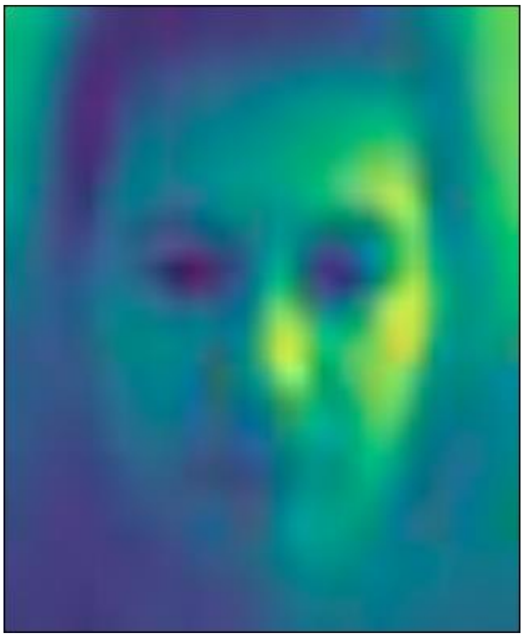
Q1:

(1) Mean face :

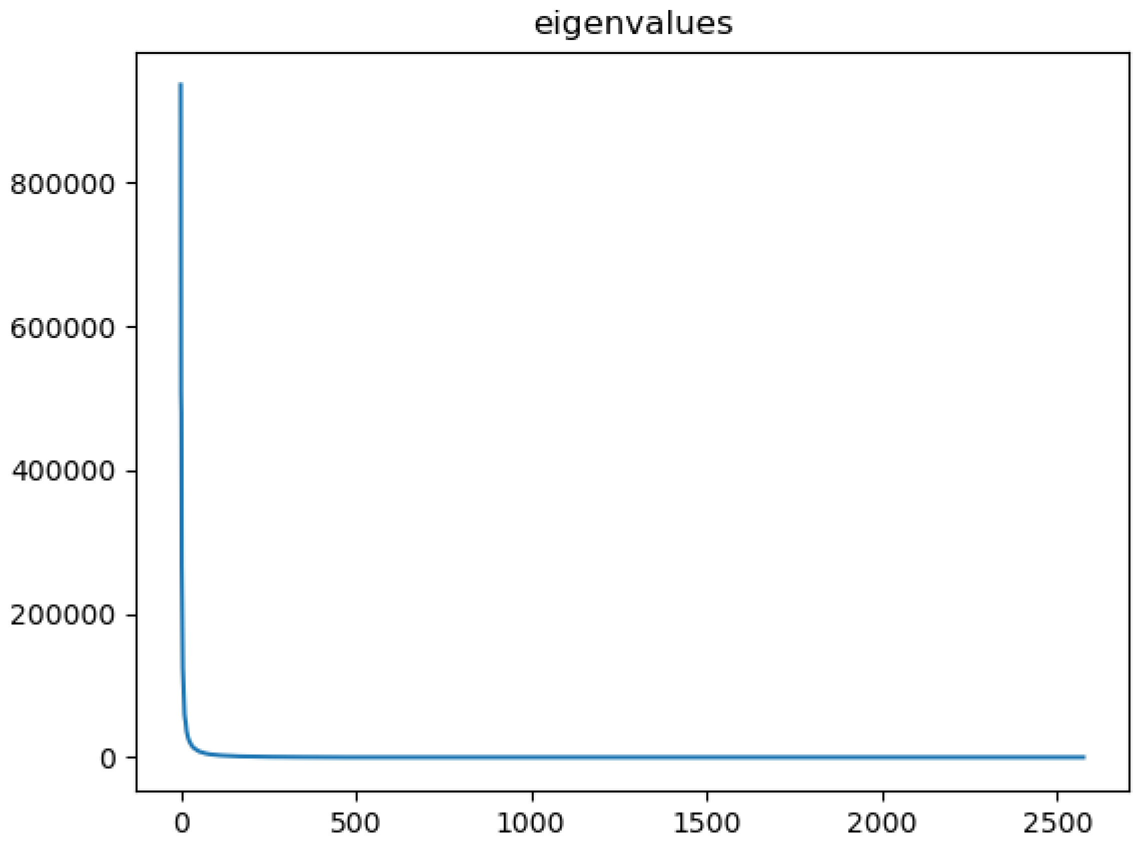


Discuss:
This is the mean face of all images in the train set, we can observe that this face looks blurred, however it still has general characteristics and components of a normal face such as eye, nose, hair, oval-shape face…

