Unmixing 4-D Ptychographic Image: Part B:Data Approach

Student: Zhen Zhang(CUHK), Huanlin Zhou(CUHK), Michaela D. Shoffner(UTK)
Mentors: R. Archibald(ORNL), S. Tomov(UTK), A. Haidar(UTK), K. Wong(UTK)

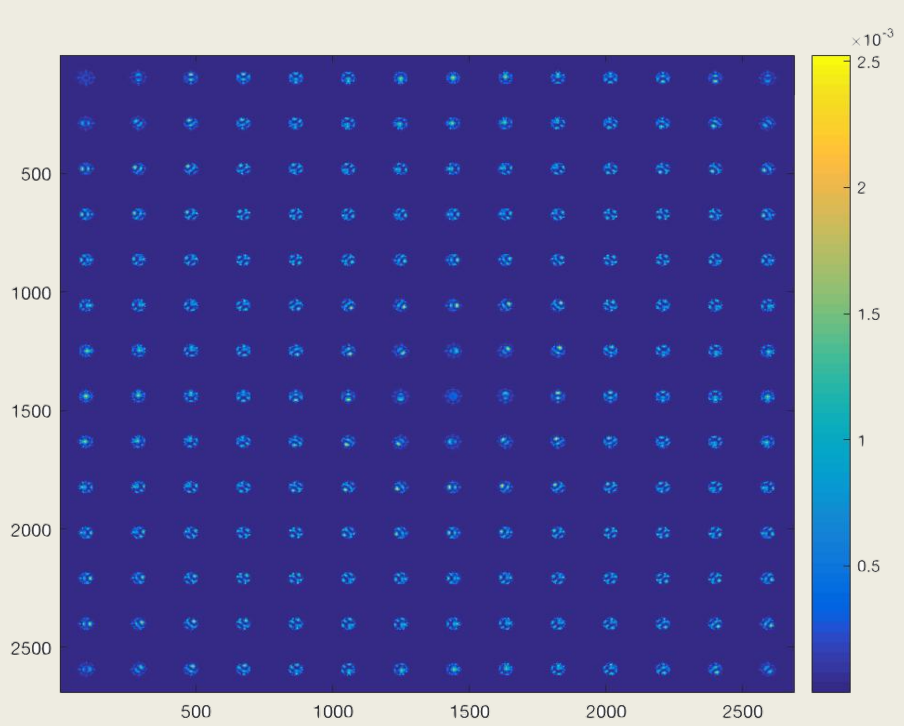
INTRODUCTION

There are three known basic modes, M_0, M_1, M_2, each of which is a 2688 by 2688 image. The problem is, for each input image I, we try to find a representation of I using the three basic modes. It is known that the input image can be closely represented as a linear combination of the three basic modes, namely,

$$I = \alpha M_0 + \beta M_1 + \gamma M_2$$

The problem can easily be solved by least square method. However, the result of least square is quite far away from what we desire. For example, for one of the input images , where the true coefficients are $(\alpha, \beta, \gamma) = (1, 1, 1)$, the output of least square method is $(0.9950, 0.8284, 0.7945)$. For $(\alpha, \beta, \gamma) = (1, -1, -1)$, the result of least square is $(0.9426, -0.3582, -0.3590)$, which has large notable error.

A machine learning method with interpolation is proposed to achieve better accuracy for current data. For example, for an image with $(\alpha, \beta, \gamma) = (1, -1, -1)$, the output of the neural network is $(0.9994, -0.9675, -0.9828)$, with 2 hidden layers, 15 nodes in each hidden layer and regularisation parameter = 0.01.



The above is M0. M1 and M2 look similar.
In future computations, we do the following assignments:
M1 - M0 -> M1, M2 - M0 -> M2.

METHOD

Currently, only 16 input images are provided and every four of them share the same set of coefficients. Namely,

