SC42500 Tensor Networks - Assignment 4 - Image Reconstruction with Tensor Completion

Floris van Seumeren (5870704), Matthias van Ogtrop (4899679)
DCSC Delft
Delft University of Technology
{fvanseumeren, mvanogtrop}@tudelft.nl

I. Introduction

This report covers our approach to the final assignment for the course Tensor Networks for Green AI and Signal Processing. The assignment covers the topic of reconstructing an image from a subset of pixels of the original image. This is called *matrix completion*. The problem of matrix completion is often ill-posed, but the assumption is made that the underlying matrix is low-rank. This allows us to formulate the image reconstruction problem as a rank minimization problem:

$$\min_{\mathbf{X}} \quad \text{rank}(X) \tag{1}$$

$$s.t \quad M(i,j) = X(i,j) \quad (i,j) \in \Omega \tag{2}$$

Here Ω is the set of pixels that remain.

The image that will be analyzed in this report is a 512×512 pixel color image (see Figure 1), and can therefore be represented by a $512 \times 512 \times 3$ tensor, with the third dimension representing the red-, green- and blue-values for every pixel. The image reconstruction problem can now be defined as a tensor completion problem:

$$\min_{\mathcal{X}} \quad \text{rank}(\mathcal{X}) \tag{3}$$

$$s.t. \quad \mathcal{X}_\Omega = \mathcal{T}_\Omega \tag{4}$$

Problem statement

The reconstruction of a color image can be formulated as three separate matrix rank minimization problems for each of the color layers of the image or as a three-dimensional tensor completion problem. The scope of this report is to evaluate, quantify and compare the performance of both methods.

II. Matrix completion method & implementation

The matrix completion algorithm used in this assignment is outlined in [1]. As was previously mentioned, theoretically the problem of matrix completion is ill-posed. However, most images can be assumed to be low rank. This is a consequence of the fact that in most images there is a high correlation between the color values of neighboring pixels, as well as the presence of repetitive spacial patterns. This means that the majority of the information is encoded in the largest singular values. This also holds for our image, see Figure 2. It shows the first 100 singular values for the three channels in the picture. It can be seen that the majority of the information is embedded within the first 20 singular values. The Singular Value Thresholding (SVT) algorithm outlined in [1] seeks to minimize the nuclear norm of the approximated matrix, while satisfying convex constraints. The exact structure of the SVT-algorithm can be found in Section 5.1.5 of [1]. It is an iterative algorithm and has two stopping criteria: a maximum number of iterations and a lower bound on the relative error. A maximum

Fig. 1: Original image

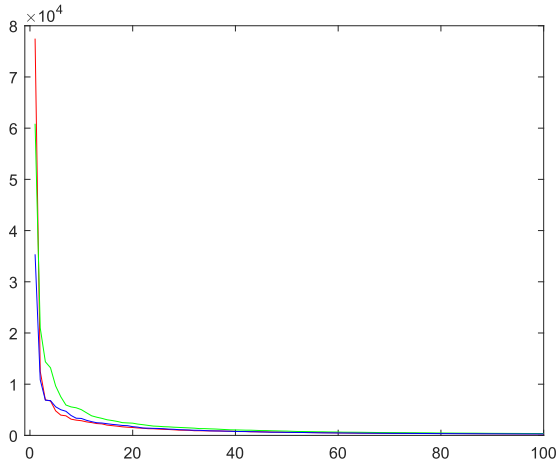

Fig. 2: First 100 Singular values of the RGB-channels. The lines have corresponding colors to the channel.

step count of 250 is chosen in our design. The relative error is defined as

$$\|\mathcal{P}_\Omega(X^k - M)\|_F / \|\mathcal{P}_\Omega M\|_F$$

and it was decided to stop the algorithm when the relative error is less than 10^{-9}. This algorithm is executed for each of the three color layers of the image.

