SHINDE SHUBHAM SUNIL
email: me18b183@smail.iitm.ac.in
Course: BT6270 - Computational Neuroscience
Instructor: Prof. V. Srinivasa Chakravarthy

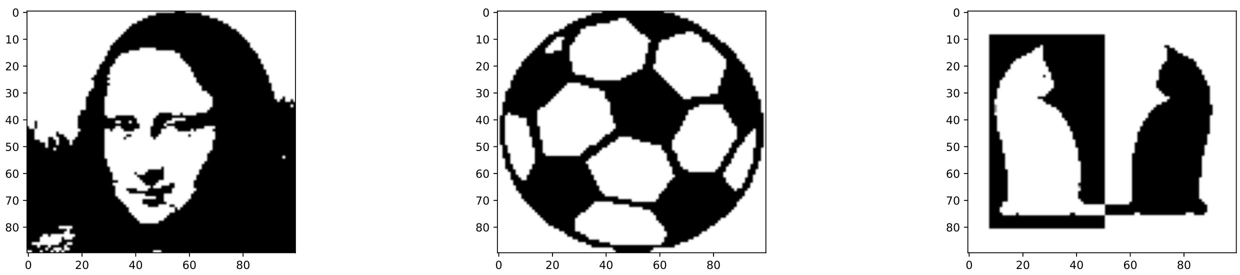
ASSIGNMENT #3
Roll No.: ME18B183
Semester: JUL - NOV 2021
Due Date: 9^{th} November, 2021

Problem 1
Three figures (mona, ball, cat) are given in *.txt* format. Each figure is a 90×100 matrix.

(a) Visualize the images and make sure that the black pixels are represented by -1 and white pixels are represented by +1.

(b) Develop a code for **Hopfield Network** with N=9000 neurons which are fully connected.

Solution:
(a) All three *.txt* files are read and visualised.



(a) Mona Lisa (b) Ball (c) Cat

Figure 1: Three images read and visualised from respective *.txt* file.

(b) The Python code of a fully connected **Hopfield Network** (*hopfield_network.ipynb*) with N=9000 neurons is attached in the submission folder.

```python
class Hopfield_Network():
    def __init__(self,niter):
        self.weights = np.zeros((9000,9000))
        self.V = np.zeros((9000,1))
        self.U = np.zeros((9000,1))
        self.U_d = np.zeros((9000,1))
        self.rmse = np.zeros((niter,1))
        self.timekeeper = np.zeros((niter,1))
        # for leading images
        self.tracker = 0

    def weights_loader(self,image_reshape):

        if self.tracker==1:
            print('Loading the images')
            self.weights = np.matmul(mona_reshape,mona_reshape.T) + np.matmul(ball_reshape,ball_reshape.T)
            + np.matmul(cat_reshape,cat_reshape.T)
        if self.tracker==0:
            print('Loading the image')
            self.weights = np.matmul(image_reshape,image_reshape.T)

    def image_loader(self,image):

        new_image = np.zeros((90,100))
        new_image[0:45,20:65] = image[0:45,20:65]
        return new_image

    def damage_weights(self,p):

        # p is the probability with which weights are damaged

        indcs = np.random.randint(0,9000*9000-1,int(9000*9000*p))
        damaged_weights = np.copy(self.weights)
        damaged_weights = np.reshape(damaged_weights,(9000*9000,1))
        print('Altering the weights')
        for i in tqdm_notebook(range(len(indcs))):
            damaged_weights[indcs[i]] = 0

        damaged_weights = np.reshape(damaged_weights,(9000,9000))
        return damaged_weights
```

Figure 2: A snippet of code from the *hopfield_network.ipynb* file

Problem 2
Save the image of ball in the network.

(a) Initialize a zero matrix of the same size as that of the input image and replace a small patch with a portion of the input image as shown in Figure 3. Use this as the cue for retrieving the image.

(b) Plot the patch which is given as the input trigger.

(c) Plot the Root Mean Squared (RMS) Error with time.

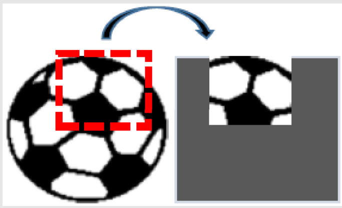


Figure 3: Image of the ball

Solution:
(b) The input image to the Hopfield Network:

Figure 4: Input image patch of the ball

(c) The root mean squared error plot for retrieving the final image from the network.

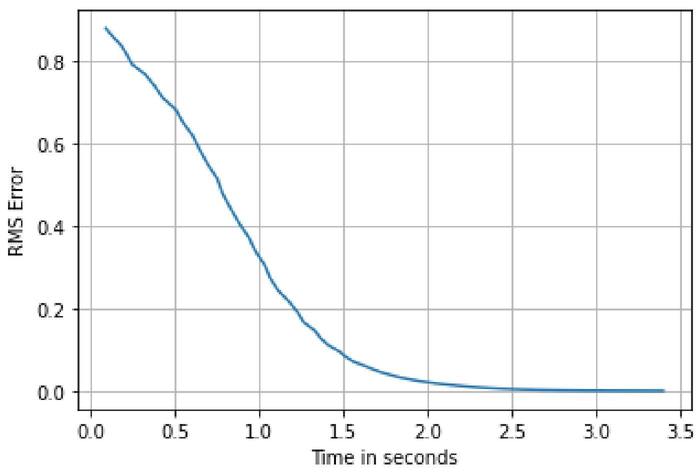
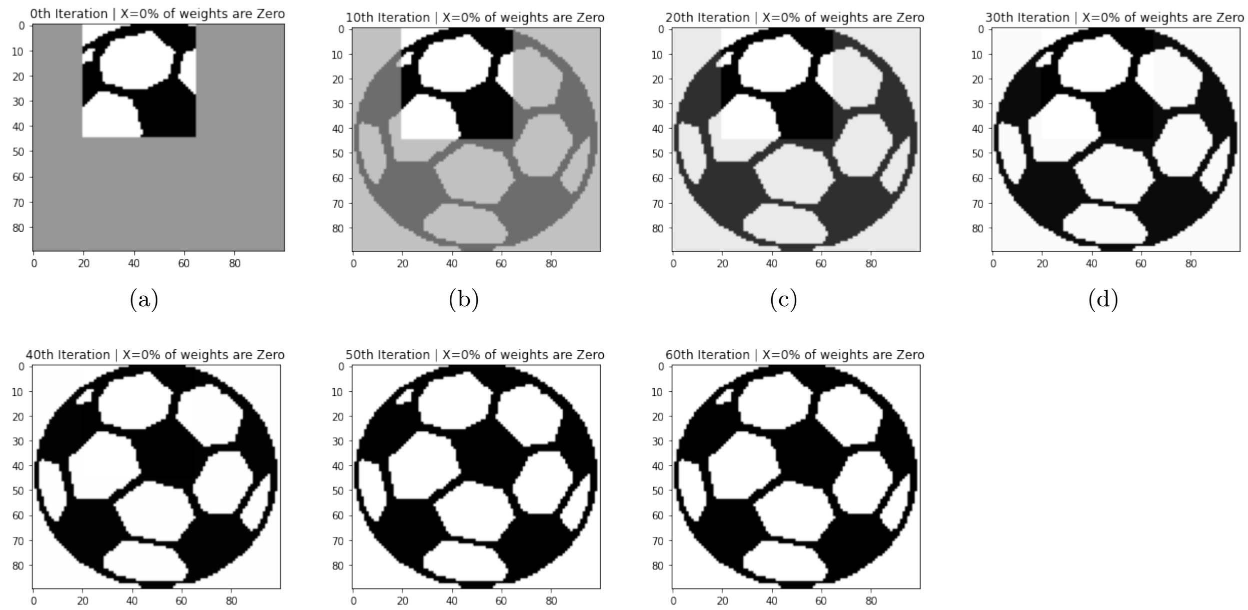


