_INPUT_PATH, os.path.join(".", 'input'), target_is_directory=True)
except FileExistsError:
    pass
try:
    os.symlink(KAGGLE_WORKING_PATH, os.path.join(".", 'working'), target_is_directory=True)
except FileExistsError:
    pass

for data_source_mapping in DATA_SOURCE_MAPPING.split(','):
    directory, download_url_encoded = data_source_mapping.split(':')
    download_url = unquote(download_url_encoded)
    filename = urlparse(download_url).path
    destination_path = os.path.join(KAGGLE_INPUT_PATH, directory)
    try:
        with urlopen(download_url) as fileres, NamedTemporaryFile() as tfile:
            total_length = fileres.headers['content-length']
            print(f'Downloading {directory}, {total_length} bytes compressed')
            dl = 0
            data = fileres.read(CHUNK_SIZE)
            while len(data) > 0:
                dl += len(data)
                tfile.write(data)
                done = int(50 * dl / int(total_length))
                sys.stdout.write(f"\r[{'=' * done}{' ' * (50-done)}] {dl} bytes downloaded")
                sys.stdout.flush()
                data = fileres.read(CHUNK_SIZE)
            if filename.endswith('.zip'):
                with ZipFile(tfile) as zfile:
                    zfile.extractall(destination_path)
            else:
                with tarfile.open(tfile.name) as tarfile:
                    tarfile.extractall(destination_path)
            print(f'\nDownloaded and uncompressed: {directory}')
    except HTTPError as e:
        print(f'Failed to load (likely expired) {download_url} to path {destination_path}')
        continue
    except OSError as e:
        print(f'Failed to load {download_url} to path {destination_path}')
        continue

print('Data source import complete.')
```

Using Gradio to make the UI interface for an user to access this Deep Learning Model to predict the pollution through satellite images.

Gradio :

Gradio is a Python library that allows you to quickly create customizable UIs for your machine learning models. It's designed to make it easy for both developers and non-developers to interact with machine learning models, enabling tasks like model deployment, testing, and visualization without needing deep technical knowledge.

Here are some key features and aspects of Gradio:

Simple Interface: Gradio provides a simple and intuitive interface for creating UIs. You can define input and output components using Python functions and decorators.

Wide Range of Components: Gradio supports various input components like text boxes, sliders, dropdowns, and file uploads. Similarly, it supports output components like text, images, plots, and more.

Support for Multiple Frameworks: Gradio is framework-agnostic, meaning you can use it with popular machine learning frameworks like TensorFlow, PyTorch, and scikit-learn.

Live Updates: The UI updates in real-time as users interact with it, providing instant feedback.

Customizability: Gradio allows you to customize the appearance of UI components and style them according to your preferences.

Deployment Options: You can deploy Gradio UIs locally on your machine, or you can share them online using Gradio's cloud hosting service. This makes it easy to share your models with others or integrate them into web applications.

Community Support: Gradio has an active community, with developers contributing new features, examples, and documentation regularly.

Integration with Model Zoo: Gradio integrates with the Gradio Model Zoo, a collection of pre-trained models across various domains, making it easy to use and deploy state-of-the-art models.

Overall, Gradio is a powerful tool for building interactive UIs for machine learning models, whether you're a seasoned developer or just getting started with machine learning. Its simplicity and versatility make it a popular choice for both prototyping and deploying ML applications.

- Integrating Model.h5 document which contains a DL model with a web interface to upload images to generate pollution and deforestation output based on user input.

```
import gradio as gr
import tensorflow as tf
from tensorflow.keras.preprocessing import image
import numpy as np

def predict_deforestation_pollution(img):
    # Create a copy of the input image array
    img_copy = np.copy(img)

    # Preprocess the input image
    img_resized = image.array_to_img(img_copy, data_format=None, scale=True, dtype=None)
    img_resized = img_resized.resize((224, 224)) # Resize to match model input shape
    img_array = image.img_to_array(img_resized) # Convert to numpy array
    x = np.expand_dims(img_array, axis=0)
    x = x / 255.0 # Normalize pixel values

    # Make predictions using the model
    preds = model.predict(x)

    return float(preds[0][0]), float(preds[0][1]) # Return two separate probabilities

