PrzemyslawOlenderPD6

June 8, 2021

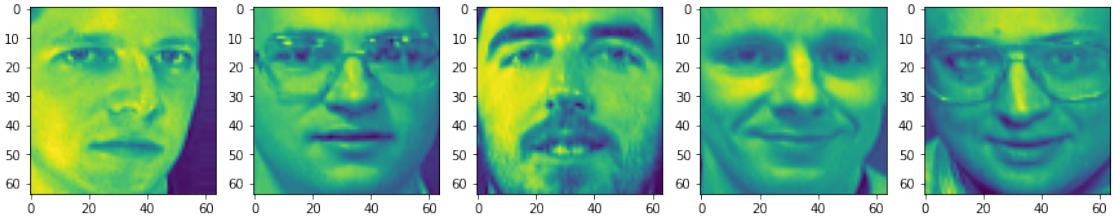
[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.datasets import fetch_olivetti_faces
```

- Narysować wybrane obrazy.
- Wykorzystać algorytm PCA do kompresji zbioru Olivetti Faces. Dobrać odpowiednią liczbę składowych. Po transformacji obliczyć stopień kompresji. Rozmiar obrazka: liczba wartości numerycznych
- Przeprowadzić transformację odwrotną (inverse_transform). Narysować, porównać z pkt. 0. Obliczyć błąd rekonstrukcji w postaci błędu RMSE dla każdego obrazu.
- Przygotować kilka / kilkanaście zmodyfikowanych obrazów (np. obróconych o 90 stopni, przyciemnionych, odbitych w poziomie).
- Korzystając z modelu wyuczonego w pkt. 1 przeprowadzić transformację, a następnie odwrotną transformację obrazów z pkt. 3.
- Obliczyć błąd rekonstrukji dla każdego typu modyfikacji. Porównać z wartościami błędu uzyskanymi w pkt. 2.
- Czy PCA może służyć do wykrywania pewnego typu anomalii w zdjęciach twarzy? Jeżeli tak to jakich?

[2]:
```
faces = fetch_olivetti_faces()
data = faces['data']
images = faces['images']
test_images = images[[1, 50, 100, 200, 300],:,:]
```

1 Rysowanie obrazów

[3]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
for i in range(5):
    axs[i].imshow(test_images[i])
```