(2) Now we consider eigenvectors and eigenvalues of covariance matrix when we compute them in 2 different ways: compute the eigenvalues and eigenvectors of covariance matrix $S = (1/N)A.\mathbf{A}^T$ directly, and use low-dimensional computation of eigenspace using matrix $S1 = (1/N).\mathbf{A}^T.A$

A) When compute the eigenvectors and eigenvalues of the covariance matrix $\mathbf{S=(1/N)AA^T}$ directly :

The graph of eigenvalues of covariance matrix S:



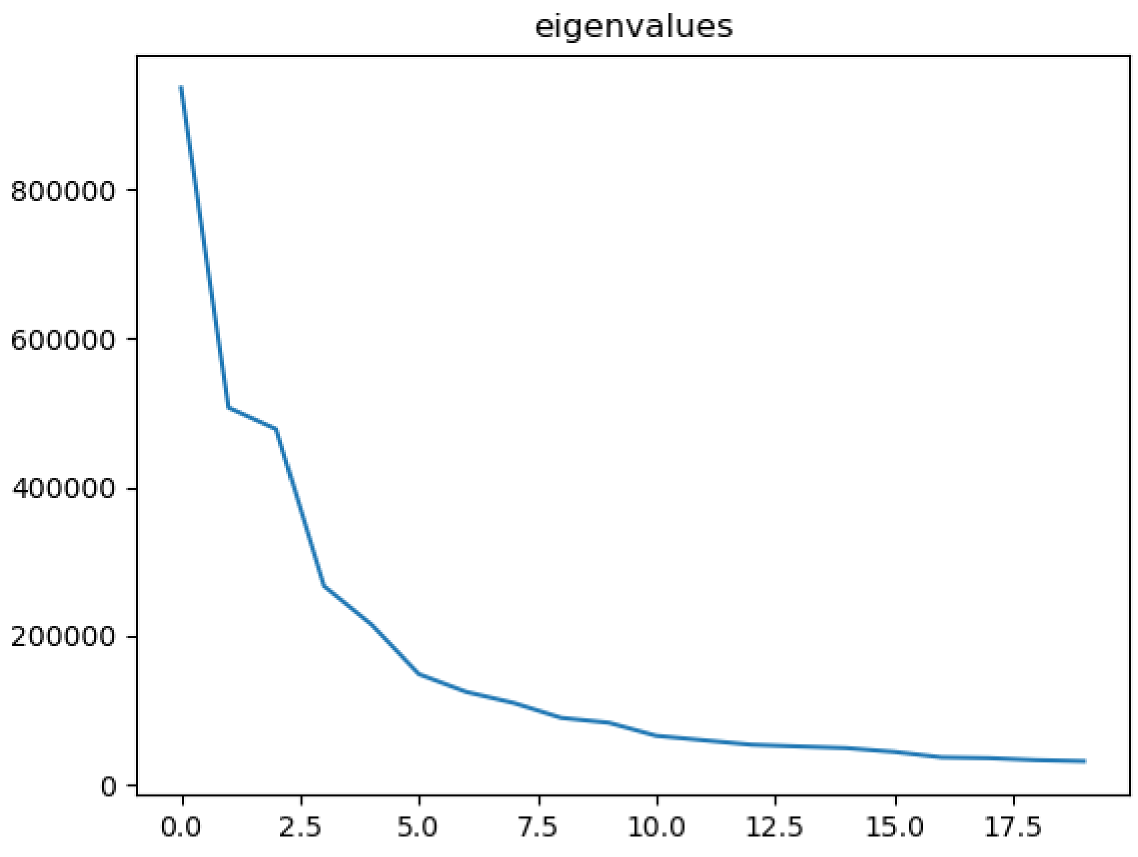
There are 2576 eigenvalues of the covariance matrix S, that's reasonable because S is a square matrix of size 2576*2576. Assume we sort these eigenvalues in the decreasing order of their absolute values : |a0| > |a1| > |a2| > … > |a2576| . Then by computation, we have absolute value of a415, a416 … a2576 are all less than 0.000001. In more detail, |a415| = 1.113e-10, |a416| = 1.066e-10, |a417| = 9.961e-11 …

Therefore, if we define zero eigenvalues as eigenvalues with absolute value less than 0.000001, then we have 2161 zero eigenvalues (a415, a416, … a2575) and 415 nonzero eigenvalues (a0,a1,a2,..a414).
The above graph looks very steep because the amount of nonzero eigenvalues is small: only 415 over 2576 eigenvalues are nonzero, and furthermore, the absolute value of the first eigenvalues are too large in comparison to the later eigenvalues.
Making some computations, we have :

$$|a0|/|a10| = 14.161$$
$$|a0|/|a50| = 103.858$$
$$|a0|/|a100| = 285.314$$
$$|a0|/|a400| = 7224.176$$

These ratios show that the value of eigenvalues decrease so quickly, and it's the reason why the graph looks very steep.

