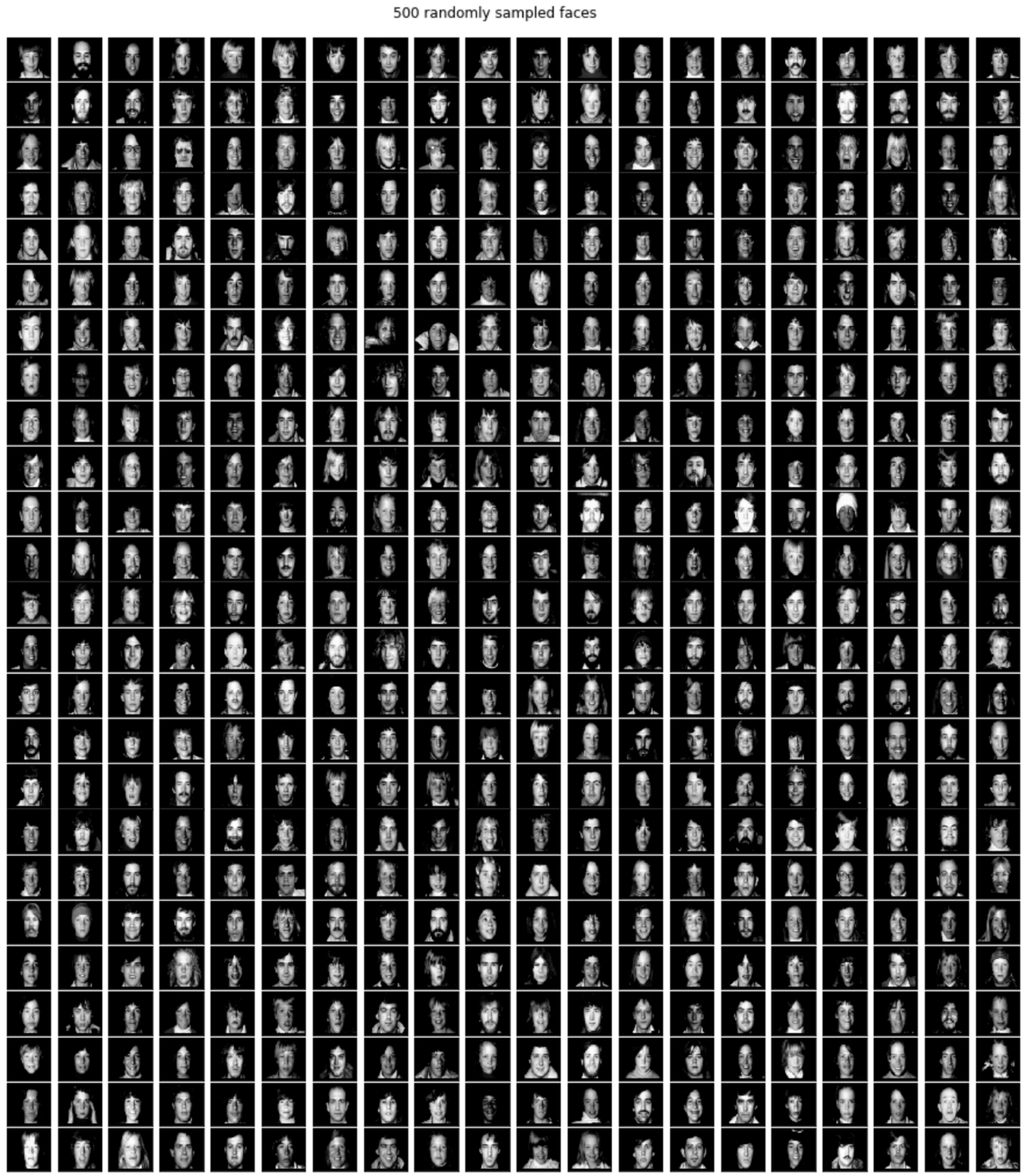
Random sample of 500 faces



Eigenface algorithm: PCA on face images

1. Load in grayscale images, all the with same width and height: $I_1, I_2, \ldots, I_N$.

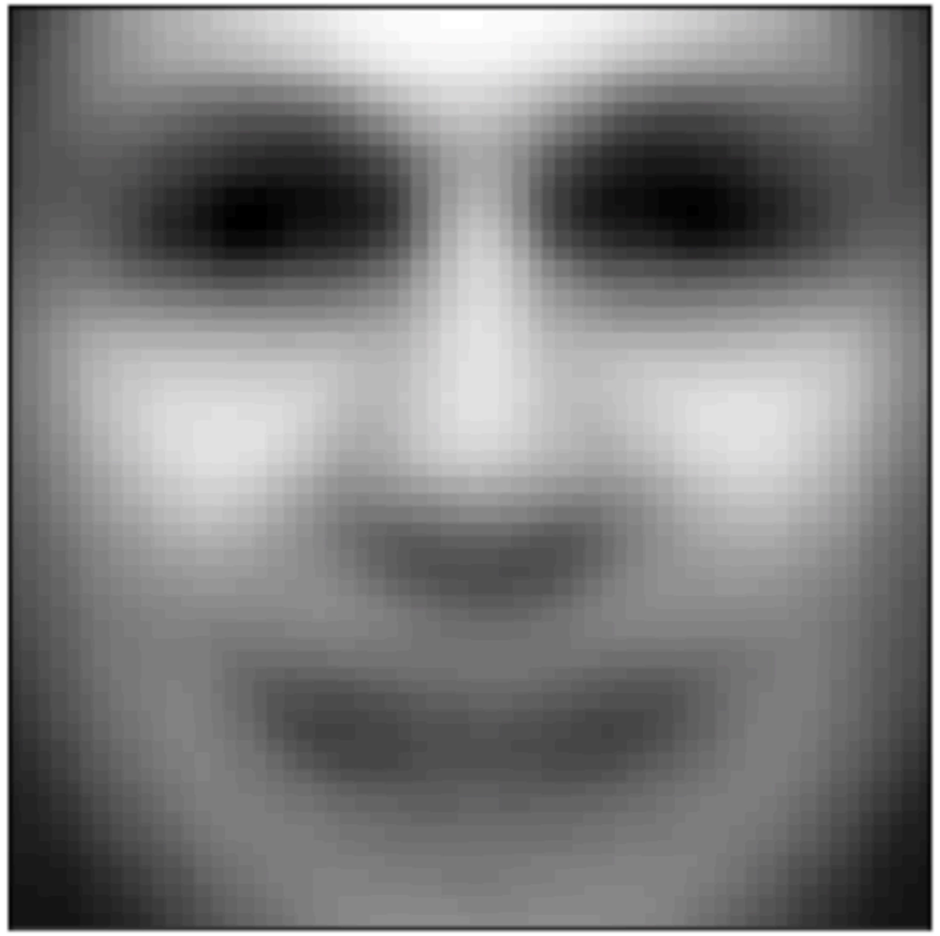
2. Collapse each 2D image into 1D vectors $\vec{x}_i$ (e.g. 16x16 2D image $\Rightarrow$ 256 1D vector). So, number of samples N = number of images. Variables are each of the pixels (e.g. if length($\vec{x}_i$) is 256. $M = 256$). Like usual, $A = [\vec{x}_1, \vec{x}_2, \ldots, \vec{x}_M]$ (*rows: images/samples, cols: 1D pixel value variables*)

3. Center the images (subtract grand mean image): $A_c = A - \vec{\mu}$, where $\vec{\mu}$ is the column means of A (i.e. the mean pixel value at the same position across all images in the dataset).

