Summarizing Visual Data Using Bidirectional Similarity

Team name : LMTG
Project-id:12
Project link : https://github.com/Computer-Vision-IIITH-2021/project-lmtg

Mohit pavan : 2018102016
Tejaswini :2018102018
Gowthami :2018102048
Lakshmi madhuri :2018101116

ALGORITHM(1/3):

- The intention of the paper is to perform image summarization by attempting to reduce redundant data in the image.
- A good visual summary should contain as much visual information as possible from the original image and shouldn’t have any other elements included in it because of the process.
- The logic of this algorithm is it to give image and dimensions of the summarized image we want as the input and get the corresponding output to those dimensions by processing the image in blocks and downscaling it after every block until we get the dimensions of the image equal to dimensions given as input.
- Paper uses bidirectional similarity to calculate the distance measure and try to minimize it to remove the redundant information.

ALGORITHM-(2/3)

- There are 4 main steps of this algorithm which happens in every block in the whole process
  - To take target image as a resized version of source image
  - Find the patches which are bidirectionally similar
  - Update the pixels in the target image with the use of formula given
  - Downscale this image by fixed value for every block until we get the final output.
- The target image according to the requirements of the algorithm should a visually similar image so that when we attempt to find the patches which are bidirectionally similar we get the patches corresponding to patches in source which are not completely different from source image.
- The pixels of the target image are updated based on the formula for each pixel which is obtained after differentiating the distance measure thereby minimizing it which is also the idea of image summary.

ALGORITHM(3/3)

- The iterative rule for first block is as follows(coherence):-
  - We take target image as resized image and calculate the bidirectionally similar patches. Let Q_1 , . . . , Q_m denote all the patches in T that contain pixel q. (for every pixel q, it can be a part of overlapping patches where pixels occupies the place of each coordinate) Let P_1 , . . . , P_m denote the corresponding (most similar) patches in S (i.e., distance between respective patches is minimum). Let p_1 , . . . , p_m be the pixels in P_1 , . . . , P_m corresponding to the position of pixel q within Q_1 , . . . , Q_m(which take the coordinate of pixel q in patch Q_i in the patch P_i corresponding to it)
  - We measure the error by giving weights with which colors of pixels in P votes for colors of pixels in Q which is denoted by Nt.This corresponds to phenomenon of Coherence
  - Error term is as follows:- $d_{\text{cohere}}(S,T)$ is $\frac{1}{N_T}\sum_{i=1}^{m}(S(p_i)-T(q))^2$

- The iterative rule for first block is as follows(Completeness):-
  - We have taken target image as resized image of source image. Let Q̃ 1 , . . . , Q̃ n denote all the patches in T that contain pixel q and serve as “the most similar patch” to some patches P̃1 , . . . , P̃ n in S (i.e., every patch of source is taken and every patch has some corresponding “most similar patch” in target image).
  - Let p̂ 1 , . . . , p̂ n be the pixels in patches P̃1 , . . . , P̃ n corresponding to the position P n of pixel q within Q̃ 1 , . . . , Q̃ n(we pick the pixels in P^i which has same coordinates of q in corresponding Q^i)
  - We measure the error by giving weights with which colors of pixels in Q votes for colors of pixels in P which is denoted by Ns.This corresponds to phenomenon of Completeness
  - The error term is

$$d_{\text{complete}}(S,T) \qquad \frac{1}{N_S}\sum_{j=1}^{n}(S(\hat{p}_j)-T(q))^2$$

- Overall error term is

$$\mathrm{Err}(T(q)) = \frac{1}{N_S}\sum_{j=1}^{n}(S(\hat{p}_j) - T(q))^2 + \frac{1}{N_T}\sum_{i=1}^{m}(S(p_i) - T(q))^2$$

- We find the update formula for each pixel by differentiating this error term. Update rule for each pixel is as follows:-

$$T(q) = \frac{\frac{1}{N_S}\sum_{j=1}^{n} S(\hat{p}_j) + \frac{1}{N_T}\sum_{i=1}^{m} S(p_i)}{\frac{n}{N_S} + \frac{m}{N_T}}$$

Important notes

- It is important to note that in iterative rule for coherence, we take target image and for every patch Q_i we find the most similar patch in source P_i whereas in iterative rule for completeness, we first find most similar patch for each patch in source S in the target image T and then for every patch Q^i one particular pixel is present, we reverse map and get all the patches P^i in source whose most similar patch was Q^i.
- Another important part of this process is, for the first block in the process we give a resized image of the source image but for next blocks in the process, we downscale the output of previous block by fixed amount every time until we get an summarized image of required dimensions.

