Feature Extraction of Hyperspectral Image

UNDER THE GUIDANCE OF

B. SUCHARITHA

BATCH **– A4**

DATE**- 9-06-2022**

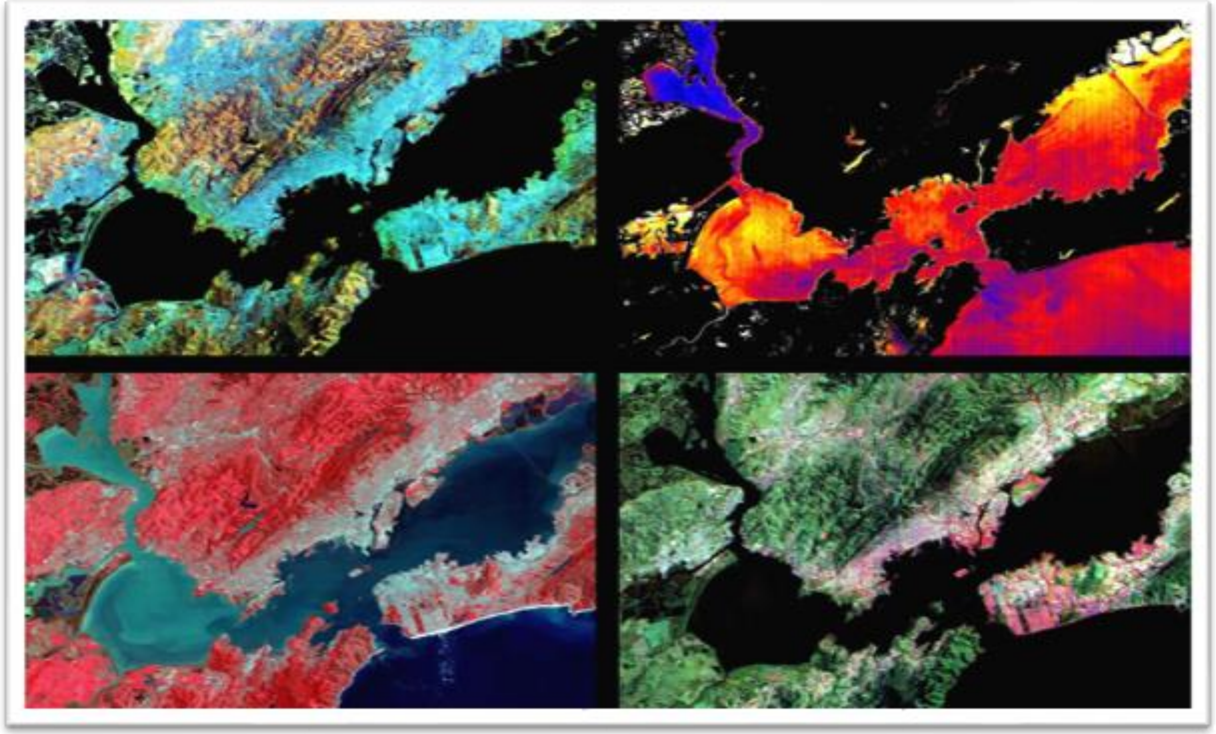
PRESENTED BY

NIDA ZIA - 1604-18-735-002

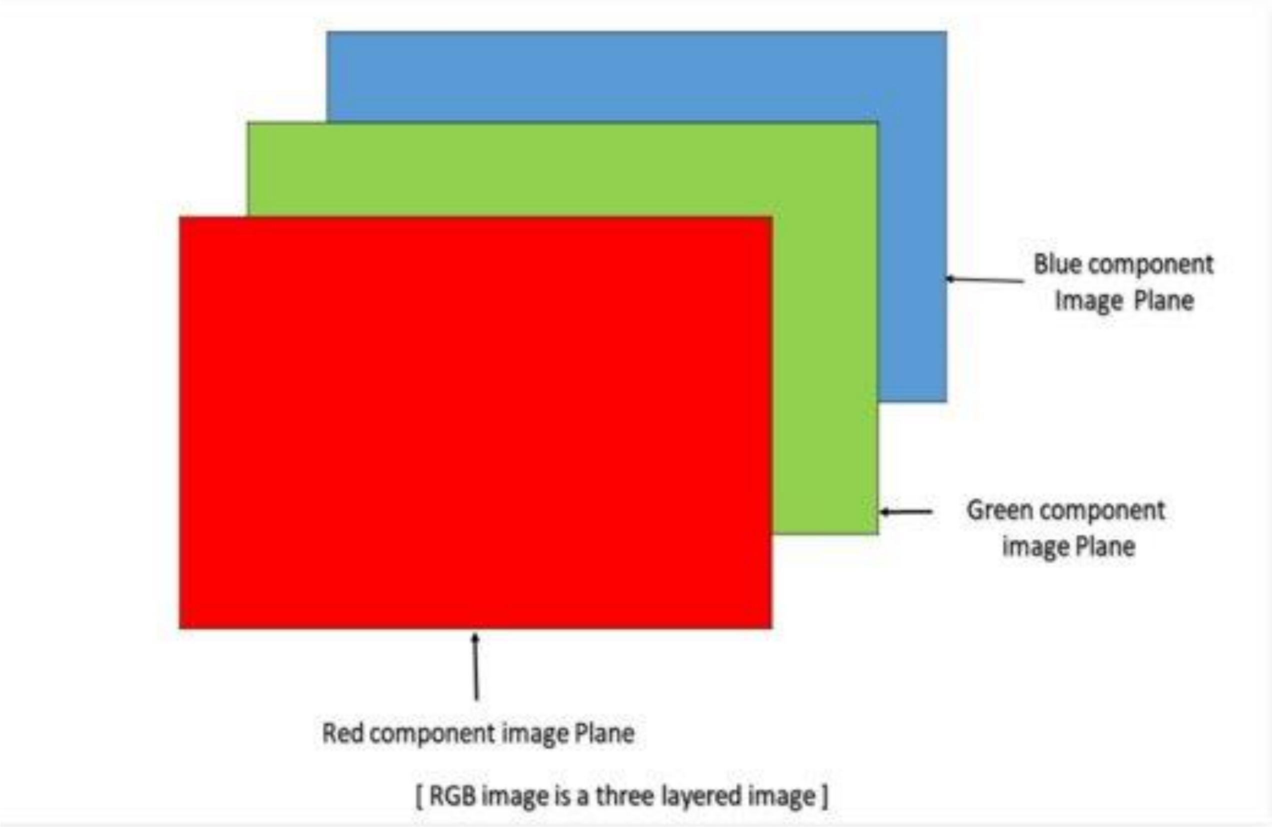
TAHEREEN RIZVI - 1604-18-735-010

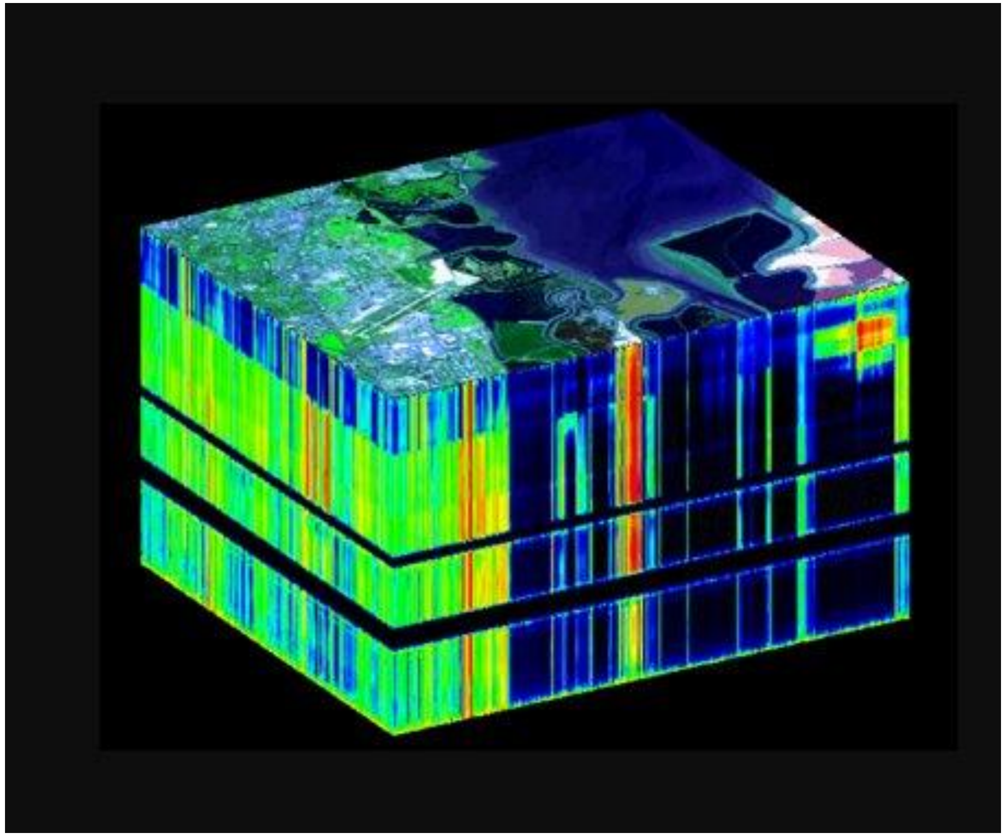
SYED DIRAAR AHMED - 1604-18-735-021

Contents



DIFFERENCE BETWEEN COLOR AND HSI IMAGE

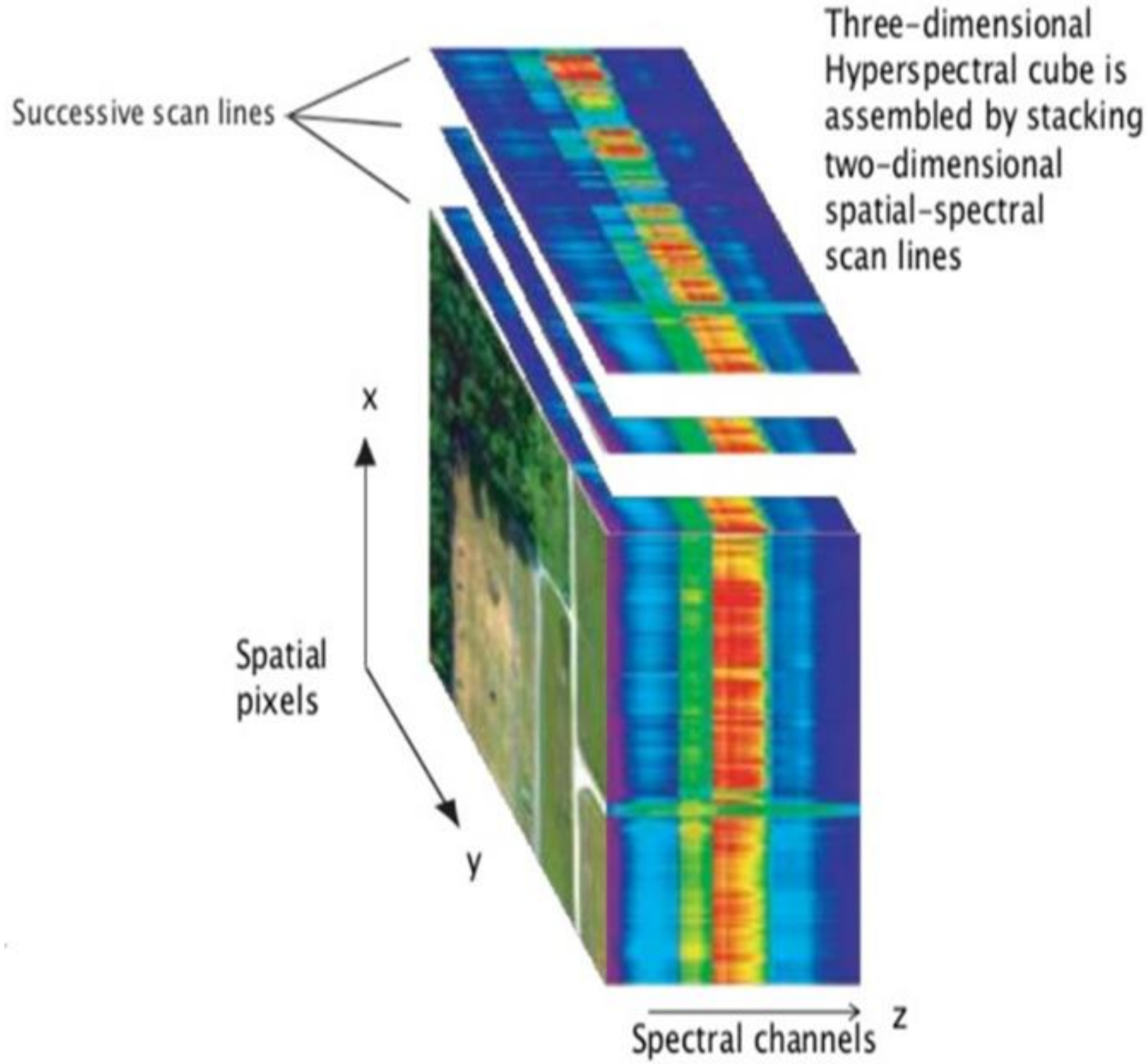




HSI

- Hyperspectral image is a three dimensional cube , which contains 2D spatial information and 1D spectral information.

