Deforestation Challenge

(WebNOVA II Hackathon)

BY VAANI BAKSHI

Introduction

Globally from 2001 to 2023, 22% of tree cover loss occurred in areas where the dominant drivers of loss resulted in deforestation.

Deforestation is a pressing concern, but with access to satellite imagery and the ability to map changes over time, we have a powerful tool to take actionable steps towards a solution.

Code

- The code utilizes a change detection algorithm to identify changes in vegetation or land cover between two satellite images. The algorithm uses Principal Component Analysis (PCA) and K-means clustering to detect changes

```python
def find_FVS(EVS, diff_image, mean_vec, new):
    i = 2
    feature_vector_set = []
    while i < new[0] - 2:
        j = 2
        while j < new[1] - 2:
            block = diff_image[i-2:i+3, j-2:j+3]
            feature = block.flatten()
            feature_vector_set.append(feature)
            j = j+1
        i = i+1
    FVS = np.dot(feature_vector_set, EVS)
    FVS = FVS - mean_vec
    print("\nfeature vector space size", FVS.shape)
    print("loop 2 done ")
    return FVS
```

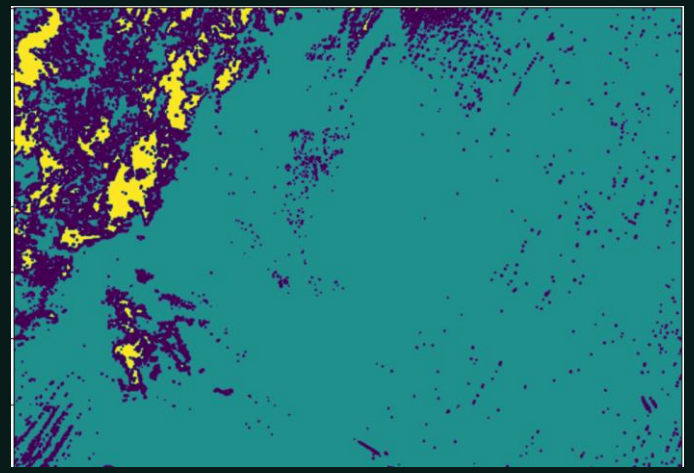
```python
def clustering(FVS, components, new):
    print("line0")
    kmeans = KMeans(components, verbose = 0,n_init=10)
    print("line1")
    kmeans.fit(FVS)
    output = kmeans.predict(FVS)
    count  = Counter(output)
    print("line2")
    least_index = min(count, key = count.get)
    change_map  = np.reshape(output,(new[0] - 4, new[1] - 4))
    print("loop 3 done ")
    return least_index, change_map,output
```

```python
def find_vector_set(diff_image, new_size):
    i = 0
    j = 0
    vector_set = np.zeros((int(new_size[0] * new_size[1] / 25), 75))
    while i < vector_set.shape[0]:
        while j < new_size[0]:
            k = 0
            while k < new_size[1]:
                block   = diff_image[j:j+5, k:k+5]
                feature = block.ravel()
                vector_set[i, :] = feature
                k = k + 5
            j = j + 5
        i = i + 1
    mean_vec   = np.mean(vector_set, axis = 0)
    vector_set = vector_set - mean_vec
    print("loop 1 done ")
    return vector_set, mean_vec
```

- The code can be broken down into several parts:
- find_vector_set: takes the diff image and its size as input and returns the vector set and mean vector
- find_FVS:function finds the Feature Vector Space(FVS) for the diff image
- clustering: applies K-means clustering to FVS

Result

- The output of the code is an image showing the detected changes between the two input images. The color scheme in the output image represents diff clusters identified by the K-means algorithm



- The meaning of the color is determined by the clustering results of the K-means algorithm in each run. You can use the average pixel value of each cluster to interpret what kind of changes each color represents

Thank you

Vaani Bakshi

bakshivaani@gmail.com