PCA for Color image compression

September 20, 2022

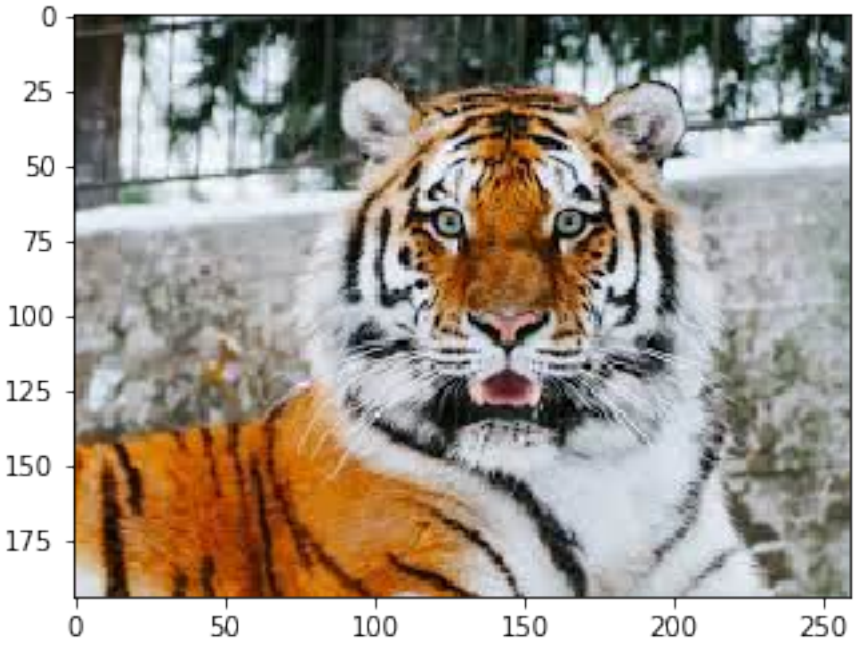
[ ]:
```
Normally images have a lot of pixels to retain their clarity, but that␣
 ↪significantly increases its size and slows down
the performance of the system when it has to process multiple images.  To␣
 ↪overcome this situation we can use
the dimensionality reduction technique which comes under Unsupervised Machine␣
 ↪Learning. We will use one picture and
reduce its dimensions or in other words compress the image using PCA in python.

Algorithm:
i) Split the image into the three channels (Blue, Green, and Red) first
ii) Perform PCA separately on each dataset representing each channel
iii) Merge them to reconstruct the compressed image.
```

[41]:
```
#Load and pre-process the image

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
import cv2
from scipy.stats import stats
import matplotlib.image as mpimg
```

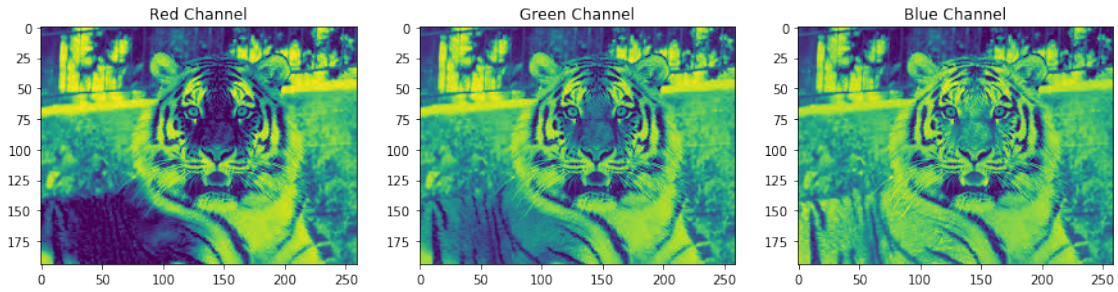
[42]:
```
img = cv2.cvtColor(cv2.imread('tiger.jfif'), cv2.COLOR_BGR2RGB)
plt.imshow(img)
plt.show()
```



[44]:
```
img.shape
```

[44]:
```
(194, 259, 3)
```

[45]:
```
#Splitting into channels
blue,green,red = cv2.split(img)
# Plotting the images
fig = plt.figure(figsize = (15, 7.2))
fig.add_subplot(131)
plt.title("Red Channel")
plt.imshow(red)
fig.add_subplot(132)
plt.title("Green Channel")
plt.imshow(green)
fig.add_subplot(133)
plt.title("Blue Channel")
plt.imshow(blue)
plt.show()
```