To have a more intuitive graph, we can look at the graph of first 20 largest eigenvalues:



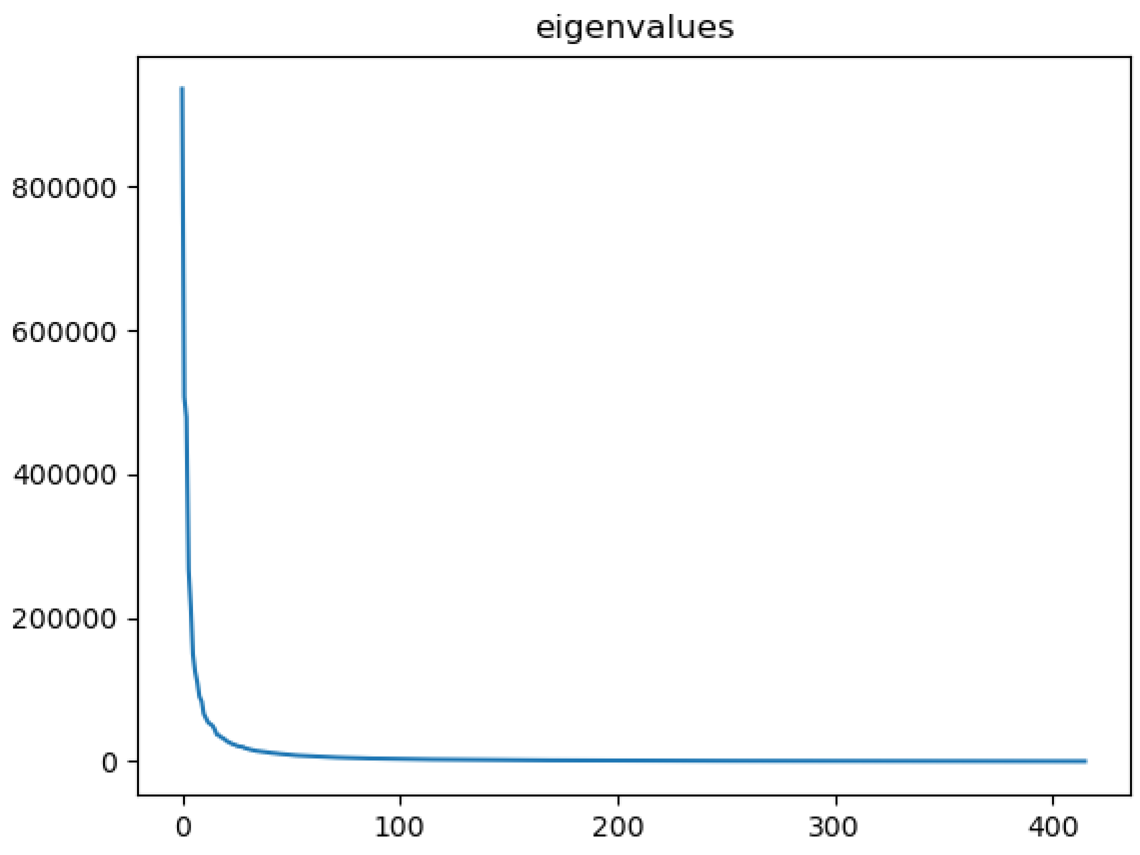
The 20 largest eigenvalues:
936388.625, 507218.260, 478365.398, 267221.967, 215489.795, 148960.417, 125026.351, 110178.655, 90206.887, 83675.736, 66122.396, 60101.236, 54124.506, 51780.229, 49634.876, 44505.252, 37134.966, 36094.535, 33406.765, 31960.005

