1. Principal Component Analysis and Dictionary Learning

(a) For PCA, I select top 64 eigen vectors.

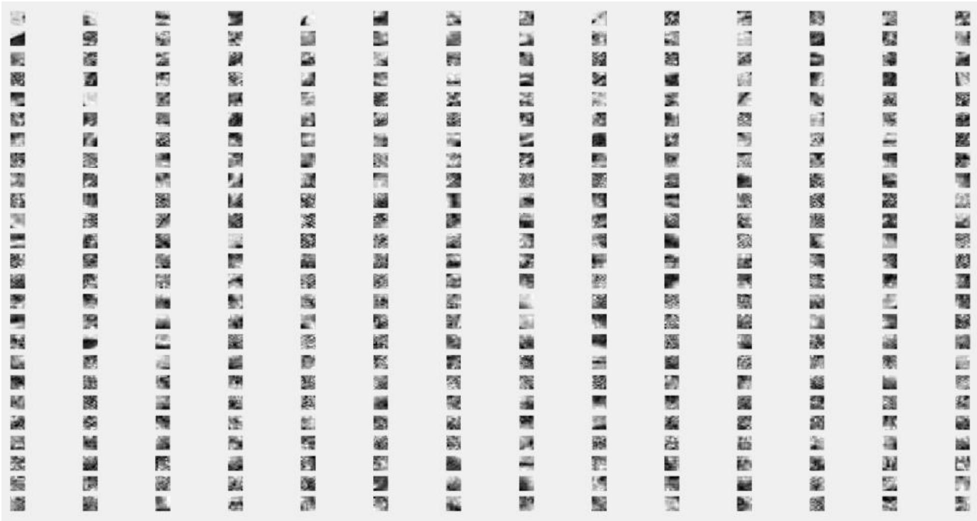
For KSVD, I set iteration as 50, and the original dictionary is a 256*256 identity matrix concatenate with the top 94 256*256 Fourier matrix.

```
Starting to  train the dictionary
Iteration   2   Total error is: 0.024187
Iteration   3   Total error is: 0.023767
Iteration   4   Total error is: 0.023441
Iteration   5   Total error is: 0.023384
Iteration   6   Total error is: 0.023227
Iteration   7   Total error is: 0.023209
Iteration   8   Total error is: 0.02303
Iteration   9   Total error is: 0.023063
Iteration   10   Total error is: 0.022969
Iteration   11   Total error is: 0.022966
Iteration   12   Total error is: 0.022886
Iteration   13   Total error is: 0.022895
Iteration   14   Total error is: 0.022841
Iteration   15   Total error is: 0.022893
Iteration   16   Total error is: 0.022836
Iteration   17   Total error is: 0.022894
Iteration   18   Total error is: 0.022806
Iteration   19   Total error is: 0.022861
Iteration   20   Total error is: 0.022826
Iteration   21   Total error is: 0.022841
Iteration   22   Total error is: 0.022773
Iteration   23   Total error is: 0.022868
Iteration   24   Total error is: 0.02275
Iteration   25   Total error is: 0.022819
Iteration   26   Total error is: 0.022712
Iteration   27   Total error is: 0.022864
Iteration   28   Total error is: 0.022715
Iteration   29   Total error is: 0.022833
Iteration   30   Total error is: 0.022712
Iteration   31   Total error is: 0.022827
Iteration   32   Total error is: 0.022663
Iteration   33   Total error is: 0.022809
Iteration   34   Total error is: 0.02269
Iteration   35   Total error is: 0.022804
Iteration   36   Total error is: 0.022666
Iteration   37   Total error is: 0.022778
Iteration   38   Total error is: 0.022653
Iteration   39   Total error is: 0.022818
Iteration   40   Total error is: 0.022665
Iteration   41   Total error is: 0.022817
Iteration   42   Total error is: 0.022647
Iteration   43   Total error is: 0.02284
Iteration   44   Total error is: 0.02264
Iteration   45   Total error is: 0.022825
Iteration   46   Total error is: 0.022667
Iteration   47   Total error is: 0.022815
Iteration   48   Total error is: 0.022665
Iteration   49   Total error is: 0.022823
Iteration   50   Total error is: 0.022668
```

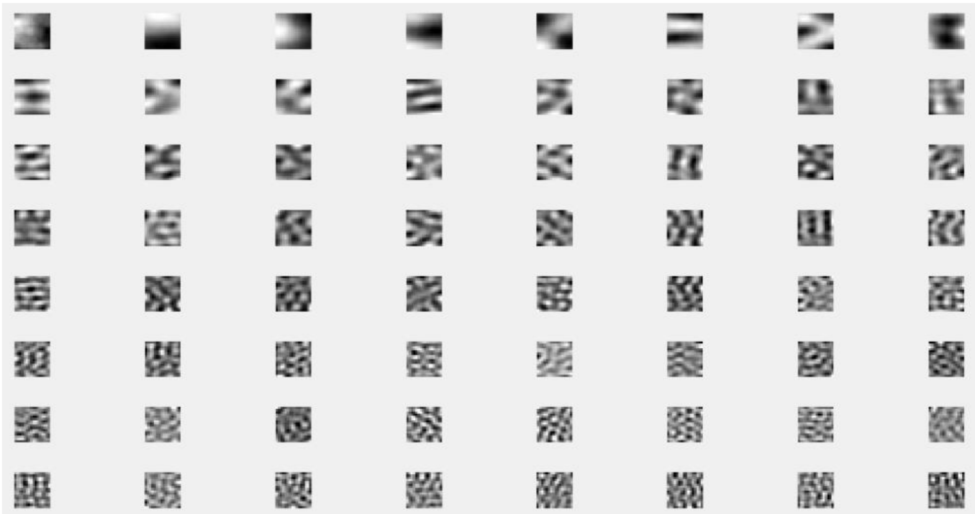
KSVD errors in each iteration

```
PCA error is 0.0937
```

