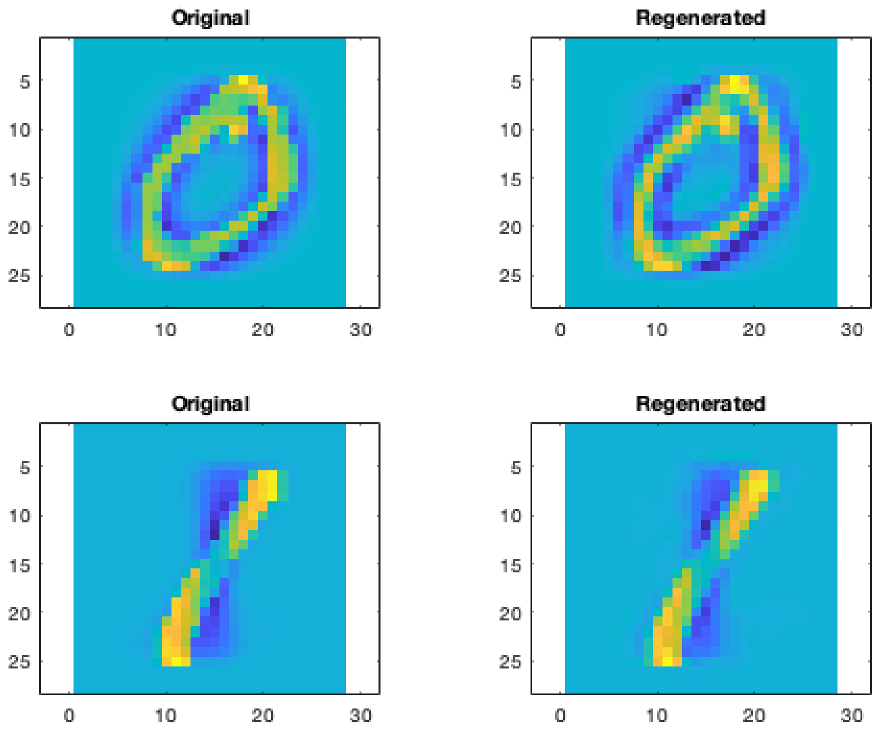
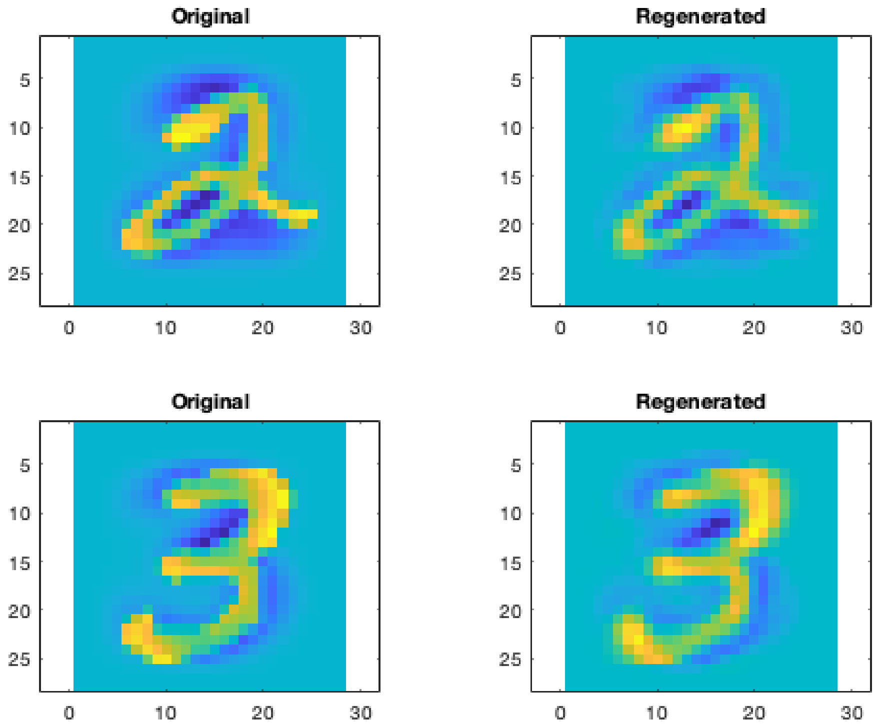
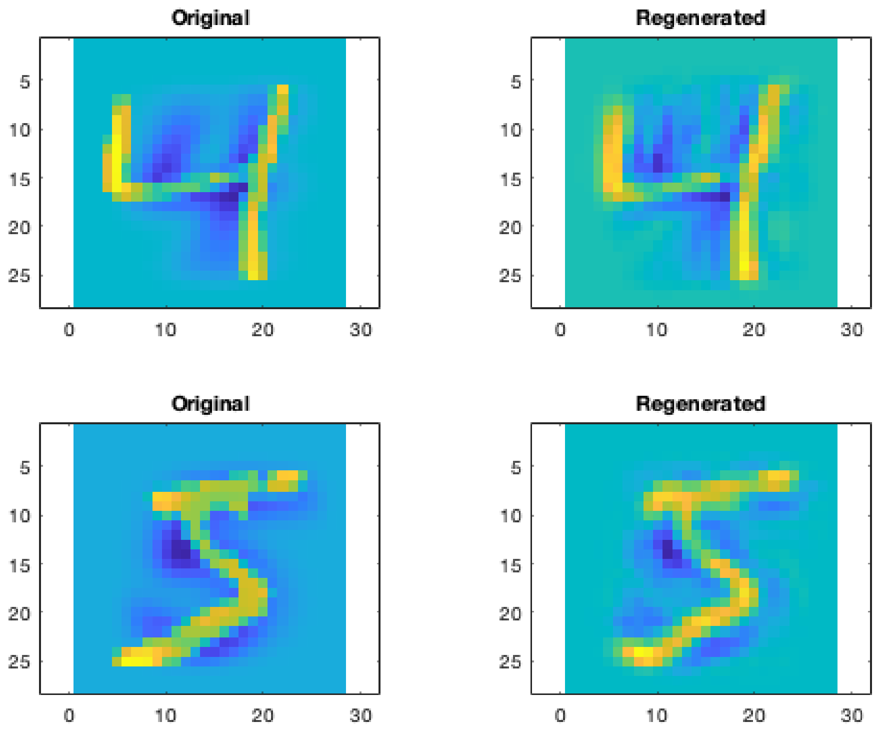
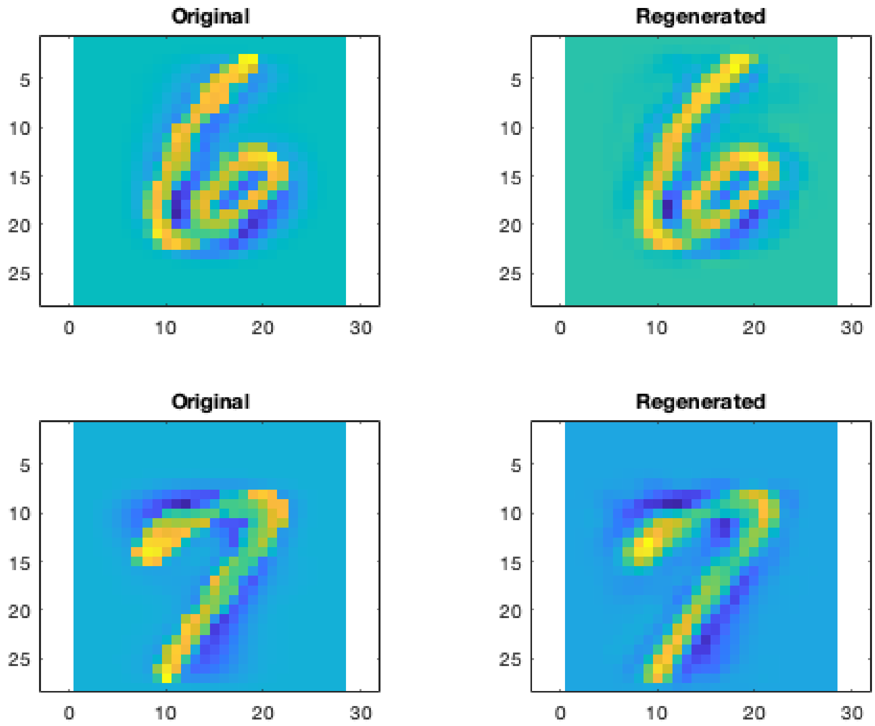
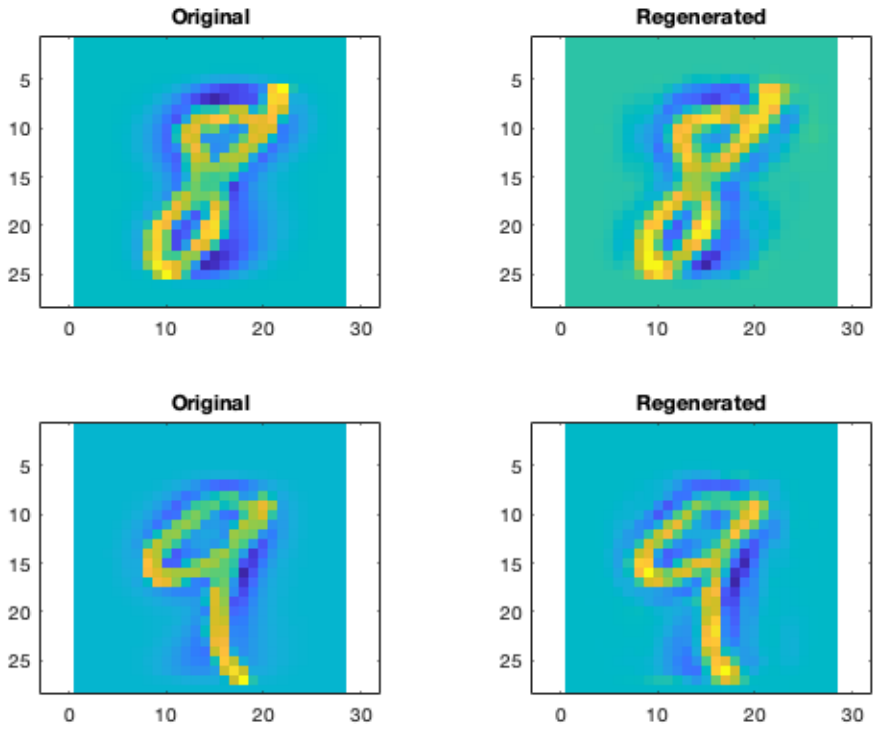
```
clear;
close all;
S = load('../data/mnist.mat');

digits = double(S.digits_train);
sample_size = size(digits, 3);                              % 60000
digits = reshape(digits, 784, sample_size);                 % digits is now a 784x60000 matrix
labels = transpose(S.labels_train);                         % labels is a 1x60000 row

Eigvecs = zeros(784, 84, 10);                               % To store the 84-D basis for each digit
reduced_images = zeros(84, sample_size);                    % To store the 84 coordinates for each image

