CHANGE DETECTION IN SATELLITE IMAGES USING CONVOLUTIONAL NEURAL NETWORKS AND PCA K MEANS CLUSTERING

Team Members:

Rudra Pratap Dhara - 16H61A0441

M.Manoj Kumar - 16H61A0430

K.Pradeep Kumar - 16H61A0426

Under the guidance of:

Mrs.Neetu Srivatsava-Assistant Professor

Contents:

- Aim of the project(Abstract)
- Block Diagram
- Flow chart
- Code(snapshot)
- Outputs & Results
- Applications
- Video(explanation of code & outputs)

Aim of the project :

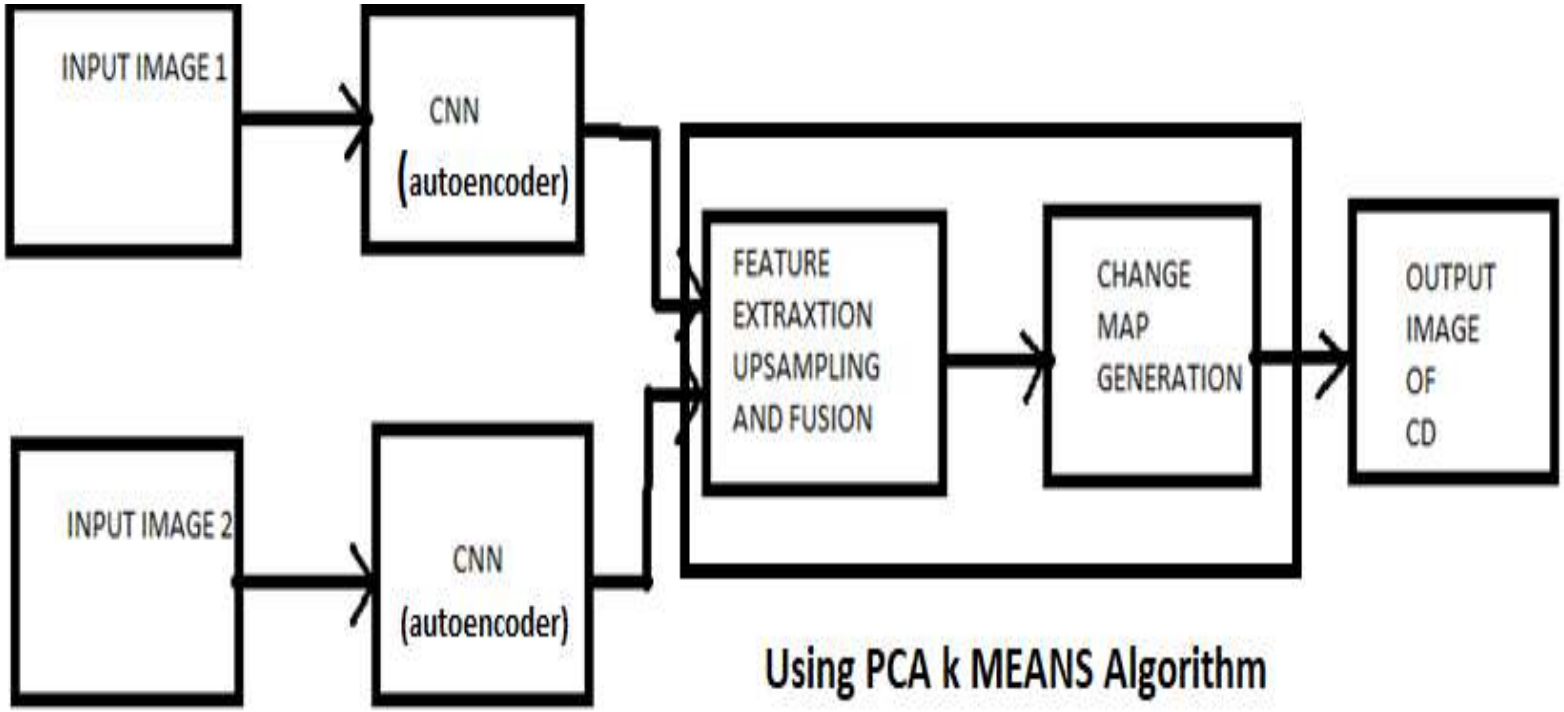
•With the popular use of high resolution remote sensing (HRRS) satellite images, a huge research efforts have been placed on change detection (CD) problem.