[46]:
```
blue_temp_df = pd.DataFrame(data = blue)
blue_temp_df
```

[46]:

	0	1	2	3	4	5	6	7	8	9	…	249	250	251	\
0	40	52	55	45	36	36	34	28	27	33	…	13	8	2	
1	48	60	61	49	38	37	33	26	27	33	…	16	12	3	
2	44	55	55	42	29	27	22	15	31	35	…	18	12	5	
3	43	54	56	45	33	33	29	22	29	33	…	12	0	6	
4	40	53	57	49	41	44	42	37	43	47	…	4	0	8	
..	…	…	…	…	…	…	…	…	…	…	…	…	…	…	
189	230	228	225	220	217	216	217	221	228	207	…	133	123	108	
190	233	230	226	221	214	213	214	217	221	204	…	113	103	92	
191	232	229	225	221	215	211	212	214	224	212	…	95	87	83	
192	224	222	221	219	216	214	215	216	215	215	…	99	90	84	
193	222	221	220	219	218	216	216	217	215	214	…	119	108	101	

	252	253	254	255	256	257	258
0	12	3	22	14	13	32	48
1	11	9	24	8	11	30	46
2	15	20	28	8	7	29	46
3	16	25	30	15	5	27	44
4	11	16	19	16	5	27	45
..	…	…	…	…	…	…	…
189	105	115	122	119	150	155	163
190	96	111	117	111	153	158	165
191	94	115	124	120	154	158	164
192	101	127	121	98	115	115	161
193	119	144	139	116	115	114	160

[194 rows x 259 columns]

[47]:
```
df_blue = blue/255
df_green = green/255
df_red = red/255
```

[48]:
```
pca_b = PCA(n_components=50)
pca_b.fit(df_blue)
trans_pca_b = pca_b.transform(df_blue)
pca_g = PCA(n_components=50)
pca_g.fit(df_green)
trans_pca_g = pca_g.transform(df_green)
pca_r = PCA(n_components=50)
pca_r.fit(df_red)
trans_pca_r = pca_r.transform(df_red)
```

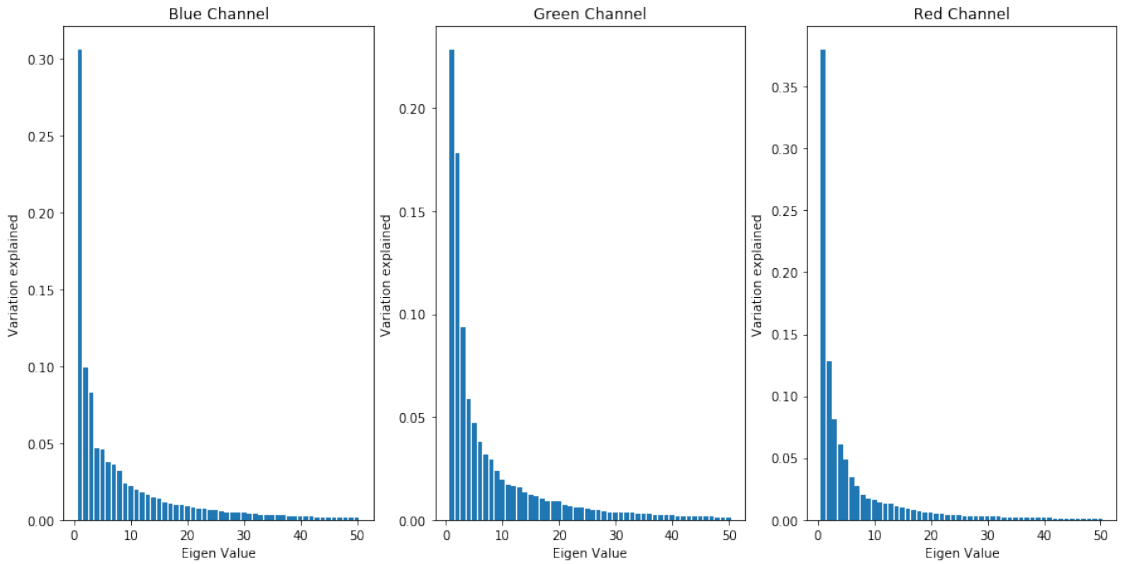
[50]:
```
print(trans_pca_b.shape)
print(trans_pca_r.shape)
print(trans_pca_g.shape)
```

```
(194, 50)
(194, 50)
(194, 50)
```

