Assignment05

April 11, 2019

Assignment05: K-means algorithm on color image
Software Engineering
20154652 Lee Dong Jae

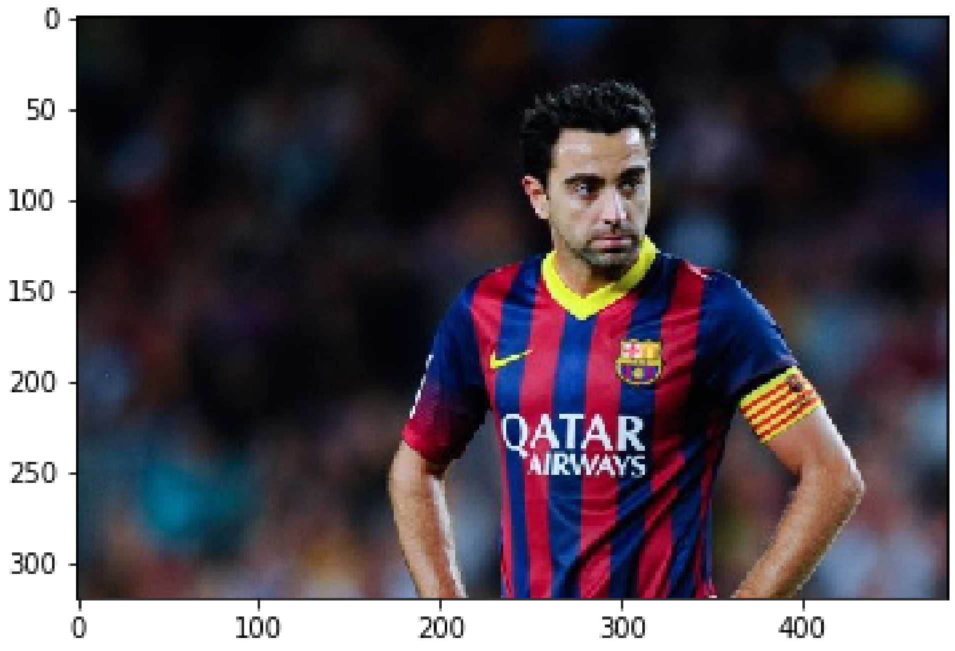
```
In [1]: import PIL.Image as pilimg
        import numpy as np
        import matplotlib.pyplot as plt
```

```
In [2]: # Read image
        im = pilimg.open('xavi.jpg')
```

```
In [3]: #change the image to numpy array
        data = np.array(im)
```

```
In [4]: plt.imshow(data)
```

```
Out[4]: <matplotlib.image.AxesImage at 0x2e4f137d8d0>
```



```
In [5]: hor = list(data.shape)[0]
        ver = list(data.shape)[1]

In [6]: def distance(x, y):

            d = (x - y) ** 2
            #s = np.sum(d)
            #r = np.sqrt(s)

            return(d)

In [7]: def initialize_label(k):

            #Assign a dictionary that contain energy
            info = {'energy_list': []}

            #initialize label randomly
            first_label = np.random.randint(k, size = (hor, ver))

            clusters_hor = {idx: [] for idx in range(k)}
            clusters_ver = {idx: [] for idx in range(k)}
            new_clusters = {idx: [] for idx in range(k)}

            #append corresponding [hor, ver] of image to cluster
            for i in range(k):
                clusters_hor[i], clusters_ver[i] = np.where(first_label == i)
                for j in range(len(clusters_hor[i])):
                    new_clusters[i].append([clusters_hor[i][j], clusters_ver[i][j]])

            make_new_centroid(new_clusters, k, info)

In [8]: def energy_function(centroid, clusters, k):
            energy = 0

            #get energy
            for m in range(k):
                for n in clusters[m]:
                    for u in range(3):
                        energy += distance(data[n[0]][n[1]][u], centroid[m][u])

            return energy / (hor*ver)

In [9]: def make_new_centroid(clusters, k, info, centroid = 0):

            new_centroid = np.zeros((k,3))
            #sum previous data contained in same cluster
            for i in range(k):
                for j in clusters[i]:
                    new_centroid[i][0] += data[j[0]][j[1]][0]
                    new_centroid[i][1] += data[j[0]][j[1]][1]
                    new_centroid[i][2] += data[j[0]][j[1]][2]

            for m in range(k):
                new_centroid[m][0] = new_centroid[m][0] / len(clusters[m])
                new_centroid[m][1] = new_centroid[m][1] / len(clusters[m])
                new_centroid[m][2] = new_centroid[m][2] / len(clusters[m])

            #if clustering does not change over, plot images and information
            if np.array_equal(centroid, new_centroid):
                print('end')
                plot_image(centroid, clusters, k)
                plot_charts(info)

            else:
                do_clustering(new_centroid, k, info)
```

```
In [10]: def do_clustering(centroid, k, info):

             #make a place for put indexes of data to each clusters
             clusters = {idx: [] for idx in range(0, k)}


             #a temporary array for keeping the distance
             temp_distance = np.zeros(k)

             for i in range(hor):
                 for j in range(ver):
                     for t in range(k):
                         for u in range(3):
                             temp_distance[t] += distance(data[i][j][u], centroid[t][u])
                     #find the argmin of distance. And append a idx of data to the cluster[arg
                     clusters[np.argmin(temp_distance)].append([i, j])
                     temp_distance = np.zeros(k)



             #calcuate energy, training accuracy, test accuracy at each time
             energy2 = energy_function(centroid, clusters, k)

             print(energy2)

             #append calculated information to each list
```

```
            info['energy_list'].append(energy2)

            make_new_centroid(clusters, k, info, centroid)
```

```
In [11]: def plot_charts(info):

             plt.figure(figsize=(10, 8))
             plt.title("Energy")
             plt.plot(range(len(info['energy_list'])), info['energy_list'])
             plt.show()
```