•An effective feature selection method can significantly boost the final result. While hand-designed features have proven difficulties to design features that effectively capture high and mid-level representations, the recent developments in machine learning (Deep Learning) omit this problem by learning hierarchical representation in an unsupervised manner directly from data without human intervention. In this letter,

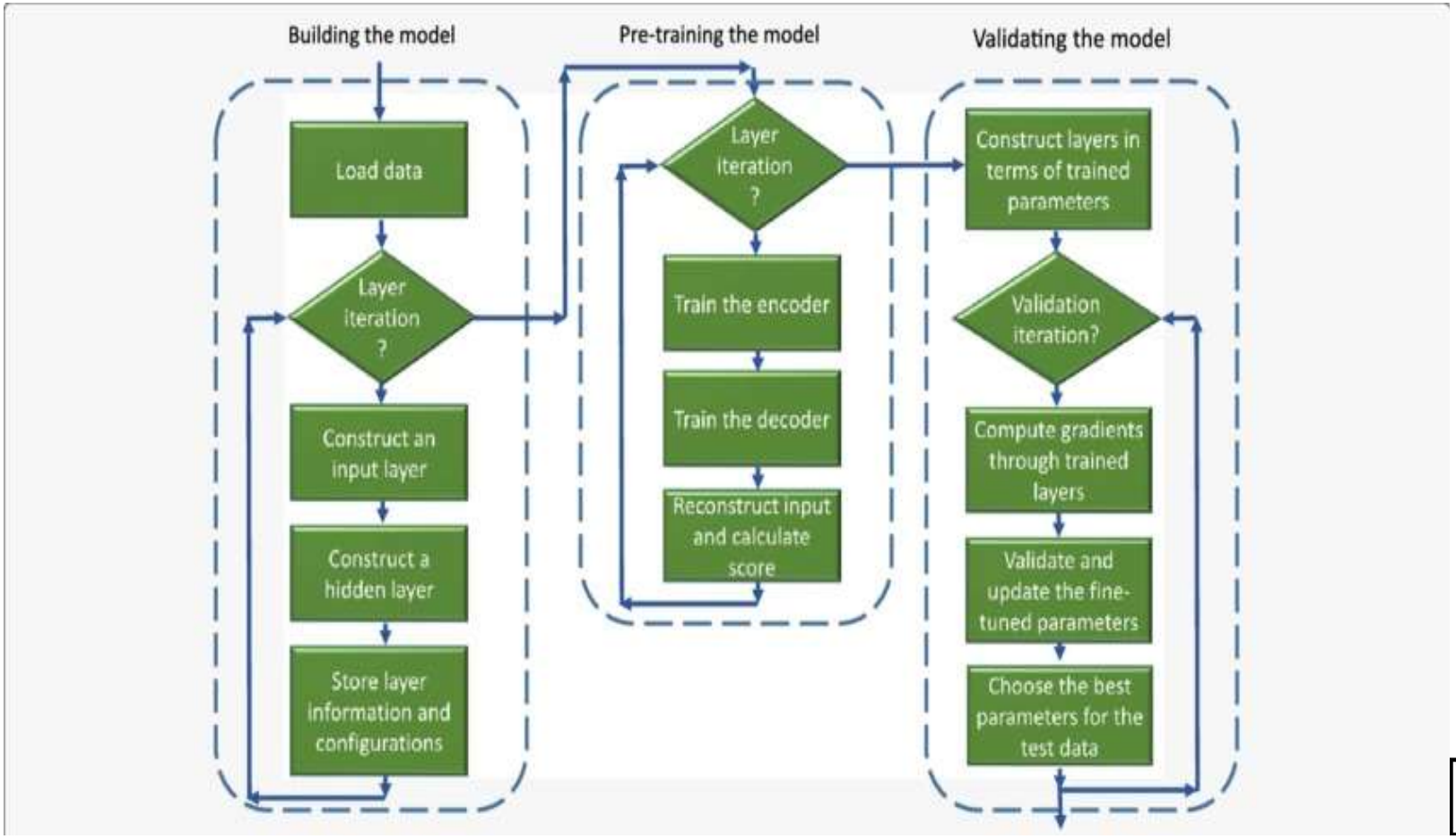
•we propose approaching the change detection problem from a feature learning perspective. A novel deep Convolutional Neural Networks (CNN) features based HR satellite images change detection method is proposed. The main guideline is to produce a change detection map directly from two images using a pre-trained CNN. This method can omit the limited performance of hand-crafted features. Firstly, CNN features are extracted through different Convolutional layers. Then, a concatenation step is evaluated after a normalization step, resulting in a unique higher dimensional feature map. Finally, a change map was computed using pixel-wise Euclidean distance.