III. Tensor completion method & implementation

For the tensor completion algorithm, multiple options were provided. [2] introduces the Low Rank Tensor Completion (LRTC) algorithm, which employs Block Coordinate Descent for the optimization. This method minimizes over the sum the mode-i unfoldings of the tensor. Three heuristic methods are also presented as a comparison. [3] presents a survey of a few different types of tensor completion algorithms, starting with several decomposition based approaches (CP-based methods and Tucker-based methods). It also mentions a few trace norm based methods, including the LRTC algorithm mentioned in [2]. It was decided to implement the LRTC algorithm for the image reconstruction problem. This was done for several reasons. Seeing as the problem of reconstructing an image of 512 by 512 pixels is sufficiently tractable, the heuristic methods were deemed excessive. According to [3], the LRTC algorithm is the first tensor completion algorithm that does not require fixing the rank of the tensor in advance, which is something that is required for decomposition-based methods. Part of the motivation for choosing this method is also that the matrix completion algorithm discussed in section II uses a trace norm method. Evaluating the performance of this matrix completion algorithm and a trace norm-based tensor completion algorithm will result in a more telling and meaningful comparison. The outline of the method is given below.

Algorithm 1 Tensor Completion Algorithm

Require: $\mathcal{T}_\Omega$
Ensure: $\mathcal{X}, \mathcal{Y}, M_i,$ for $i = 1, \ldots, n$
1: Set $\mathcal{Y}_\Omega = \mathcal{T}_\Omega$, $\mathcal{Y}_{\bar{\Omega}} = 0$, $\mathcal{X} = \mathcal{Y}$, $M_i = \mathcal{Y}(i)$
2: **while** no convergence **do**
3: **for** $i = 1$ to n **do**
4: $M_i = D_{\frac{\gamma_i}{\alpha_i + \beta_i}} \left(\frac{\alpha_i X_{(i)} + \beta_i Y_{(i)}}{\alpha_i + \beta_i} \right)$
5: **end for**
6: $\mathcal{X} = \frac{\sum_{i=1}^n \alpha_i \, \text{fold}_i(M_i)}{\sum_{i=1}^n \alpha_i}$
7: $\mathcal{Y}_{\bar{\Omega}} = \left(\frac{\sum_{i=1}^n \beta_i \, \text{fold}_i(M_i)}{\sum_{i=1}^n \beta_i} \right)_{\bar{\Omega}}$
8: **end while**

IV. Results

Simulation 1 - Same mask for all layers

The first simulation is performed by applying a mask to the original image. This mask removes a certain percentage of pixels by setting the RGB-values to 0, transforming them into black pixels. This masked image is then sent to both the matrix rank minimization algorithm and the tensor completion algorithm. These both produce a reconstruction of the original image. The algorithms are run for four different black pixel percentages, namely 15%, 25%, 50% and 70%. The resulting images can be seen in figures 3-6. The runtimes for the algorithms for each pixel percentage can be seen in table I.

Fig. 3: masked image with 15% black pixels (left), matrix algorithm reconstruction (middle), tensor algorithm reconstruction (right)

Fig. 4: masked image with 25% black pixels (left), matrix algorithm reconstruction (middle), tensor algorithm reconstruction (right)

Fig. 5: masked image with 50% black pixels (left), matrix algorithm reconstruction (middle), tensor algorithm reconstruction (right)

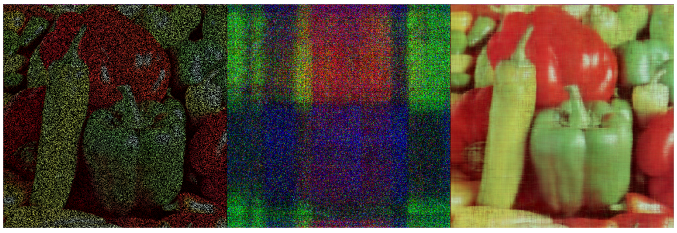

Fig. 6: masked image with 70% black pixels (left), matrix algorithm reconstruction (middle), tensor algorithm reconstruction (right)