RESULTS

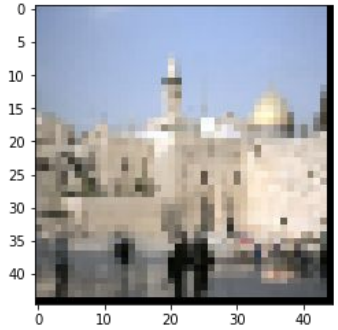
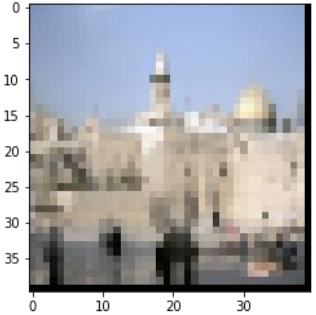
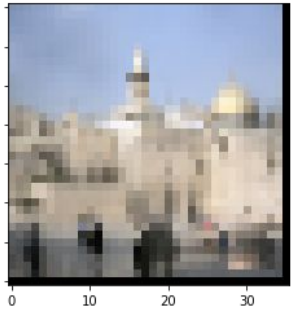
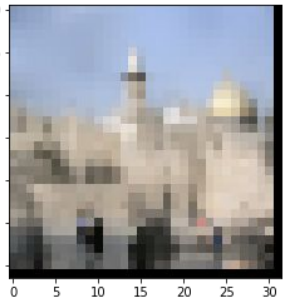
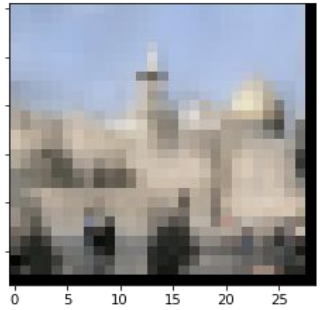
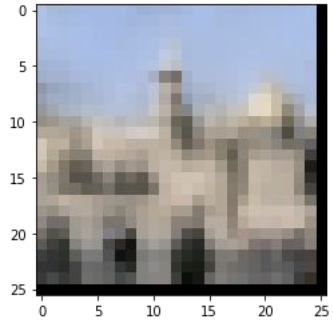
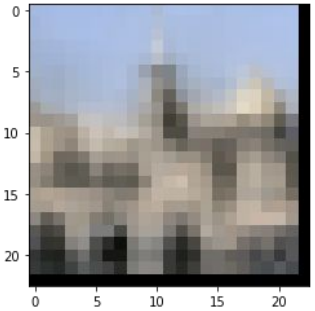
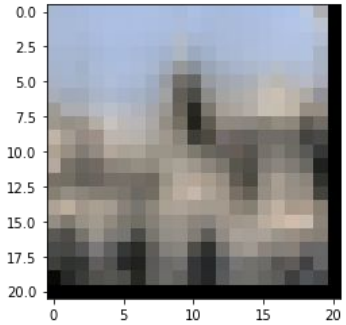
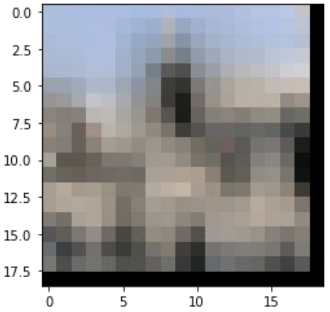
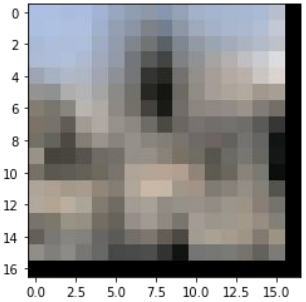