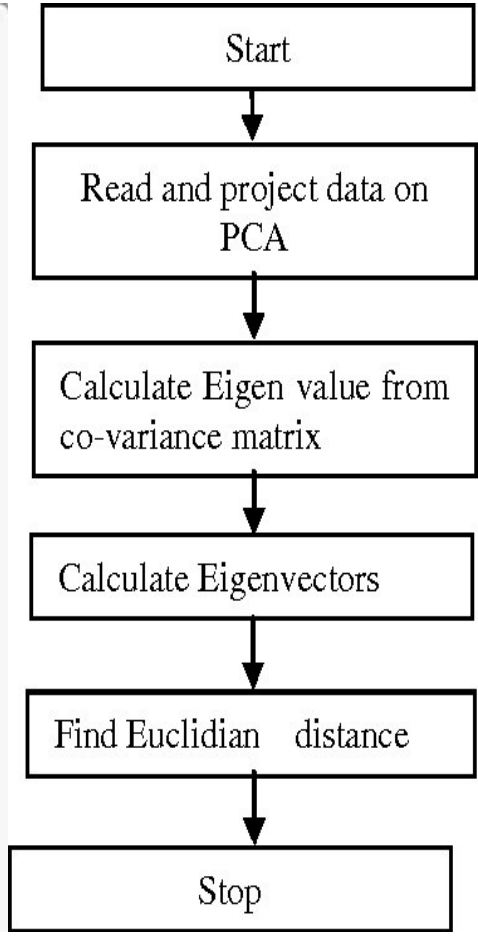
• Our method has been validated on real bi-temporal HRRS satellite images according to qualitative and quantitative analyses. The results obtained confirm the interest of the proposed method.

Block Diagram :



Flow Chart:


Building the model
Load data
Layer iteration ?
Construct an input layer
Construct a hidden layer
Store layer information and configurations
Pre-training the model
Layer iteration ?
Train the encoder
Train the decoder
Reconstruct input and calculate score
Validating the model
Construct layers in terms of trained parameters
Validation iteration?
Compute gradients through trained layers
Validate and update the fine-tuned parameters
Choose the best parameters for the test data


Start
Read and project data on PCA
Calculate Eigen value from co-variance matrix
Calculate Eigenvectors
Find Euclidian distance
Stop

Code:

maincode.py - C:\Users\DESKTOP PC\Desktop\New folder (3)\mlproject\maincode.py (3.6.8)

