18-660: Project 2: Image Recovery

Akshay Rajhans
arajhans@ece.cmu.edu

***Abstract*— In this project, we apply compressed sensing techniques to recover full image from a small number of sampled pixels. We seek a sparse solution that approximates the discrete cosine transform (DCT) coefficients for an image. This sparse solution gives us a small number of 'sample' coefficients that we then use to (approximately) reconstruct the original image using inverse DCT. This sparse solution is obtained using *Orthogonal Matching Pursuit (OMP)*, which minimizes the ℓ_1 norm of the error made by the sparse solution in approximating a given vector of ground-truth values. The parameter λ, which is the number of non-zero entries returned by OMP algorithm is chosen using randomized cross-validation. Experimental results are presented for cases where five different number of sampled pixels are chosen.**

I. Overview

In this project, we study the problem of recovering an image from its discrete cosine transform (DCT) coefficients, hereafter transform coefficients. The image can be uniquely reconstructed from given transform coefficients by applying inverse DCT. For each image pixel located at coordinates (x,y), the greyscale intensity $g(x,y)$ is given by

$$g(x,y) = \sum_{u=1}^{P}\sum_{v=1}^{Q} \alpha_u \cdot \beta_v \cdot \cos\frac{\pi(2x-1)(u-1)}{2P} \cdot \frac{\pi(2y-1)(v-1)}{2Q} \cdot G(u,v) \tag{1}$$

where $G(u,v)$ is the transform domain coefficient at coordinates (u,v) in the transformed image; and P and Q are the image width and height respectively. If the transform domain coefficients $G(\cdot,\cdot)$ can be arranged in a column vector α[1], then each pixel intensity $g(\cdot,\cdot)$ is a linear combination of the transform coefficients in the vector α. If the pixel intensities are arranged in a column vector c, we can write the above equation as

$$[c]_{PQ\times 1} = [T]_{PQ\times PQ} \times [\alpha]_{PQ\times 1} \tag{2}$$

However, for this project, we consider that only a few samples, say S, are available from the actual image. In other words, only a few rows sampled from vector c are available, which we call vector b. We can use only those rows from the matrix T for which the b values are available. We call such a matrix formed by selecting these rows from T as matrix A. Therefore, Eq. 2 becomes the following under-determined system of equations:

$$[b]_{S\times 1} = [A]_{S\times PQ} \times [\alpha]_{PQ\times 1} \tag{3}$$

We solve this under-determined sysetm of equations and get the "best" solution $\hat{\alpha}$ for the vector α, which we then use to estimate the vector $\hat{c} = T \cdot \hat{\alpha}$ using equation 2. This vector $[\hat{c}]_{PQ\times 1}$ when reshaped into a $P \times Q$ matrix gives us the pixel intensities of the recovered image.

For this project, we define "best" as the best sparse solution that has at most λ nonzero entries such that it minimizes the ℓ_2-norm of the error. In other words, we get the α that solves the following minimization problem with ℓ_1 regularization.

$$\min_{\alpha} ||A\cdot\alpha - b||_2^2$$
$$s.t. ||\alpha||_1 \leq \lambda$$

In this project, we use *Orthogonal Matching Pursuit (OMP)* to find an efficient[2] solution to the above minimization problem. The OMP implementation is explained in Section II-A. The right value for λ is determined using randomized cross-validation, which is explained in Sec. II-B. Sec. III presents the results obtained during this exercise for different values of S and Sec. IV concludes this report by mentioning some discussions about the project.

II. Mathematical Formulation

In this project, we seek a sparse solution to the vector of DCT coefficients. In a big image, since there is a lot of information, a lot of DCT coefficients can be of comparable magnitude. Therefore, using a spase solution, we may lose a lot of information. However, locally, in small-enough sections of an image, there is not a whole lot of information. Therefore, the "DC coefficient (top left corner)" and a few "low-frequency coefficients" (mainly concentrated around top left corner) in the transform domain are of significance and others are of fairly small magnitude.

To leverage this fact, we split the image in $K \times K$ blocks of pixels on which we use the reconstruction algorithm. Once we have these recovered "sub-images", we stitch them to form a complete recovered image. Here K was set to be 8.