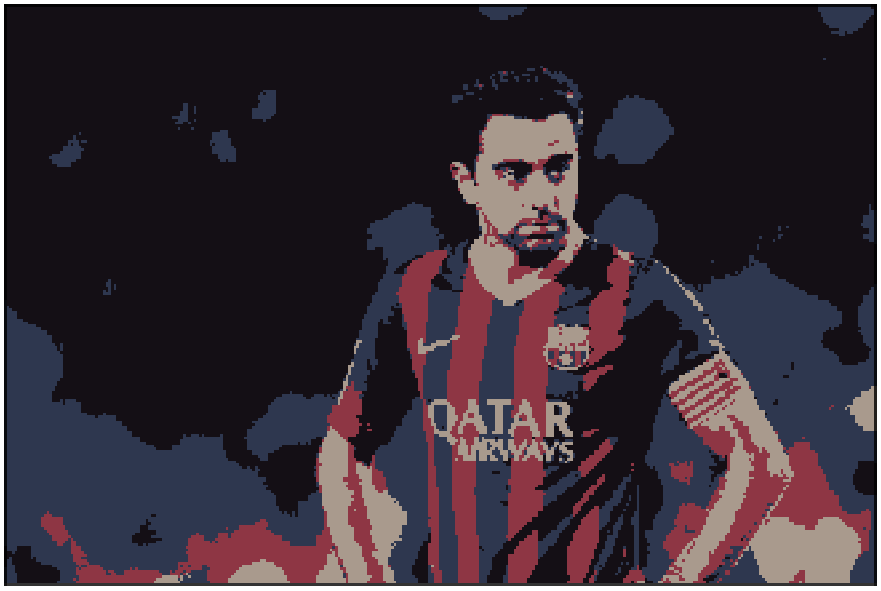
```
In [12]: def plot_image(centroid, clusters, k):
             plt.figure(1)
             new_image = np.zeros((hor, ver, 3), dtype = np.uint8)
             for i in range(k):
                 for j in clusters[i]:
                     for u in range(3):
                         new_image[j[0]][j[1]][u] = centroid[i][u]
             plt.title("new image")
             plt.imshow(new_image)
             frame = plt.gca()
             frame.axes.get_xaxis().set_visible(False)
             frame.axes.get_yaxis().set_visible(False)
             plt.show()
```