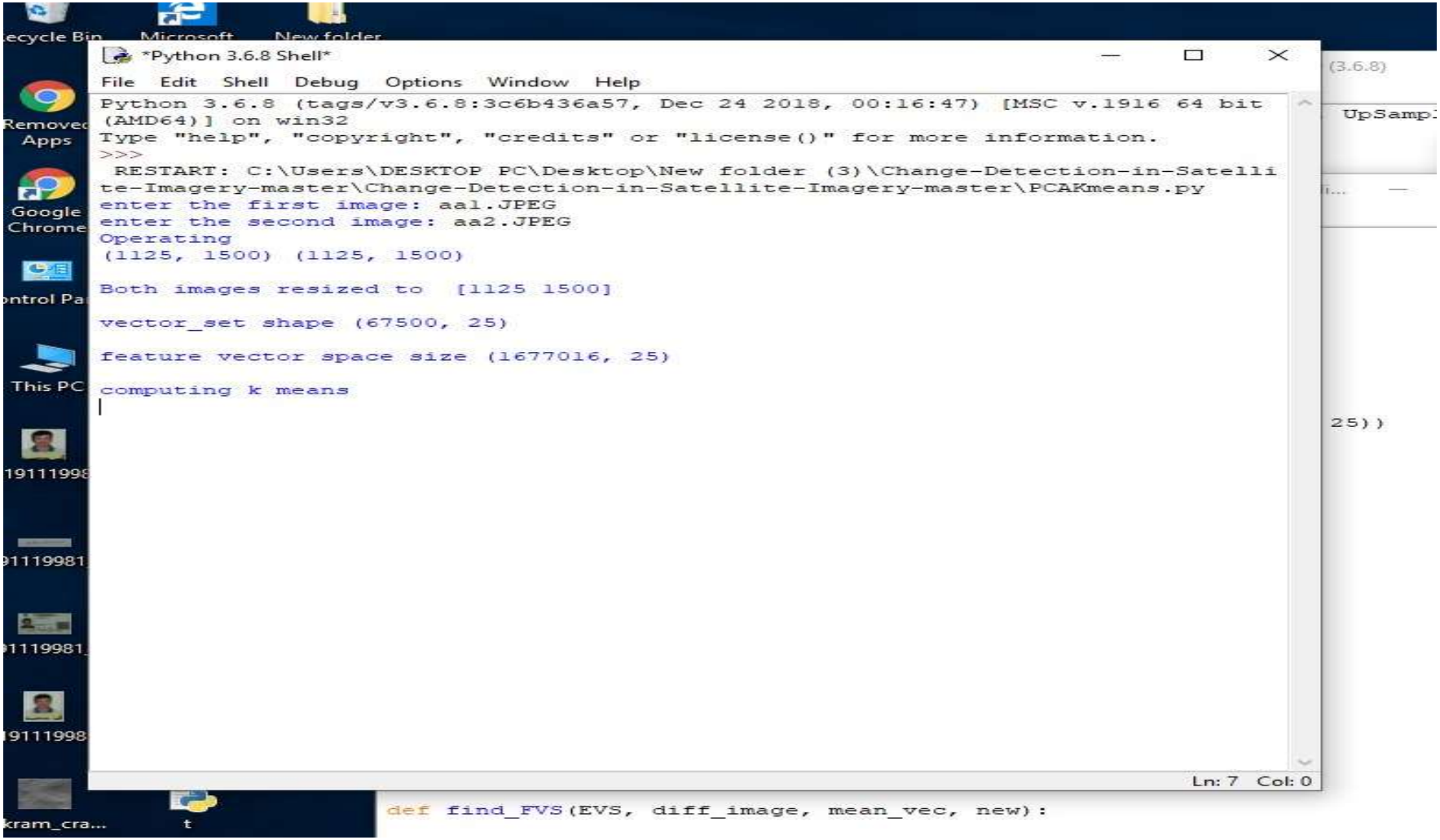
File Edit Format Run Options Window Help

```python
from keras.layers import Input, Dense, Conv2D, MaxPooling2D, UpSampling2D
from keras.models import Model
from keras.datasets import mnist
from keras import backend as K
import numpy as np
import os
import cv2
x_train = []
for i in os.listdir("train images"):
    img = cv2.imread("train images/"+i)
    img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
    img = cv2.resize(img, (500, 500))
    x_train.append(img)
x_train = np.array(x_train)
input_img = Input(shape=(500, 500, 3))  # adapt this if using `channels_first` i

x = Conv2D(16, (3, 3), activation='relu', padding='same')(input_img)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
x = MaxPooling2D((2, 2), padding='same')(x)
x = Conv2D(8, (3, 3), activation='relu', padding='same')(x)
encoded = MaxPooling2D((2, 2), padding='same')(x)

y = Conv2D(8, (3, 3), activation='relu', padding='same')(encoded)
y = UpSampling2D((2, 2))(y)
y = Conv2D(8, (3, 3), activation='relu', padding='same')(y)
y = UpSampling2D((2, 2))(y)
y = Conv2D(16, (3, 3), activation='relu')(y)
y = UpSampling2D((2, 2))(y)
decoded = Conv2D(3, (3, 3), activation='sigmoid', padding='same')(y)

autoencoder = Model(input_img, decoded)
autoencoder.compile(optimizer='adadelta', loss='sparse_categorical')
encoder = Model(input_img, encoded)
autoencoder.fit(x_train, x_train, epochs=100)
model_data = autoencoder.to_json()
fl = open("Model.JSON", "w")
fl.write(model_data)
fl.close()
autoencoder.save("Model.h5")
```