A. Orthogonal Matching Pursuit

Orthogonal Matching Pursuit (OMP) tries to find the λ most correlated columns of a given A matrix in order to match a given vector b. Overall, the algorithm follows these steps:

1) Set $f = b$, $\Omega = \{\}$ and $p = 1$.
2) Calculate the inner product values $\theta_i = < A_i, f >$
3) Identify the index s for which $|\theta_s|$ takes the largest value
4) Update Ω: $\Omega = \Omega \cup \{s\}$

[1] This α has nothing to do with the α_u in Eq. 1.

[2] but possibly sub-optimal

5) Approximate f by the linear combination of $\{A_i; i \in \Omega\}$ $\min_{\alpha_i, s.t. i \in \Omega} ||\Sigma_{i \in \Omega} \alpha_i \cdot A_i - b||_2^2$
6) Update f: $f = b - \Sigma_{i \in \Omega} \alpha_i \cdot A_i$
7) • If $p < \lambda$, $p = p + 1$ and go to Step 2.
   • Otherwise, stop.
8) $\alpha_i = 0$ $(i \notin \Omega)$

The minimization problem in step 5 can be efficiently solved in Matlab by using the 'backslash' command, e.g.,

```
alpha(Omega) = A(:,Omega)\ b;
```

where `Omega` is a vector that has the indices in Ω so far.

B. Finding the right λ by cross-validation

In this project, we use randomized averaged cross-validation, which is performed as follows.

We iterate the values of λ from 1, $1 + \Delta\lambda$, $1 + 2\Delta\lambda, \cdots$.[3] For each λ, we do the following process:

For each sub-image, we select S samples for doing OMP. Of these S samples, we set aside $m = \text{floor}(S/6)$ samples as a test set and we use the remaining $\text{ceil}(5S/6)$ samples for the actual training set, i.e. send these samples to the OMP algorithm described above. Once we get the sparse solution α from the OMP algorithm, we measure the mean-squared error made by the recovered values of the test set w.r.t the actual values of the test set. We repeat this process $M = 20$ times and record the average mean-squared error over these 20 tries.

Then we increment the λ values in steps of $\Delta\lambda$ and repeat the process for the remaining value of λ. In the end, we choose λ^* that has the least average mean-squared error.

Then we again sample S points one last time, and use the OMP algorithm with $\lambda = \lambda^*$ and get $\hat{\alpha}$ which has at most λ^* non-zero entries. We reconstruct the sub-image from these $\hat{\alpha}$ values and reshape the resulting image vector into an image matrix. We do this exercise for all sub-images and finally stitch together all the recovered sub-images to form the complete recovered image.

This experiment is repeated five times, with S taking values 10, 20, 30, 40 and 50 respectively.

III. Experimental Results

The recovered images for the five different experiments are shown in Fig. 1 through 5.

A. Qualitative analysis of the results

This mainly visual analysis tells us that as S increases, the quality of reconstruction improves. For the $S = 10$ case, the reconstructed image looks somewhat 'patchy'. This is an artifact of the fact that the value of λ^* for a lot of the sub-images turns out to be 1. Therefore, the OMP selects only 1 best DCT frequency that best represents that block. Since the size of the test set for this case is that of a single randomly selected pixel of of the 64 (averaged over 20 trials), the λ^* value returned by the randomized cross-validation may not

[3] We keep $\Delta\lambda = 1$, so that we get to try out all possible λ values. Setting $\Delta\lambda$ to an integer value > 1 might result in a speed-up, but the best λ may not be found. See Sec. IV-C.2 for a discussion.