Image resized to : 50x50
The resizing factor for every iteration : 0.9

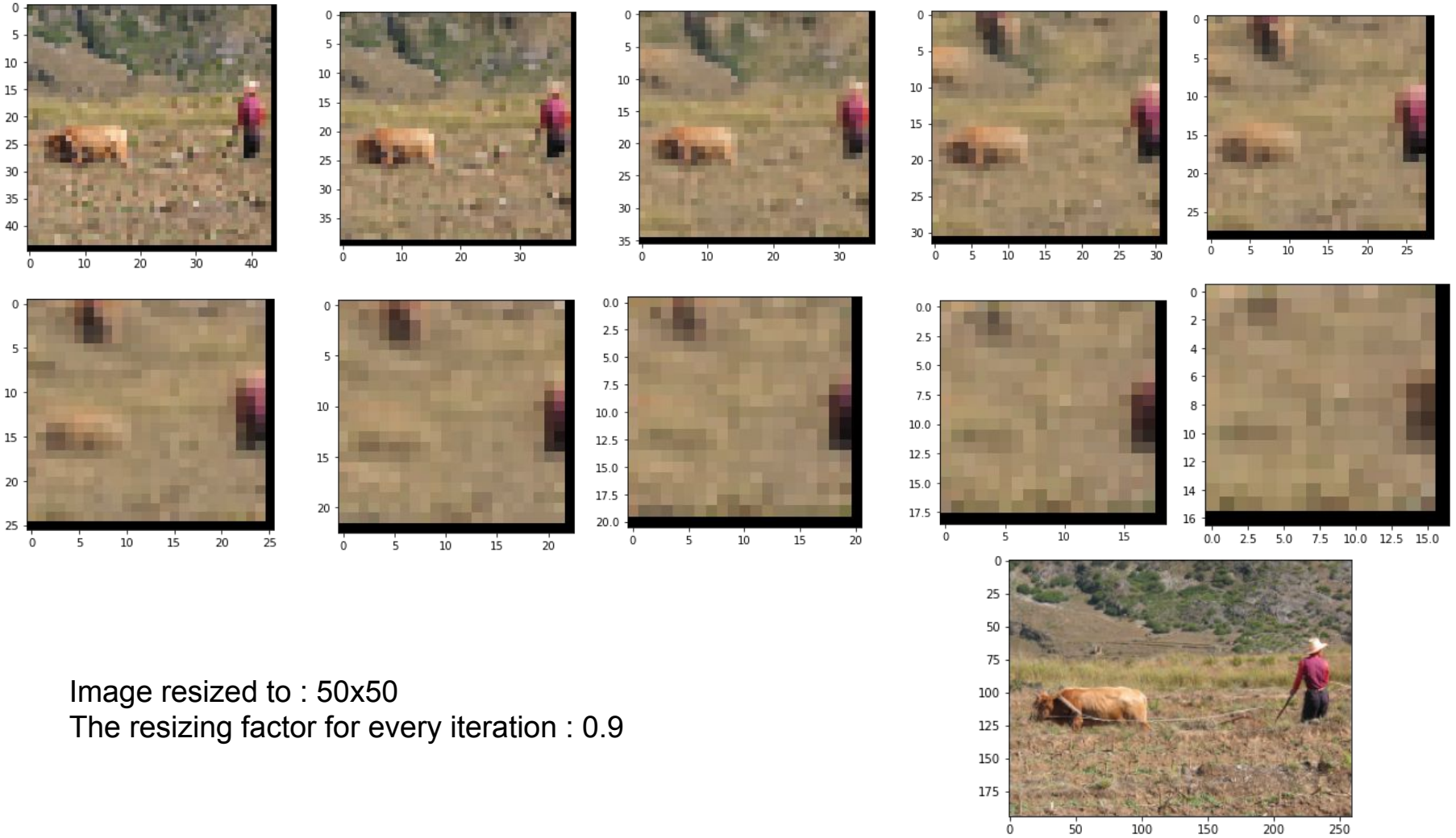
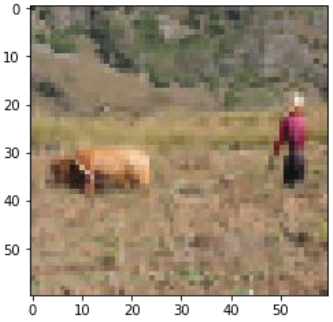