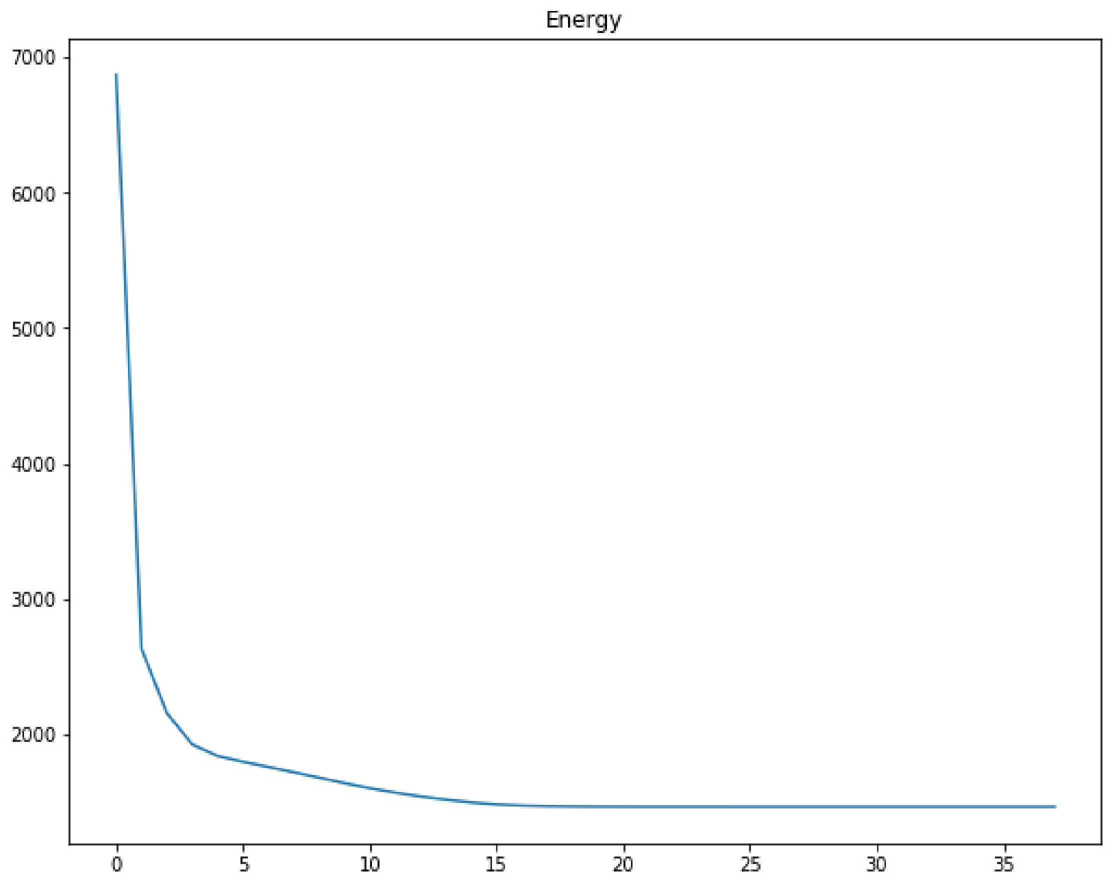
```
In [14]: initialize_label(4)
```

```
6869.347145829909
2632.3436863733496
2156.7014151616113
1925.5632320079064
1839.7810285356727
1796.7839507007075
1759.2190443437503
1719.933738742148
1679.1925231701755
1639.4560519961944
1602.5394862145065
1569.5916946911132
1540.7853650136726
1517.319957454133
1496.9830975237378
1482.2068080044485
1473.7848105219116
1469.6899814944143
1467.6212540682548
1466.5799718565652
1466.1350378451189
1465.9126944474576
```

```
1465.8100874181919
1465.7520904499872
1465.725012997234
1465.7129018934477
1465.7069835486
1465.7033975805637
1465.700760211445
1465.6987400011617
1465.6964498050959
1465.6947154969462
1465.6935003966457
1465.6928674434214
1465.6923717896977
1465.6921019325973
1465.692032569517
1465.6920228778195
end
```