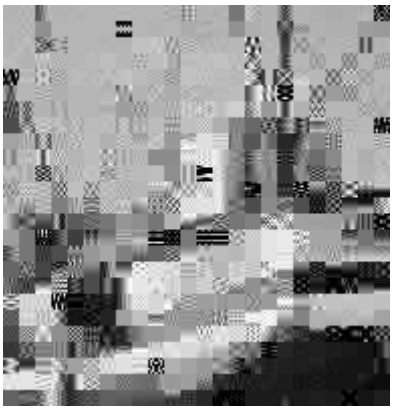

Fig. 1. Recovered image for S = 10

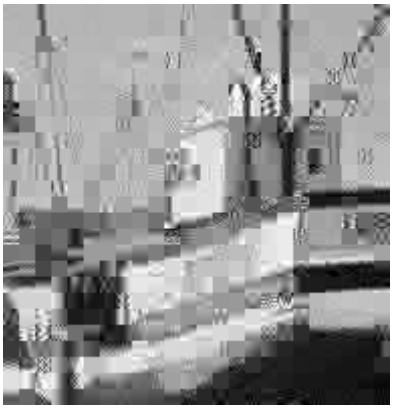

Fig. 2. Recovered image for S = 20

Fig. 3. Recovered image for S = 30

Fig. 4. Recovered image for S = 40

Fig. 5. Recovered image for S = 50

be be the true best value. Furthermore, the OMP technique can inherently be sub-optimal. This all adds up in case of those blocks where this choice of 1 single DCT frequency can be a grossly inappropriate reconstruction of the blocks.

As the S value increases, the visual quality of the reconstructed image noticably becomes better and better. Each image is progressively less patchier than the earlier one. Fig. 5 depicts the best possible reconstruction of the image, which is for the case $S = 50$.

B. Quantitative analysis of the results

We can quantify the quality of the recovered images using the following formula. The recovery error $\mathscr{E}$ is defined as

$$\mathscr{E} = \frac{1}{P \times Q} \sum_{u=1}^{P} \sum_{v=1}^{Q} (\hat{g}(x,y) - g(x,y))^2 \tag{4}$$

where, $\hat{g}(\cdot,\cdot)$ are the pixels of the recovered image, while $g(\cdot,\cdot)$ are the pixels of the original image. This quantitative analysis is summarized in Table I, where $\mathscr{E}$ is shown in the last column. The interpretation of the recovery error $\mathscr{E}$ is that it is the average squared error per pixel in the original intensity and the recovered intensity.

TABLE I

TABLE SUMMARIZING THE QUALITY OF RECONSTRUCTION

S	Test Set Size ($m = ceil(S/6)$)	Training Set Size ($S-m$)	% Training Set $(S-m)/(K*K)$	Error Metric ($\mathscr{E}$)
10	1	9	14.06 %	22.0783
20	3	17	26.56 %	0.0355
30	5	25	39.06 %	0.0562
40	6	34	53.12 %	0.0024
50	8	42	65.62 %	0.0001

First row depicts the case $S = 10$, where only 14.06% of pixels are available for training set, the average squared error per pixel is 22.07; which on a scale of 0-255 is still less than 10%. As we go down the table, the percentage of pixels available increases, and the average squared error per pixel drops pretty shaprly. For the best case, where $S = 50$ pixels are selected out of every 64, and of these, 42 are used for training set, the average squared error per pixel is as low as 0.0001, which on the scale of 0-255 is an error of 3.9216e-05%. This result is quite remarkable!

IV. DISCUSSION

A. Factors that may impact quality of the recovered image

1) Number of samples selected: As is shown in case of Fig. 1 through 5, the quality of reconstruction varies with the number of samples selected for the training and test sets.

2) λ value selected for OMP: In our algorithm, the best λ value, i.e. λ^* is determined using randomized cross-validation, for every single sub-image of size $K \times K$. For $K = 8$, there are 25×24, i.e. 600 sub-images.

Table II summarizes the results for the average value of λ^*. In general, λ can take any value from 1 through $S-m$, which were the values tried out during the randomized cross-validation. Due to the lack of space in this paper, we report only the average value of λ^* per reconstruction experiment[4]. The average is calculated over the 600 sub-images for each experiment. What this means, for example in case of the last row in the table, is that although the α vector can have as many as 42 non-zero entries, about 8 non-zero entries are sufficient to reconstruct the image fairly well.

This λ^* fed to the OMP algorithm determines the number of non-zero entries returned in every sparse solution by the OMP algorithm. As shown in the table, the average value of λ^* grows as we go down the last column. This has a correlation with improved reconstruction. The higher the λ^*, the better the reconstruction.

3) Different values for K, m, M: Changing the value of K from 8 is likely to affect the quality of reconstruction. There is a tradeoff between increasing and decreasing the K value. If we increase the K value, it will be the case that not a lot of DCT coefficients will be zero. Therefore, when we look

[4]Bar graphs capturing the frequency of occurence of different values of λ^* in these 600 sub-images were also created. They can be found in the .zip file.