for i = 0:9
    indices = find(labels == i);                            % extracting indices of current digit
    current_digit = digits(:, indices);
    N = size(current_digit, 2);                             % Number of samples
    mean = sum(current_digit, 2)/N;                         % 784x1 vector
    current_digit = current_digit - mean;
    C = current_digit * current_digit' / max(1, N-1);       % 784x784 covariance matrix

    [U, S] = eig(C);
    [D, ind] = sort(diag(S), 'descend');
    Us = U(:, ind);                                         % U in sorted order

    Eigvecs(:, :, i+1) = Us(:, 1:84);                       % 84-D basis for digit i
    reduced_images(:, indices) = Eigvecs(:, :, i+1)' * current_digit;

    % Now using reduced_images to regenerate the image
    original_image = reshape(current_digit(:, 1), 28, 28);
    regenerated_image = reshape(Eigvecs(:, :, i+1) * reduced_images(:, indices(1)), 28, 28);

    if mod(i, 2) == 0
        figure;
        tiledlayout(2, 2);
    end

    nexttile;
    imagesc(original_image);
    axis equal;
    title('Original');

    nexttile;
    imagesc(regenerated_image);
    axis equal;
    title('Regenerated');
end
```


Original
Regenerated
Original
Regenerated
5
10
15
20
25
0
10
20
30


Original
Regenerated
Original
Regenerated
5
10
15
20
25
0
10
20
30


Original
Regenerated
5
10
15
20
25
0
10
20
30


Original
Regenerated
5
10
15
20
25
0
10
20
30



Published with MATLAB® R2020a