1 Main vars

First we will define some of the variables for clarification:

1. X: original data matrix containing all images, each image is represented as a column vector.
2. $X_{normalized}$: normalized data matrix, obtained by subtracting the mean of each column/image from the data and then dividing it by its std dev.
3. D: dictionary matrix, contains a set of basis vectors that can be linearly combined to reconstruct the images.
4. Z: sparse coding matrix, contains sparse coefficients that represent each image as a linear combination of a few dictionary atoms. We calculate this with the following equation:

$$\underset{Z}{\operatorname{argmin}} ||X - DZ||_2^2 + \lambda ||Z||_1$$

   where $||.||_2$ and $||.||_1$ are the L_2 and L_1 norms. This equation has the least squares term and the LASSO penalty. So, λ is a param that control the strength of the penalty.

5. I_j: set of indices corresponding to the non-zero elements in the jth column of Z, notes which dictionary atoms are used to represent the jth image.
6. D_j: jth column of the dictionary matrix
7. R_j: residual error matrix for jth image, obtained by subtracting the contribution of all other dictionary atoms from the original data. It is used to update the jth atom.
8. n_{atoms}: the number of atoms used to reconstruct each image.
9. *reconstructed_data* : the reconstructed data matrix, which is obtained by multiplying the dictionary matrix by the sparse coding matrix and then denormalizing the data.
10. *reconstructed_images* : the reconstructed images, which are obtained by reshaping each column of the reconstructed data matrix back into the original image shape.

2 Future

Because of lack to time, currently the program only works for input images of the same dimensions. So, I would add more preprocessing steps to crop or resize images so that they are the same dimensions for dictionary creation.

I would also like to add the ability to automatically determine the optimal number of atoms. I thought about the possibility of doing this using calcualted residual error. But, an issue with that might be that different data might be visually differentiable with higher residuals than others, so it might not be a perfect solution.

The algorithm that updates the dictionary is pretty elementary. It would be fun to implement other dictionary update algorthms that I've read about like online dictionary learning.

I never had issues with runtime because of the preprocessing and the small data input, but if this became an issue I could have used CUDA to send some of the operations to my GPU. Also, I could have parallelized some of the loops that wouldn't have had issues with race conditions or shared memory.

Also, I could implement alternative dictionary learning algorithms, like the ones in the prompt and etc.

And, althought there is significant work that would need to be done, I could make a dictionary learning package. I could wrap the functions in a class, provide methods for training the dictionary, sparse coding, and etc. And then, also implement methods for saving and loading the dictionaries and sparse representatoins and parameter control. So, here I could implement alternative diciontary learning algorithms that can be used in different cases depending on what is best for the user.