- The images are combined to form a three-dimensional (x,y,λ) hyperspectral data cube

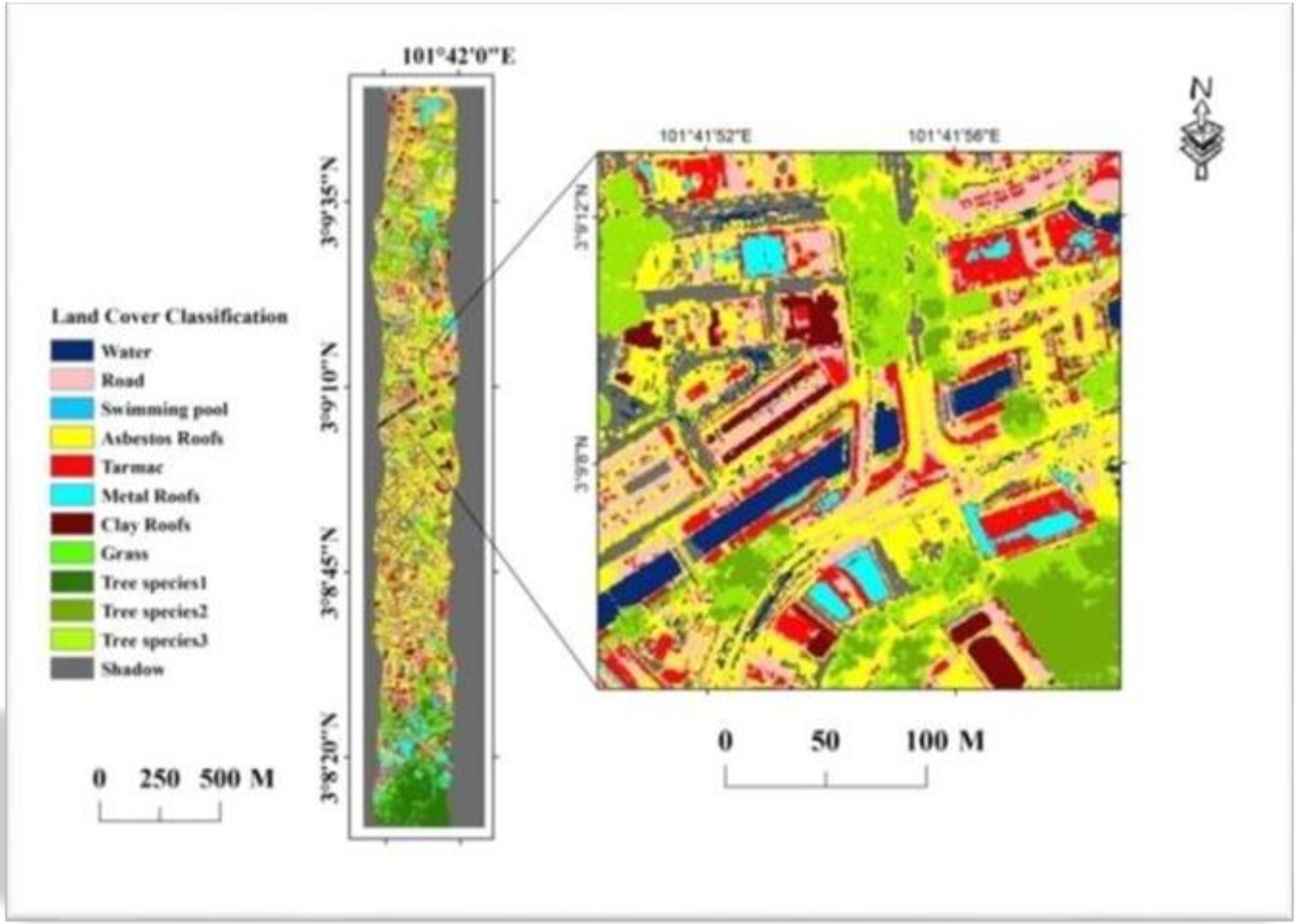


NEED FOR HYPERSPECTRAL IMAGE COMPRESSION

- Less storage space
- Data transmission time
- Bandwidth

NEED FOR HYPERSPECTRAL IMAGE CLASSIFICATION

- Object/image identification
- Identify geologic terrains
- Mineral exploration
- Land use and land cover



DESIGN METHODOLOGY

1. Read the hyper spectral image (from a data set)

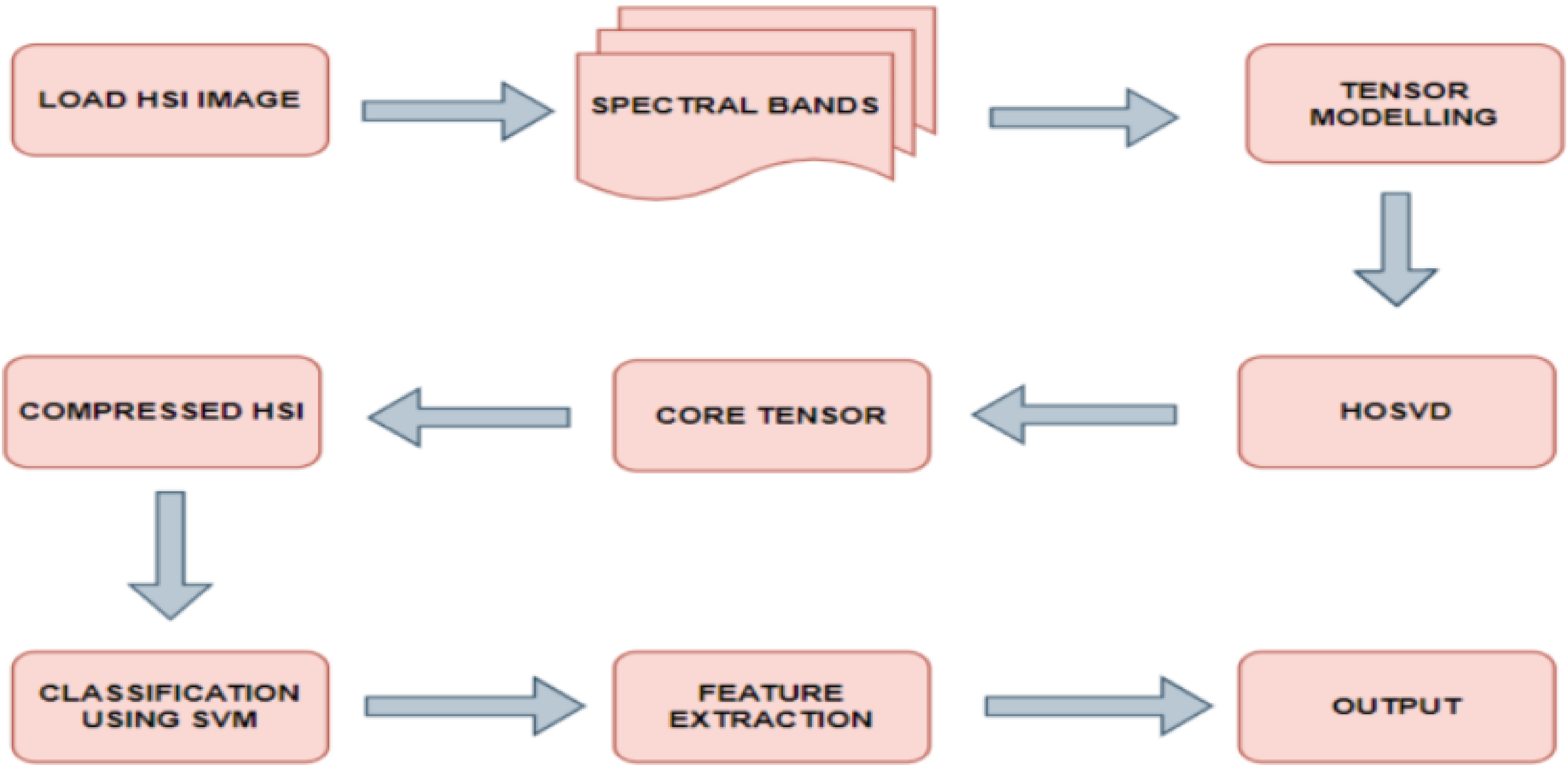
2. Apply **Tensor decomposition** on the image.

3. Compressed image is obtained.

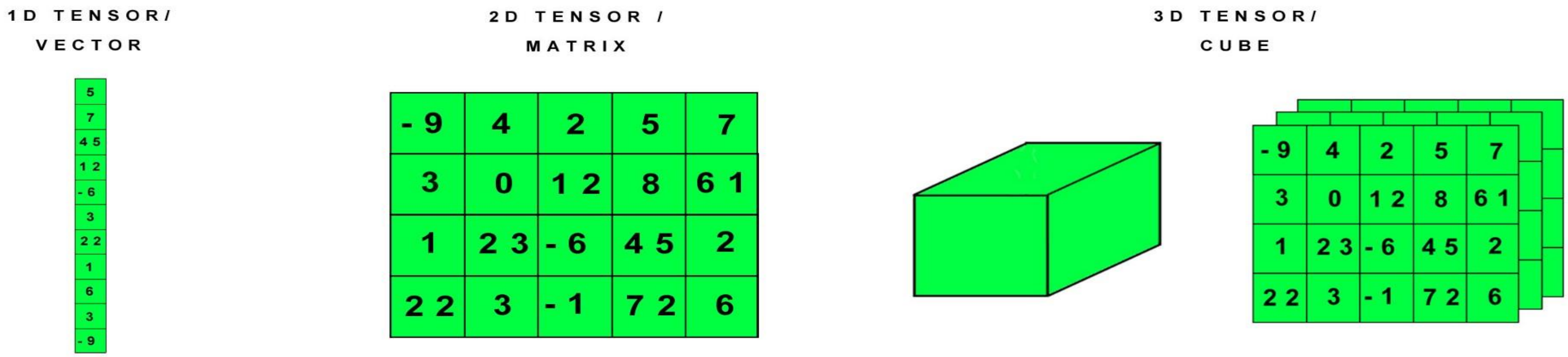
4. Classification of the image is done using **Support Vector Machine (SVM)** algorithm.

