DEPARTMENT OF ELECTRICAL AND COMPUTER ENGINEERING, UCLA
ECE 211A: DIGITAL IMAGE PROCESSING I

**INSTRUCTOR:** Prof. Achuta Kadambi
**TA:** Yunhao Ba

**NAME:** Guy Ohayon
**UID:** 205461330

# HOMEWORK 2: SPARSE CODING AND DICTIONARY LEARNING

| PROBLEM | TOPIC | MAX. POINTS | GRADED POINTS | REMARKS |
|---|---|---|---|---|
| 2.1 | Training and Testing Images | 1.0 | | |
| 3.1 | Image Patches | 1.0 | | |
| 3.2 | Random Dictionary | 1.0 | | |
| 3.3 | Reconstruction Function | 1.0 | | |
| 3.4 | Reconstruction Result | 1.0 | | |
| 4.1 | K-SVD Implementation | 2.0 | | |
| 4.2 | K-SVD Dictionary | 1.5 | | |
| 4.3 | K-SVD Reconstruction | 1.5 | | |
| | **Total** | 10.0 | | |

# 1 Problem Setup

In sparse coding, the goal is to find a sparse representation $\mathbf{x}$ of the input data $\mathbf{y}$ with a dictionary $\mathbf{A}$. The whole process can be characterized by the following optimization problem:

$$\begin{aligned} &\min_{\mathbf{x}} \|\mathbf{A}\mathbf{x} - \mathbf{y}\|_2 \\ &s.t. \|\mathbf{x}\|_0 \leq S. \end{aligned} \tag{1}$$

Orthogonal Matching Pursuit (OMP) is one of the common methods to solve the above sparse coding problem, and the dictionary $\mathbf{A}$ can be learned using the K-SVD algorithm. In this homework, we will use these two algorithms for an image inpainting task, where our goal is to remove the extra text content on an image. Similarly, we will use PSNR and SSIM as the metrics to evaluate the reconstruction performance in this homework.

# 2 Image Preparation

First, we need to prepare some images for the inpainting task. Here are the procedures:

1. Pick two images from the BSDS500 dataset[1], and convert them to gray scale. Crop a $256 \times 256$ region from each of the selected images. Denote these two image as $y_{train}(n_1, n_2)$ and $y_{test}(n_1, n_2)$.
2. Put some text on $y_{test}(n_1, n_2)$. Denote the obtained image as $y_{test\_missing}(n_1, n_2)$.
3. Based on $y_{test\_missing}(n_1, n_2)$, generate a mask map to indicate whether a pixel in $y_{test\_missing}(n_1, n_2)$ is missed or not. Denote this mask as $m_{test\_missing}(n_1, n_2)$.

[1] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

### 2.1 Training and Testing Images (1.0 points)

Plot $y_{train}(n_1,n_2)$, $y_{test}(n_1,n_2)$, $y_{test\_missing}(n_1,n_2)$, and $m_{test\_missing}(n_1,n_2)$. Report PSNR($y_{test}$, $y_{test\_missing}$) and SSIM($y_{test}$, $y_{test\_missing}$).