Figure 5: RMSE v/s Time (in seconds)



(a) (b) (c) (d) (e) (f) (g)

Figure 6: A small patch with a portion of the input image as shown in Figure 6(a) is being used as a cue for retrieving the ball image.

> **Problem 3**
> Save all three images (mona, ball and cat) in the network.
>
> (a) Give small patches of each image to retrieve the corresponding saved image.
>
> (b) Plot the RMS error with time and the final retrieved image for all three inputs.
>
> (c) Make X% of weights to be zero and repeat questions 3(a) and 3(b) for X=25%, X=50% and X=80%.
>
> (i) Plot the RMS error with time for each case.
>
> (ii) Plot the final retrieved image for each case.

Solution:
(a) The following image patches are used for retrieving the final image using the network.

(a) Input image patch of Mona Lisa

(b) Input image patch of the ball

(c) Input image patch of the cat

Figure 7: Input image patches

(b) Mona Lisa

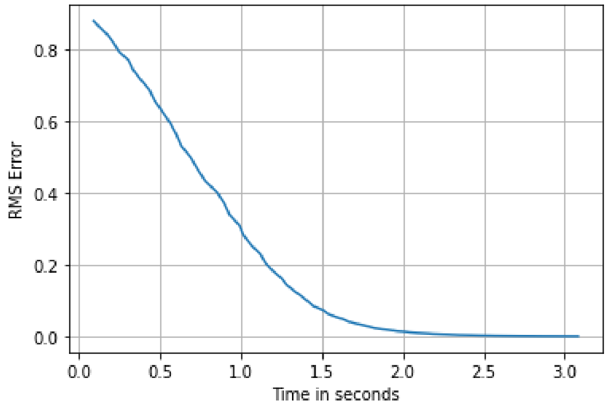
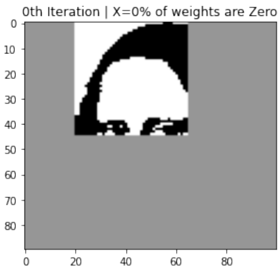
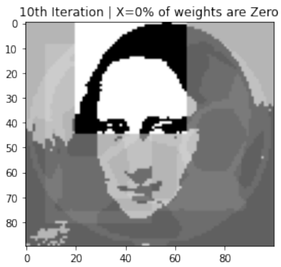


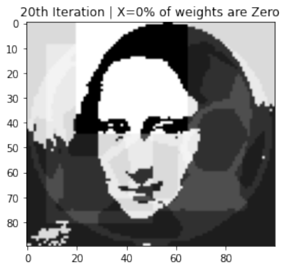
Figure 8: RMSE v/s Time (in seconds) for image *mona*



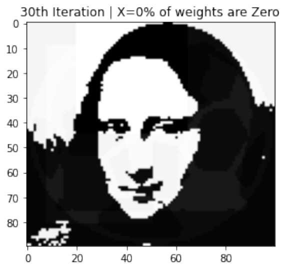
(a) Initial Input



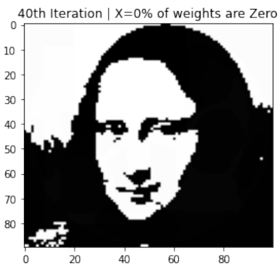
(b) 10^{th}iteration



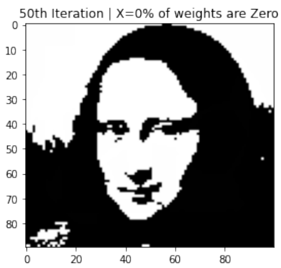
(c) 20^{th}iteration



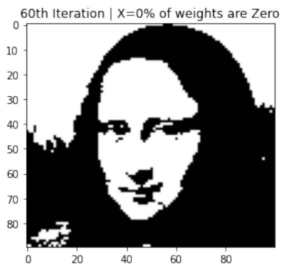
(d) 30^{th}iteration



(e) 40^{th}iteration



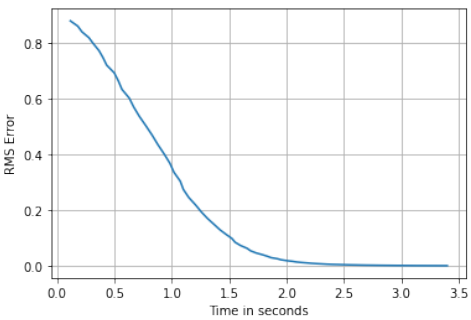
(f) 50^{th}iteration



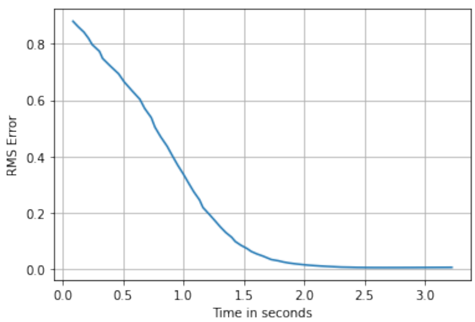
(g) Final Image

Figure 9: A small patch with a portion of the input image as shown in Figure 9(a) is being used as a cue for retrieving the *mona* image.