As a result, the error of PCA is about 0.09488, and the error of KSVD is about 0.022668



Visualize final dictionary (350-d)



Top 64 eigen vectors

(b) Code for majorization function:

```
function [ result ] = majDic( x,W,z,epsilon,lambda )
oldobj = 0;
for i=1:10
    u = abs(z);
    for n = 1:1000
        z(:,n) = pinv(W'*W+lambda*diag(1/(2*u(:,n))))*W'*x(:,n);
    end
    sum = 0;
    for j = 1:1000
        for k = 1:350
            if u(k,j) < epsilon
                sum = sum + (z(k,j)^2+u(k,j)^2)/(2*epsilon);
            else
                sum = sum + (z(k,j)^2+u(k,j)^2)/(2*u(k,j));
            end
        end
    end
    E = norm(x-W*z)^2;
    F = lambda*sum;
    obj = E+F;
    if mod(100,i)==0
        disp(["epcho = ",i," obj = ",obj])
    end
    if abs(obj - oldobj)<epsilon
        break;
    end
    oldobj = obj;
end
result = z;
end
```

Code for SPAMS:

```
param.K=350;  % learns a dictionary with 100 elements
param.lambda=0.15;
param.numThreads=4; % number of threads

param.iter=100;  % let us see what happens after 100 iterations.

tic
D = mexTrainDL(X,param);
t=toc;
fprintf('time of computation for Dictionary Learning: %f\n',t);

fprintf('Evaluating cost function...\n');
alpha=mexLasso(X,D,param);
R=mean(0.5*sum((X-D*alpha).^2)+param.lambda*sum(abs(alpha)));
ImD=displayPatches(D);
subplot(1,3,2);
imagesc(ImD); colormap('gray');
fprintf('objective function: %f\n',R);
```

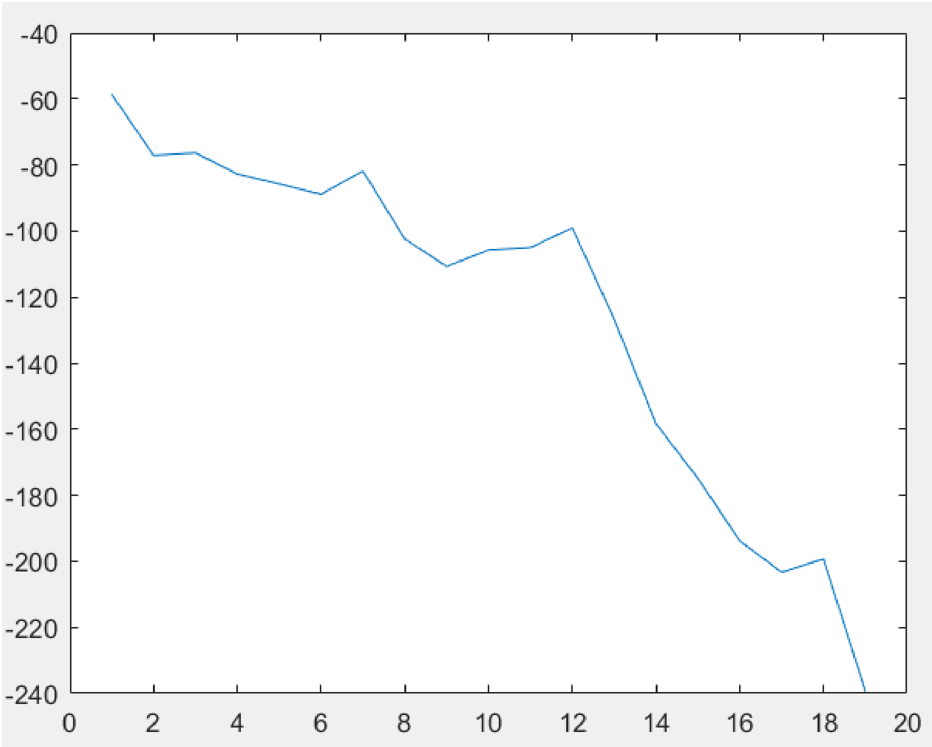
```
>> spams
time of computation for Dictionary Learning: 3.288415
Evaluating cost function...
objective function: 0.224341
```

Result from SPAMS

By **z_SPAMS = D'*X;** we can calculate the z vectors that produced by SPAMS.

The L2 distance between z vectors from my function and the vectors I get from SPAMS is around 1

2. Clustering using mixture models



GMM negative log-likelihood

As we can see. The "complete data" negative log-likelihood objective function is decreased.

According to the result of GMM and K-Means. We find that GMM can make data fitter than K-Means. In K-Means, there is much data get together in one cluster. However, GMM makes data spread beautifully, we can see that points are uniform scattered on the clusters.