The 20 eigenvectors that correspond to 20 largest eigenvalues above:

```
array([-0.01228713, -0.01349973, -0.01402377, ...,  0.0029651 , 0.00260247,  0.00274912]),
array([-0.0192394 , -0.01952942, -0.02006991, ...,  0.04474409, 0.04445823,  0.04521234]),
array([-0.01088869, -0.01215637, -0.01382132, ..., -0.02270606, -0.02206885, -0.02091973]),
array([-3.57635935e-03, -1.01472866e-03, -1.42966746e-05, ...,3.607635e-02,  3.61897205e-02]),
array([0.00761264, 0.00938116, 0.01236837, ..., 0.00798879, 0.01117107, 0.01306759]),
array([-0.01599602, -0.01976549, -0.02473501, ...,  0.01027646, 0.01164444,  0.0145276 ]),
array([-0.02961436, -0.03232598, -0.03585976, ..., -0.00794767, -0.00588496, -0.00622126]),
array([-0.03203351, -0.03479692, -0.03619378, ...,  0.02891417, 0.03035104,  0.02824698]),
array([-0.02750365, -0.02835834, -0.03318466, ..., -0.00737587, -0.00531902, -0.00327706]),
array([-0.02647438, -0.02474755, -0.02326615, ...,  0.0681444 , 0.06178841,  0.05722392]),
array([0.04730973, 0.04374813, 0.0415116 , ..., 0.00529897, 0.00539652, 0.00646058]),
array([-0.00596232, -0.00889039, -0.01126274, ..., -0.00497306,  -0.01058243, -0.01227784]),
array([-0.04191433, -0.03548517, -0.02852901, ..., -0.03237877, -0.03579812, -0.04056137]),
array([-0.00744906, -0.00382175,  0.00511729, ...,  0.00032466, -0.00415377, -0.01279311]),
array([-0.01415264, -0.01495962, -0.01715392, ..., -0.0029764 , -0.00384384, -0.00791951]),
array([0.02432119, 0.03267958, 0.03552327, ..., 0.00997197, 0.00458816, 0.01045138]),
array([ 0.00128105,  0.00064928, -0.00527637, ..., -0.04277924,  -0.04505291, -0.04579452]),
array([-0.0163748 , -0.023374  , -0.03073449, ...,  0.00751814, 0.00125748,  0.00265255]),
array([-0.04210698, -0.04243713, -0.03779016, ..., -0.009805  ,  -0.00511903, -0.00498672]),
array([-0.02593623, -0.02481796, -0.02165989, ..., -0.00900535, -0.00413227, -0.00836397])
```

B) When use **low-dimensional computation** of eigenspace :
In this approach, we use the matrix $S1 = (1/N)A^TA$

The graph of eigenvalues of matrix S1:



There are 416 eigenvalues of S1, that's reasonable because S1 is a square matrix of size 416*416. Assume we sort these eigenvalues in the decreasing order of their absolute values: |b0| >|b1| >

|b2| > … > |b415|. Then by computation, we have |b413| = 90.095, |b414| = 77.860, |b415| = 5.807e-12

We have defined above that zero eigenvalues are eigenvalues with absolute value less than 0.000001. Thus, among these 416 eigenvalues, there is only one zero eigenvalue, it is b415, and we have total 415 non-zero eigenvalues (b0, b1, … b414).

Now, we **compare the eigenvalues and eigenvectors obtained by two above methods:**

- The number of nonzero eigenvalues of two methods A (compute S = (1/N)AAT) and B (compute S1 = (1/N)**A**^{T}A) are the same.
- By doing further computation, I observe that all 416 eigenvalues of the low-dimensional computation method (use matrix S1 = (1/N)**A**^{T}.A) are identical to 416 largest eigenvalues of method A (use matrix S = (1/N)A**A**^{T}). Therefore, we can conclude that all eigenvalues of covariance matrix S1 = (1/N)**A**^{T}.A are rqual to 416 largest eigenvalues of the covariance matrix S = (1/N)A**A**^{T}.

Discuss:
This is a very meaningful conclusion because instead of compute eigenvalues of the matrix S with shape (2576,2576) that is very costly, we can compute 416 largest eigenvalues of S by compute the eigenvalues of S1 , a matrix with much smaller shape (416,416) , because we have concluded that eigenvalues of S1 are equal to 416 largest eigenvalues of S.