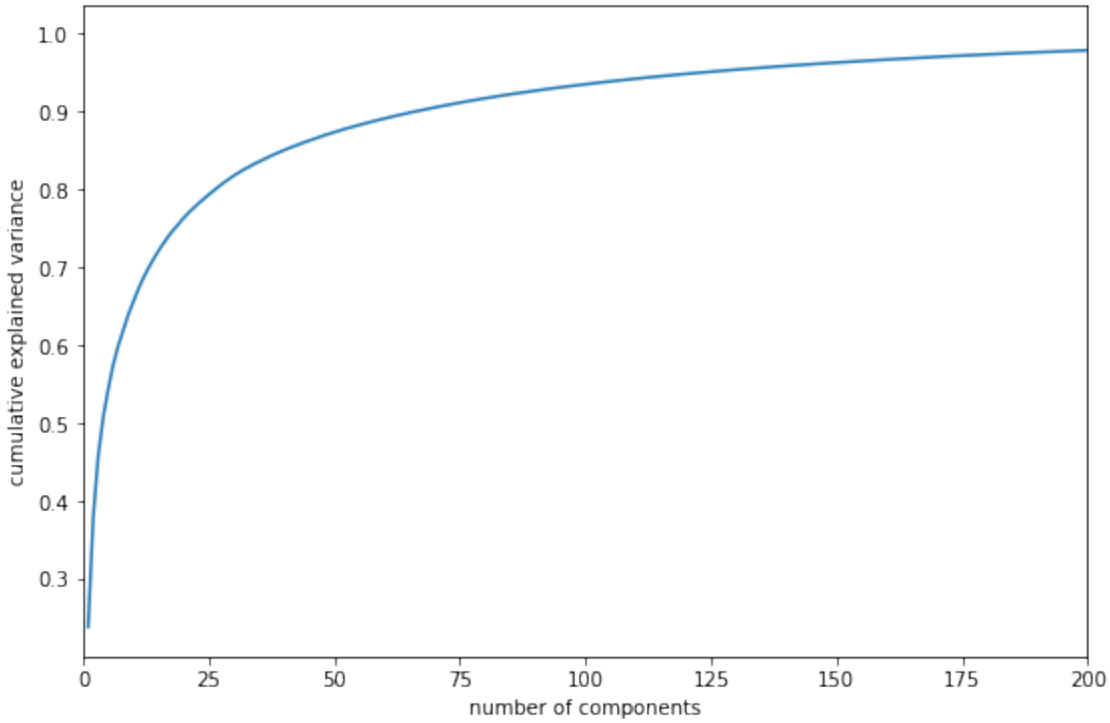
2 Kompresja

[4]:
```
from sklearn.decomposition import PCA
```

[5]:
```
pca = PCA(n_components=400)

data_pca = pca.fit_transform(data)
```

[6]:
```
plt.figure(figsize=(9,6))
plt.plot(range(1, len(pca.explained_variance_ratio_)+1), np.cumsum(pca.
 ↪explained_variance_ratio_))
plt.xlabel('number of components')
plt.xlim(0, 200)
plt.ylabel('cumulative explained variance')
```

[6]:
```
Text(0, 0.5, 'cumulative explained variance')
```



Przy około 100 komponentach wyjaśnionych jest 90% warinacji.

[7]:
```
pca = PCA(n_components=100)
data_pca2 = pca.fit_transform(data)
```

[8]:
```
data.shape
```

[8]: (400, 4096)

[9]:
```
data_pca2.shape
```

[9]: (400, 100)

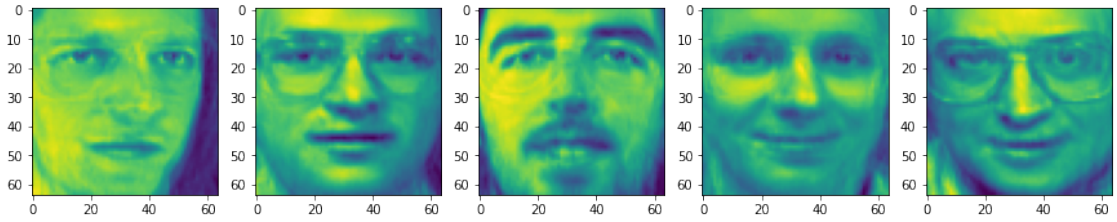
[10]:
```
print(f"Stopień kompresji to {(data.shape[0] * data.shape[1]) / (data_pca2.
 ↪shape[0] * data_pca2.shape[1])}")
```

```
Stopień kompresji to 40.96
```

3 Transformacja odwrotna

[11]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
inv_t = pca.inverse_transform(data_pca2)
idx = [1, 50, 100, 200, 300]
```

```
for i in range(5):
    axs[i].imshow(inv_t[idx[i]].reshape(64, 64))
```



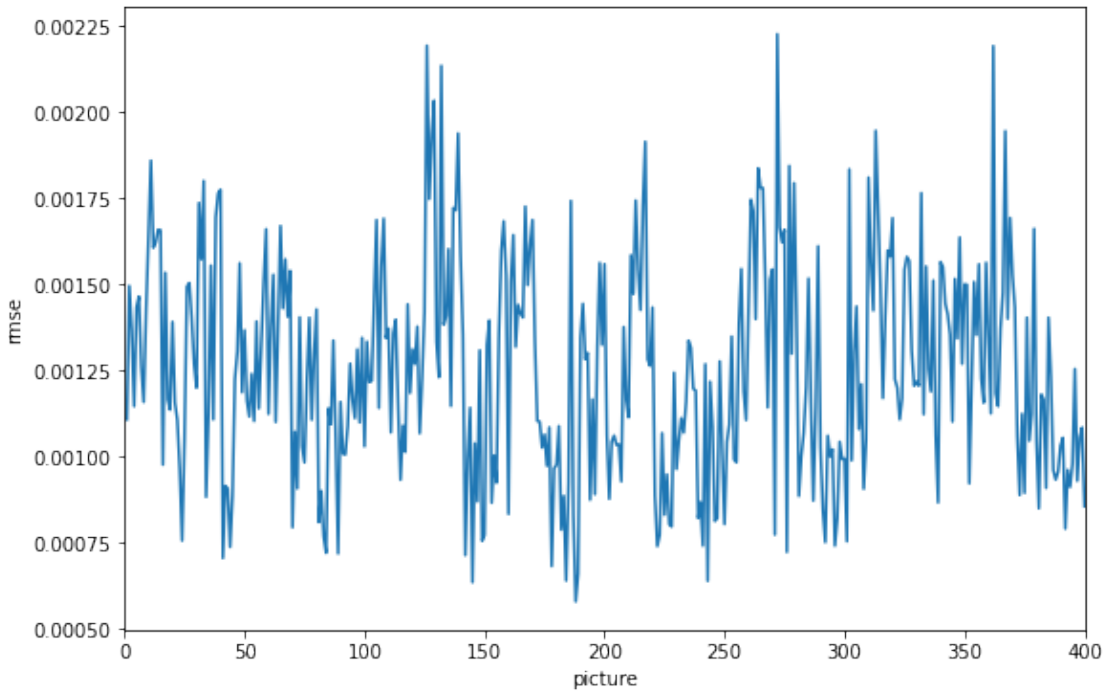
Różnica po inverse transforamtion jest niezauważalna.

[12]:
```
from sklearn.metrics import mean_squared_error as rmse
```

[13]:
```
rmse_arr = []