Similarly, for **Ball and Cat**

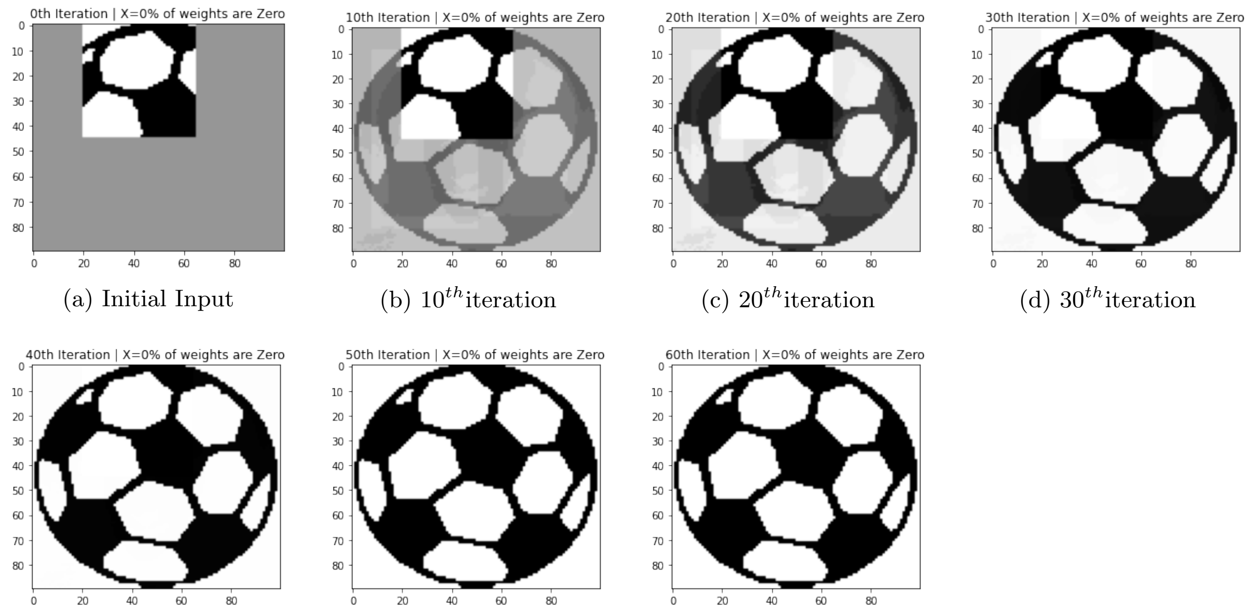


(a) For ball image



(b) For cat image

Figure 10: RMSE v/s Time (in seconds) plots for image *ball* and *cat*.

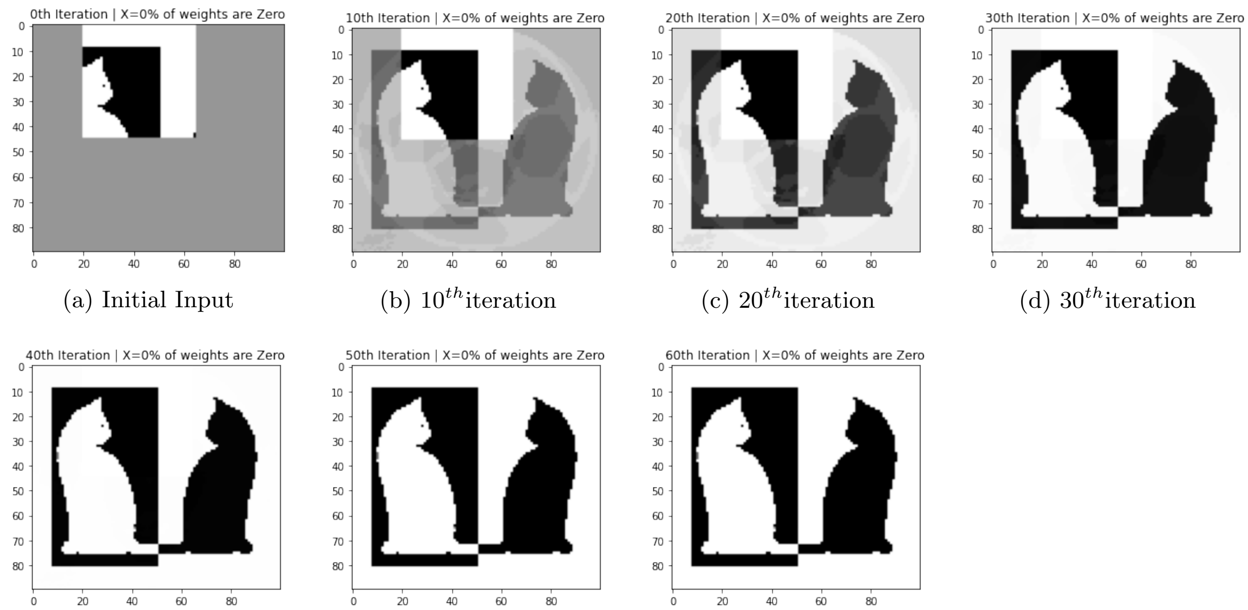


(a) Initial Input (b) 10^{th}iteration (c) 20^{th}iteration (d) 30^{th}iteration

(e) 40^{th}iteration (f) 50^{th}iteration (g) Final Image

Figure 11: A small patch with a portion of the input image as shown in Figure 11(a) is being used as a cue for retrieving the *ball* image.

Similarly, our **Hopfield Network** takes Figure 12(a) as input and returns the final image of cat.



(a) Initial Input (b) 10^{th}iteration (c) 20^{th}iteration (d) 30^{th}iteration

(e) 40^{th}iteration (f) 50^{th}iteration (g) Final Image

Figure 12: A small patch with a portion of the input image as shown in Figure 12(a) is being used as a cue for retrieving the *cat* image.

(c) Mona Lisa: X=25%

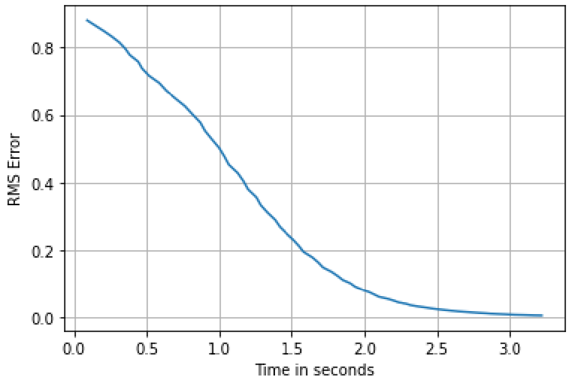
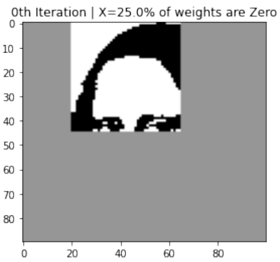
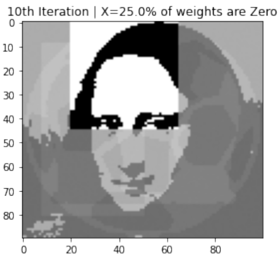


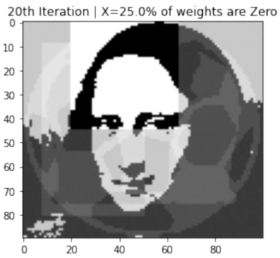
Figure 13: RMSE v/s Time (in seconds) for image *mona* (X=25%)



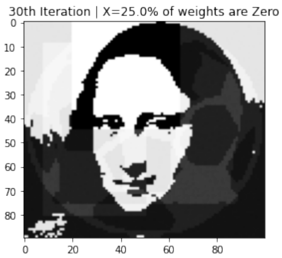
(a) Initial Input



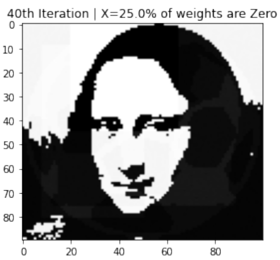
(b) 10^{th}iteration



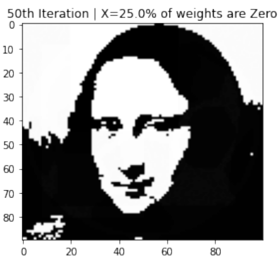
(c) 20^{th}iteration



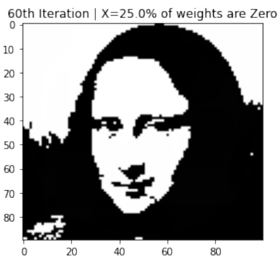
(d) 30^{th}iteration



(e) 40^{th}iteration



(f) 50^{th}iteration



(g) Final Image

Figure 14: A small patch (X=25%) with a portion of the input image as shown in Figure 14(a) is being used as a cue for retrieving the *mona* image.