$$(\alpha, \beta, \gamma) = (1, \pm 1, \pm 1)$$

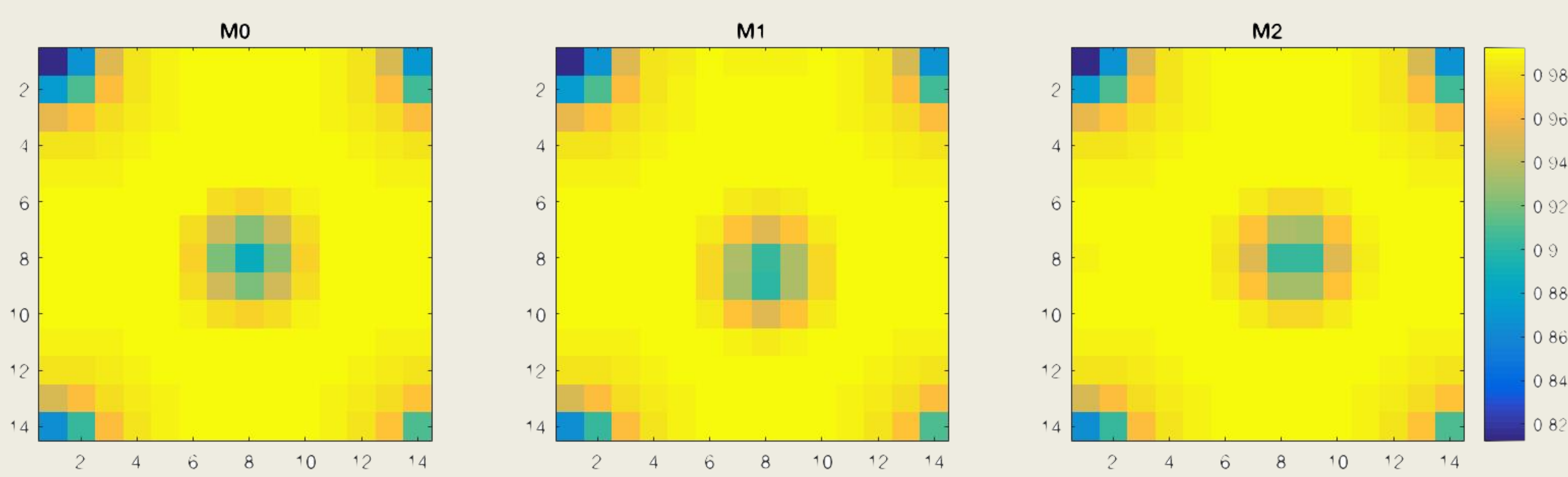
This shortage of data makes it impossible to train a neural network with what we now have. The remedy is to generate synthetic data with interpolation. For each of the pixels in an input image, we know the bias of linear approximation. It is assumed that the bias is a result of mutual effect of β and γ . Namely, the bias for a pixel (x, y) can be written as following:

$$B = B_{x,y}(\beta, \gamma)$$

We can interpolate the bias using the four points for each pixel. If we take M_1 and M_2 also as input images, we can interpolate using six points.

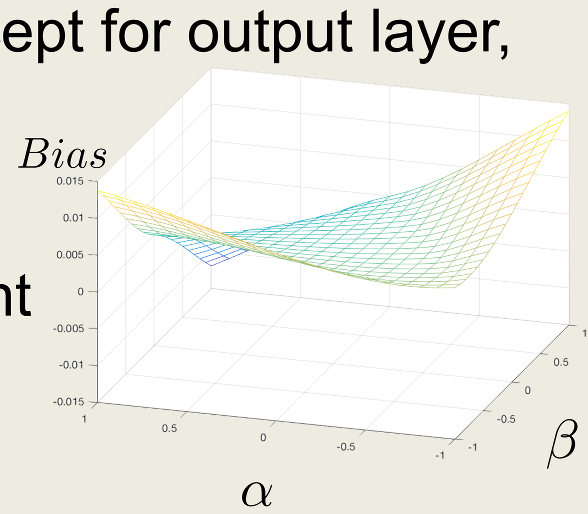
COMPUTATIONS&RESULTS

To simplify the inputs we sum up all pixel in a 192 by 192 block in an input image or basic mode; we will only consider the 14 by 14 summed image.