new image

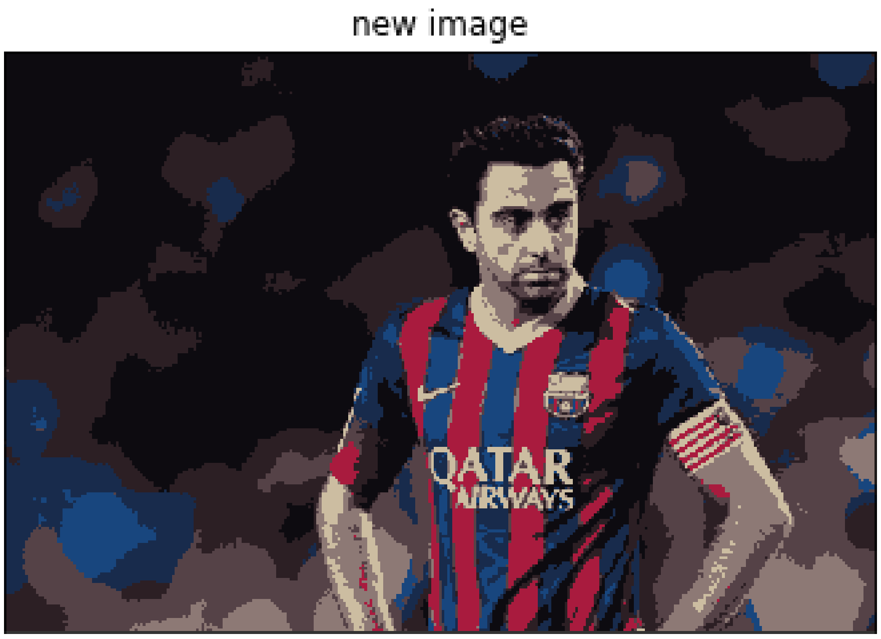


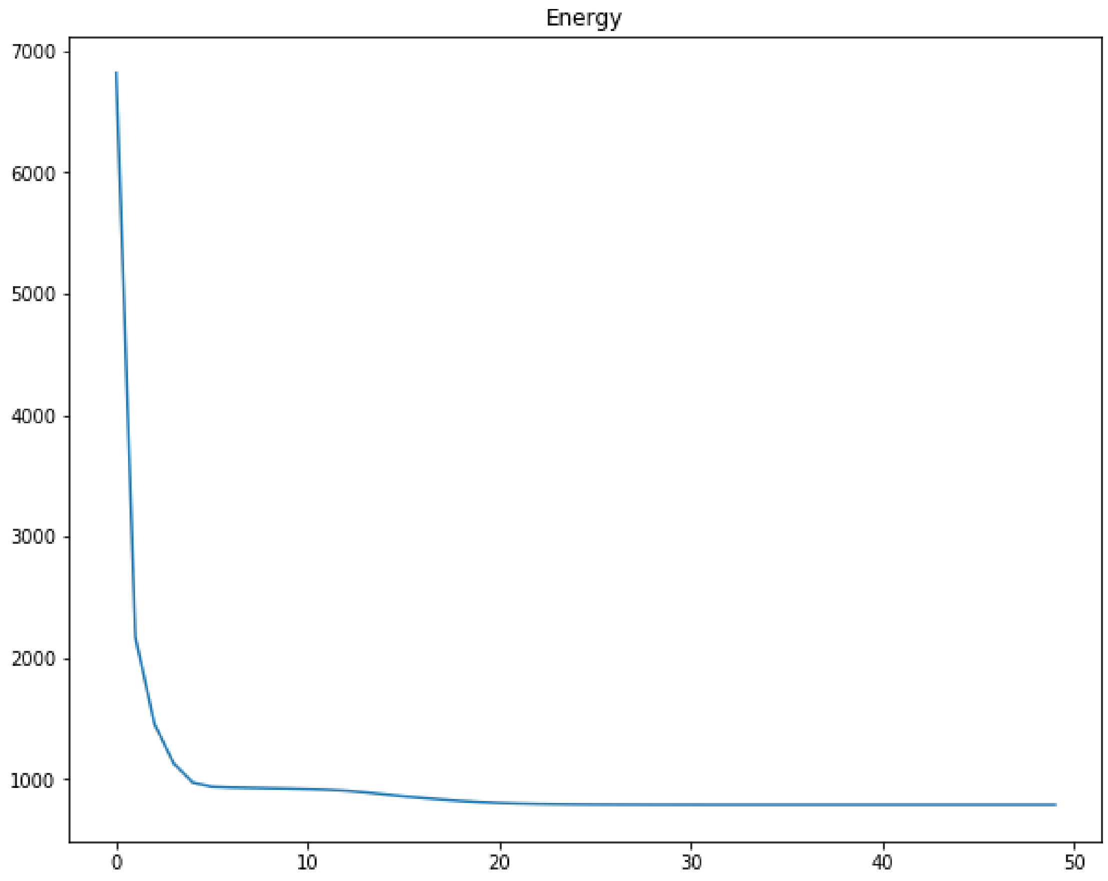


```
In [13]: initialize_label(8)
```

```
6818.508878821344
2171.027567719056
1454.2400775765177
1130.550739323829
972.3903000420127
938.45151379406
932.3785488116956
928.8938866107779
925.7166721369014
922.4704766763533
918.5866222927281
913.2517584393966
904.9931546206287
891.4203780201897
874.7268051882824
858.9224305464621
845.8096972477477
832.6247578496118
```

```
820.8891827066001
811.2240896245225
804.3900664849914
799.871961306284
796.7536431685215
794.6823822846449
793.2482915504431
792.2810672625548
791.6823492873867
791.3208123458546
791.0694567178322
790.874811658933
790.7312856303241
790.6417142808629
790.5782885424502
790.5354933256189
790.5028868603781
790.4795823557183
790.4656119384291
790.4513172593504
790.4438202903559
790.4390301198158
790.4354176139917
790.4324901327739
790.4310848206418
790.4298358856993
790.4285969657419
790.427908896846
790.4276833213734
790.4272819431948
790.4270748425315
790.4270726179872
end
```