Mona Lisa: X=50%

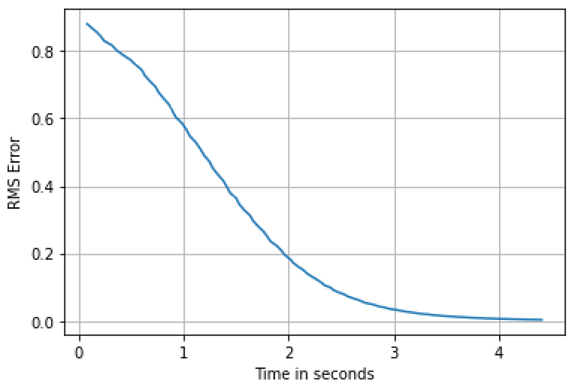
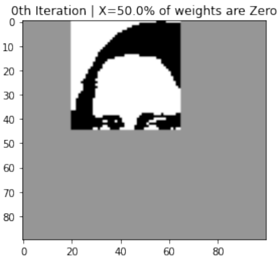
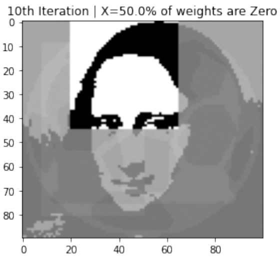


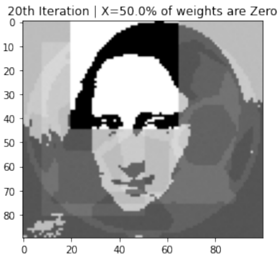
Figure 15: RMSE v/s Time (in seconds) for image *mona* (X=50%)



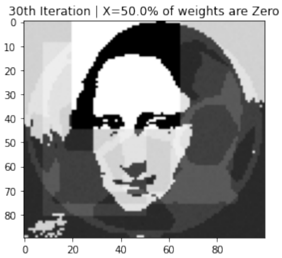
(a) Initial Input



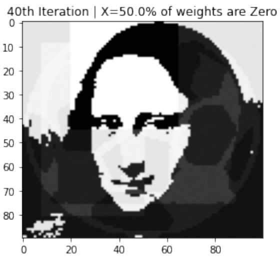
(b) 10^{th}iteration



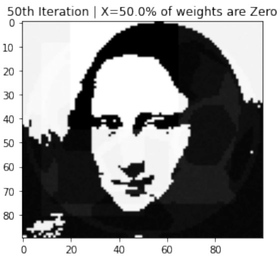
(c) 20^{th}iteration



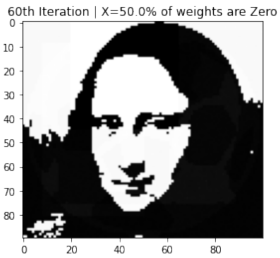
(d) 30^{th}iteration



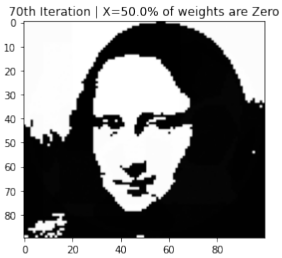
(e) 40^{th}iteration



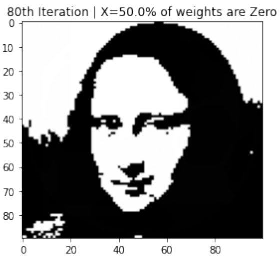
(f) 50^{th}iteration



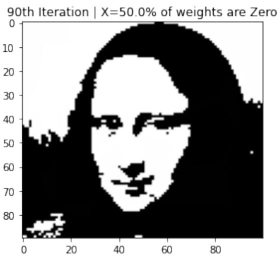
(g) 60^{th}iteration



(h) 70^{th}iteration



(i) 80^{th}iteration



(j) Final Image

Figure 16: A small patch (X=50%) with a portion of the input image as shown in Figure 16(a) is being used as a cue for retrieving the *mona* image.

Mona Lisa: X=80%

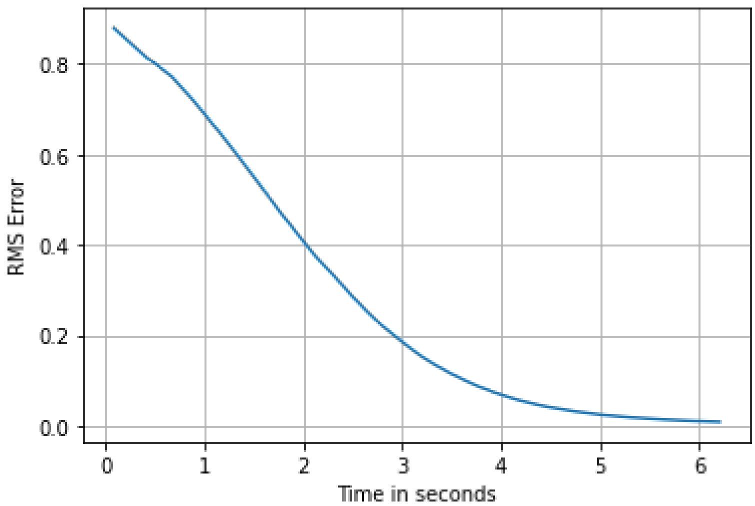
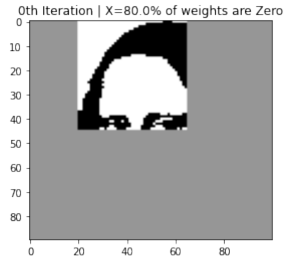
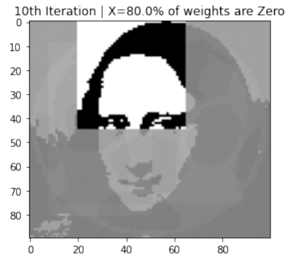


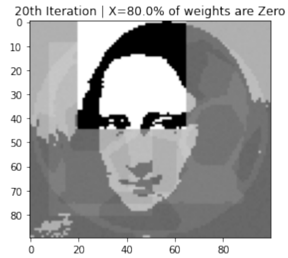
Figure 17: RMSE v/s Time (in seconds) for image *mona* (X=80%)