for i in range(400):
    rmse_arr.append(rmse(data[i], inv_t[i], squared = True))
```

[14]:
```
plt.figure(figsize=(9,6))
plt.plot(range(1, len(rmse_arr)+1), rmse_arr)
plt.xlabel('picture')
plt.xlim(0, 400)
plt.ylabel('rmse')
```

[14]:
```
Text(0, 0.5, 'rmse')
```



[15]:
```
np.mean(rmse_arr)
```

[15]: 0.0012535453

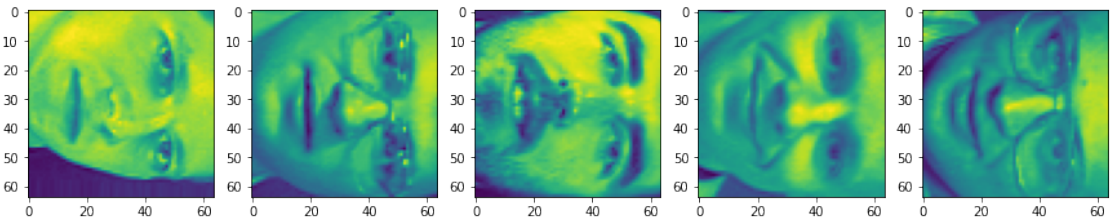
Błąd rekonstrukcji jest bardzo niski, wynosi troche ponad 0.001.

4 Modyfikacja obrazów

4.1 Obrót

[16]:
```
from skimage.transform import rotate
```

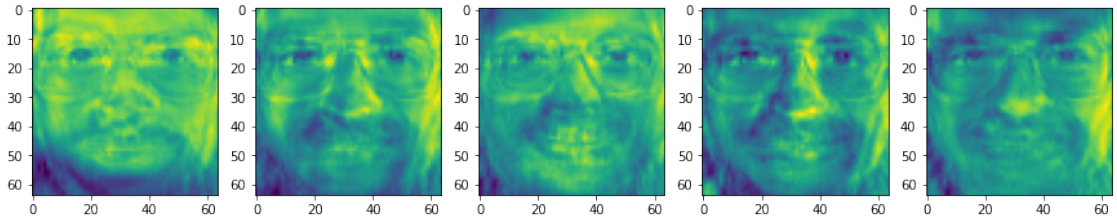
[17]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
rotated_images = []
for i in range(5):
    rotated_images.append(rotate(test_images[i], -90))
    axs[i].imshow(rotated_images[i])
```



4.2 Obrót po PCA i inverse trasformation

[18]:
```
rotated_images_pca = pca.inverse_transform(pca.transform(np.
↪array(rotated_images).reshape(5, -1)))
```