Image resized to : 50x50
The resizing factor for every iteration : 0.9



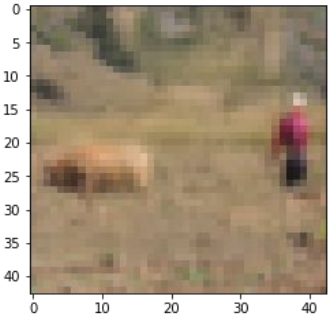
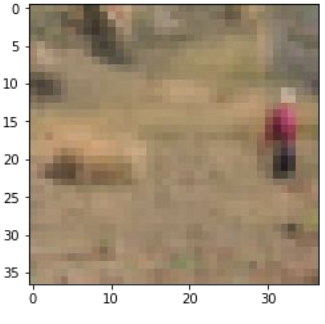
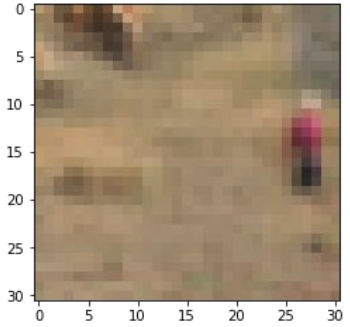
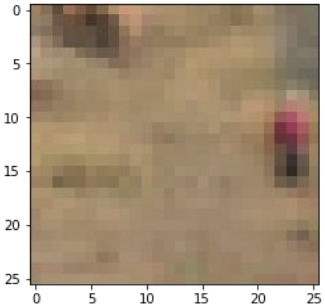
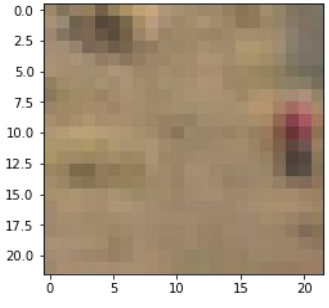
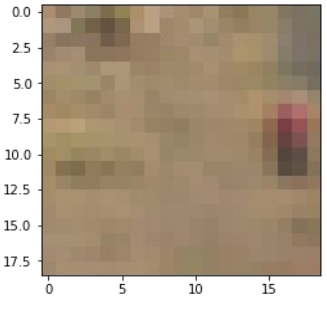
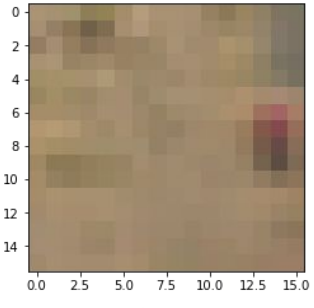
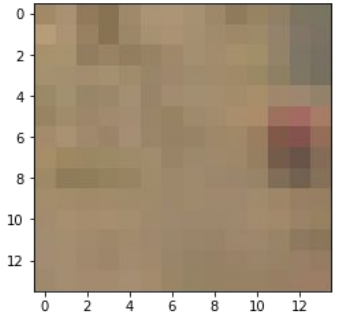