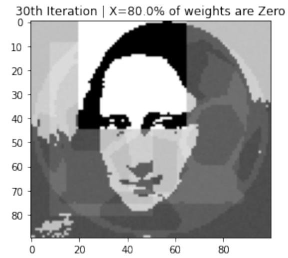
(a) Initial Input



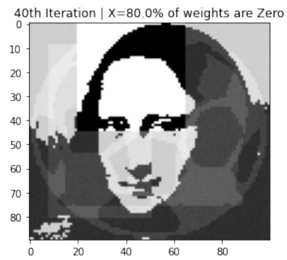
(b) 10^{th}iteration



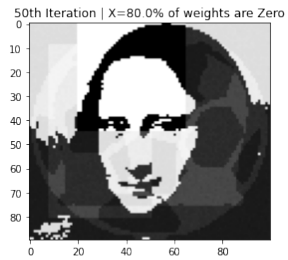
(c) 20^{th}iteration



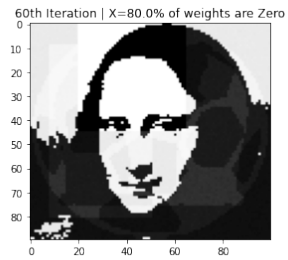
(d) 30^{th}iteration



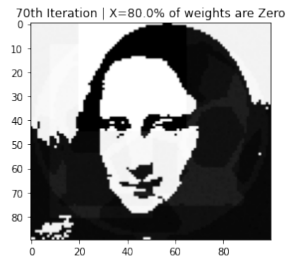
(e) 40^{th}iteration



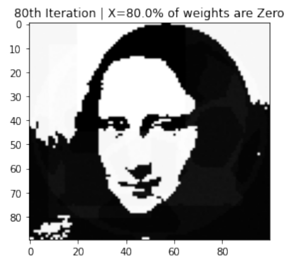
(f) 50^{th}iteration



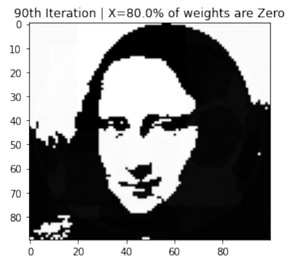
(g) 60^{th}iteration



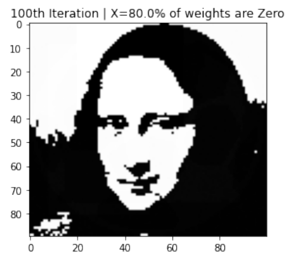
(h) 70^{th}iteration



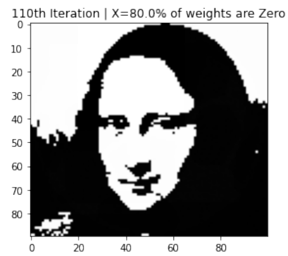
(i) 80^{th}iteration



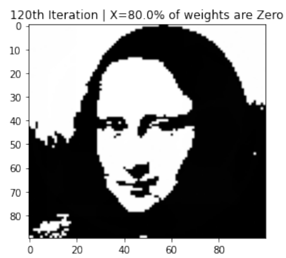
(j) 90^{th}iteration



(k) 100^{th}iteration



(l) 110^{th}iteration



(m) Final Image

Figure 18: A small patch (X=80%) with a portion of the input image as shown in Figure 18(a) is being used as a cue for retrieving the *mona* image.

Ball: X=25%

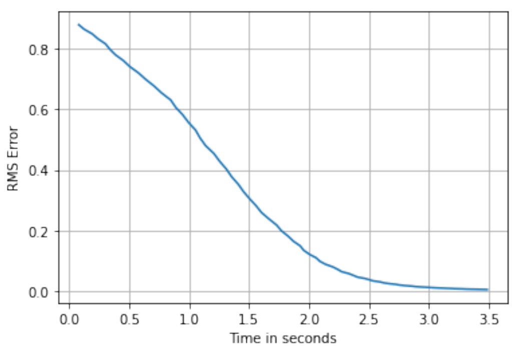


Figure 19: RMSE v/s Time (in seconds) for image *ball*
(X=25%)

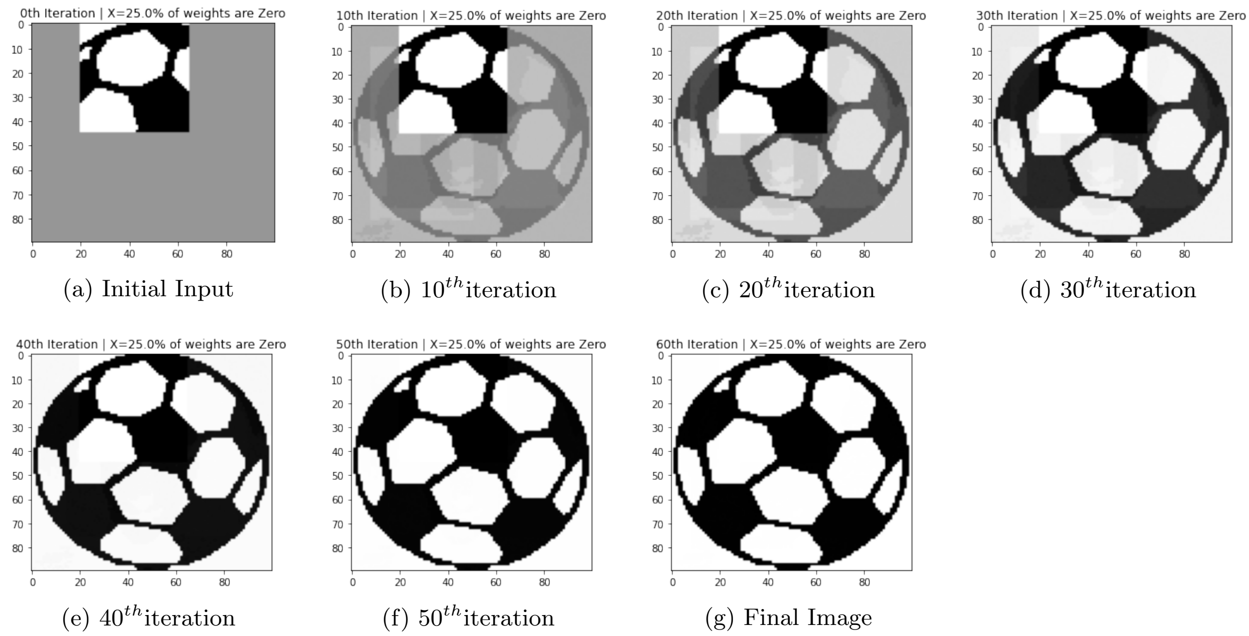


Figure 20: A small patch (X=25%) with a portion of the input image as shown in Figure 20(a) is being used as a cue for retrieving the *ball* image.