Ln: 1 Col: 0

```python
    print(image1.shape,image2.shape)
    new_size = np.asarray(image1.shape) / 5
    new_size = new_size.astype(int) * 5
    image1 = imresize(image1, (new_size)).astype(np.int16)
    image2 = imresize(image2, (new_size)).astype(np.int16)

    diff_image = abs(image1 - image2)
    imsave('diff.jpg', diff_image)
    print('\nBoth images resized to ',new_size)

    vector_set, mean_vec = find_vector_set(diff_image, new_size)

    pca     = PCA()
    pca.fit(vector_set)
    EVS = pca.components_

    FVS     = find_FVS(EVS, diff_image, mean_vec, new_size)

    print('\ncomputing k means')

    components = 3
    least_index, change_map = clustering(FVS, components, new_size)

    change_map[change_map == least_index] = 255
    change_map[change_map != 255] = 0

    change_map = change_map.astype(np.uint8)
    kernel     = np.asarray(((0,0,1,0,0),
                             (0,1,1,1,0),
                             (1,1,1,1,1),
                             (0,1,1,1,0),
                             (0,0,1,0,0)), dtype=np.uint8)
    cleanChangeMap = cv2.erode(change_map,kernel)
    imsave("cm.jpg", change_map)
    imsave("ccm.jpg", cleanChangeMap)


if __name__ == "__main__":
    a = input("enter the first image: ")
    b = input("enter the second image: ")
    a = cv2.cvtColor(cv2.imread(a), cv2.COLOR_BGR2GRAY)
    b = cv2.cvtColor(cv2.imread(b), cv2.COLOR_BGR2GRAY)
    imsave("forest_1986.jpg",a)
    imsave("forest_1992.jpg",b)
    #a = 'forest_1986.jpg'
    #b = 'forest_1992.jpg'
    #b1 = 'Dubai_11272000.jpg'
    #a2 = 'Andasol_09051987.jpg'
    #b2 = 'Andasol_09122013.jpg'
    find_PCAKmeans(a,b)
```

Code:


Python 3.6.8 Shell
File Edit Shell Debug Options Window Help
Python 3.6.8 (tags/v3.6.8:3c6b436a57, Dec 24 2018, 00:16:47) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
RESTART: C:\Users\DESKTOP PC\Desktop\New folder (3)\Change-Detection-in-Satellite-Imagery-master\Change-Detection-in-Satellite-Imagery-master\PCAKmeans.py
enter the first image: aa1.JPEG
enter the second image: aa2.JPEG
Operating
(1125, 1500) (1125, 1500)
Both images resized to [1125 1500]
vector_set shape (67500, 25)
feature vector space size (1677016, 25)
computing k means
Ln: 7 Col: 0
def find_FVS(EVS, diff_image, mean_vec, new):