Image resized to : 70x70
The resizing factor for every iteration : 0.85

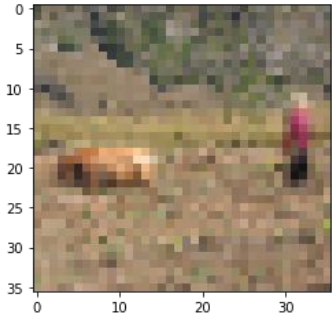
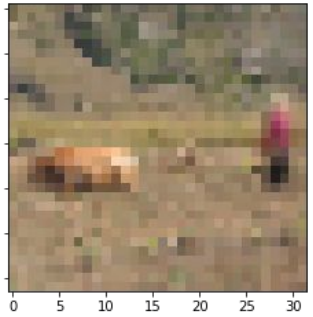
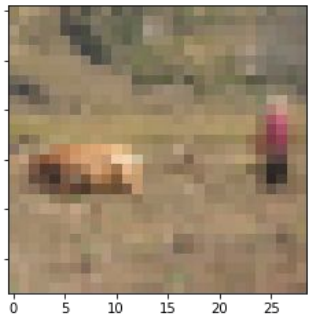
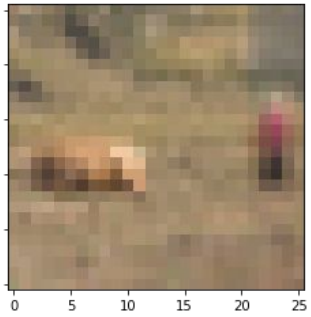
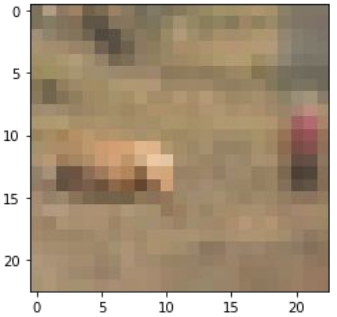
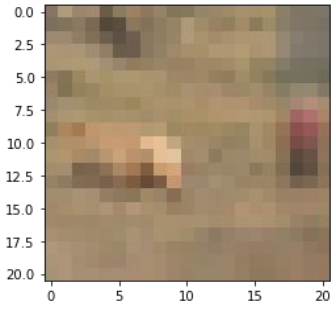
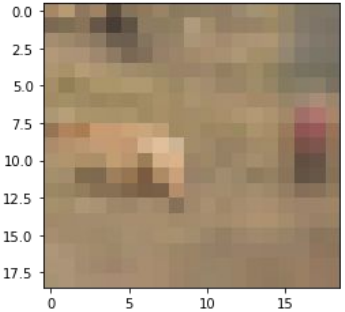
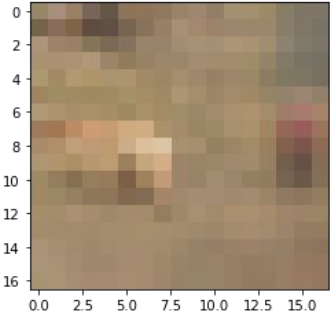
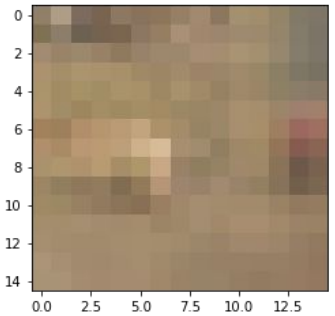
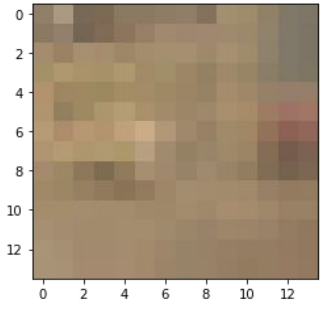












Image resized to : 40x40
The resizing factor for every iteration : 0.85

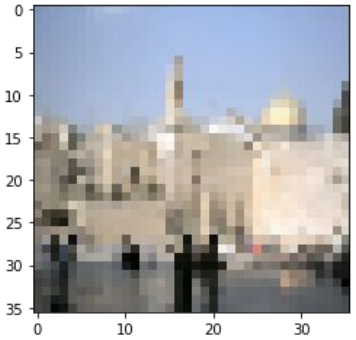
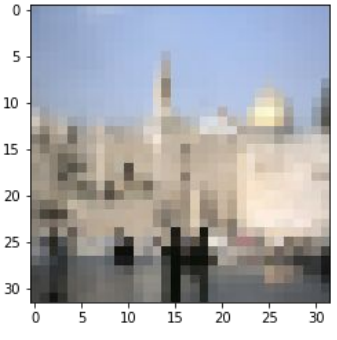
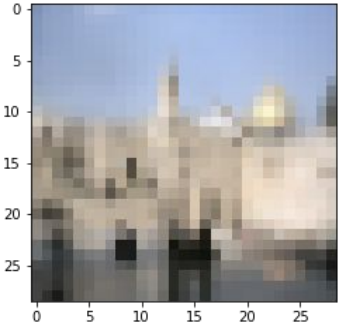
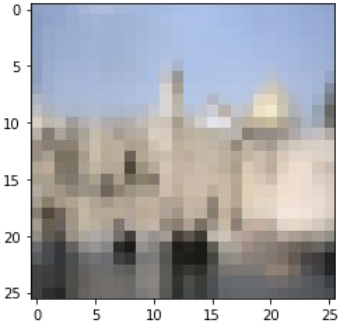
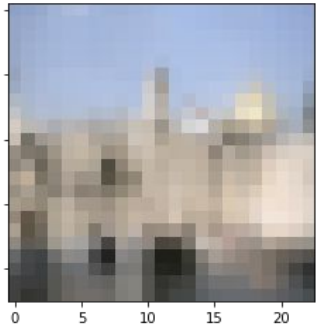
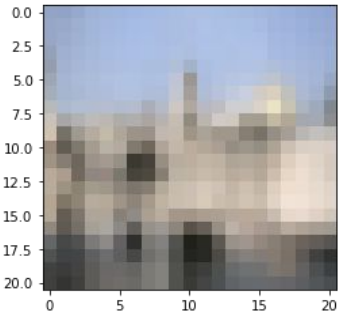
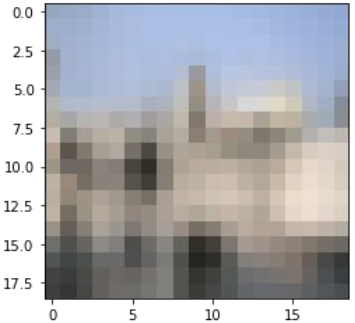
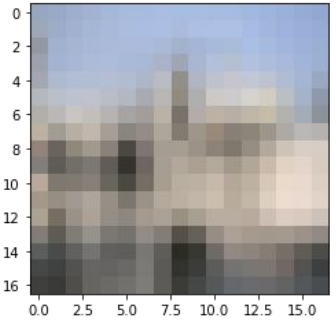
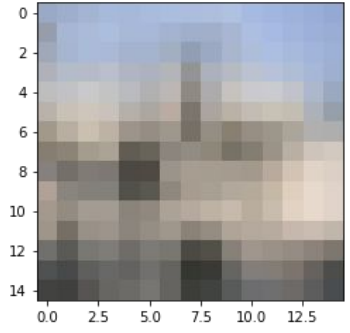
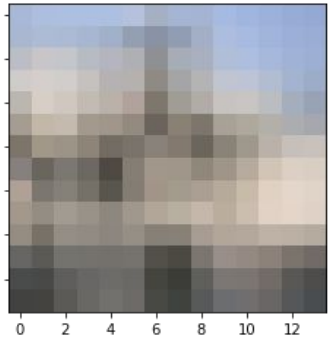
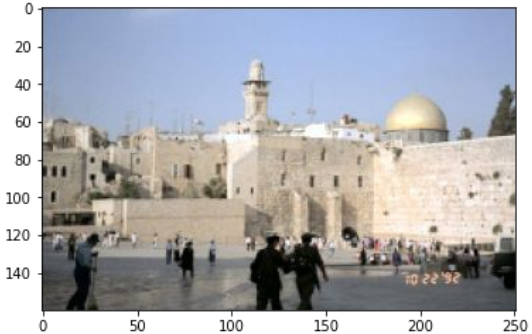
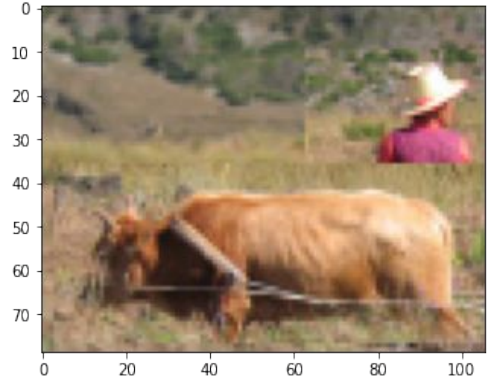












Image resized to : 40x40
The resizing factor for every iteration : 0.85

Results of another approach(using interactive gui)



To be improved by final evaluation:

1. Image quality and improvement of results.
2. Speed up the code
3. Making report on different sets of images
4. Try to make 2-3 applications if possible .

THANK YOU