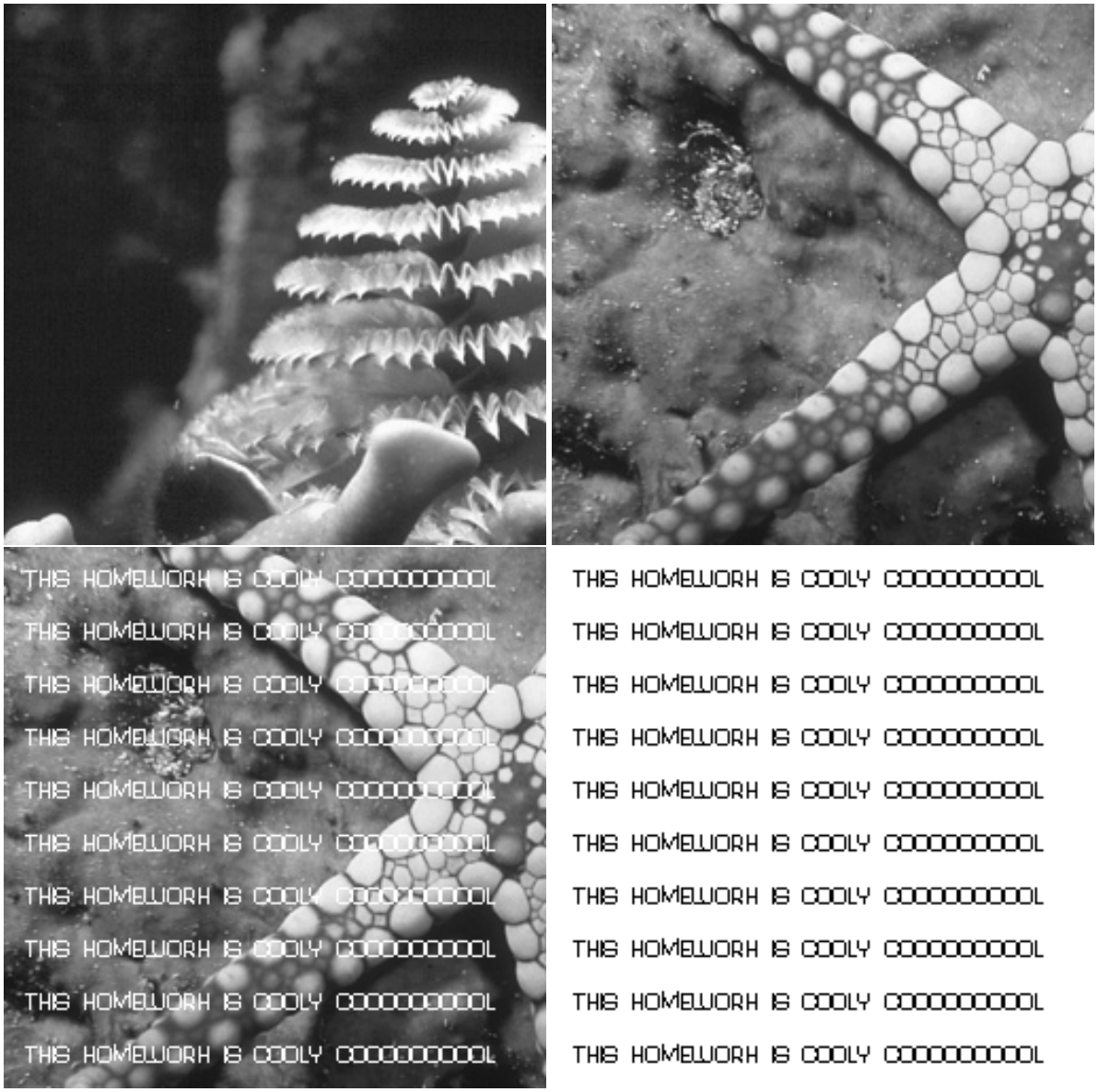


Figure 1: Insert images of $y_{train}(n_1,n_2)$, $y_{test}(n_1,n_2)$, $y_{test\_missing}(n_1,n_2)$, and $m_{test\_missing}(n_1,n_2)$ (replace our example).
$PSNR(y_{test}, y_{test\_missing}) = 15.25291392854199$
$SSIM(y_{test}, y_{test\_missing}) = 0.6323040957575629$

# 3 Random Dictionary Reconstruction

## 3.1 Image Patches (1.0 points)

Considering the dimension of images, it is common to split the images into smaller block patches in sparse coding problems. Split $y_{test\_missing}(n_1, n_2)$ into multiple block patches of size $8 \times 8$ pixels. How many non-overlapping block patches can you get? Plot any 25 block patches extracted together with their associated masks.

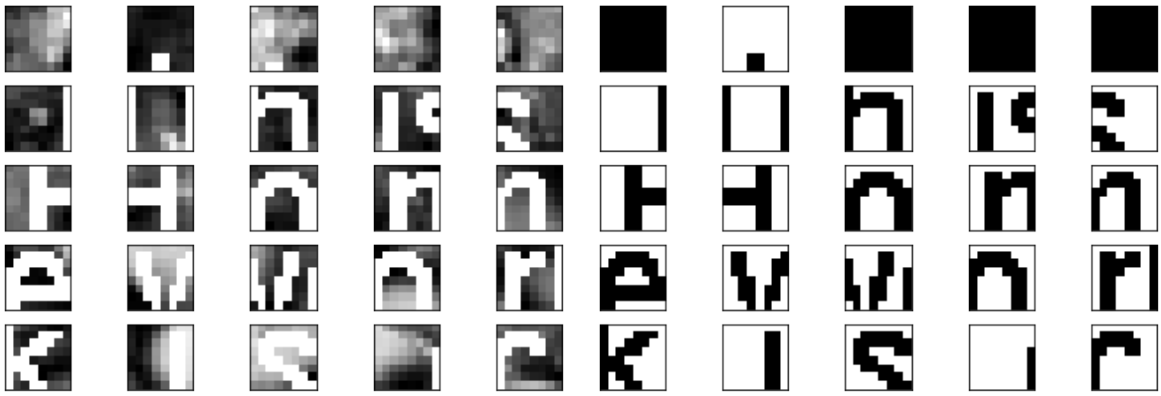

Figure 2: Insert the plotted patches and their masks (replace our example).

I can get a total of 1024 non-overlapping block patches, since $\frac{256 \cdot 256}{8 \cdot 8} = 1024$.