5. Features of classified image are extracted

BLOCK DIAGRAM

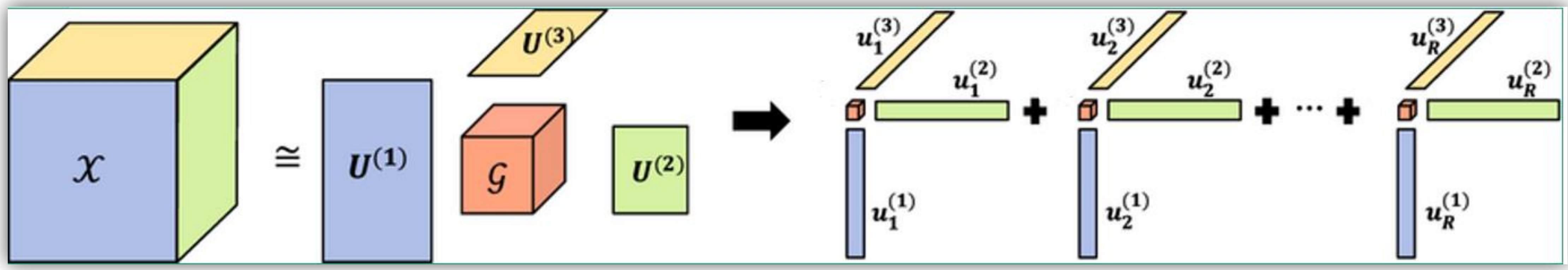


TENSOR



- A tensor is a multidimensional (n-order) array.
- 1rst order tensor is a vector
- 2nd order tensor is a matrix
- If the order is 3 and above, then they are known as higher order tensors.

TENSOR DECOMPOSITION



$$X \approx G \times U_1 \times U_2 \times U_3$$

CODE OF COMPRESSION

Editor - C:\Users\rizvi\Compression_Code.m

Compression_Code.m

```
clc;clear all;close all;
 addpath tensor_toolbox-v3.1;
  load('Indian_pines_corrected.mat');
[M N C]=size(indian_pines_corrected);
for i=1:10
  figure(1);
  imagesc(indian_pines_corrected(:,:,i))
  colormap('jet'); brighten(0.5);
  title([' Original Image Band- 10']);
  xlabel(' No of Columns')
  ylabel('No of Rows')
end
 tic;
for i = 1:C
    %***********Spectral Band Display*************
    slice = indian_pines_corrected(:,:,i);
    figure(2);
   imshow(slice,[]);
   colormap('gray'); brighten(0.5); title(['Band - ',num2str(i)]);
 [cA1,cH1,cV1,cD1] = dwt2(slice,'bior6.8');
 ca1(:,:,i)=cA1;
 ch1(:,:,i)=cH1;
 cv1(:,:,i)=cV1;
 cd1(:,:,i)=cD1;
 toc;
tic;
 x1=tensor(indian_pines_corrected);
    T1 = hosvd(x1,0.05);
  coresize1 = size(T1.core)
 u1=size(T1.U{1})
 u2=size(T1.U{2})
 u3=size(T1.U{3})
for i=1:5
 T11 = ttensor(T1.core,T1.U);
    t11=double(T11);
 end
 toc;
 s1=size(T1.core)
 s2=[M N C]
 CR=(s2/s1)*100
```

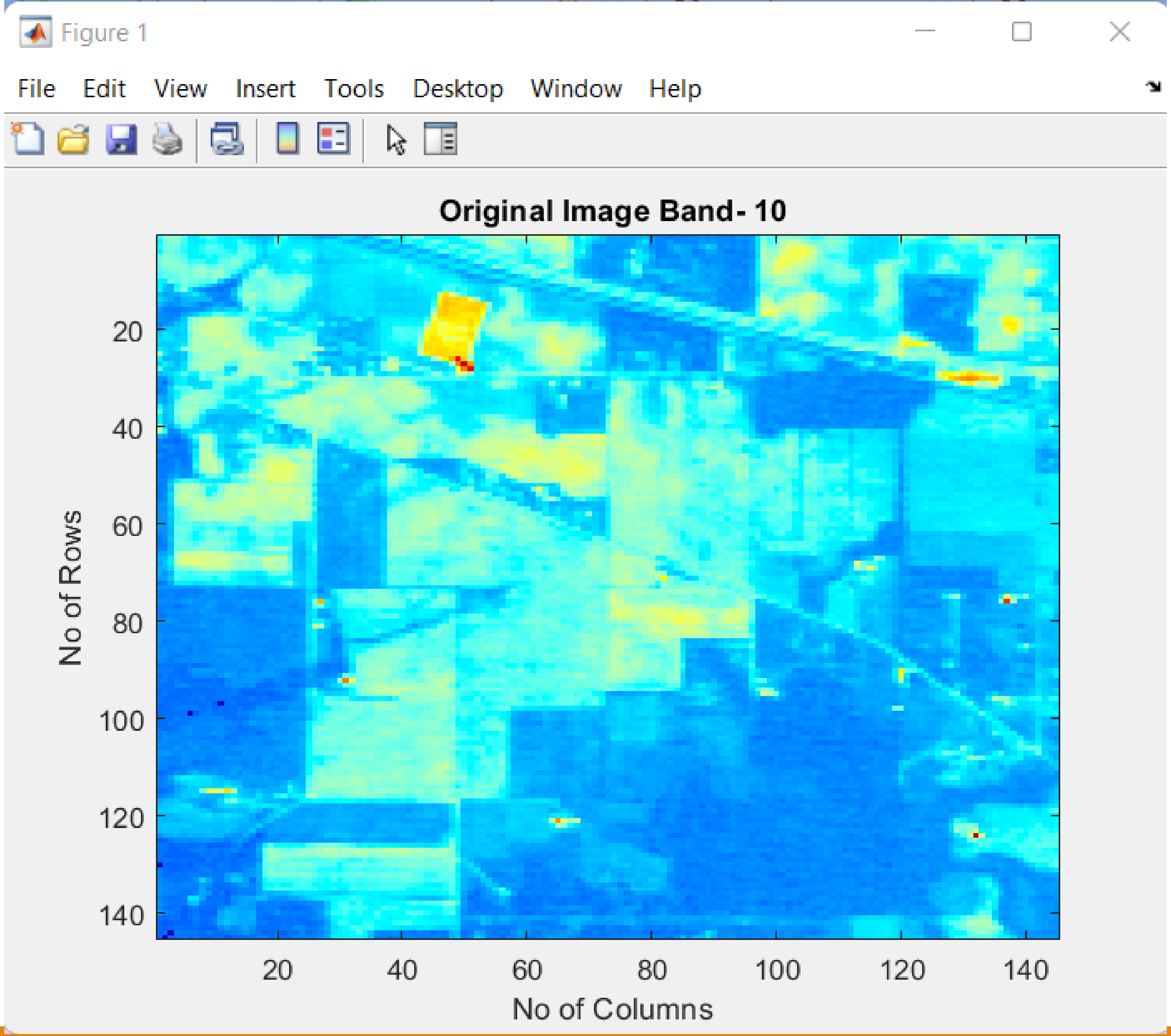


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image Band- 10
No of Rows
20
40
60
80
100
120
140
No of Columns
20
40
60
80
100
120
140

```
Elapsed time is 12.763669 seconds.
Computing HOSVD...
Size of core: 55 x 33 x 3
||X-T||/||X|| = 0.0487458 <=0.050000 (tol)


coresize1 =

    55    33     3


u1 =

   145    55


u2 =

   145    33


u3 =

   200     3

Elapsed time is 0.452287 seconds.
```

```
s1 =

    55    33     3


s2 =

   145   145   200


CR =

  324.0359
```

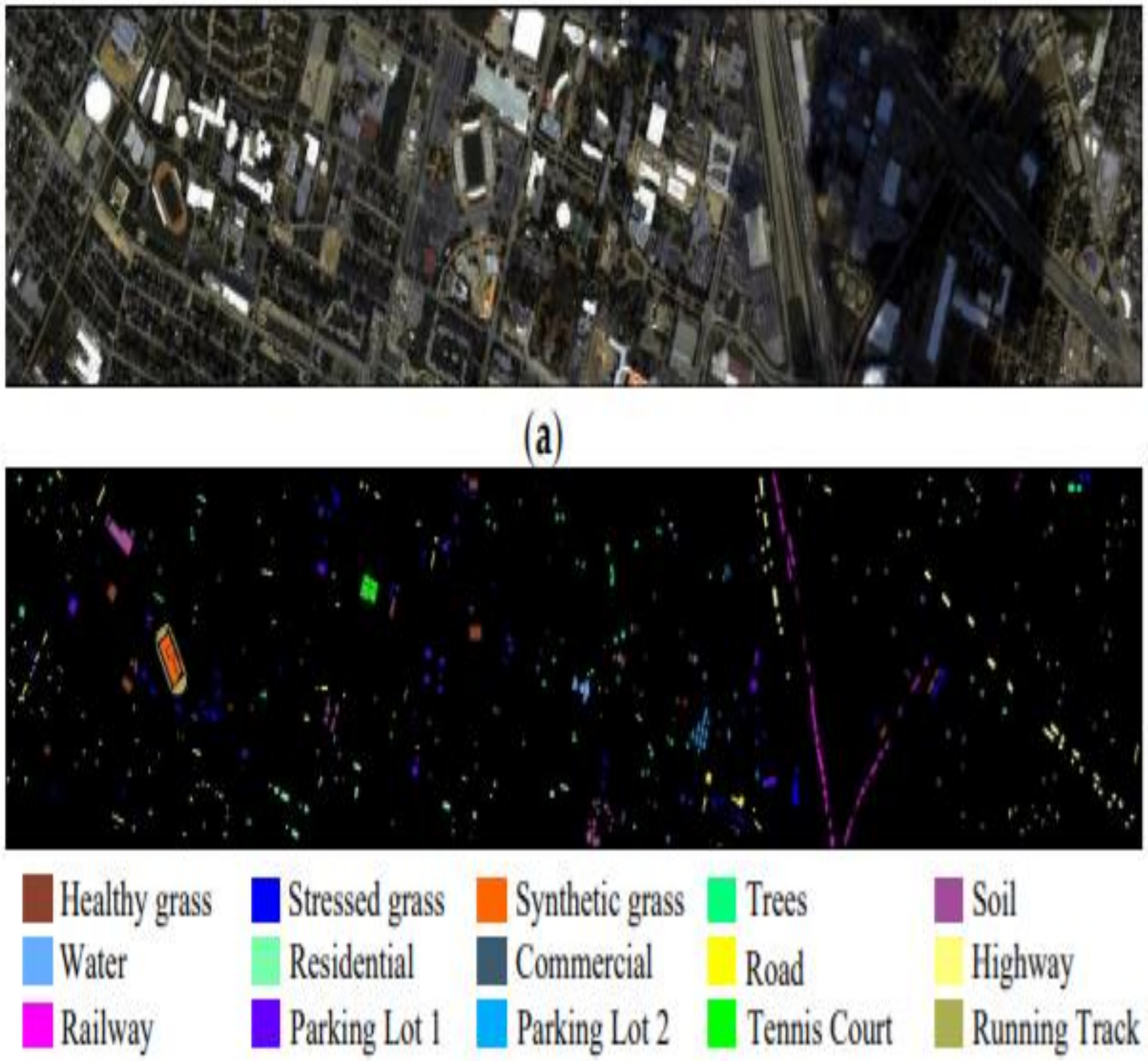
Feature Extraction by Classification of HSI

The basic goal of hyperspectral image classification is to assign a class label to each pixel.

Supervised learning trains models using labelled data.

Y=f(X)

Unsupervised learning trains the model using unsupervised data.