new image
QATAR
AIRWAYS

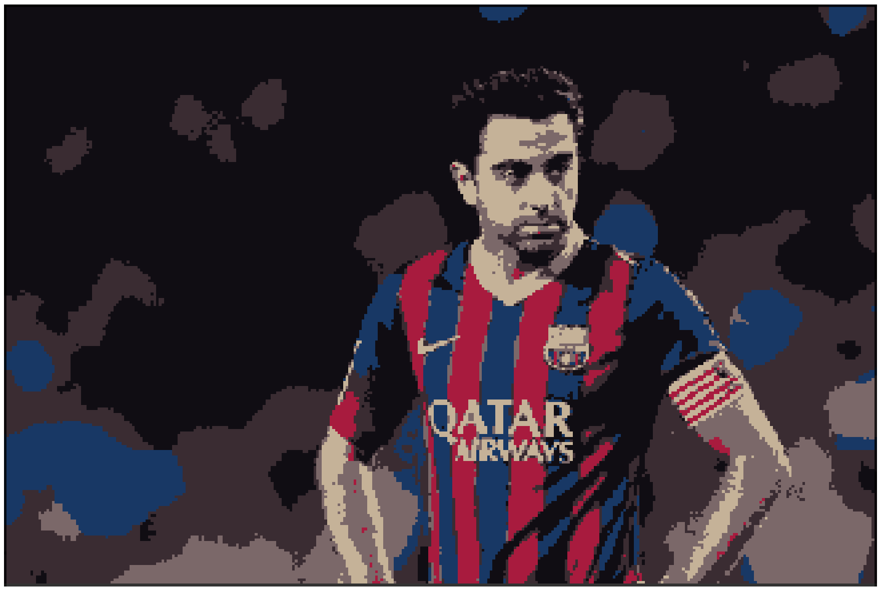

Energy
7000
6000
5000
4000
3000
2000
1000
0
10
20
30
40
50

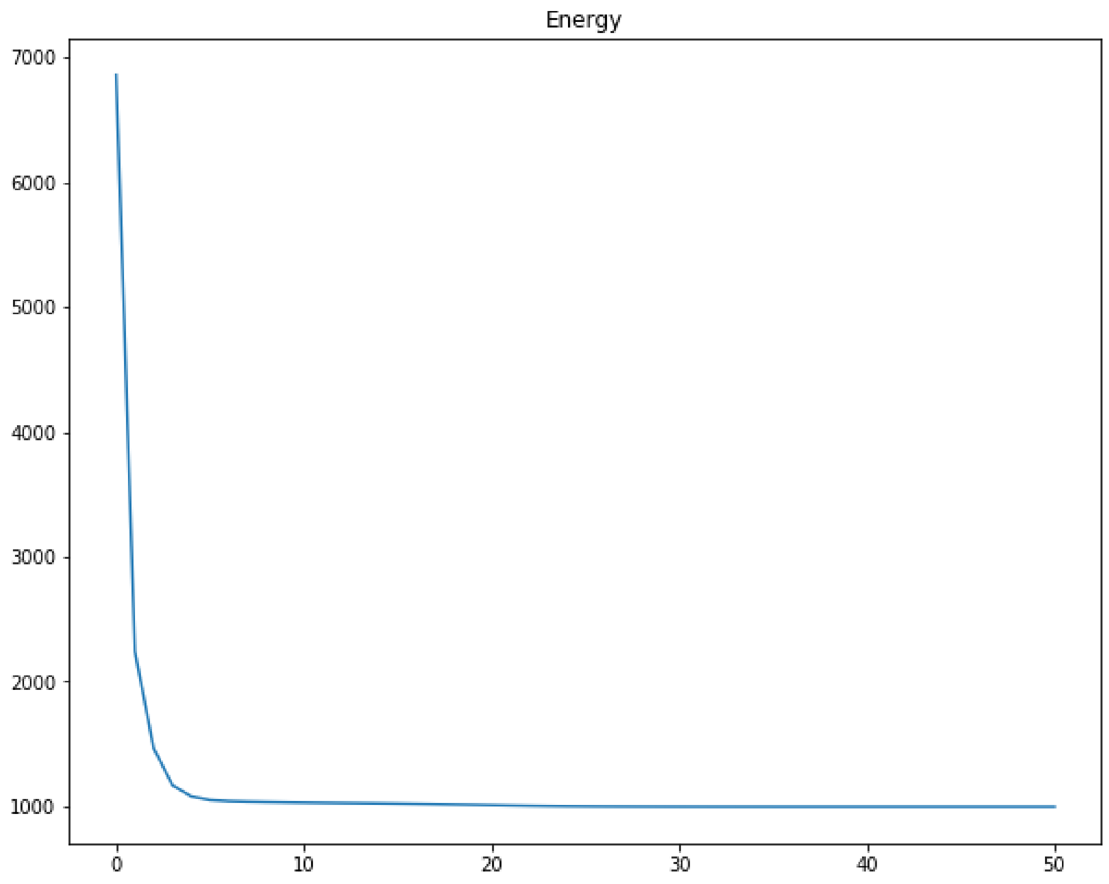
```
In [15]: initialize_label(6)
```

```
6857.188169556307
2241.5153072699695
1466.2006817547642
1171.128211294202
1080.8937352663406
1051.9189897727645
1042.648311613631
1038.6413229186264
1036.0300174900296
1033.7600275764635
1031.668468741558
1029.7496385258737
1028.0382873636095
1026.4445388931485
1024.7702863309457
1022.8565383912006
1020.7348261064482
1018.3726820355429
1015.8728990733243
1013.5275124236996
1010.8927955253913
1008.1934194449321
1005.8823311896048
1004.0076919655661
1002.6109542282444
1001.5463874233889
1000.7588502420014
1000.199191736293
999.8190065548052
999.5292557455305
999.289455619304
999.1383262001704
999.0038728430345
998.9231097003111
998.878319162669
998.850165533962
998.8354117048776
998.8253861233113
998.8191733525517
998.8156058644868
998.8127265083461
998.8108415198955
998.8099656368821
```

```
998.8080439148315
998.8069651178537
998.806309412233
998.8054797458691
998.8051576221903
998.8050158380197
998.8048434926222
998.8048428401086
end
```