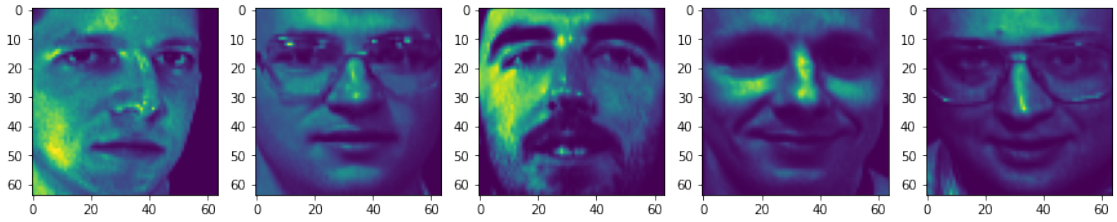
[19]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
for i in range(5):
    axs[i].imshow(rotated_images_pca[i].reshape(64,64))
```



4.3 Przyciemnienie

[20]:
```
from skimage import exposure
```

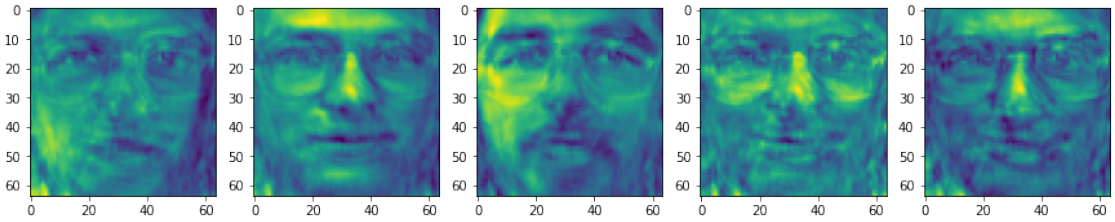
[21]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
darker_images = []
for i in range(5):
    darker_images.append(exposure.adjust_gamma(test_images[i], 5))
    axs[i].imshow(darker_images[i])
```



4.4 Przyciemnienie po PCA i inverse trasformation

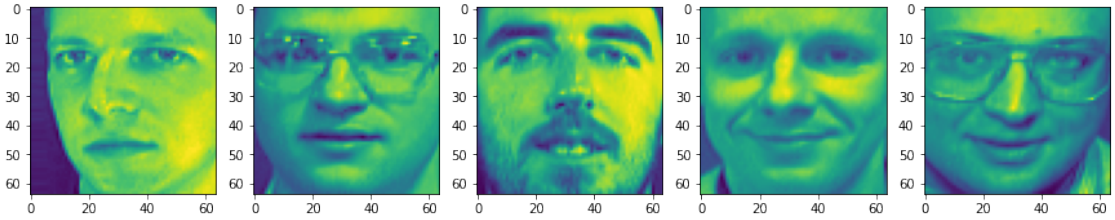
[22]:
```
darker_images_pca = pca.inverse_transform(pca.transform(np.array(darker_images).
↪reshape(5, -1)))
```

[23]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
for i in range(5):
    axs[i].imshow(darker_images_pca[i].reshape(64,64))
```



4.5 Odbicie w poziomie

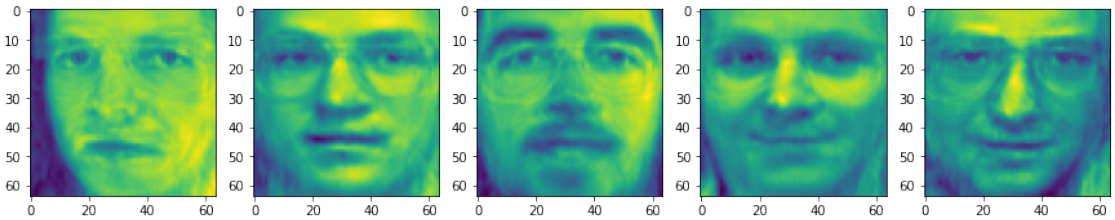
[24]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
fliped_images = []
for i in range(5):
    fliped_images.append(test_images[i][:,::-1])
    axs[i].imshow(fliped_images[i])
```



4.6 Odbicie w poziomie po PCA i inverse trasformation

[25]:
```
fliped_images_pca = pca.inverse_transform(pca.transform(np.array(fliped_images).
↪reshape(5, -1)))
```

[26]:
```
fig, axs = plt.subplots(1, 5, figsize=(15, 3))
for i in range(5):
    axs[i].imshow(fliped_images_pca[i].reshape(64,64))
```