[51]:
```
print(f"Blue Channel : {sum(pca_b.explained_variance_ratio_)}")
print(f"Green Channel: {sum(pca_g.explained_variance_ratio_)}")
print(f"Red Channel  : {sum(pca_r.explained_variance_ratio_)}")
```

```
Blue Channel : 0.967393998073789
Green Channel: 0.9685616475342463
Red Channel  : 0.9786243808465617
```

[52]:
```
fig = plt.figure(figsize = (15, 7.2))
fig.add_subplot(131)
plt.title("Blue Channel")
plt.ylabel('Variation explained')
plt.xlabel('Eigen Value')
plt.bar(list(range(1,51)),pca_b.explained_variance_ratio_)
fig.add_subplot(132)
plt.title("Green Channel")
plt.ylabel('Variation explained')
plt.xlabel('Eigen Value')
plt.bar(list(range(1,51)),pca_g.explained_variance_ratio_)
fig.add_subplot(133)
plt.title("Red Channel")
plt.ylabel('Variation explained')
plt.xlabel('Eigen Value')
plt.bar(list(range(1,51)),pca_r.explained_variance_ratio_)
plt.show()
```



[54]:
```
b_arr = pca_b.inverse_transform(trans_pca_b)
g_arr = pca_g.inverse_transform(trans_pca_g)
r_arr = pca_r.inverse_transform(trans_pca_r)
print(b_arr.shape, g_arr.shape, r_arr.shape)
```

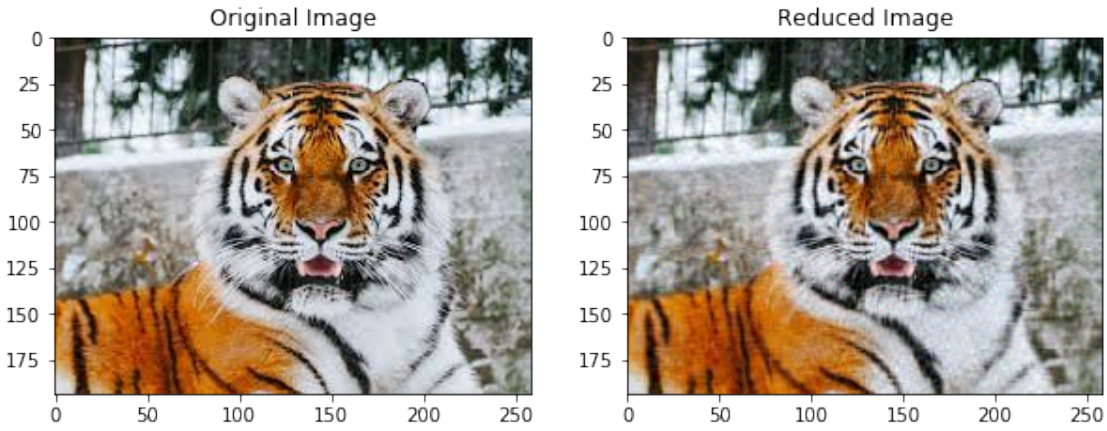
```
(194, 259) (194, 259) (194, 259)
```

[55]:
```
img_reduced= (cv2.merge((b_arr, g_arr, r_arr)))
print(img_reduced.shape)
```

```
(194, 259, 3)
```

[56]:
```
fig = plt.figure(figsize = (10, 7.2))
fig.add_subplot(121)
plt.title("Original Image")
plt.imshow(img)
fig.add_subplot(122)
plt.title("Reduced Image")
plt.imshow(img_reduced)
plt.show()
```

```
Clipping input data to the valid range for imshow with RGB data ([0..1] for
floats or [0..255] for integers).
```


Original Image
Reduced Image
0
25
50
75
100
125
150
175
0
50
100
150
200
250

[ ]: