

# **GNR652 : COURSE PROJECT**

Anwesh Mohanty : 170070009

Koustav Jana : 170070051

Rohan Bansal : 170070058

# **IMAGE COMPRESSION USING K-MEANS CLUSTERING**

# INTRODUCTION

- K-means algorithm is used to cluster  $n$  objects based on attributes into  $k$  partitions where  $k < n$ .
- The grouping is done by minimising the sum of squares of distances between data and the corresponding cluster centroid.

# PROBLEM STATEMENT

We are given a JPEG/png image. Our aim is to compress the image still retrieving majority of the attributes. We also try to prove our claim by running the compressed and the original images through an image classification program and observe the results. The image classification program is implemented using neural networks from the keras python library.

# IMPLEMENTATION

**Compute centroids :** Returns the new centroids by computing the means of the data points assigned to each centroid.

```
def computeCentroids(X,idx,k):  
    m=X.shape[0]  
    n=X.shape[1]  
    centroids=np.zeros((k,n))  
    itr=np.zeros((k,1))  
  
    for i in range(0,len(idx)):  
        l=int(idx[i,0])  
        centroids[l,:]=centroids[l,:]+X[i,:]   
        itr[l]=itr[l]+1;  
    centroids=np.divide(centroids,itr)  
    return centroids;
```

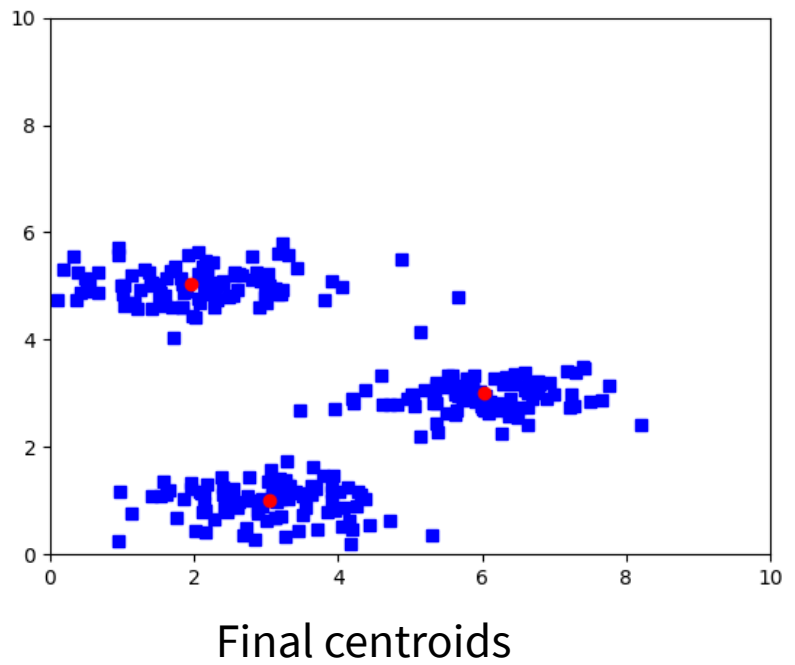
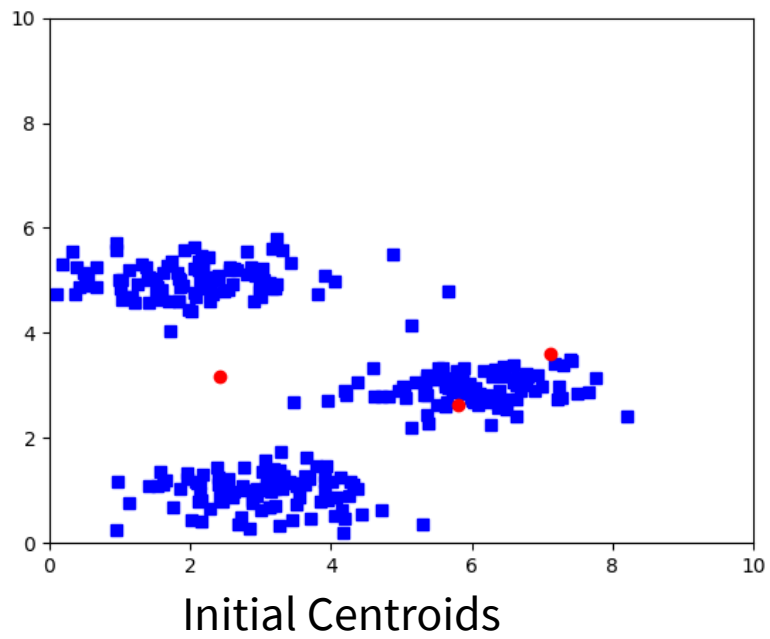
**Find closest centroid :**  
Assigns the centroid memberships to each datapoint.

```
def findClosestCentroids(X,centroids):  
    K=centroids.shape[0]  
    idx=np.zeros((X.shape[0],1))  
    for i in range(0,X.shape[0]):  
        min=1000000000  
        for j in range(0,K):  
            if(pow(LA.norm(X[i,:]-centroids[j,:]),2)<min):  
                min=pow(LA.norm(X[i,:]-centroids[j,:]),2);  
                idx[i,0]=j;  
  
    return idx;
```

# IMPLEMENTATION(cont.)

**Run K-means** : Implements the previous two functions for a fixed number of iterations to find the final centroids.

```
def runKMeans(X, initial_centroids, max_iters):  
    m=X.shape[0]  
    n=X.shape[1]  
    K=initial_centroids.shape[0]  
    centroids=initial_centroids  
    previous_centroids=centroids  
    idx=np.zeros((m,1))  
  
    for i in range(1,max_iters):  
        idx=findClosestCentroids(X,centroids);  
        centroids=computeCentroids(X,idx,K)  
        print("Iteration number:",i);  
  
    return centroids,idx;
```



\*plotted using matplotlib library

# RESULTS



Original Image(250.1Kb)



Compressed Image(73.6Kb)



Original Image(173Kb)



Compressed Image(53.1Kb)



## RESULTS(cont.)



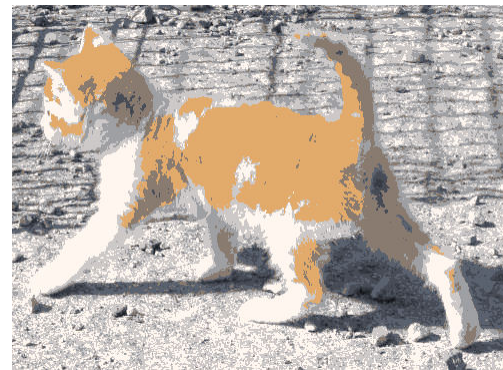
Original Image(117.7Kb)



Compressed Image(23.8Kb)



Original Image(299.2Kb)

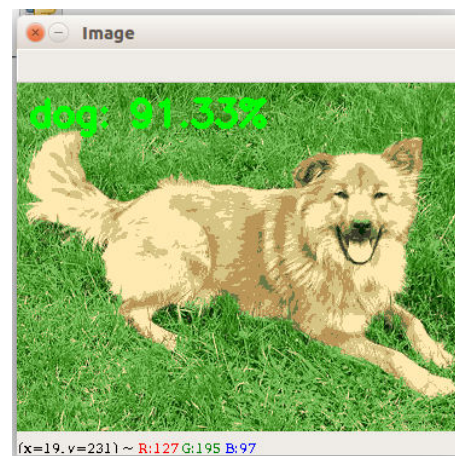


Compressed Image(62.8Kb)

# IMAGE CLASSIFICATION RESULTS



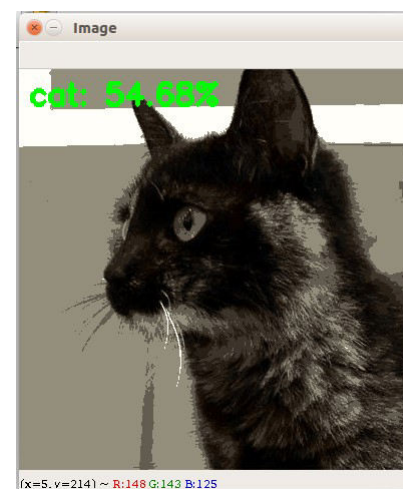
Original Image



Compressed Image



Original Image



Compressed Image

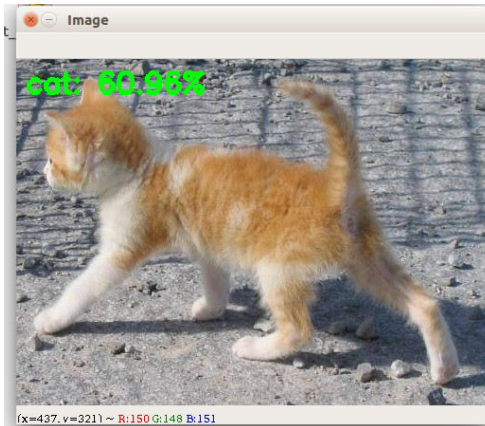
# IMAGE CLASSIFICATION RESULTS(cont.)



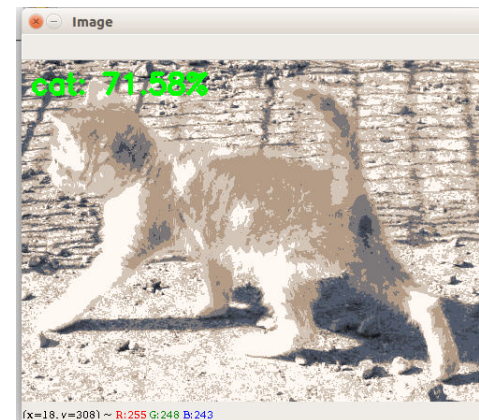
Original Image



Compressed Image



Original Image



Compressed Image

# Conclusions

- The sizes of the images have decreased significantly. For some images the decrease is upto 5 times.
- When passed through the image classifier, we observe that there isn't significant difference between the classifying accuracy of the original and compressed images.
- This proves that the compression has retrieved majority of the attributes or properties of the image, while also significantly reducing the image size.

# References

- <https://www.pyimagesearch.com/2016/09/26/a-simple-neural-network-with-python-and-keras/>
- <https://www.geeksforgeeks.org/python-numpy/>
- [https://matplotlib.org/users/pyplot\\_tutorial.html](https://matplotlib.org/users/pyplot_tutorial.html)
- <https://www.guru99.com/scipy-tutorial.html>

THANK YOU!

# CONTRIBUTIONS

- **Anwesh Mohanty : Implementing the functions of K-Means Clustering**
- **Rohan Bansal : Exploring the data and results. Making the presentation.**
- **Koustav Jana : Image Classification**