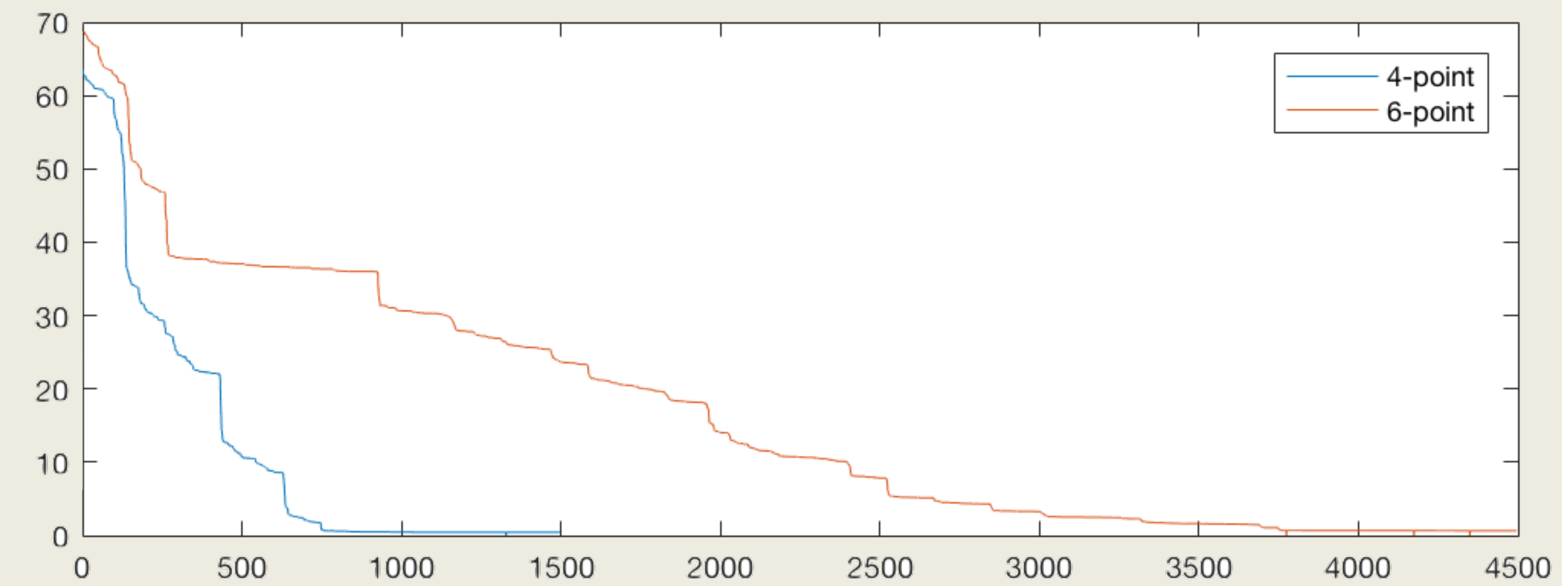


The activation function is tanh except for output layer, which is the identity.

Different interpolation methods can be chosen. The figure on the right shows the 6-point cubic interpolation of one pixel.



The cost function is minimised by a CG method.[1]
With 200 training examples, regularisation parameter = 0.01, and linear interpolation, the changes of cost functions are shown in the following figure.



Note that in the 4-point case the training is much faster.

Results:
(4-point case, one input image for each set of coefficients)

coef \ True coef	(1,1,1)	(1,1,-1)	(1,-1,1)	(1,-1,-1)	(1,1,0) (M1)	(1,0,1) (M2)
α	0.9891	1.0053	0.9823	0.9925	0.9911	1.0051
β	1.0010	0.9735	-0.9755	-1.0396	-0.0494	1.1258
γ	0.9946	-0.9931	0.9914	-1.0123	1.1155	0.0359

(6-point case, one input image for each set of coefficients)

coef \ True coef	(1,1,1)	(1,1,-1)	(1,-1,1)	(1,-1,-1)	(1,1,0) (M1)	(1,0,1) (M2)
α	0.9934	1.0019	0.9972	0.9953	0.9947	1.0025
β	0.8718	1.0750	-1.0476	-0.9736	0.9907	0.0199
γ	1.0464	-1.0962	0.9614	-0.9915	0.0310	1.0205

Recall: M1 and M2 are two of the basic modes
Note that in the 4-point case, predictions on M1 and M2 failed.

ANALYSIS&FUTURE WORK

A better testing of the algorithm is to directly input M_0 and check if the output is (1,0,0). In fact, the output is (1.0050, -0.0829, 0.0054), which is quite close. This successful prediction on M_0 gives us confidence that this interpolation-training method can work for other test data.

In the future, if more data can be experimentally acquired, this algorithm can be more rigorously tested. If it does not work well with the new data, we can interpolate with the new data and thus improve the model.

Implementation of the algorithm using MAGMA is being worked on.

Essentially, this neural network is solving this problem similar to using least square method, except that there is a non-linear bias of unknown mathematical form. Therefore this idea can possibly be used to solve other linear algebra problems based on the NN formulation, such as to compute the matrix inverse.

ACKNOWLEDGEMENTS

This program is sponsored by the National Science Foundation, the Joint Institute of Computational Sciences, and the Chinese University of Hong Kong.

REFERENCE

1. fmincg(), Carl Edward Rasmussen, 2003