new image



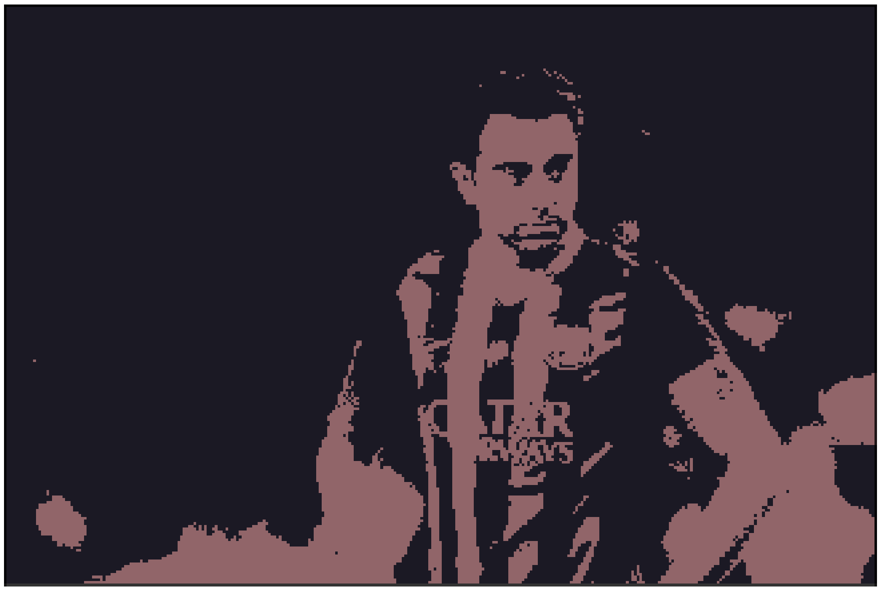


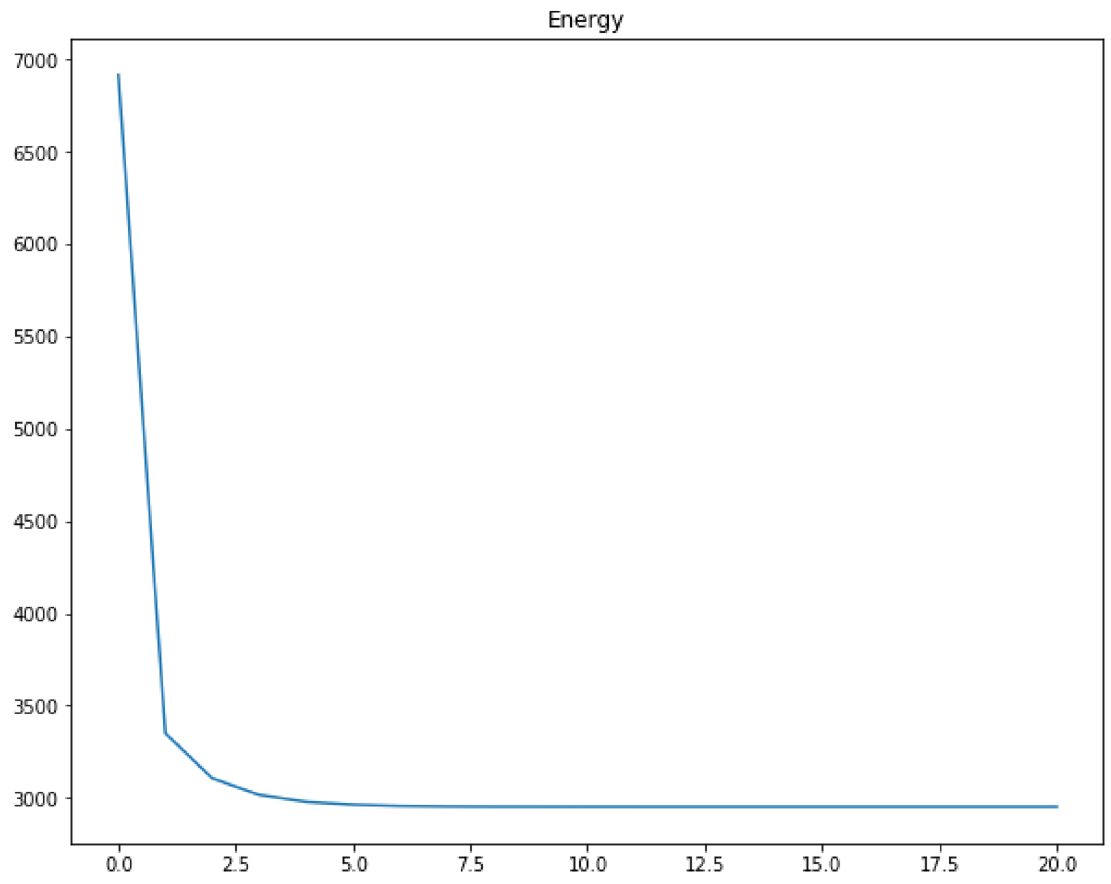
```
In [13]: initialize_label(2)
```

```
6916.552024507432
3350.1244937035394
3105.9615917426827
3015.78655090859
2978.326949881929
2962.636000170432
2955.9351202025746
2953.392506901132
2952.505476838586
2952.2497508718393
2952.152741001759
2952.116256445785
2952.099447548904
2952.092607404813
2952.0911018739553
2952.0908195228526
2952.090721736558
2952.0906863018017
```

```
2952.090656784123
2952.090652824963
2952.090651103949
end
```

new image




Energy
7000
6500
6000
5500
5000
4500
4000
3500
3000
0.0
2.5
5.0
7.5
10.0
12.5
15.0
17.5
20.0

```
In [ ]:
```