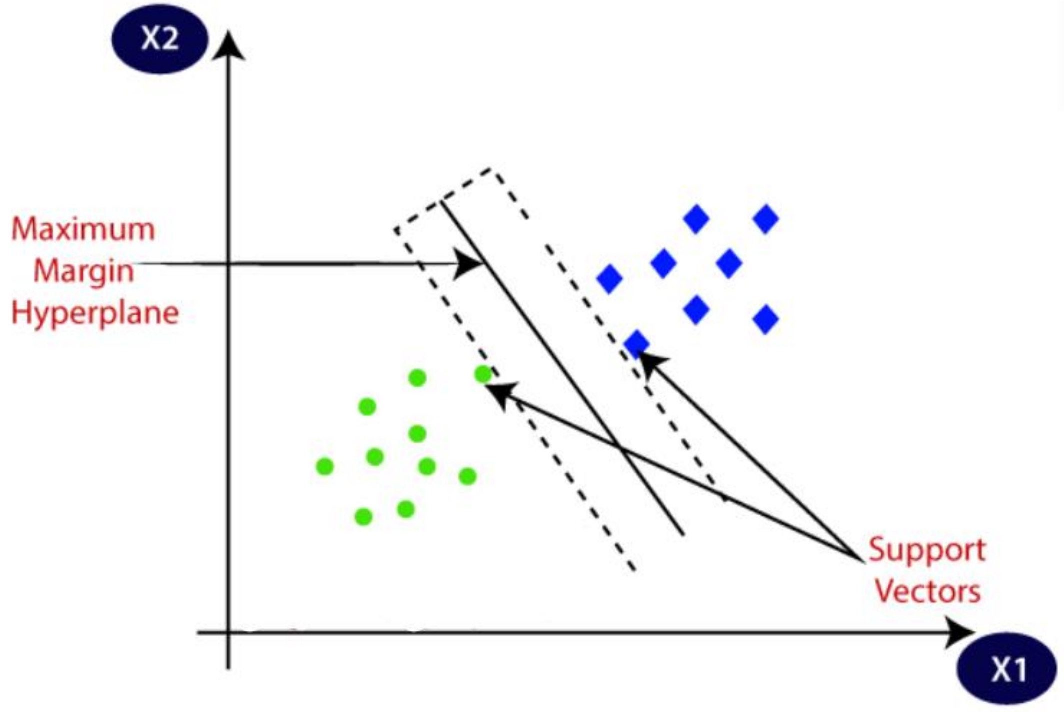


Support Vector Machine (SVM)

It is a supervised learning algorithm used for classification.

The goal of the SVM algorithm is to create a decision boundary that can segregate similar models into classes as shown.

The boundary line that separates the classes is called a Hyperplane.



CLASSIFICATION CODE

```
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
from sklearn.svm import SVC
from sklearn.decomposition import PCA
from sklearn.model_selection import train_test_split
from scipy.io import loadmat
```

```
dataset = loadmat('Indian_pines_corrected.mat')['indian_pines_corrected']
ground_truth = loadmat('Indian_pines_gt.mat')['indian_pines_gt']
```

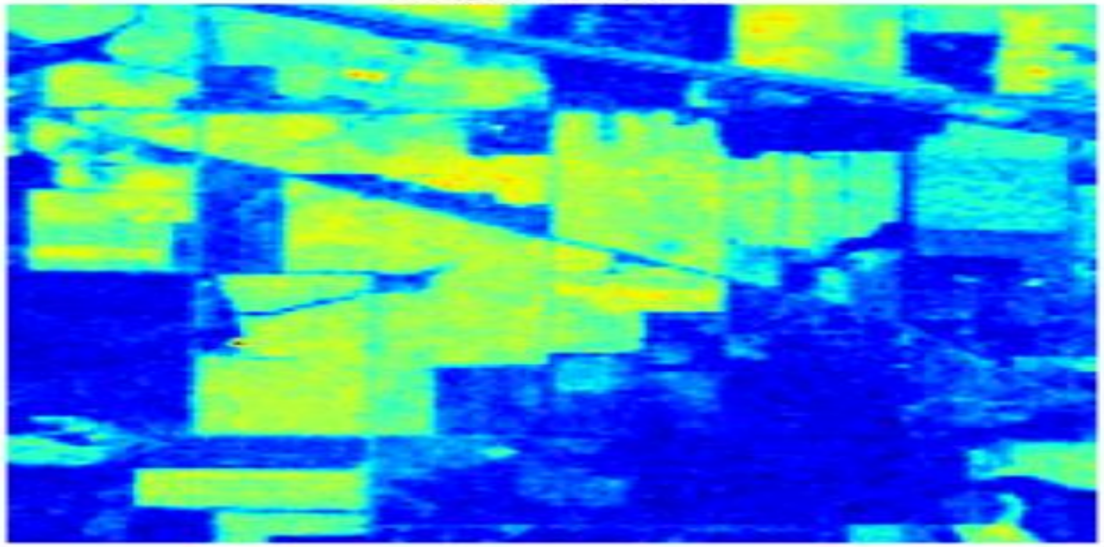
```
print('Dataset:'+ str(dataset.shape))
print('Ground Truth:'+ str(ground_truth.shape))
```

```
Dataset:(145, 145, 200)
Ground Truth:(145, 145)
```

```
def plot_band(dataset):
    plt.figure(figsize=(8, 6))
    image=dataset[:,:, 150]
    plt.imshow(dataset[:,:,150], cmap='jet')
    plt.title('Band-{150}', fontsize=14)
    plt.axis('off')
    plt.colorbar()
    plt.show()
```

```
plot_band(dataset)
```

Original Band



```
df = pd.read_csv('Dataset.csv')
```

```
X = df.iloc[:, :-1].values

y = df.iloc[:, -1].values
```

```
print(y)
```

```
[3 3 3 ... 0 0 0]
```

```
pca = PCA(n_components = 200)

principalComponents = pca.fit_transform(X)
#compressed image
```

```
X_train, X_test, y_train, y_test, indices_train, indices_test  = train_test_split(principalComponents, y,  range(X.shape[0]),test_size = 0.0007, random_state = 0)
```

```
X_train.shape, X_test.shape,len(indices_train),len(indices_test)
```

```
((21010, 200), (15, 200), 21010, 15)
```

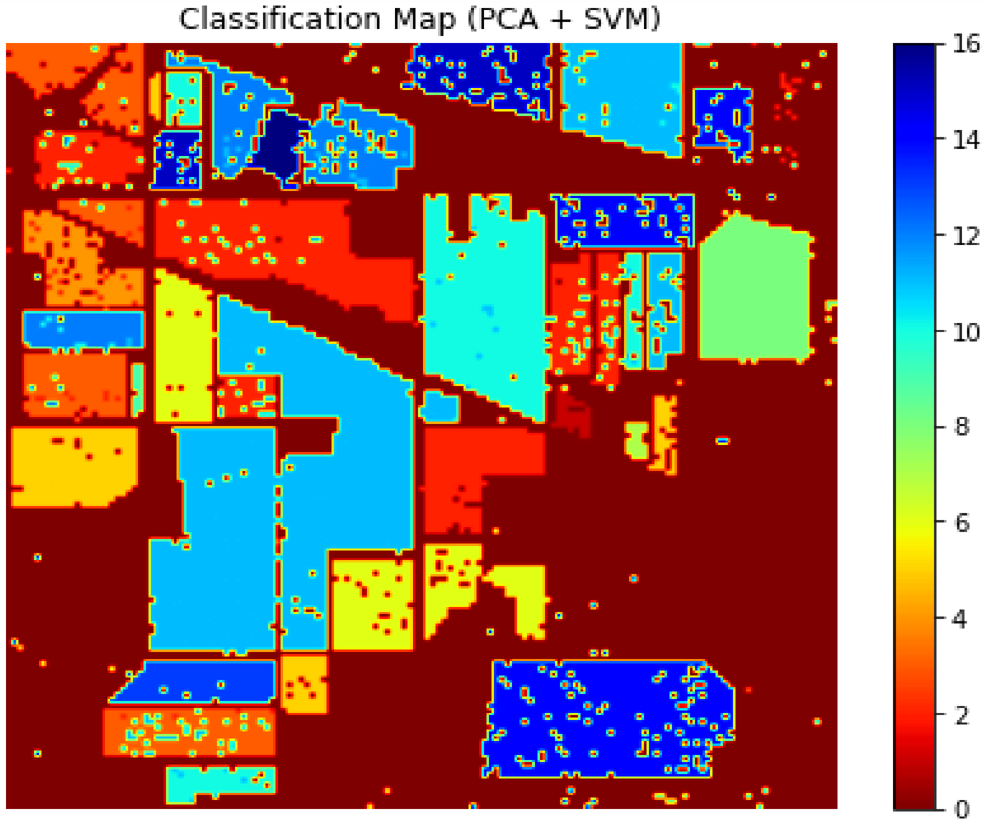
```
svm = SVC(cache_size=1024*7)
svm.fit(X_train, y_train)
```

```
SVC(cache_size=7168)
```

```
y_pred = svm.predict(X_test)
```

```
pre = y_pred

clmap = [0]*X.shape[0]

for i in range(len(indices_train)):
    clmap[indices_train[i]] = y[indices_train[i]]

for i in range(len(indices_test)):
    clmap[indices_test[i]] = pre[i]
```

```
plt.figure(figsize=(8, 6))
img=np.array(clmap).reshape((145, 145))
plt.imshow(img,cmap='jet_r')
plt.colorbar()
plt.axis('off')
plt.title('Classification Map (PCA + SVM)')
plt.savefig('Classification_map.png')
```




```
!pip3 install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\rizvi\anaconda3\lib\site-packages (4.5.5.64)
Requirement already satisfied: numpy>=1.19.3 in c:\users\rizvi\anaconda3\lib\site-packages (from opencv-python) (1.20.3)
```

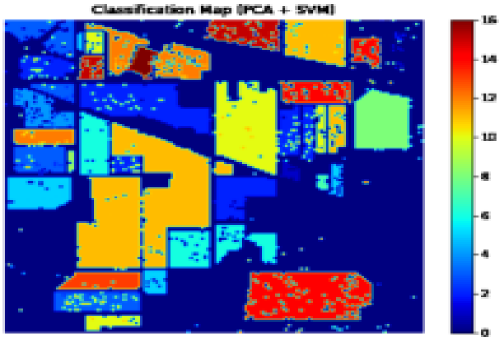
```
import cv2
```

```
img = cv2.imread("Classification_map.png")
```

```
plt.imshow(img,cmap='jet')
plt.axis('off')
```

```
plt.imshow(img,cmap='jet')
plt.axis('off')
```