Outputs & Results :

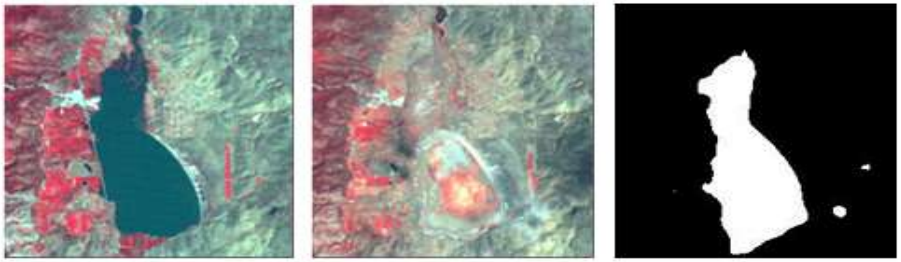
RESULT 1:

Image 1 | image 2 | output

forest_1986.png | forest_1992.png | forest.jpg

RESULT 2:

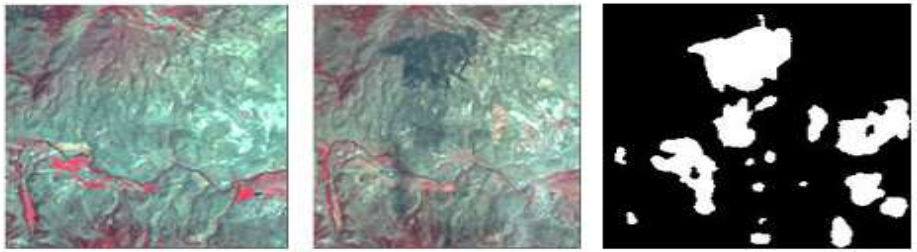
Image 1 | image 2 | output



lake_1986.png | lake_1992.png | lake.jpg

RESULT 3:

Image 1 | image 2 | output



burn_1986.png | burn_1992.png | burn.jpg

RESULT 4:

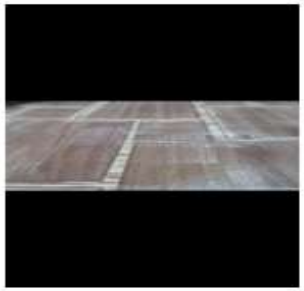
Image 1 | image 2 | output

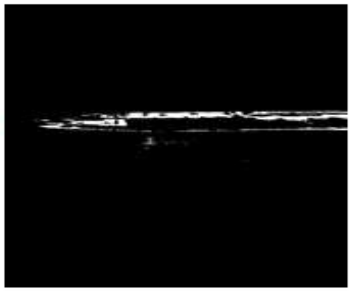
conifer_1986.png | conifer _1992.png | conifer.jpg

Outputs & Results :

RESULT 5:

Image 1 **image 2** **output**





pen_bef.png **pen_aft.png** **pen.jpg**

RESULT 6:

Image 1 **image 2** **output**



team_bef.png **team_aft.png** **team.jpg**

ACCURACY:

The accuracy for above images are as follows:

Image Name	Image Number	Accuracy (in %)
Forest	Result 1	82
Lake	Result 2	84
Burn	Result 3	85.36
Conifer	Result 4	87.74
Pen	Result 5	88.3
Team	Result 6	91.07