4. Compute covariance matrix Σ then recover eigenvalues and eigenvectors.

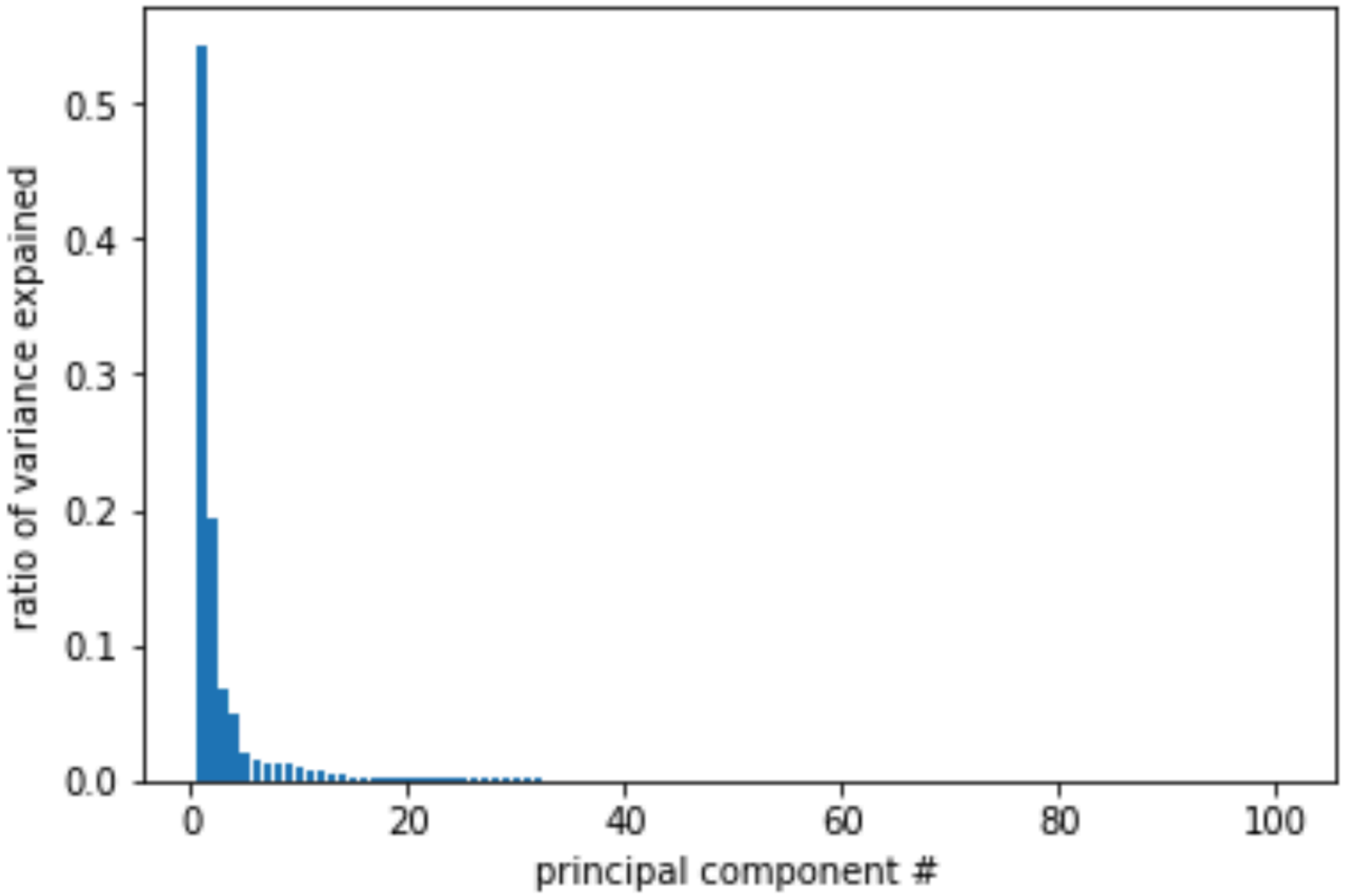
5. Project images onto top k of principal components.

Grand mean of 500 faces ($\vec{\mu}$)