(-0.5, 575.5, 431.5, -0.5)




```
img.shape
```

(432, 576, 3)

```
hsv_img = cv2.cvtColor(img, cv2.COLOR_RGB2HSV)
```

```
#Wheat 1
lower1=np.array([7,100,100])
up1=np.array([11,255,255])
mask1 = cv2.inRange(hsv_img, lower1, up1)
result1 = cv2.bitwise_and(img,img,mask=mask1)

# woods 2
lower2=np.array([0,255,240])
up2=np.array([1,255,255])
mask2 = cv2.inRange(hsv_img, lower2, up2)
result2 = cv2.bitwise_and(img,img,mask=mask2)

# alfalfa 3
lower3=np.array([110,140,60])
up3=np.array([255,255,180])
mask3 = cv2.inRange(hsv_img, lower3, up3)
result3 = cv2.bitwise_and(img,img,mask=mask3)
```

```
# soybean-clean 4
up4 = np.array([17,255,255])
lower4 = np.array([10, 100, 20])
mask4 = cv2.inRange(hsv_img, lower4, up4)
result4 = cv2.bitwise_and(img,img,mask=mask4)

# soybean mintill 5
up5 = np.array([25,255,255])
lower5 = np.array([20,255,255])
mask5= cv2.inRange(hsv_img, lower5, up5)
result5 = cv2.bitwise_and(img,img,mask=mask5)

# soybean-notill 6
up6=np.array([66,  240,  255])
lower6=np.array([30, 200, 150])
mask6= cv2.inRange(hsv_img, lower6, up6)
result6 = cv2.bitwise_and(img,img,mask=mask6)

# hay-windrowed 7
lower7=np.array([51, 71, 100])
up7=np.array([66,  240,  255])
mask7= cv2.inRange(hsv_img, lower7, up7)
result7 = cv2.bitwise_and(img,img,mask=mask7)

# Stone steel towers 8
lower8=np.array([0,0,0])
up8=np.array([100,255,180])
mask8= cv2.inRange(hsv_img, lower8, up8)
result8 = cv2.bitwise_and(img,img,mask=mask8)

# grass trees 9
lower9=np.array([80,100,100])
up9=np.array([88,255,255])
mask9= cv2.inRange(hsv_img, lower9, up9)
result9 = cv2.bitwise_and(img,img,mask=mask9)

# grass pasture 10
lower10=np.array([94,100,100])
up10=np.array([100,255,255])
mask10= cv2.inRange(hsv_img, lower10, up10)
result10 = cv2.bitwise_and(img,img,mask=mask10)

# corn 11
lower11=np.array([97,100,100])
up11=np.array([105,255,255])
mask11= cv2.inRange(hsv_img, lower11, up11)
result11 = cv2.bitwise_and(img,img,mask=mask11)
```

```python
# corn-mintilla 12
lower12=np.array([105,100,100])
up12=np.array([114,255,255])
mask12= cv2.inRange(hsv_img, lower12, up12)
result12 = cv2.bitwise_and(img,img,mask=mask12)

# corn-notill 13
lower13=np.array([114,100,100])
up13=np.array([118,255,255])
mask13= cv2.inRange(hsv_img, lower13, up13)
result13 = cv2.bitwise_and(img,img,mask=mask13)

# building grass-trees 14
lower14=np.array([0,100,150])
up14=np.array([5,255,220])
mask14= cv2.inRange(hsv_img, lower14, up14)
result14 = cv2.bitwise_and(img,img,mask=mask14)

# grass-pastured-mowed 15
lower15=np.array([72, 100, 100])
up15=np.array([77,  255,  255])
mask15= cv2.inRange(hsv_img, lower15, up15)
result15 = cv2.bitwise_and(img,img,mask=mask15)

# oats 16
lower16=np.array([43, 100, 100])
up16=np.array([45,  255,  255])
mask16= cv2.inRange(hsv_img, lower16, up16)
result16 = cv2.bitwise_and(img,img,mask=mask16)
```

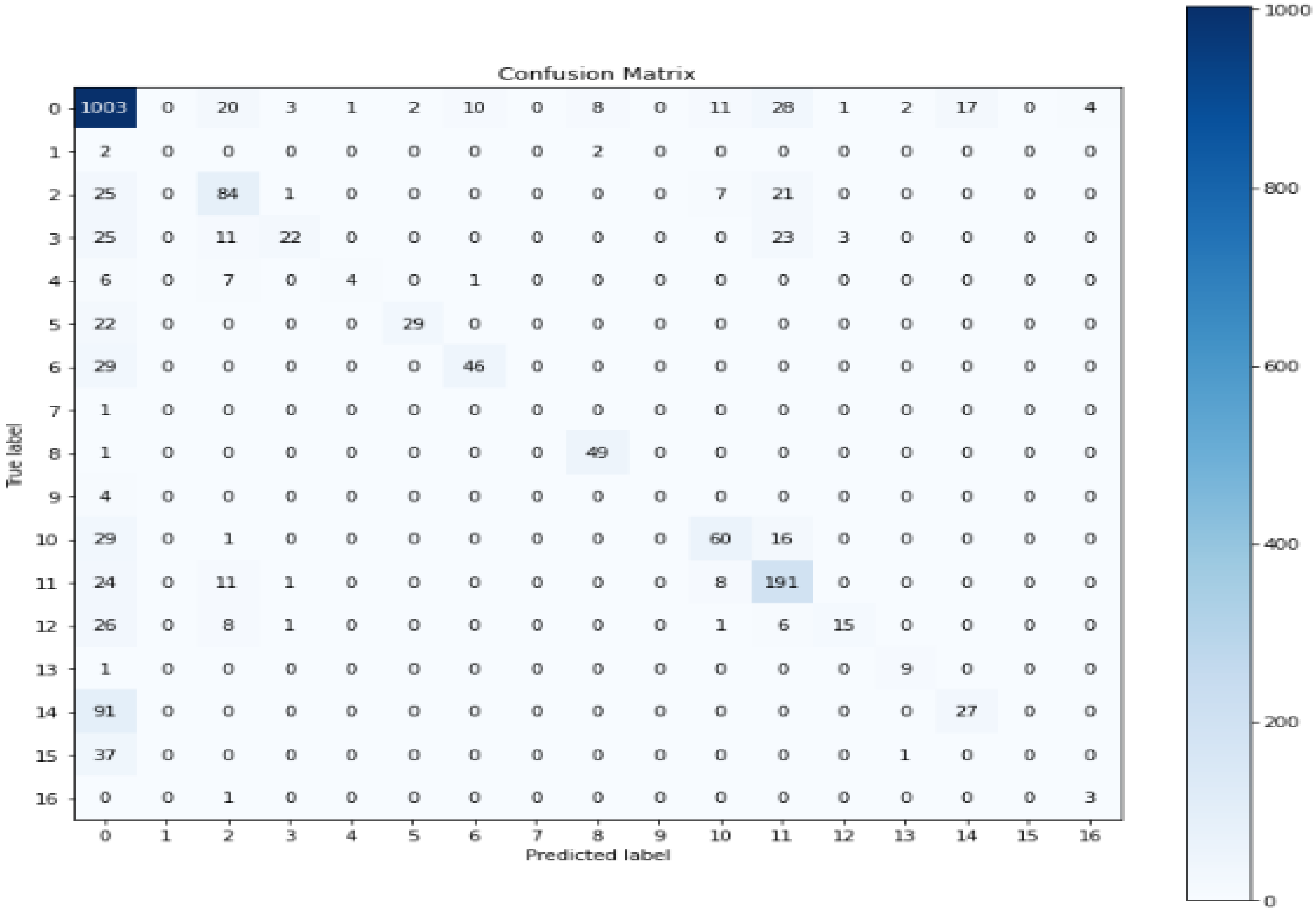
```python
#plt.imshow(result,cmap='jet')
```

```python
from sklearn.metrics import accuracy_score

print(f'Accuracy: {(accuracy_score(y_test, y_pred))*100}%)')
```

```
Accuracy: 73.32382310984308%)
```

```python
! pip install -q scikit-plot
```

```python
import scikitplot as skplt

skplt.metrics.plot_confusion_matrix(
    y_test,
    y_pred,
    figsize=(12,12));
```



```python
from PIL import Image
#wheat
img1 = Image.new('RGB', (10, 10), (255, 66, 3))

#wood
img2 = Image.new('RGB', (10, 10), (255, 9, 0))

#alfalfa
img3 = Image.new('RGB', (10, 10), (0, 65, 200))

#soybean-clean 4
img4 = Image.new('RGB', (10, 10), (255, 137, 0))

# soybean mintill 5
img5 = Image.new('RGB', (10, 10), (255, 209, 0))

# soybean-notill 6
img6 = Image.new('RGB', (10, 10), (226, 255, 3))

# hay-windrowed 7
img7 = Image.new('RGB', (10, 10), (128, 255, 139))

# Stone steel towers 8
img8 = Image.new('RGB', (10, 10), (161, 3, 0))

# grass trees 9
img9 = Image.new('RGB', (10, 10), (61, 255, 237))

# grass pasture 10
img10 = Image.new('RGB', (10, 10), (61, 207, 255))

# corn 11
img11 = Image.new('RGB', (10, 10), (64, 173, 255))

# corn-mintilla 12
img12 = Image.new('RGB', (10, 10), (49, 111, 255))

# corn-notill 13
img13 = Image.new('RGB', (10, 10), (41, 80, 255))

# building grass-trees 14
img14 = Image.new('RGB', (10, 10), (218, 9, 30))

# grass-pastured-mowed 15
img15 = Image.new('RGB', (10, 10), (102, 248, 141))

# oats 16
img16 = Image.new('RGB', (10, 10), (190, 248, 89))
```

```
fig = plt.figure(figsize=(10, 10))
fig.add_subplot(8,2,1)
plt.imshow(img1)
plt.ylabel('Wheat', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,2)
plt.imshow(img2)
plt.ylabel('Wood', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,3)
plt.imshow(img3)
plt.ylabel('Alfalfa', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,4)
plt.imshow(img4)
plt.ylabel('Soybean-clean', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,5)
plt.imshow(img5)
plt.ylabel('Soybean-mintill', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,6)
plt.imshow(img6)
plt.ylabel('Soybean-notill', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,7)
plt.imshow(img7)
plt.ylabel('Hay-windrowed', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])
```

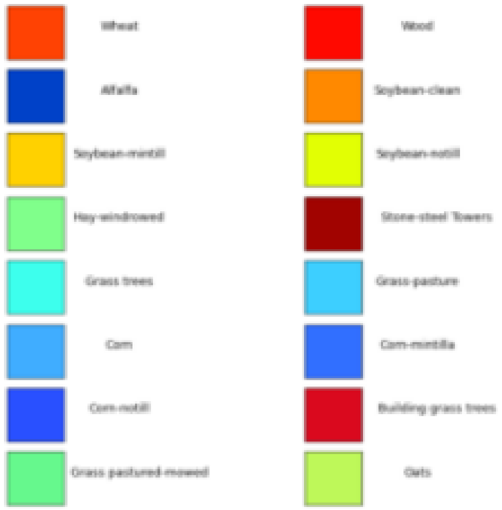
```
fig.add_subplot(8,2,8)
plt.imshow(img8)
plt.ylabel('Stone-steel Towers', labelpad=-135, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,9)
plt.imshow(img9)
plt.ylabel('Grass trees', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,10)
plt.imshow(img10)
plt.ylabel('Grass-pasture', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,11)
plt.imshow(img11)
plt.ylabel('Corn', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,12)
plt.imshow(img12)
plt.ylabel('Corn-mintilla', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,13)
plt.imshow(img13)
plt.ylabel('Corn-notill', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,14)
plt.imshow(img14)
plt.ylabel('Building grass trees', labelpad=-135, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])

fig.add_subplot(8,2,15)
plt.imshow(img15)
plt.ylabel('Grass pastured-mowed', labelpad=-135, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])
```

```
fig.add_subplot(8,2,16)
plt.imshow(img16)
plt.ylabel('Oats', labelpad=-115, fontsize=12,rotation=360)
plt.xticks([])
plt.yticks([])
```