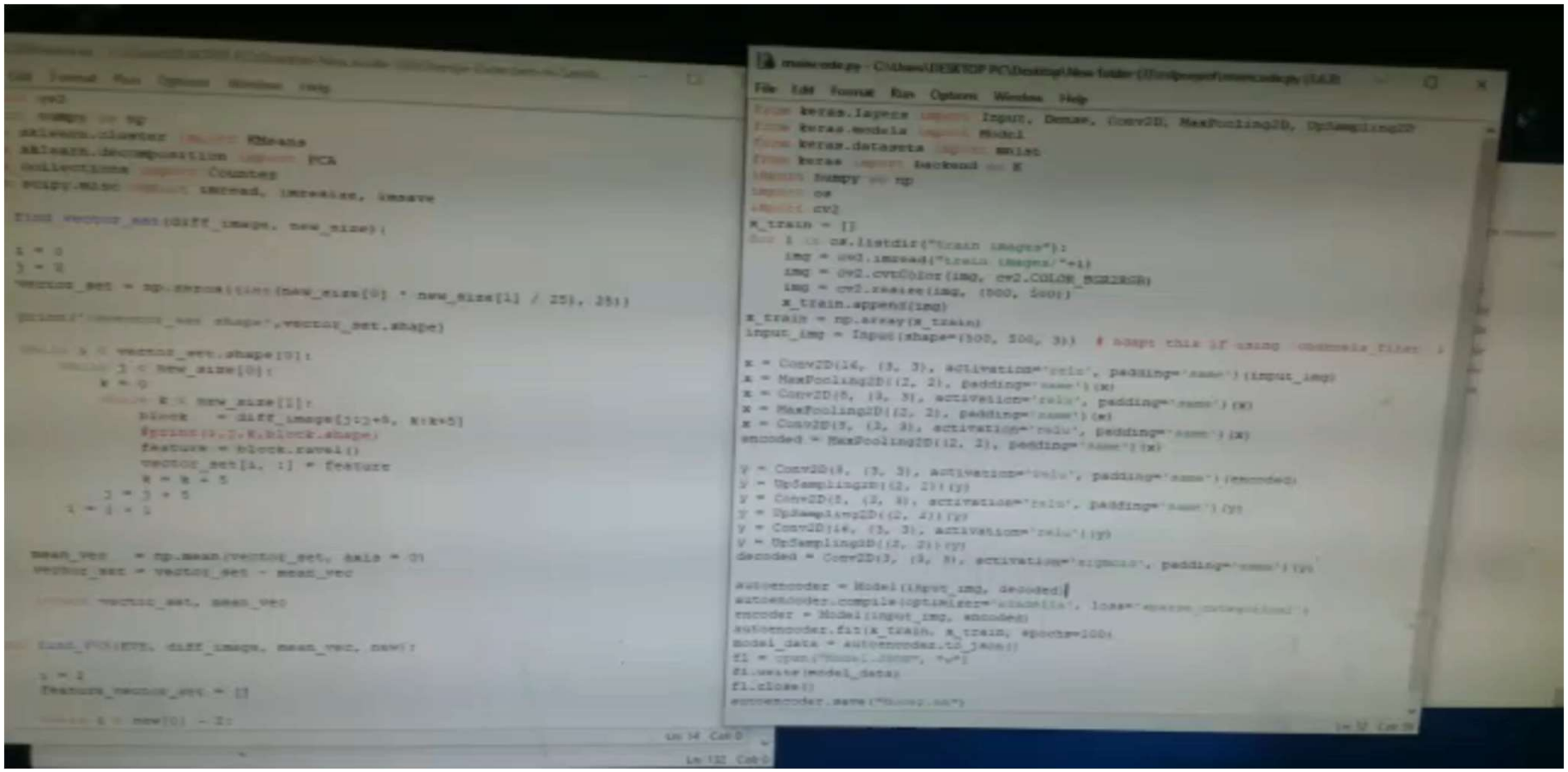
Applications :

- Medical image diagnosis
- Military surveillance
- Agriculture land monitoring
- Astronomical phenomenon
- Civilization density

Conclusion :

In this paper, a novel CD is presented, based on CNN hyper features of two registered images, covering the same area took at different times, t1 and t2. Experiments has demonstrated the effectiveness of the CNN features on CD task. By fusing the information from several layers, the CD performances are greatly enhanced. The technique has a drawback of requirement of more computational time and the necessity of the registration step.

Output Video :



THANK YOU