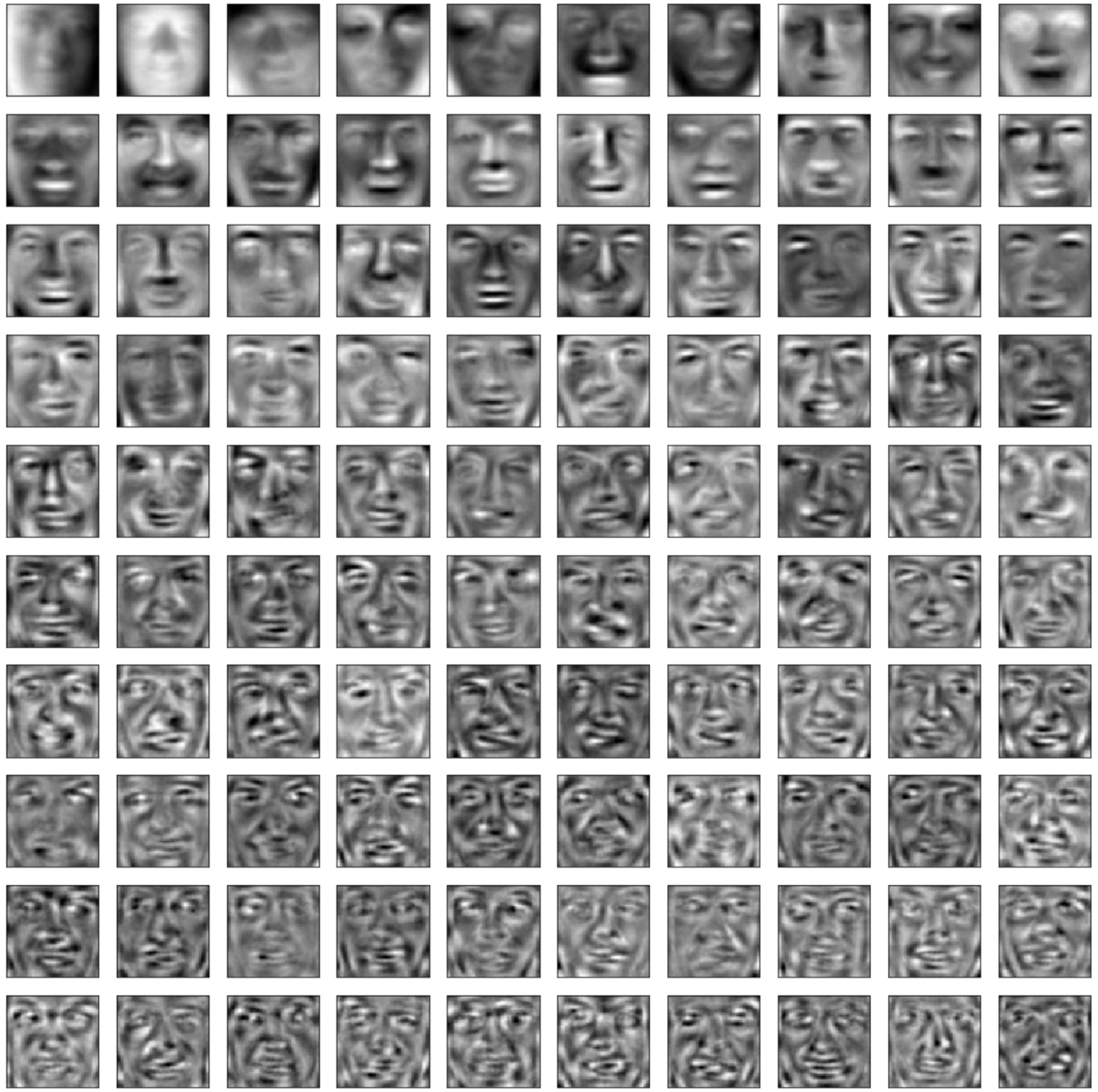


Because $\vec{\mu}$ is 1D vector, I had to **reshape** it into a 2D image format (e.g. 256 1D vector -> 16x16 2D image)