Pros and Cons of each method:

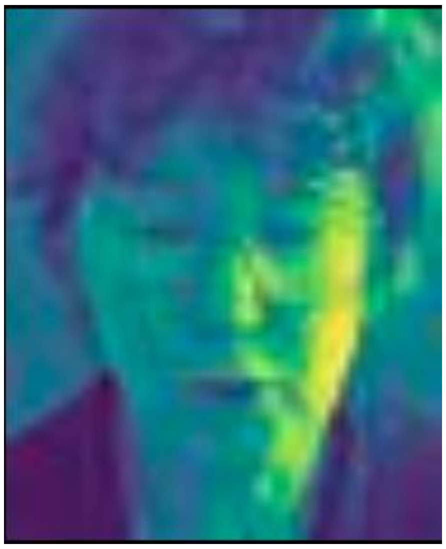
Method A : compute eigenvalues and eigenvectors of covariance matrix **S = (1/N)AA^T**
Pros: By compute eigenvectors of S, we can use these eigenvectors for the task of face reconstruction.
Cons: The size of matrix S is too large : 2576*2576, so the time required to compute the eigenvalues and eigenvectors of S is quite large. In fact, I write a function to compute the eigenvalues and eigenvectors of covariance matrix S in question 1 using python, and it costs 20.04 seconds, that's really a large amount of time to run a python script.

Method B : compute eigenvalues and eigenvectors of low-dimensional matrix **S1 = (1/N)A^T.A**
Pros: the size of S1 is small : 416*416, so the required amount of time to compute eigenvalues and eigenvectors of S1 is much smaller than method A. In fact, I write a function to compute the eigenvalues and eigenvectors of the matrix S1 = (1/N)**A**^{T}.A) in question 1, and it costs only about 0.12922 seconds to execute this function, that's about 1/150 amount of time computing eigenvalues and eigenvectors of matrix S in method A.
Cons: Eigenvectors computed by this method cannot be used for the task face reconstruction.

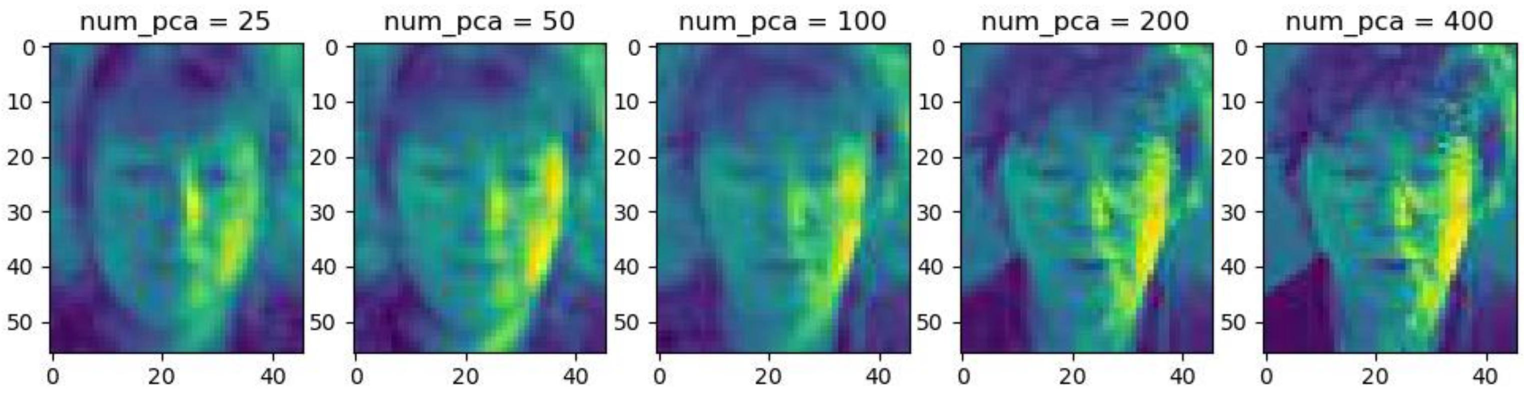
(3) Face reconstruction by PCA:

Now we implement face instruction for some faces in the dataset:

- Real face from the dataset with id 0:



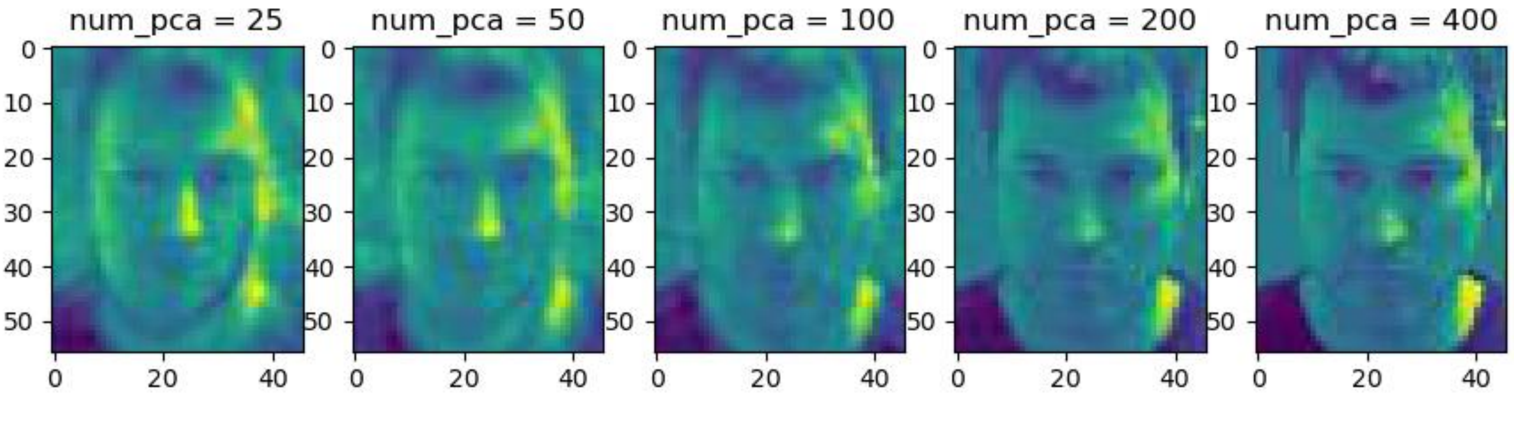
Reconstructed faces when varying the number of PCA bases:



The left most image is the face reconstruction by using 25 most important PCA bases, the second one reconstructed by using 50 most important PCA bases, similarly for remaining images

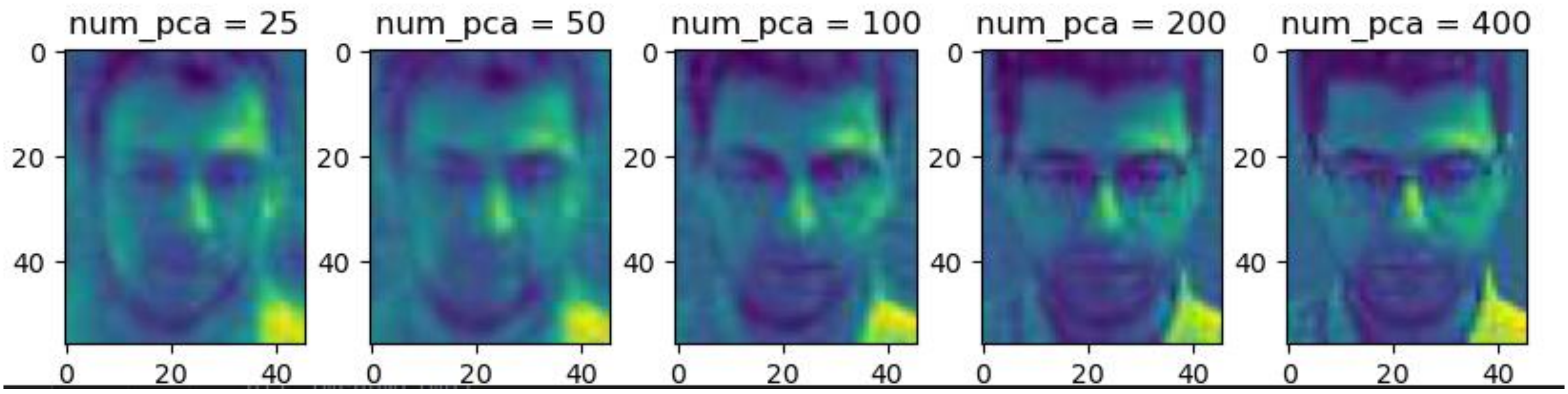
- Real face from the dataset with id 28:

Reconstructed faces when varying the number of PCA bases:



Real face from the dataset with id 48:

Reconstructed faces when varying the number of PCA bases:



Discuss:
In the above faces of 3 different people, we can observe that when we increase the number of PCA bases, the reconstructed face would look better and better, and with 400 PCA bases, the reconstructed faces are nearly identical to the real face. We can explain this as following: when we use little number of PCA bases, we can only reconstruct most general features of the face, and a lot of information in the face will be lost, as a result the reconstructed face would be very blurred, when we keep adding more and more number of PCA bases, we can reconstruct more information of the original face and reduce the amount of lost information, therefore, increasing number of PCA bases make the reconstructed faces look better and better.