Ball: X=50%

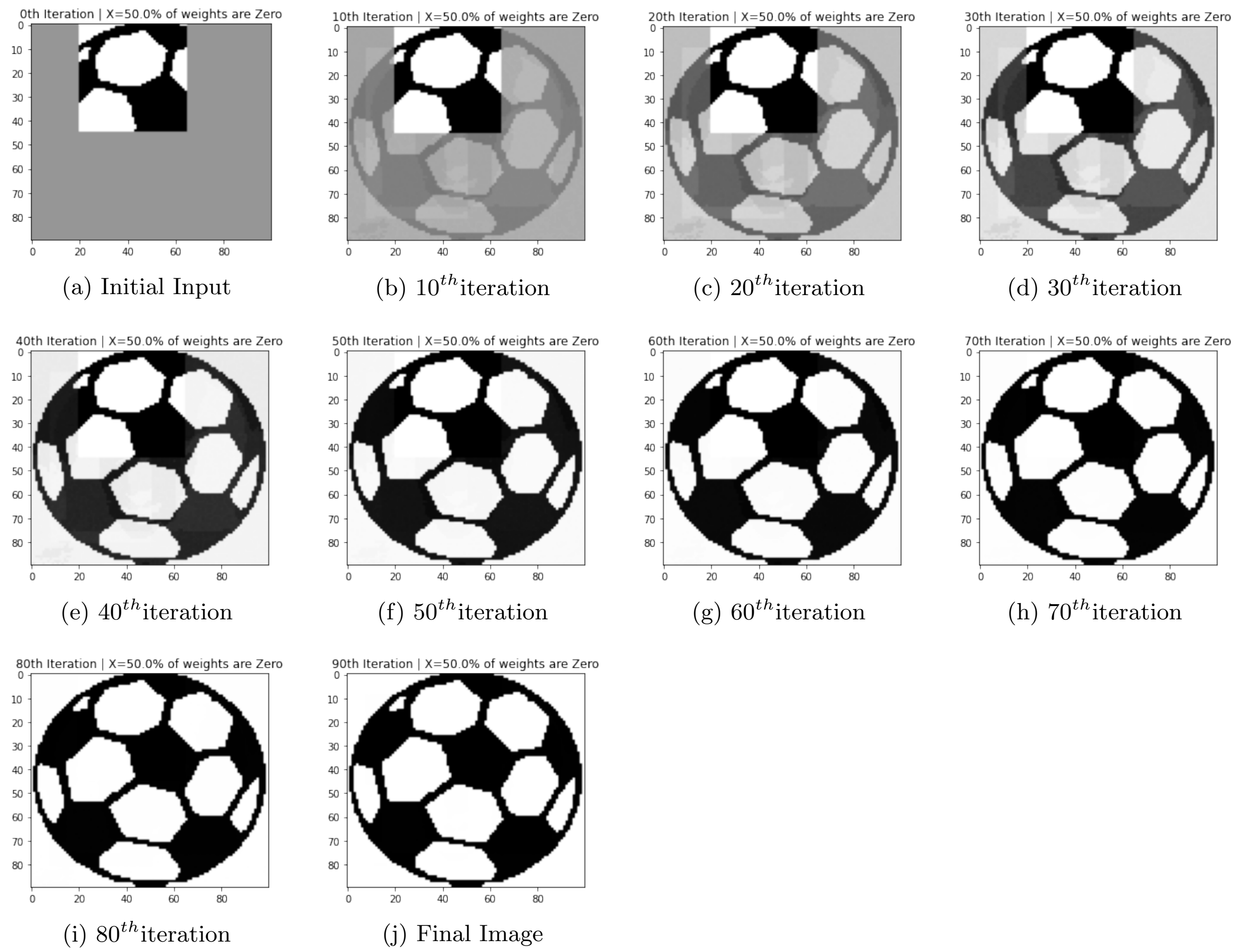


Figure 21: A small patch (X=50%) with a portion of the input image as shown in Figure 21(a) is being used as a cue for retrieving the *ball* image.

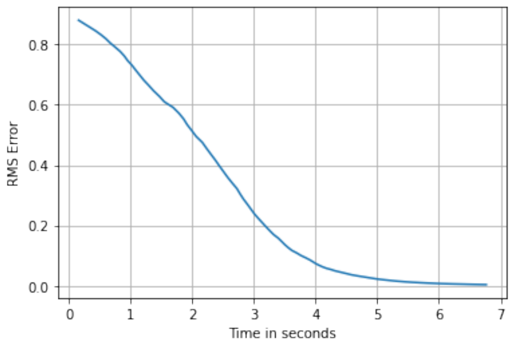
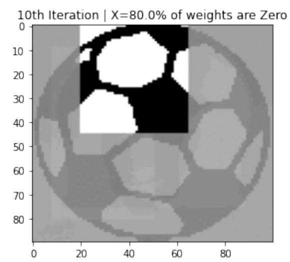


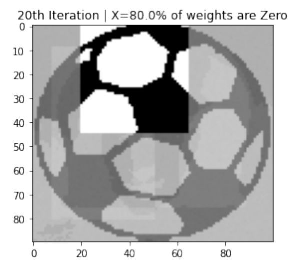
Figure 22: RMSE v/s Time (in seconds) for image *ball* (X=50%)