TABLE II

TABLE SUMMARIZING THE AVERAGE λ^* FOUND BY RANDOMIZED CROSS-VALIDATION

S	Test Set Size $(m = ceil(S/6))$	Training Set Size $(S-m)$	% Training Set $(S-m)/(K*K)$	Avg. λ^*
10	1	9	14.06 %	3.9550
20	3	17	26.56 %	3.4000
30	5	25	39.06 %	4.3133
40	6	34	53.12 %	6.4733
50	8	42	65.62 %	8.4150

for a λ-dimensional sparse solution, i.e. selecting λ non-zero DCT coefficients, there will be higher loss of accuracy. On the other hand, if we reduce the K value, we will not get enough sample points for the randomized cross-validation.

Increasing the value of m would mean that a higher percentage of sample points will be set aside for testing. This will be particularly useful for the $S = 10$ case where we currently have just a single pixel. But what that means is that fewer percentage of sample points will be available for training set, i.e., for OMP computation.

Increasing the value of M will have the effect of yielding better averaging of error results for the randomizd cross-validation. Lowering this number would give a coarser approximation of the average error, and in turn possibly worse reconstruction, but will imrove speed.

B. Limits or problems of the current approach

1) Nested `for` loops: The biggest problem with the current implementation is too many `for` loops in the code. While the result is good, it takes a long time to compute because Matlab is not optimized for `for` loops.

In the current approach, we have a for loop at the very top level that is for five different values of S, as follows:

```
for iS = 1:5
    recover_arajhans1(iS);
    ...
end
```

where the function `recover_arajhans1` has the following nested `for` loop.

```
T = compute_transform_coefficients(...);
...
for nImage = 1:nP*nQ
  ...
  for lambda = 1:S-m
    ...
    alpha = OMP_arajhans(...);
    ...
  end
end
```

Here, the top-most loop is done for each sub-image and there are `nP*nQ`, i.e. 600 of these. Once we ge the transform coefficients, which in itself is another series of nested `for` loops, we need to solve the OMP for every λ from `1` to `S-m`.

TABLE III

COMPUTATION TIME FOR THE FIVE EXPERIMENTS

S	10	20	30	40	50
Time in minutes	1.04	2.88	6.96	13.72	21.88

Computation of the transform coefficients is nothing but doing the computation in Eq. 1 $P \times Q$ times, which has the following form.

```
for x = 1:P
    for y = 1:Q
        ...
        for u = 1:P
            for v = 1:Q
                ...
                T(iRow,iCol) = ...;
            end
        end
    end
end
```

In this dumbest implementation, the computation `T(iRow,iCol) = ...;` is done 20,480 number of times.[5] The OMP computation is done a minimum of 27000 times for the $S = 10$ case, and a maximum of 126000 times for the $S = 50$ case.[6] Furthermore, each instance of OMP does its computation to find λ best non-zero entries in a vector, and therefore has a `for` loop that runs λ times, minimum being 1 and maximum being 42.

The time taken by the `recover_arajhans1` function for the five values of S is summarized in Table III.

C. Possible improvements that can be done

1) Vectorize the code for speed-up: Although we get correct results using nested `for` loops, it is inefficient. In order to improve the efficiency of the code, the nested `for` loops need to be converted into matrix-vector multiplications, for which Matlab is optimized. However, how to do this is not trivial since the resulting coefficients are not only a function of u and v, but also of x and y. And a cosine is needed *after* these functions of u, v, x and y are computed. Even the other than the computations are not easily amenable to getting rid of the `for` loops.

2) Not checking every single λ: Another place where to speed up the computation would be to not check every single λ while doing the cross-validation. Even if we checked every other λ instead of every λ, we save on up to 21 $\times$ 42 computations. However, there is a tradeoff between the efficiency improved by not checking every λ and the accuracy of the result, since we'll only get an approximate $\hat{\lambda}^*$ instead of λ^* if we didn't check every single λ during the cross-validation.

[5] 5*8*8*8*8 = 20,480.

[6] 5*600*9 = 27000 and 5*600*42 = 126000.