Simulation 2 - Different masks for each layer

The second simulation is performed by applying a different mask to each individual layer of pixels, as opposed to applying the same mask to all the layers like in simulation 1. Three different simulations are performed. In the first simulation, a mask that removes 25% of the pixels is applied to each of the color layers. In the second simulation, the red and green layers are left almost intact by only removing 10% of the pixels, whereas 75% of the pixels are removed from the blue layer. In the third simulation, the red layer is left almost intact with only 10% of the pixels removed, whereas 75% of pixels of the green and blue layers are removed.

Scenario 1 - 25% removed from each layer:

Fig. 7: original image (top left), masked image with 75% of R, G, and B pixels remaining (top right), matrix algorithm reconstruction (bottom left), tensor completion reconstruction (bottom right)

Scenario 2 - 10% removed from red and blue layer, 75% from green layer:

Fig. 8: original image (top left), masked image with 90% of R and B pixels, 25% of G pixels remaining (top right), matrix algorithm reconstruction (bottom left), tensor completion reconstruction (bottom right)

Scenario 3 - 10% removed from red layer, 75% from green and blue layers:

Fig. 9: original image (top left), masked image with 90% of R pixels, 25% of B and G pixels remaining (top right), matrix algorithm reconstruction (bottom left), tensor completion reconstruction (bottom right)

Simulation 3 - Uniform Noise distortion

To see how the matrix and tensor methods handle a noise disturbance, a third simulation is performed, where a noise

mask is added. The noise is added adding a random number between -40 and 40 to each value in the image tensor, while keeping the noised value between 0 and 255. The black pixel generating mask from section IV is then added. Two simulations are run: one where 25% of the pixels are removed and one where 60% of the pixels are removed. The results can be seen in figures 10-11.

Scenario 1 - Image reconstruction with noise, 25% of pixels removed:

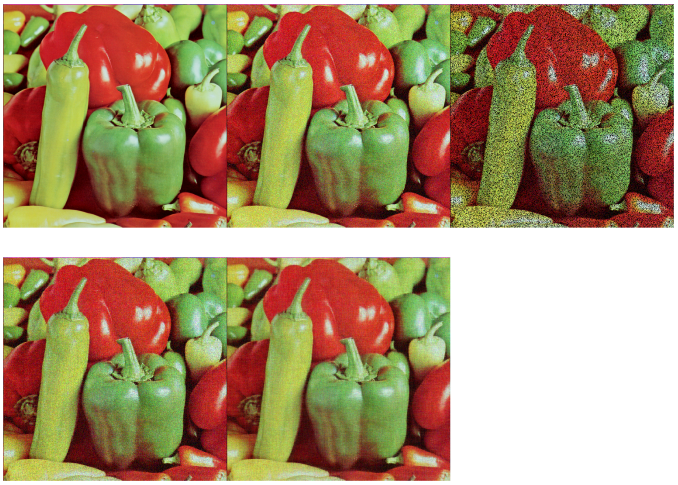

Fig. 10: Original image (top left), image with noise (top middle), image with noise and black pixel mask, 25% of pixels removed (top right), matrix reconstruction (bottom left), tensor reconstruction (bottom right)

Scenario 2 - Image reconstruction with noise, 60% of pixels removed:

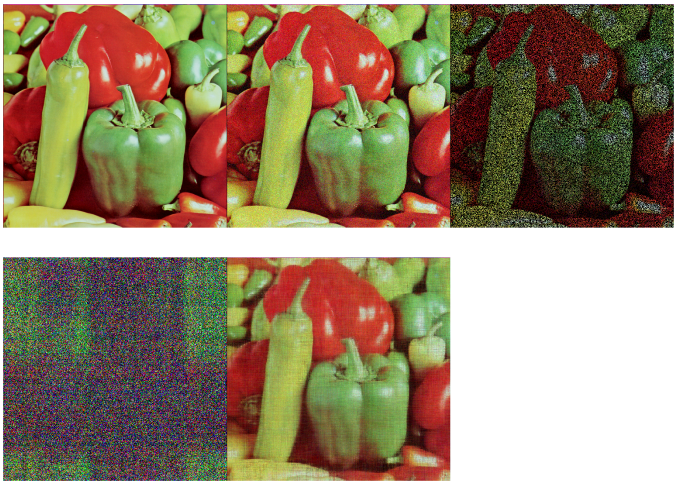

Fig. 11: Original image (top left), image with noise (top middle), image with noise and black pixel mask, 60% of pixels removed (top right), matrix reconstruction (bottom left), tensor reconstruction (bottom right)

V. Discussion

After performing several simulations for different types of image reconstruction challenges, it is clear that the matrix completion algorithm works fine as long as a significant percentage of the image is intact. However, figures 3-6 clearly show that the matrix algorithm struggles with reconstructing images with larger percentages of pixels missing. On the other hand, the tensor algorithm does relatively well even with up to 70 % of pixels missing. The tensor method also outperforms the matrix method when the three layers of the images are compromised to different extents. This is especially notable in Figure 9, where the tensor method does a far better job at restoring the image to the same color than the matrix method. This is due to the tensor approach looking at all layers of pixels, and therefore being able to extract information from less affected layers, which is something that the layer-by-layer approach of the matrix method isn't able to do. Finally, the tensor method also outperforms the matrix method when noise is added to the image. Even when a low amount of pixels is missing, which is where the matrix method should compare relatively well to the tensor method, the latter still manages to outperform the former at both reconstructing the sharpness of the image and denoising it. The second effect is quite subtle, but can be seen when zooming on the bottom two pictures of 10. The computation times and other performance metrics for the different methods and different scenarios can all be seen in table I. From the results it is clear that the tensor completion method outperforms the matrix rank minimization method by a substantial margin. However, there are scenarios in which the matrix method comes close to matching the tensor method in performance. In the very first simulation (85% pixels remaining), the matrix method's performance metrics indicate that the image quality is superior to the that of the tensor method. The matrix method performs well when it has a high proportion of pixels to work with for each layer. From the results we can make the assumption that the relative margin of the tensor method over the matrix method increases the more pixels are missing. This margin further increases when the layers are missing values from different pixels with respect to each other, as demonstrated in the different layers section.

References

[1] J.-F. Cai, E. J. Candes, and Z. Shen, "A singular value thresholding algorithm for matrix completion," 2008.

[2] J. Liu, P. Musialski, P. Wonka, and J. Ye, "Tensor completion for estimating missing values in visual data," in *2009 IEEE 12th International Conference on Computer Vision*, pp. 2114–2121, 2009.

[3] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor completion algorithms in big data analytics," 2018.

		Same layers				Different layers			Same layers with noise	
		85%	75%	50%	30%	*1	*2	*3	75%	40%
Compute time (seconds)	Matrix	58.0	57.0	43.0	204.4	38.6	38.7	46.2	38.6	320.1
	Tensor	6.6	7.6	11.6	11.5	6.3	12.1	11.6	5.6	10.0
PSNR (-)	Matrix	35.8	32.6	26.2	7.3	20.8	19.8	12.6	23.6	6.5
	Tensor	31.2	30.0	26.7	22.4	30.2	28.0	24.0	23.9	20.0
Relative error (-)	Matrix	0.0319	0.0463	0.0967	1.3380	0.1809	0.2030	0.5980	0.1290	1.8517
	Tensor	0.0541	0.0627	0.0918	0.1499	0.0614	0.0789	0.1251	0.1239	0.1956

TABLE I: Computation times, Peak Signal-to-noise ratio and relative error for all simulations.
Remaining pixels:
*1:R=75%,G=75% B=75%.
*2:R=90%,G=90% B=25%.
*3:R=90%,G=25% B=25%.