Ball: X=80%

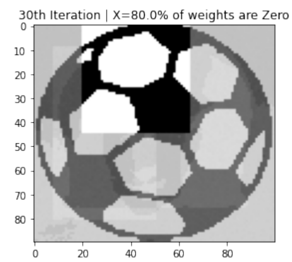
(a) Initial Input



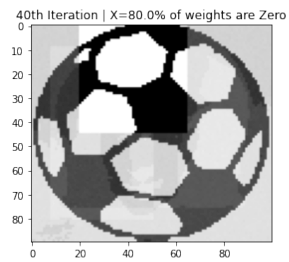
(b) 10^{th}iteration



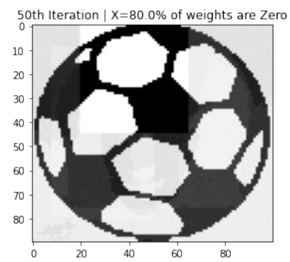
(c) 20^{th}iteration



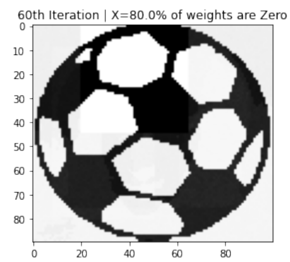
(d) 30^{th}iteration



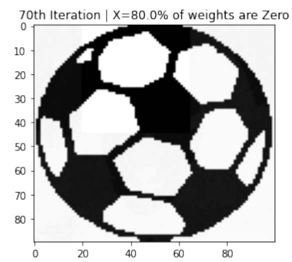
(e) 40^{th}iteration



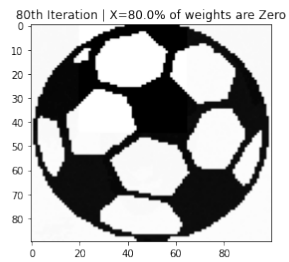
(f) 50^{th}iteration



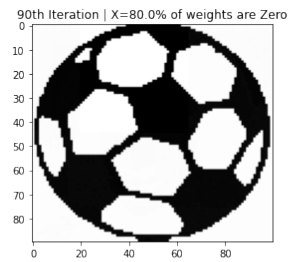
(g) 60^{th}iteration



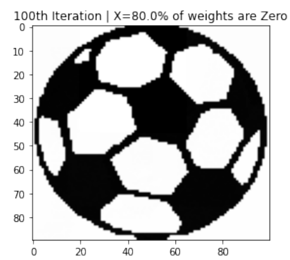
(h) 70^{th}iteration



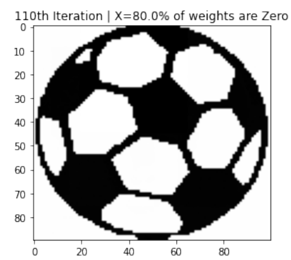
(i) 80^{th}iteration



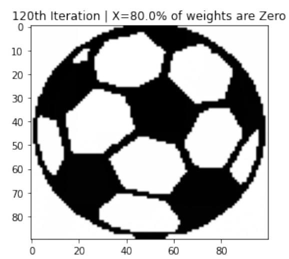
(j) 90^{th}iteration



(k) 100^{th}iteration



(l) 110^{th}iteration



(m) Final Image

Figure 23: A small patch (X=80%) with a portion of the input image as shown in Figure 23(a) is being used as a cue for retrieving the *ball* image.

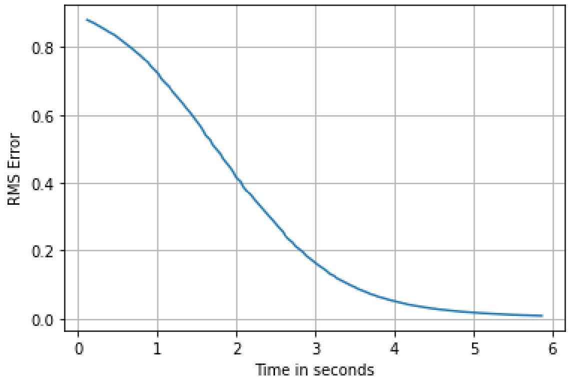


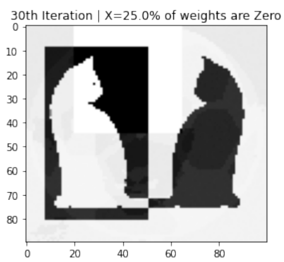
Figure 24: RMSE v/s Time (in seconds) for image *ball* (X=80%)

Cat: X=25%

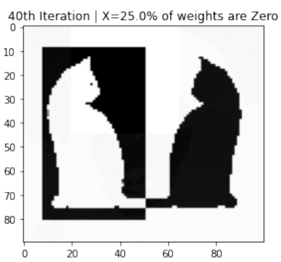
(a) Initial Input

(b) 10^{th}iteration

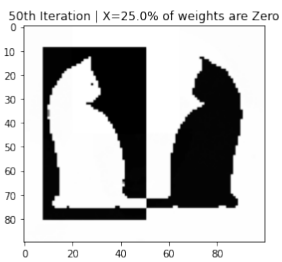
(c) 20^{th}iteration



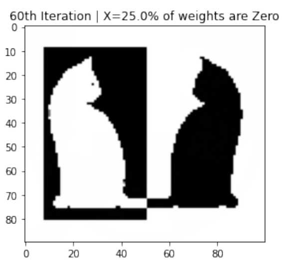
(d) 30^{th}iteration



(e) 40^{th}iteration



(f) 50^{th}iteration



(g) Final Image

Figure 25: A small patch (X=25%) with a portion of the input image as shown in Figure 25(a) is being used as a cue for retrieving the *cat* image.

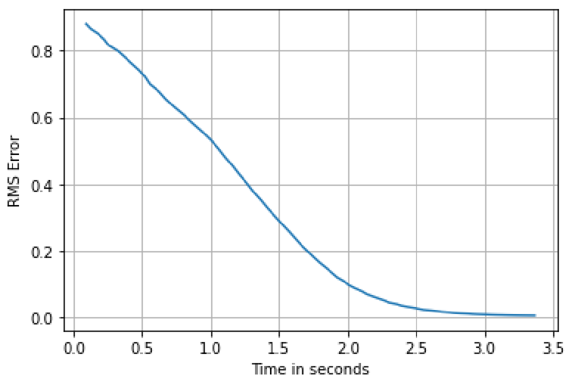
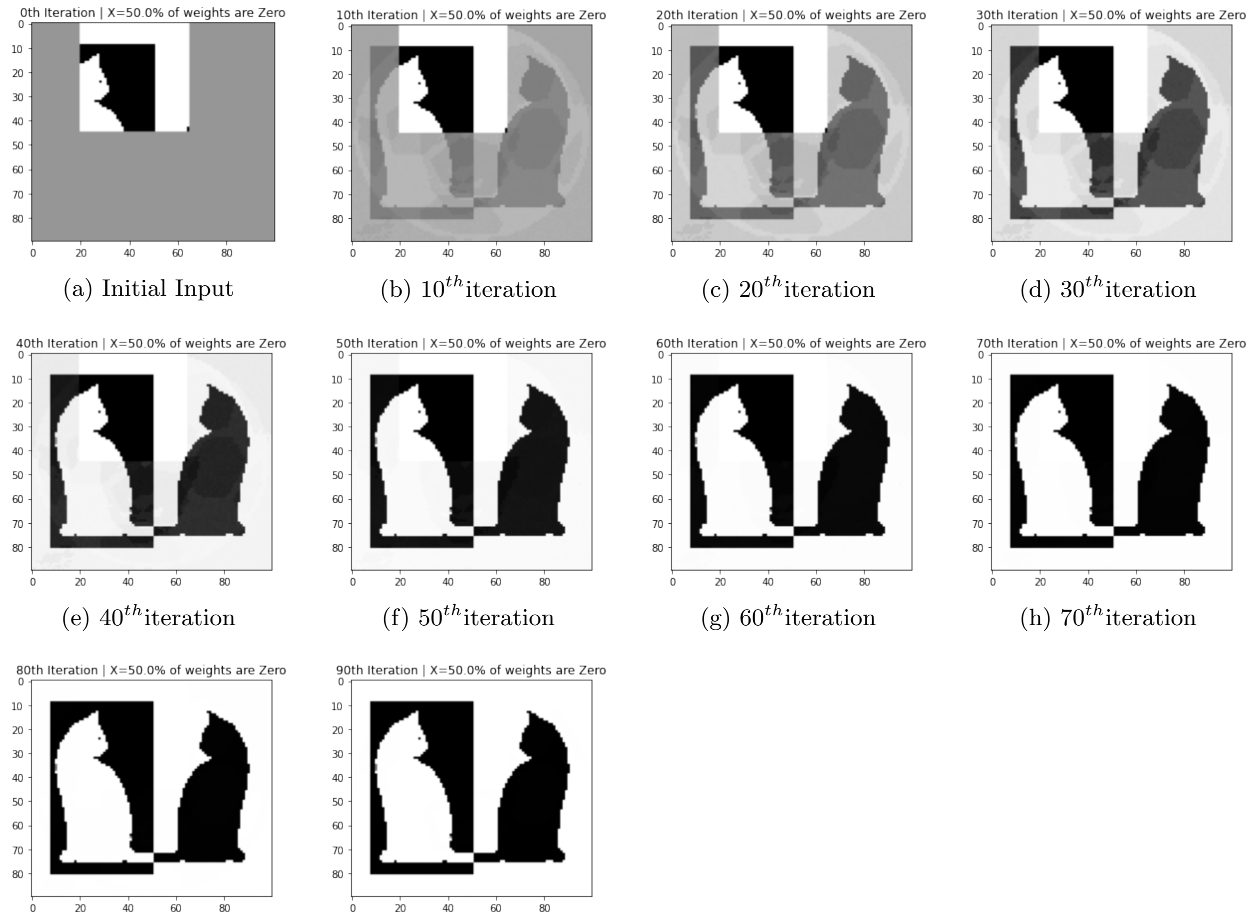


Figure 26: RMSE v/s Time (in seconds) for image *cat* (X=25%)

Cat: X=50%



(a) Initial Input (b) 10^{th}iteration (c) 20^{th}iteration (d) 30^{th}iteration

(e) 40^{th}iteration (f) 50^{th}iteration (g) 60^{th}iteration (h) 70^{th}iteration

(i) 80^{th}iteration (j) Final Image

Figure 27: A small patch (X=50%) with a portion of the input image as shown in Figure 27(a) is being used as a cue for retrieving the *cat* image.

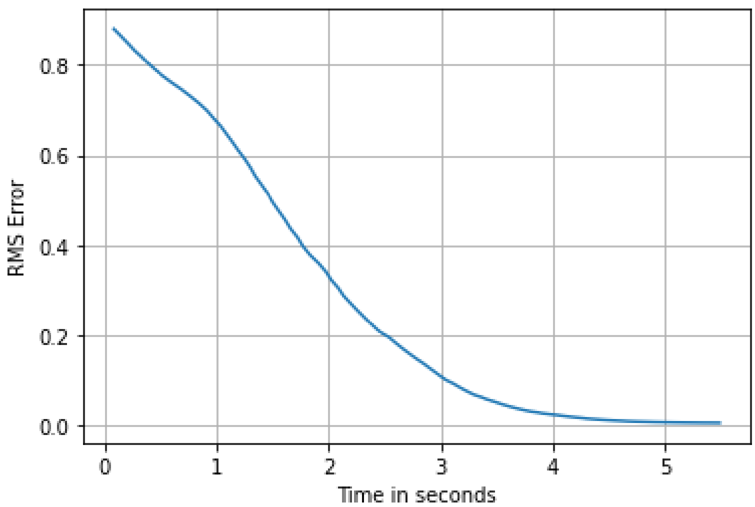


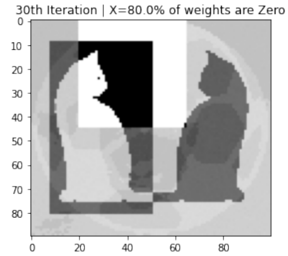
Figure 28: RMSE v/s Time (in seconds) for image *cat* (X=50%)

Cat: X=80%

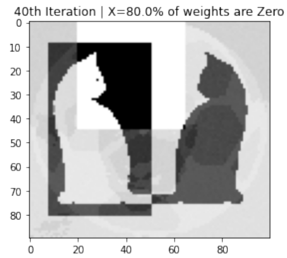
(a) Initial Input

(b) 10^{th}iteration

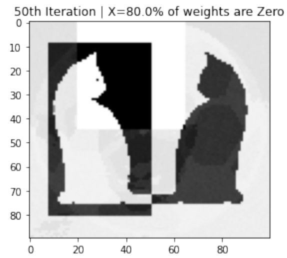
(c) 20^{th}iteration



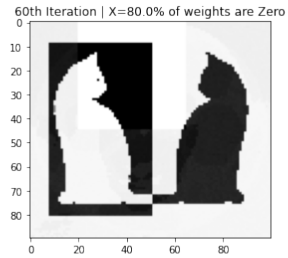
(d) 30^{th}iteration



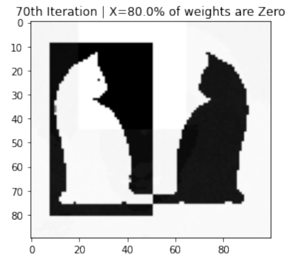
(e) 40^{th}iteration



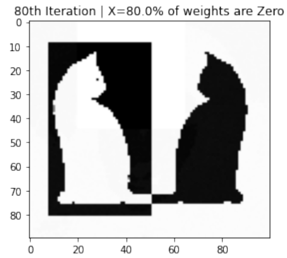
(f) 50^{th}iteration



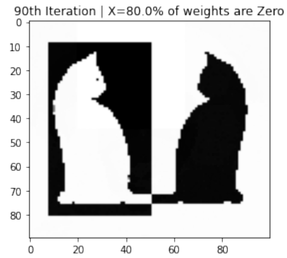
(g) 60^{th}iteration



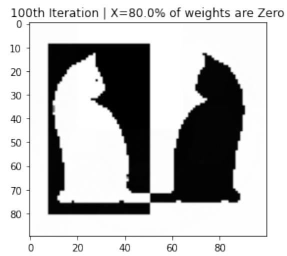
(h) 70^{th}iteration



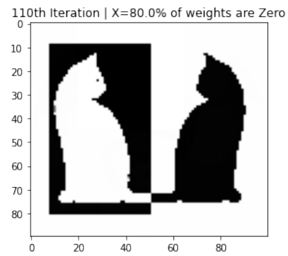
(i) 80^{th}iteration



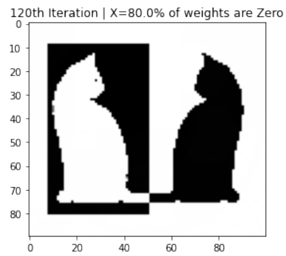
(j) 90^{th}iteration



(k) 100^{th}iteration



(l) 110^{th}iteration



(m) Final Image

Figure 29: A small patch (X=80%) with a portion of the input image as shown in Figure 29(a) is being used as a cue for retrieving the *cat* image.

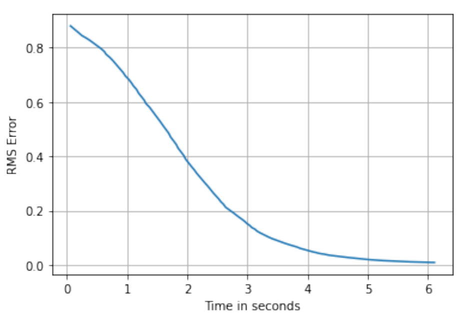


Figure 30: RMSE v/s Time (in seconds) for image *cat* (X=80%)