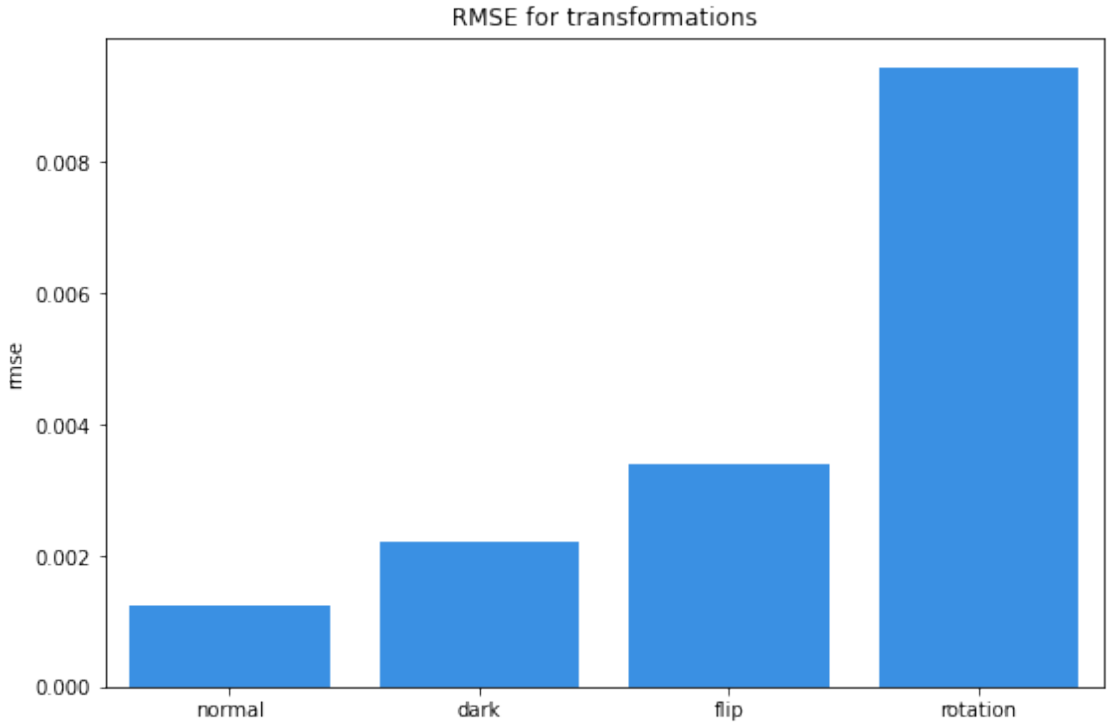
5 Błąd transformacji

[27]:
```
# błąd dla wszystkich obrazów przy obrocie
rotated_all = np.array([rotate(images[i], -90) for i in range(len(images))]).
 ↪reshape(len(images), -1)
rotated_all_pca = pca.inverse_transform(
    pca.transform(
        rotated_all
    )
)

rmse_rotated = np.array([rmse(rotated_all[i], rotated_all_pca[i], squared =␣
 ↪True) for i in range(len(rotated_all))])

# błąd dla wszystkich obrazów po przciemnieu
darker_all = np.array([exposure.adjust_gamma(images[i], 5) for i in␣
 ↪range(len(images))]).reshape(len(images), -1)
darker_all_pca = pca.inverse_transform(
    pca.transform(
        darker_all
    )
)

rmse_darker = np.array([rmse(darker_all[i], darker_all_pca[i], squared = True)␣
 ↪for i in range(len(darker_all_pca))])

# błąd dla wszystkich obrazów po odbiciu
flipped_all = np.array([images[i][:,::-1] for i in range(len(images))]).
 ↪reshape(len(images), -1)
flipped_all_pca = pca.inverse_transform(
    pca.transform(
        flipped_all
    )
)

rmse_flipped = np.array([rmse(flipped_all[i], flipped_all_pca[i], squared =␣
 ↪True) for i in range(len(flipped_all))])
```

[28]:
```
df = pd.DataFrame({
    'transformation' : ['normal', 'rotation', 'dark', 'flip'],
    'rmse' : [np.mean(rmse_arr), np.mean(rmse_rotated), np.mean(rmse_darker),␣
 ↪np.mean(rmse_flipped)]
    })
```

[29]:
```
fig, ax = plt.subplots(1, 1, figsize=(9,6))

sns.barplot(data = df, x = 'transformation', y = 'rmse',
            color = 'dodgerblue',
            order = df.sort_values('rmse')['transformation'], ax = ax)
ax.set_title('RMSE for transformations')
ax.set_xlabel('')
```

[29]:
```
Text(0.5, 0, '')
```



Największe RMSE jest dla obrócenia zdjęcia o 90, można było to wywnioskować już po narysowanych obrazach, po transformacji i transformacji odwrotenj obrazy w ogóle nie były obrócone i prawie wcale nie są do siebie podobne. Przyciemnione obrazy również straciły na jakości jednak wciąż można je rozpoznać a efekt przyciemnienia się zachował. Na obazach odbitych w poziomie widać podobieństow i zachowanie efektu.

PCA może być wykorzytywane w wykrywaniu anomalii w zdjęciach, szczególnie w obrotu (można na przykład stworzyć narzędzie do automatycznego obracania zdjęć w dobrą stronę).