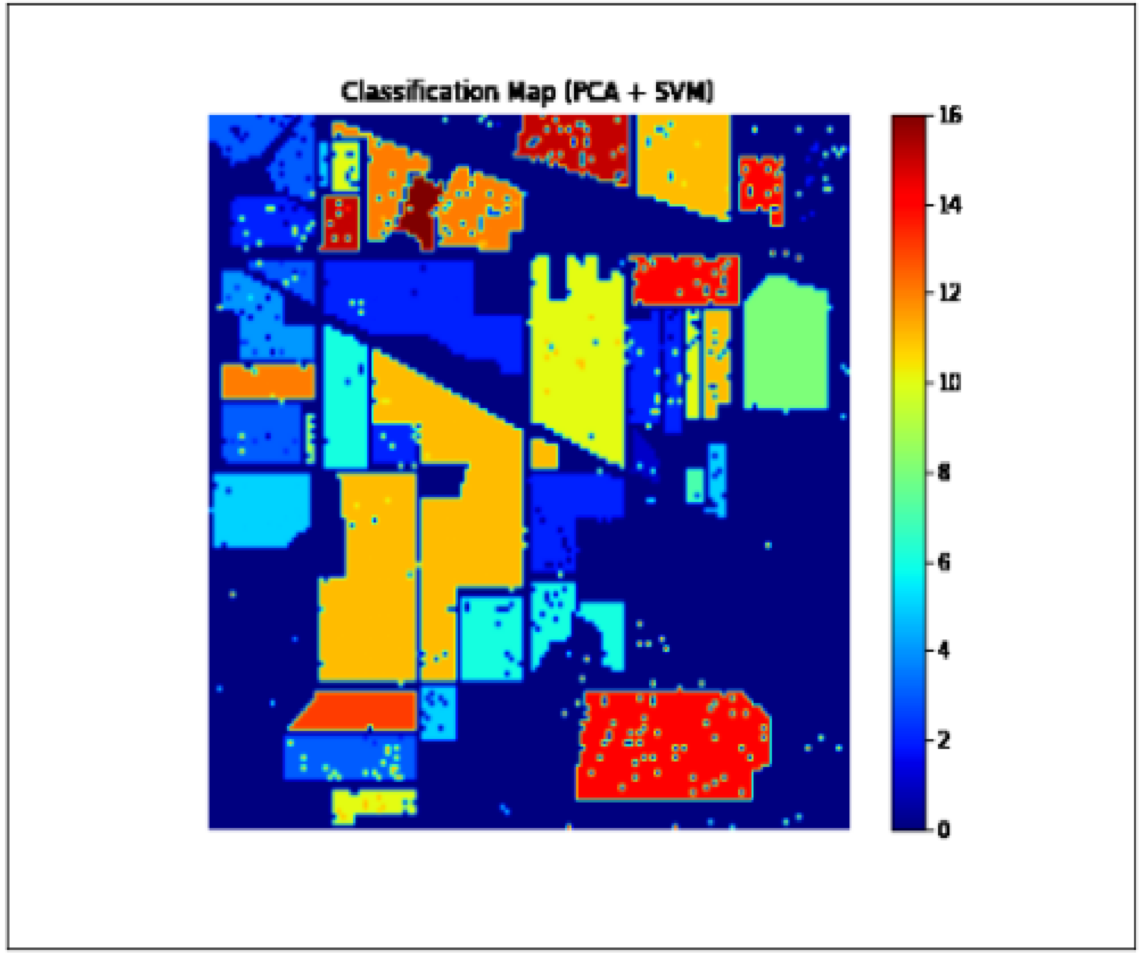
```
([], [])
```

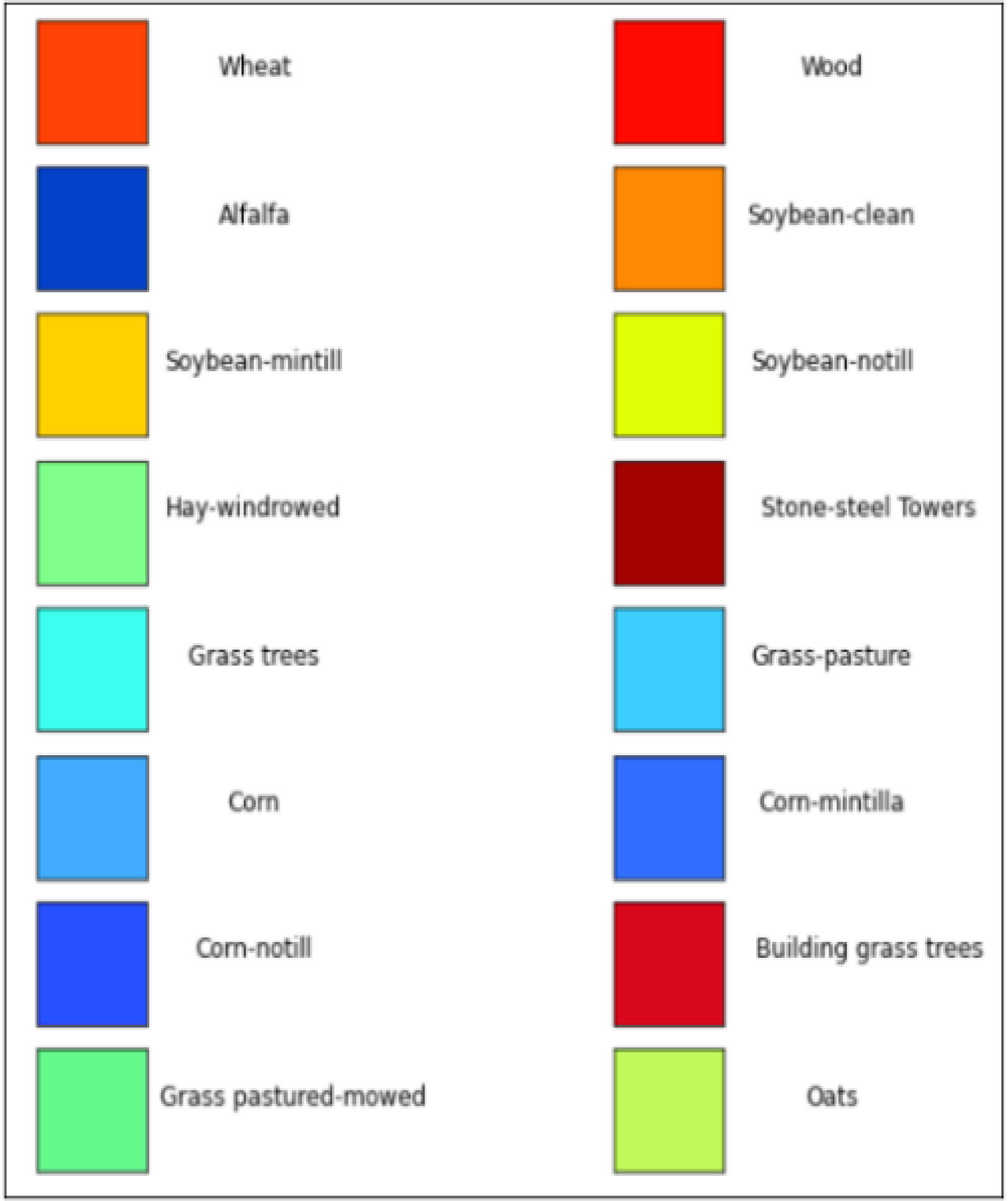
```python
from PIL import Image
im=Image.open('labels.png')
plt.imshow(im)
plt.axis('off')
```

```
(-0.5, 591.5, 634.5, -0.5)
```

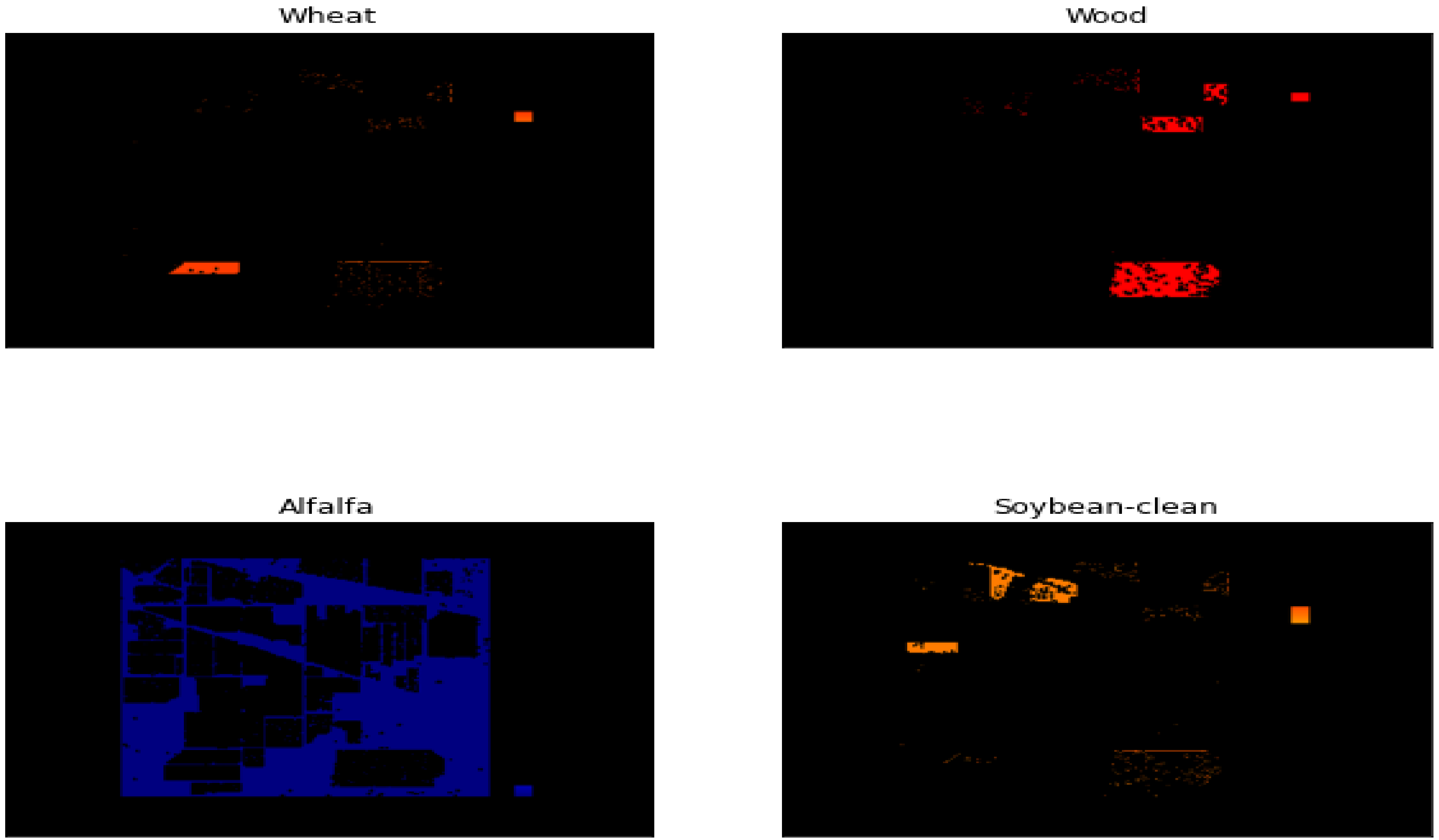



```python
img = cv2.imread("Classification_map.png")

fig = plt.figure(figsize=(20, 20))
fig.add_subplot(1,2,1)
plt.imshow(img)
plt.xticks([])
plt.yticks([])

fig.add_subplot(1,2,2)
plt.imshow(im)
plt.xticks([])
plt.yticks([])
```

