**Dataset Link :** <https://drive.google.com/drive/folders/1D2twZVdDM4HNuDNr_Vrsg_ObBIFN_gXA?usp=sharing>

**Source Code For Thyroid Detection:**

import numpy as np

import os

from sklearn import metrics

from sklearn.model\_selection import train\_test\_split

from skimage import io, color, feature, exposure

from skimage.transform import resize

from sklearn.neighbors import KNeighborsClassifier

# Load images and labels

image\_dir = "/content/drive/MyDrive/dataset"

images = []

labels = []

for class\_folder in os.listdir(image\_dir):

class\_folder\_path = os.path.join(image\_dir, class\_folder)

if os.path.isdir(class\_folder\_path):

for image\_path in os.listdir(class\_folder\_path):

if image\_path.endswith(('.jpg', '.png', '.jpeg')): # Ensure you are loading only image files

img = io.imread(os.path.join(class\_folder\_path, image\_path))

img\_gray = color.rgb2gray(img) # Convert to grayscale

img\_resized = resize(img\_gray, (128, 128)) # Assuming you want to resize images

images.append(img\_resized)

labels.append(class\_folder)

# Extract HOG features

hog\_features = []

for image in images:

fd = feature.hog(image, orientations=8, pixels\_per\_cell=(8, 8),

cells\_per\_block=(2, 2), multichannel=False)

hog\_features.append(fd)

# Run the process multiple times to get 40 accuracy values

num\_iterations = 20

accuracy\_values = []

# Initialize variables to keep track of the maximum accuracy and its corresponding iteration

max\_accuracy = 0

max\_accuracy\_iteration = 0

for i in range(num\_iterations):

# Split the dataset into training and testing sets

X\_train, X\_test, y\_train, y\_test = train\_test\_split(hog\_features, labels, test\_size=0.3, random\_state=None)

# Create a KNN classifier

clf = KNeighborsClassifier(n\_neighbors=5) # You can adjust the number of neighbors as needed

# Train the KNN classifier

clf.fit(X\_train, y\_train)

# Predict using the trained KNN classifier

y\_pred = clf.predict(X\_test)

# Compute accuracy

accuracy = metrics.accuracy\_score(y\_test, y\_pred)

accuracy\_values.append(accuracy)

# Update the maximum accuracy and its corresponding iteration

if accuracy > max\_accuracy:

max\_accuracy = accuracy

max\_accuracy\_iteration = i + 1

# Print the maximum accuracy and its corresponding iteration

print(f"\nMaximum Accuracy: {max\_accuracy \* 100:.2f}% (Iteration {max\_accuracy\_iteration})")