### 3.2 Random Dictionary (1.0 points)

To recover the missing pixels using OMP, we need a dictionary $\mathbf{A}_{random}$. In this question, you need to create a random dictionary with 512 atoms. What is the size of this dictionary? Plot any 25 atoms of $\mathbf{A}_{random}$.

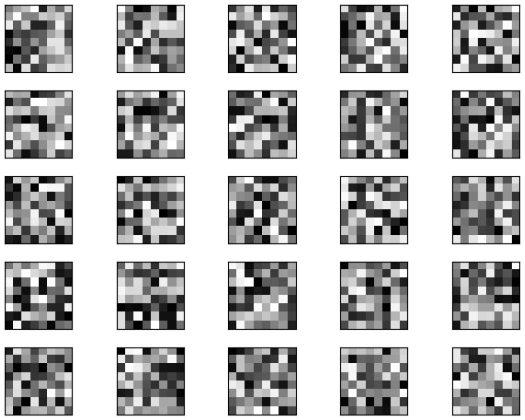

Figure 3: Insert the image of different atoms (replace our example).

Our dictionary contains 512 atoms, where each atom is a vector of size $8 \cdot 8 = 64$. Thus, the dictionary size is $64 \cdot 512 = 32,768$.

### 3.3 Reconstruction Function (1.0 points)

Implement a function to reconstruct a block patch using OMP. You can use the OMP function in scikit-learn package[2]. Make sure all the atoms in the dictionary have unit norm before using this function, and the maximal number of non-zero elements $S$ is 20. Place your code in the box below.

```
#OMP
from sklearn.linear_model import OrthogonalMatchingPursuit
def OMP_reconstruction(mask_patches,
                       y_patches, dictionary, n_nonzero_coefs=20, num_of_patches=1024,
                        result_img_shape=(256, 256)):
    omp = OrthogonalMatchingPursuit(n_nonzero_coefs=n_nonzero_coefs, fit_intercept=False)
    reconstructed_y = np.zeros(shape=result_img_shape)
    for patch_index in range(num_of_patches):
        patch_col = int(patch_index % 32)
        patch_row = int(patch_index / 32)
        bool_mask = mask_patches[patch_index].flatten().astype(bool)
        y_relevant_info_patch = y_patches[patch_index].flatten()[bool_mask]
        omp.fit(dictionary[bool_mask,:], y_relevant_info_patch)
        coef = omp.coef_
        y_predict = dictionary.dot(coef)
        reconstructed_y[patch_row * 8:patch_row * 8 + 8,
                        patch_col * 8:patch_col * 8 + 8] = y_predict.reshape(8, 8)
    return reconstructed_y
```

[2] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.orthogonal_mp.html

### 3.4 Reconstruction Result (1.0 points)

Use your reconstruction function in Section 3.3 to reconstruct all the block patches of $y_{test\_missing}(n_1, n_2)$. Concatenate these reconstructed patches together to form $\hat{y}_{test\_random}(n_1, n_2)$. Plot $\hat{y}_{test\_random}(n_1, n_2)$, and report PSNR($y_{test}$, $\hat{y}_{test\_random}$) and SSIM($y_{test}$, $\hat{y}_{test\_random}$).

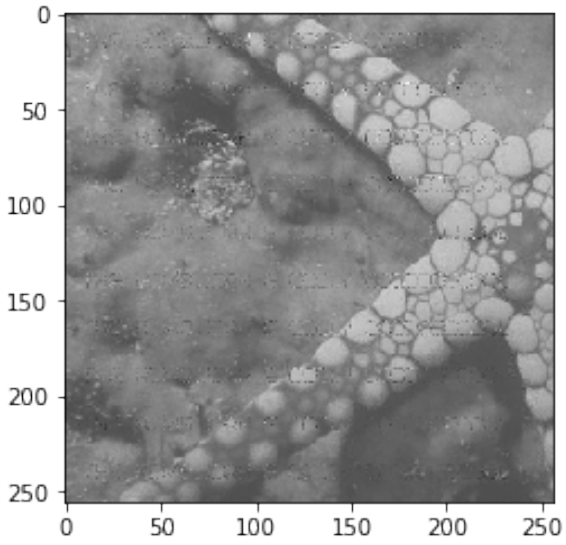


Figure 4: Insert the image of $\hat{y}_{test\_random}(n_1, n_2)$ (replace our example).