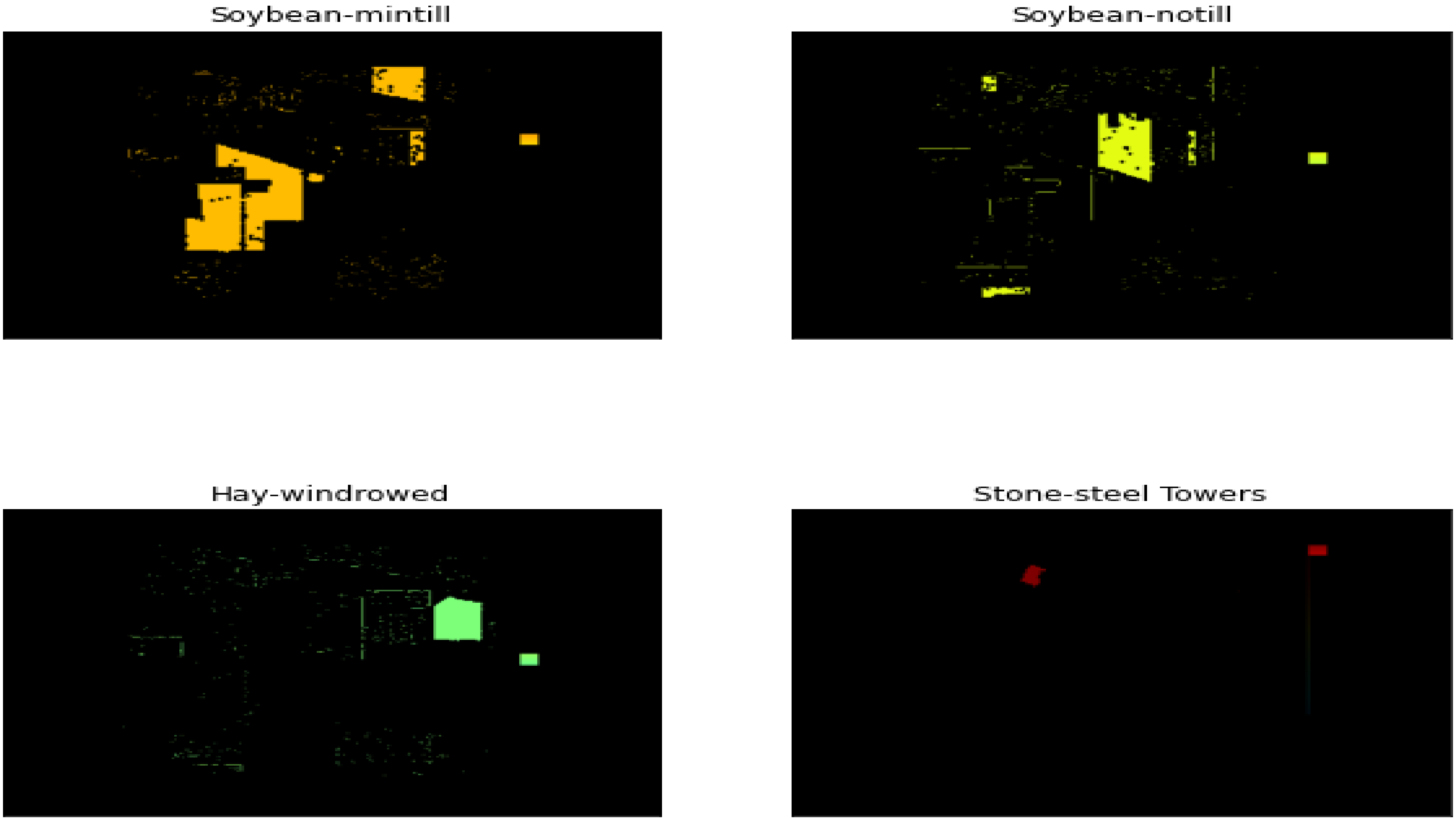

Classification Map (PCA + SVM)
16
14
12
10
8
6
4
2
0


Wheat
Wood
Alfalfa
Soybean-clean
Soybean-mintill
Soybean-notill
Hay-windrowed
Stone-steel Towers
Grass trees
Grass-pasture
Corn
Corn-mintilla
Corn-notill
Building grass trees
Grass pastured-mowed
Oats

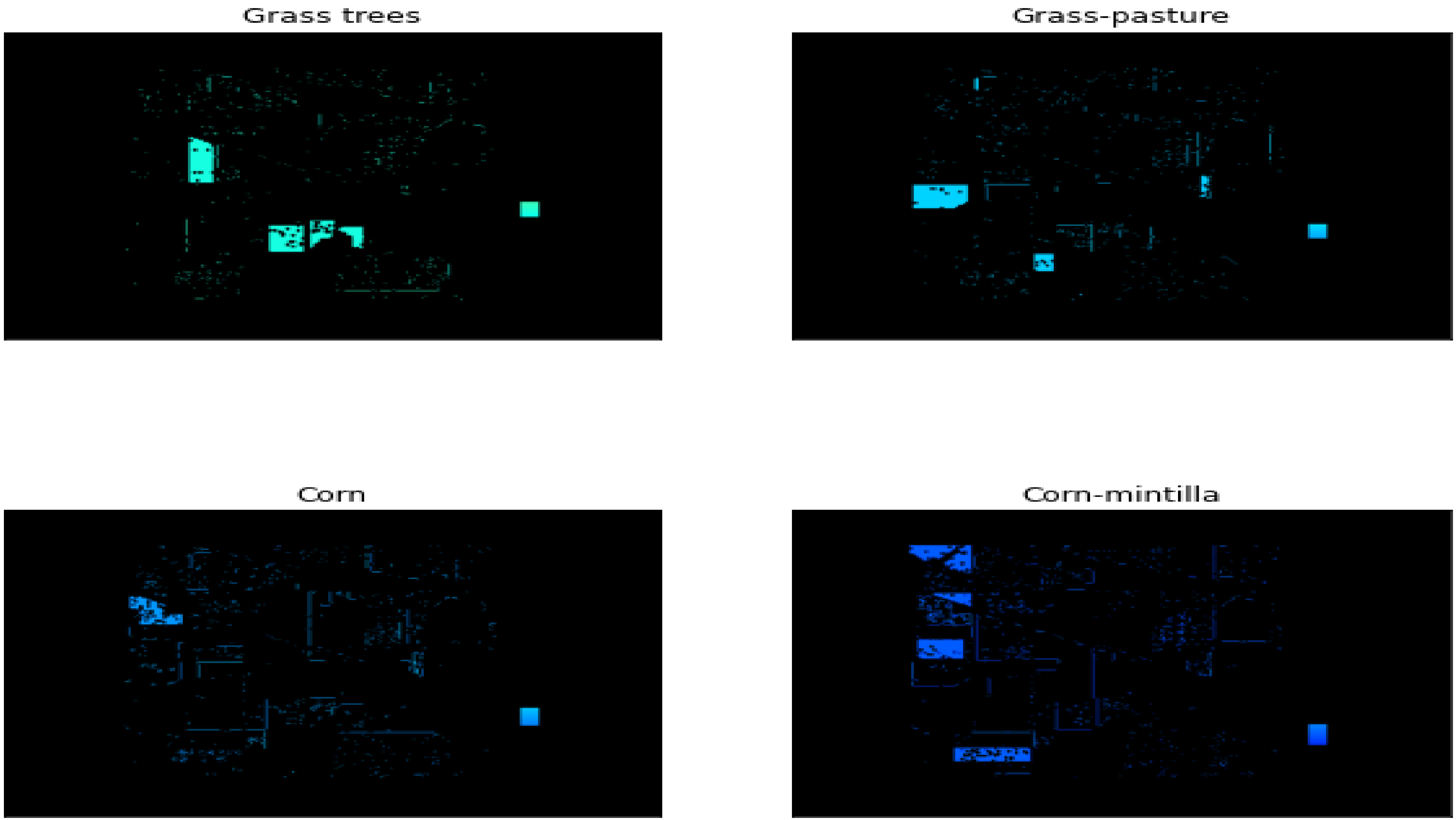
```
fig = plt.figure(figsize=(10, 10))
fig.add_subplot(2,2,1)
plt.imshow(result1)
plt.title('Wheat')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,2)
plt.imshow(result2)
plt.title('Wood')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,3)
plt.imshow(result3)
plt.title('Alfalfa')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,4)
plt.imshow(result4)
plt.title('Soybean-clean')
plt.xticks([])
plt.yticks([])
```


Wheat
Wood
Alfalfa
Soybean-clean

```
fig = plt.figure(figsize=(10, 10))
fig.add_subplot(2,2,1)
plt.imshow(result5)
plt.title('Soybean-mintill')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,2)
plt.imshow(result6)
plt.title('Soybean-notill')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,3)
plt.imshow(result7)
plt.title('Hay-windrowed')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,4)
plt.imshow(result8)
plt.title('Stone-steel Towers')
plt.xticks([])
plt.yticks([])
```

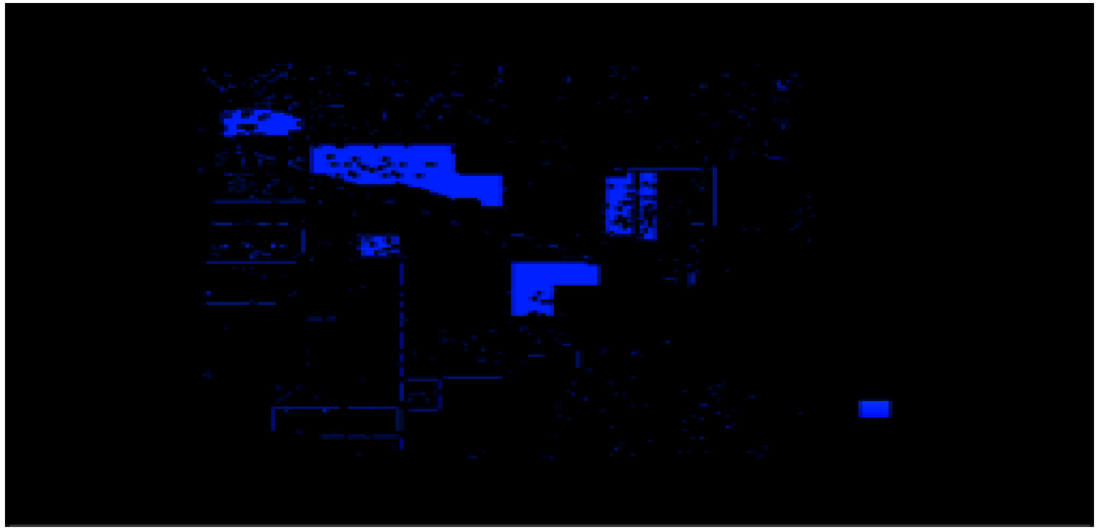

Soybean-mintill
Soybean-notill
Hay-windrowed
Stone-steel Towers

```
fig = plt.figure(figsize=(10, 10))
fig.add_subplot(2,2,1)
plt.imshow(result9)
plt.title('Grass trees')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,2)
plt.imshow(result10)
plt.title('Grass-pasture')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,3)
plt.imshow(result11)
plt.title('Corn')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,4)
plt.imshow(result12)
plt.title('Corn-mintilla')
plt.xticks([])
plt.yticks([])
```

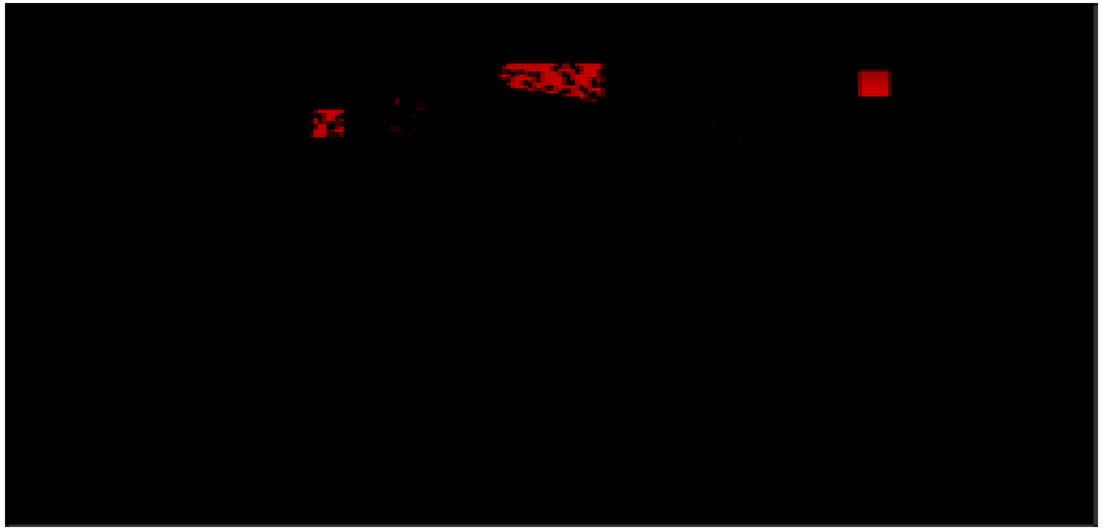

Grass trees
Grass-pasture
Corn
Corn-mintilla

```
fig = plt.figure(figsize=(10, 10))
fig.add_subplot(2,2,1)
plt.imshow(result13)
plt.title('Corn-notill')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,2)
plt.imshow(result14)
plt.title('Building grass trees')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,3)
plt.imshow(result15)
plt.title('Grass pastured-mowed')
plt.xticks([])
plt.yticks([])

fig.add_subplot(2,2,4)
plt.imshow(result16)
plt.title('Oats')
plt.xticks([])
plt.yticks([])
```