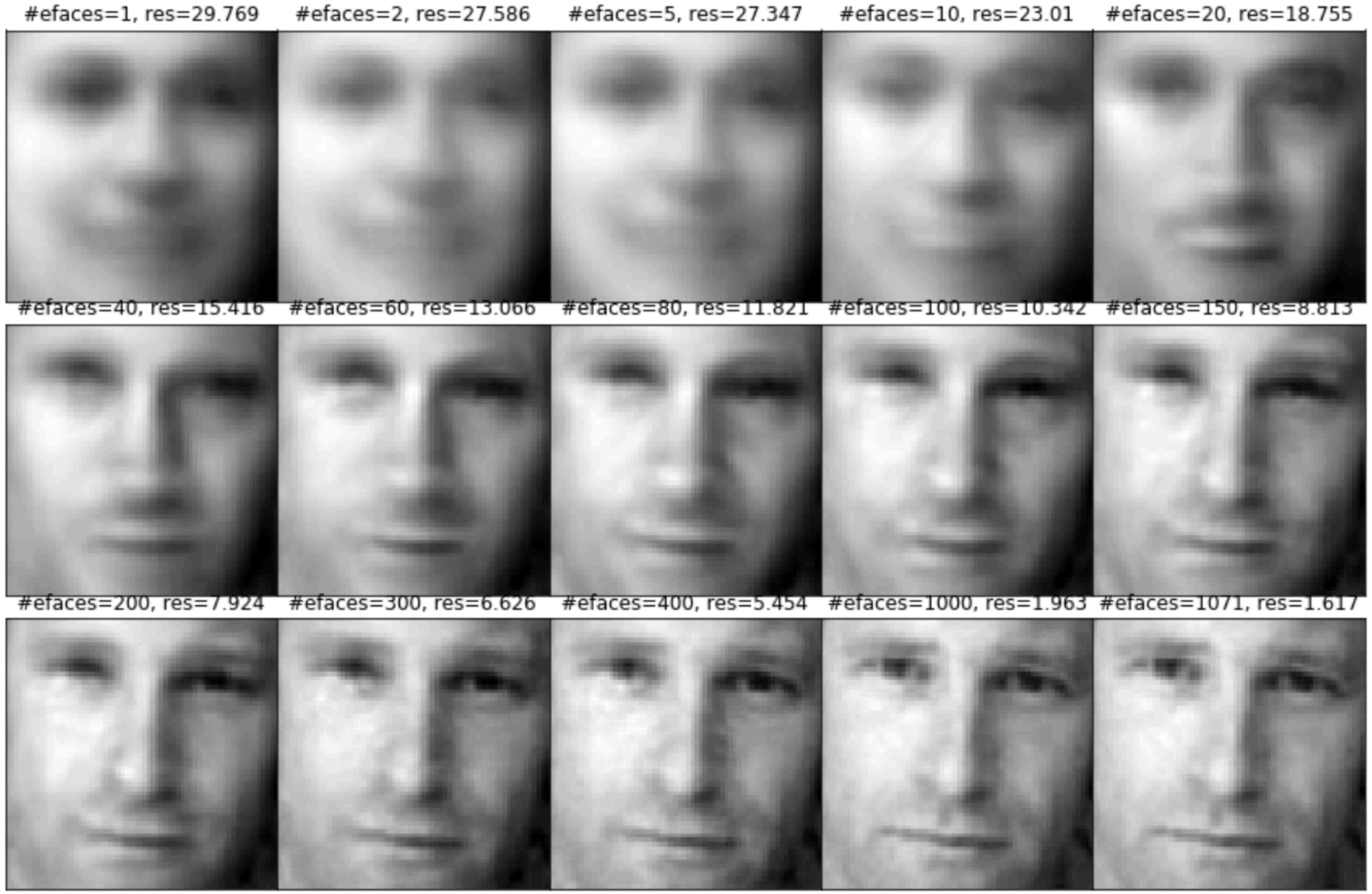
Variance explained by top eigenvalues/PCs



Face principal components: top k eVecs (Reshaped 1D -> 2D)



Project one face image onto top K PCs



Facial recognition using Eigenface algorithm

1. Do PCA on faces in your "known face database", get PC vectors (2 slides back).
2. Project **query** image you want to recognize into PCA space, keeping top `k` PCs (1 slide back, except no need to reconstruct data space). `shape=(1, k)`
3. Re-project images in your known face database PCA space one-by-one (1 slide back, except no need to reconstruct data space). `shape=(1, k)`
4. Compute the distance (scalar) between projected query image and current project database image (both `(1, k)` vectors).
5. If match is close enough (distance < threshold), you recognize the face! Otherwise repeat step 3.
6. If you exhaust all images in known faces dataset, you do not recognize the query face.

Application: PCA on handwritten digits (Optdigits dataset)