# Load your HDF5 model
model = tf.keras.models.load_model('/content/model.h5')

iface = gr.Interface(fn=predict_deforestation_pollution, inputs="image", outputs=["number", "number"], title='Deforestation & Pollution Prediction')
iface.launch(debug=True)
```

➤ UI interface of Webpage:

Deforestation & Pollution Prediction

img

Drop Image Here
- OR -
Click to Upload

Clear Submit

output 0
0

output 1
0

Flag

Use via API · Built with Gradio

Output1 showcases: Deforestation level on a scale of 0-1

Output2 showcases: Pollution level on a scale of 0-1

➤ OUTPUT1: When a polluted image is given as an input

Deforestation & Pollution Prediction

img

0.4819202721118927

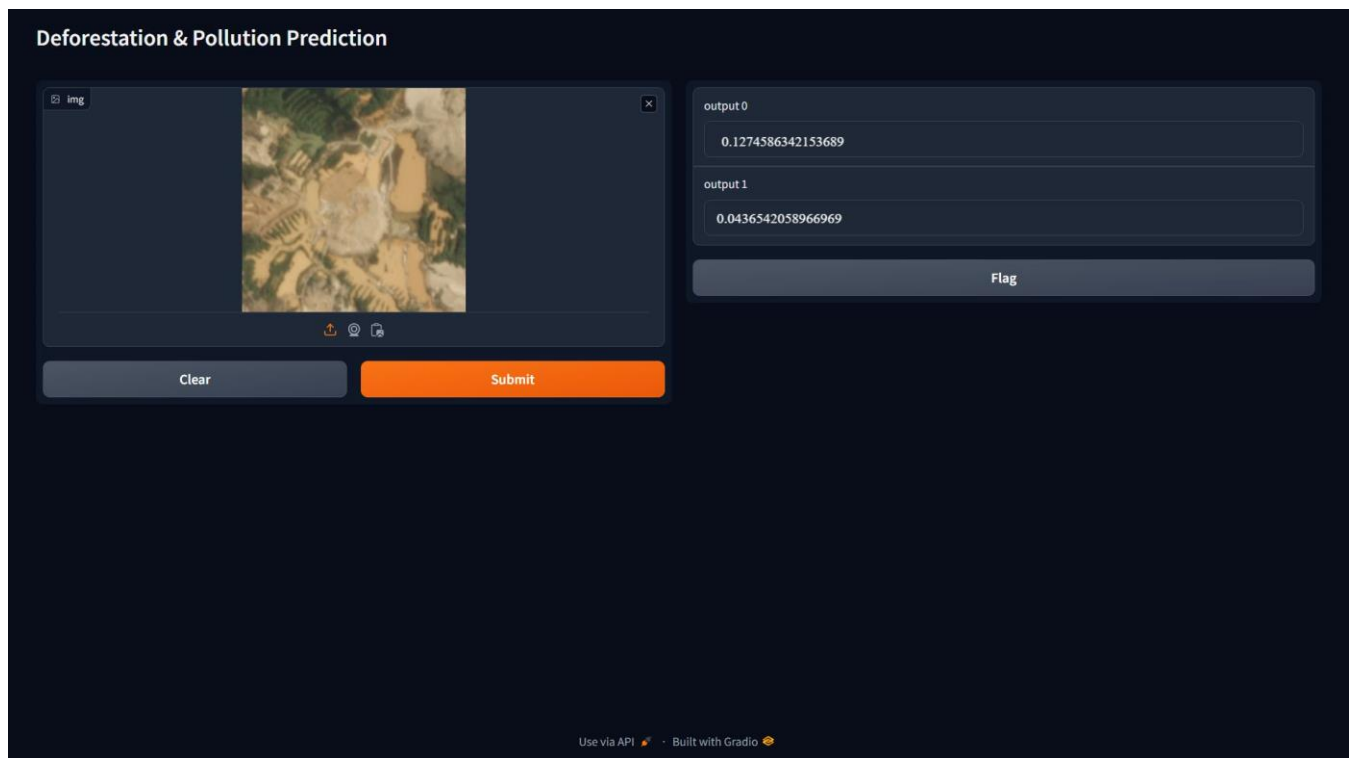
0.6230955123901367

Flag

Clear Submit

Use via API · Built with Gradio

- OUTPUT2: When a green land surface is given as an input



Conclusion –

We successfully completed the project with UI interface integrated with deep learning model which is being trained on more than 40,000 images of dataset which can be used to predict pollution and deforestation level in an area based on user input image.

This various deforestation and pollution levels can be used to regulate pollution and various environmental factors during numerous government projects such highway and industrialization.