$PSNR(y_{test}, \hat{y}_{test\_random}) = 24.791504848779937$
$SSIM(y_{test}, \hat{y}_{test\_random}) = 0.8156806182557758$

# 4 K-SVD

## 4.1 K-SVD Implementation (2.0 points)

Split $y_{train}(n_1, n_2)$ into multiple block patches of size $8 \times 8$ pixels, and implement your own K-SVD algorithm to learn a dictionary $\mathbf{A}_{ksvd}$ from block patches of $y_{train}(n_1, n_2)$. The size of $\mathbf{A}_{ksvd}$ should be the same as the size of $\mathbf{A}_{random}$. In your implementation, you can use the OMP function in scikit-learn and SVD function in NumPy[3]. The maximal number of non-zero elements $S$ is still 20. Place your code in the box below.

```
# n_samples = number of pixels
# n_features = number of atoms
# n_targets = number of training examples
# targets shape=(n_samples, n_targets)
def KSVD(targets, n_samples=64, max_iter=15, n_nonzero_coefs=20,
         thres=1e-6, n_features=512):

    omp = OrthogonalMatchingPursuit(n_nonzero_coefs=n_nonzero_coefs, fit_intercept=False)
    D = np.random.randn(n_samples, n_features)

    for i in range(max_iter):
        omp.fit(D, targets)
        coef = omp.coef_.T #shape=(n_features, n_targets)
        y_predict = D.dot(coef) #shape=(n_samples, n_targets)
        err = np.linalg.norm(targets - y_predict)
        if err < thres:
            break
        # Update dictionary
        for j in range(n_features):
            I = coef[j, :] != 0
            if np.sum(I) == 0:
                continue
            E_k = targets[:, I] - D.dot(coef[:, I]) + np.outer(D[:, j], coef[j, I])
            U, S, V = np.linalg.svd(E_k)
            D[:, j] = U[:, 0]
            coef[j, I] = S[0] * V[:, 0]

    #Normalize D:
    D /= np.linalg.norm(D, axis=0)[np.newaxis, :]
    return D
```

Note that I normalized D only at the end of KSVD because it led to better results without normalizing in the beginning.

[3] https://docs.scipy.org/doc/numpy/reference/generated/numpy.linalg.svd.html

### 4.2 K-SVD Dictionary (1.5 points)

Sort the atoms of $\mathbf{A}_{ksvd}$ based on their standard deviation in ascending order. Plot the first 25 atoms in the box below.

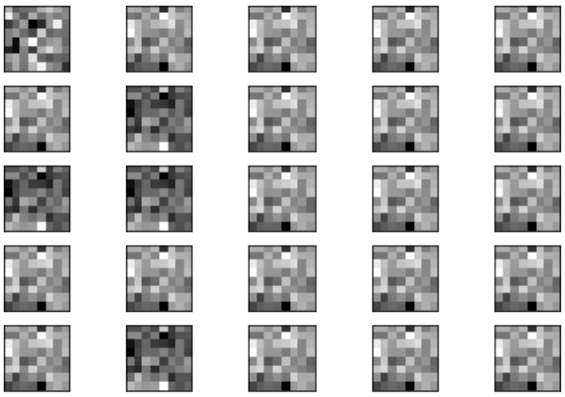

Figure 5: Insert the image of atoms with small standard deviation (replace our example).

### 4.3 K-SVD Reconstruction (1.5 points)

Use $\mathbf{A}_{ksvd}$ to reconstruct all the block patches of $y_{test\_missing}(n_1, n_2)$. Concatenate these reconstructed patches together to get $\hat{y}_{test\_ksvd}(n_1, n_2)$. Plot $\hat{y}_{test\_ksvd}(n_1, n_2)$, and report PSNR($y_{test}$, $\hat{y}_{test\_ksvd}$) and SSIM($y_{test}$, $\hat{y}_{test\_ksvd}$).

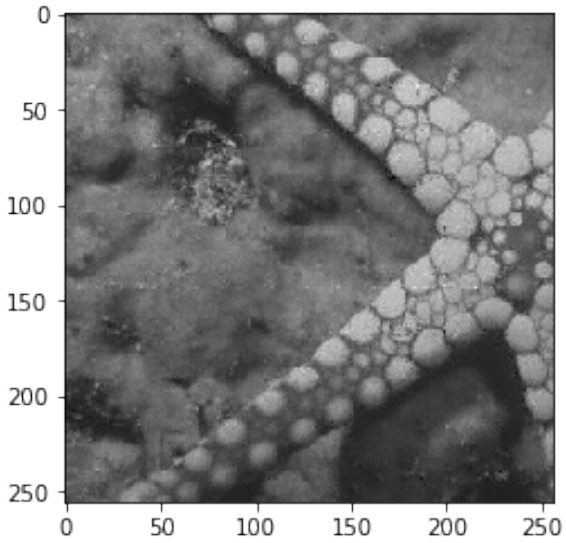


Figure 6: Insert the image of $\hat{y}_{test\_ksvd}(n_1, n_2)$ (replace our example).

$PSNR(y_{test}, \hat{y}_{test\_ksvd}) = 27.129191978582995$
$SSIM(y_{test}, \hat{y}_{test\_ksvd}) = 0.8507635116094617$