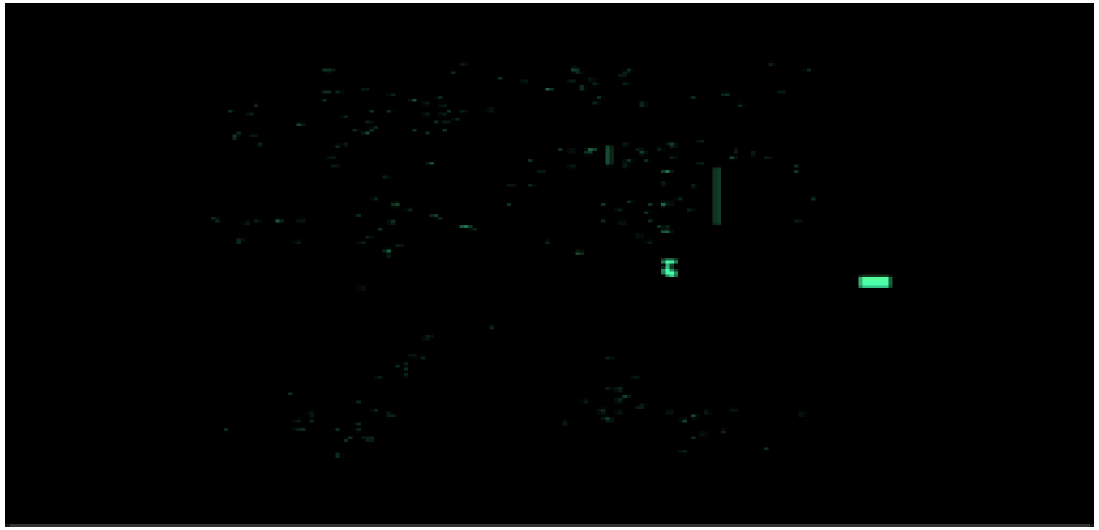
Corn-notill



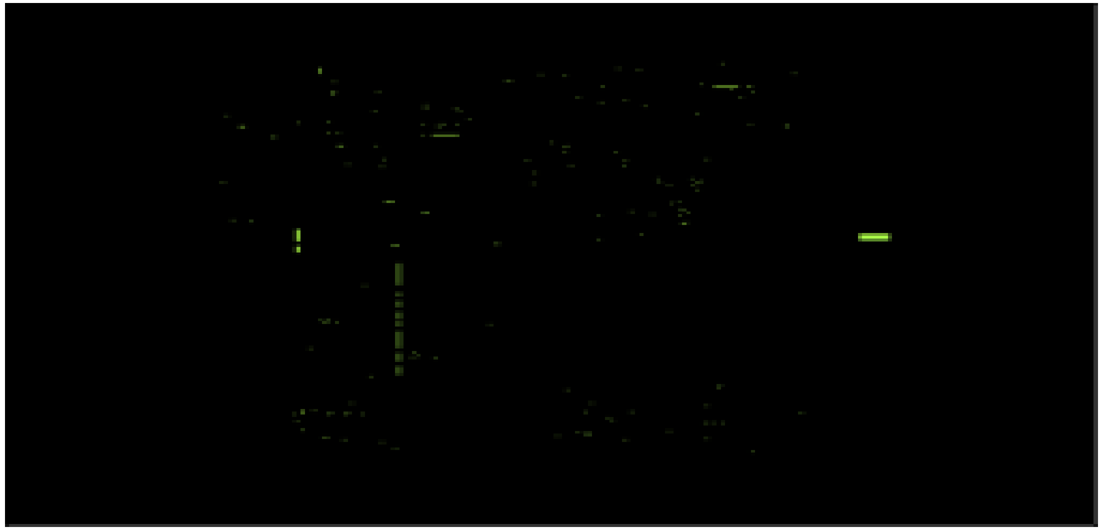
Building grass trees



Grass pastured-mowed



Oats



PERFORMANCE PARAMETERS

Compression Ratio: Ratio of the size of the original image to the compressed image.

$$\mathbf{CR} = \frac{\boldsymbol{Original\ Size}}{\boldsymbol{Compressed\ Size}} = 324.0359$$

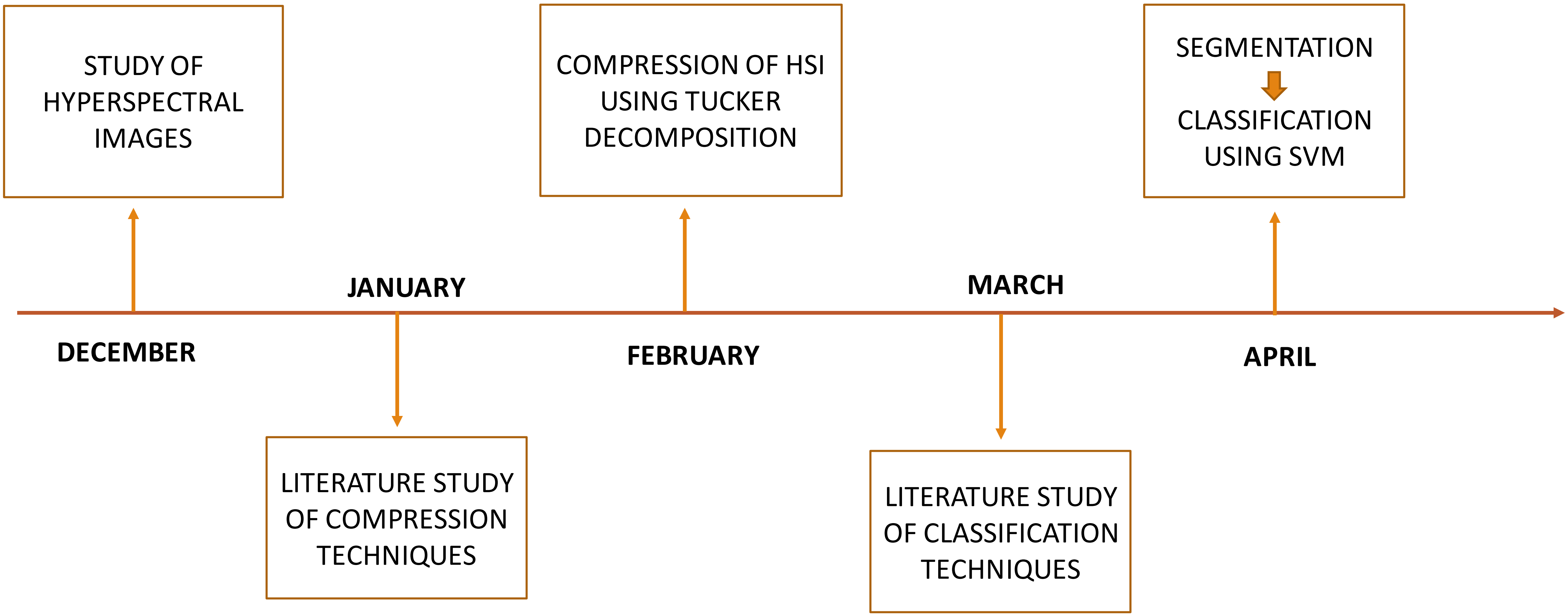
Elapsed Time: It is the amount of time that passes from the start of an event to its finish or it is the duration from when the process was started until the time it terminated.

Elapsed time= 0.45227

Accuracy: The accuracy calculation (AC) is used to compare the efficiency of the system. It is a measure of closeness between the predicted value and the obtained value.

Accuracy = $\frac{T\mathrm{P}+TN}{TP+FP+FN+TN} = 73.33\%$

TIMELINE



STUDY OF HYPERSPECTRAL IMAGES
DECEMBER
JANUARY
LITERATURE STUDY OF COMPRESSION TECHNIQUES
COMPRESSION OF HSI USING TUCKER DECOMPOSITION
FEBRUARY
MARCH
LITERATURE STUDY OF CLASSIFICATION TECHNIQUES
SEGMENTATION
CLASSIFICATION USING SVM
APRIL

CONCLUSION

It can be deduced that Tensor decomposition can reduce the spatial and spectral domains simultaneously, at lesser elapsed time. Hence Tensor Decomposition was used to compress the hyperspectral image. Classification has also been successfully done using the supervised learning algorithm: SVM and all the features of Indian Pines hyperspectral image were extracted.

REFERENCES

1. Compression of Hyperspectral Images Using Discerete Wavelet Transform and Tucker Decomposition Azam Karami, Student Member, IEEE, Mehran Yazdi, Member, IEEE, and Grégoire Mercier, Senior Member, IEEE

2. On the Impact of Lossy Compression on Hyperspectral Image Classification and Unmixing Fernando García-Vílchez, Jordi Muñoz-Marí, Maciel Zortea, Ian Blanes, Student Member, IEEE, Vicente González-Ruiz, Gustavo Camps-Valls, Senior Member, IEEE, Antonio Plaza, Senior Member, IEEE, and Joan Serra-Sagristà, Member, IEEE

3. Hyperspectral Image Compression Using JPEG2000 and Principal Component Analysis Qian Du, Senior Member, IEEE, and James E. Fowler, Senior Member, IEEE

4. Least Square based fast denoising approach to Hyperspectral Imagery Srivatsa S, V Sowmya, and Soman K P

5. Lossless to Lossy Dual-Tree BEZW Compression for Hyperspectral Images Kai-jen Cheng, Student Member, IEEE, and Jeffrey Dill, Member, IEEE

6. Hyperspectral Data Compression Tradeoff Emmanuel Christophe

7. X. Tang and W. A. Pearlman, “Three-dimensional wavelet-based compression of hyperspectralimages,” in Hyperspectral Data Compression, G. Motta, F. Rizzo, and A. Storer, Eds. : Springer,2006.

8. Q. Du and J. E. Fowler, “Hyperspectral image compression using JPEG2000 and principalcomponent analysis,” IEEE Geosci. RemoteSens. Lett. , vol. 4, no. 2, Apr. 2007

8. J.Wangand C.Chang,“Independent component analysis-based dimensionality reduction withapplications in hyperspectral image analysis,” IEEE Trans. Geosci. Remote Sens , vol. 44, no. 6,pp. 1586–1600, Jun.2006.

9. Azam Karami, Mehran Yazdi, Gregoire Mercier, “Compression of Hyperspectral Images Using Discrete Wavelet Transform and Tucker Decomposition”, IEEE Journal, vol. 5, no. 2, April 2012, pp. 444-450.

10. Behcet Ugur Toreyin, Ozan Yilmaz, Yakup Murat Mert, Fethi Turk, “Lossless Hyperspectral Image Compression Using Wavelet Transform Based Spectral Decorrelation”, IEEE, 2015 pp.251-254.

11. R. E. Chaudhari, S. B. Dhok, “Wavelet Transformed based Fast Fractal Image Compression”,IEEE, 2014, pp.65-69.

12. J. Serra-Sagristà and F. Aulí-Llinàs, “Remote sensing data compression,” in Computational Intelligence for Remote Sensing. Berlin,Germany: Springer-Verlag, Jun.2008, pp. 27–61.

13. J. Robinson and V. Kecman, “Combining support vector machine learning with the discrete cosine transform in image compression,” IEEE Trans. Neural Networks , vol. 14, no. 4, Jul. 2003.

14. W. Kim and C.-C. Li, “A study on core conditioning multiwavelet systems for image compression,”in Wavelet Analysis and Its Application ,Y.Y.Tang,P.C.Yuen,NewYork,2001

15. A. Cohen, I. Daubechies, and J.-C. Feauveau, “Biorthogonal bases of compactly supported wavelets,” Commun. Pure and Appl. Math., vol.45, no. 5, May 1992.

THANK YOU