Contents lists available at ScienceDirect

Neural Networks

journal homepage: www.elsevier.com/locate/neunet

Towards holographic "brain" memory based on randomization and Walsh–Hadamard transformation[☆]

Daniel Berend[a,b], Shlomi Dolev[a,*], Sergey Frenkel[c], Ariel Hanemann[a]

[a] Computer Science Department, Ben Gurion University of the Negev, Beer Sheva, Israel
[b] Mathematics Department, Ben Gurion University of the Negev, Beer Sheva, Israel
[c] Institute of Informatics Problems, Federal Research Center "Computer Science and Control" of the Russian Academy of Sciences, Moscow, Russia

A R T I C L E I N F O

Article history:
Received 9 August 2015
Received in revised form 11 December 2015
Accepted 2 February 2016
Available online 12 February 2016

Keywords:
Holographic memory
Walsh–Hadamard
Neural networks
Neural computation

A B S T R A C T

The holographic conceptual approach to cognitive processes in the human brain suggests that, in some parts of the brain, each part of the memory (a neuron or a group of neurons) contains some information regarding the entire data.

In Dolev and Frenkel (2010, 2012) we demonstrated how to encode data in a holographic manner using the Walsh–Hadamard transform. The encoding is performed on randomized information, that is then represented by a set of Walsh–Hadamard coefficients. These coefficients turn out to have holographic properties. Namely, any portion of the set of coefficients defines a "blurry image" of the original data.

In this work, we describe a built-in error correction technique—enlarging the width of the matrix used in the Walsh–Hadamard transform to produce a rectangular Hadamard matrix. By adding this redundancy, the data can bear more errors, resulting in a system that is not affected by missing coefficients up to a certain threshold. Above this threshold, the loss of data is reflected by getting a "blurry image" rather than a concentrated damage. We provide a heuristic analysis of the ability of the technique to correct errors, as well as an example of an image saved using the system. Finally, we give an example of a simple implementation of our approach using neural networks as a proof of concept.

1. Introduction

1.1. Holographic brain

The Holographic Brain Theory, developed by Pribram (1971, 1991), suggests that the brain holds memories in a holographic manner. In a hologram, the data is not localized, but rather distributed. Each part of the holographic recording film contains some information about the entire image. Thus, a reconstruction of the image by a small piece of the holographic recording film results in a noisy version of the entire original image (Gabor, 1972), as opposed to a classical photograph, where a small piece of the film yields a sharp image of the specific piece, but no information about the rest of the picture.

1.2. Walsh–Hadamard transform

A Hadamard matrix is an $n \times n$ orthogonal matrix with entries in $\{-1, 1\}$. Hadamard matrices are widely used for many applications, e.g., communication systems, signal and image processing, digital logic, and fault-tolerant system design (Agaian, Sarukhanayan, Egiazarian, & Astola, 2011). The Walsh–Hadamard matrix is a specific Hadamard matrix (Agaian et al., 2011) (defined only for sizes n of the form $n = 2^k$). The Walsh–Hadamard transform (multiplication by the Walsh–Hadamard matrix) is a holographic transform in the sense that each *coefficient* (entry in the output vector) is a result of a simple computation involving *all* the entries of the original input vector. Indeed, since each entry in the matrix is either 1 or −1, multiplication by the matrix is obtained by additions and subtractions. Thus, each coefficient contains some information about all the entries of the input vector. Moreover, each entry has the same weighted effect on each coefficient. There are other holographic data representations, where any portion of the data contains some information on the entire data (Bruckstein, Holt, & Netravali, 1998, 2001; Dovgard, 2004; Jayalakshmi & Ananthashayana, 2007).

[☆] Partially supported by Rita Altura Trust Chair in Computer Sciences, Lynne and William Frankel Center for Computer Sciences, and Israel Science Foundation (grant number 428/11). Partially supported by the Russian Foundation for Basic Research under grant RFBR 15-07-05316.

[*] Corresponding author.
E-mail address: dolev@cs.bgu.ac.il (S. Dolev).

http://dx.doi.org/10.1016/j.neunet.2016.02.001
0893-6080/© 2016 Elsevier Ltd. All rights reserved.

However, it seems that, due to the fact that the Walsh–Hadamard transform can be calculated by additions and subtractions only, it can be very easily implemented by the simplest neural model, whereas other holographic transforms require more complicated neurons or a more complicated network of simple neurons. For example, Velik (2008) used linear threshold neurons to calculate the discrete Fourier transform. Moreover, these transforms are not completely holographic in the sense that there is a major difference between the images obtained by removing different parts of the transform data. For example, in the Fourier transform, removing the high frequencies from the transformed data results in a significantly different image than removing the low frequencies. In our use of the Walsh–Hadamard transform and randomization, all parts of the transformed data will have almost the same effect. There are also holographic memory systems that save data in a holographic manner without prior transformation such as the Hopfield network (Hopfield, 1982) and Bidirectional Associative Memory networks (Kosko, 1988). However, these systems are not completely holographic. For example, removing several edges that are connected to the same node in the Hopfield network will result in losing information in that specific node (as well as some other nodes in some cases), which is not in line with the holographic desired property. Fig. 1 provides an example of the action of the Walsh–Hadamard transform on a $\{-1, 1\}^8$ vector.

The output vector is the result of multiplying the input vector by the Hadamard matrix. The input vector can be reconstructed from the output vector by multiplying the output vector by the transposed matrix (which is actually the same matrix) and dividing each entry by the width n (8 in the case of the example in Fig. 1).

2. Our contribution

Hadamard transforms are known and well researched transforms. In this work, we show how this transform can be used to convert data from classic representation to a holographic representation. Moreover, we provide a novel method for a single step error correction using a Hadamard transform. While Hadamard transforms are commonly used for error correction, our method combines the error correction with the holographic features of the transform to create a transform that corrects errors up to a certain threshold and provide graceful degradation of the data when this threshold is crossed. We also provide a method for creating Hadamard matrices without limitations on the length of the matrix, i.e., the only limitation on the length is that it must be less than the width. Except for this limitation, the length of the matrix can be any integer value. The main goal of our use of Hadamard transform is the graceful degradation in the presence of errors that characterizes holographic memory. We also provide a neural network computation of the transform as a proof of concept for feasibility.

3. Holographic memory using the Walsh–Hadamard transform

One can have a memory model with holographic features simply by saving the output vector (the result of the multiplication of the input vector by the Walsh–Hadamard matrix) instead of saving the input vector itself. For example, instead of saving a 2D image, one can save the transformed image obtained by concatenating the rows of the image array to a 1D vector, multiply the vector by the Walsh–Hadamard matrix, and save the result.

As mentioned above, the Walsh–Hadamard transform may be viewed as a holographic transform, as each coefficient contains information regarding the entire input vector. We show next how a corruption of parts of the transformed data results in a graceful degradation.

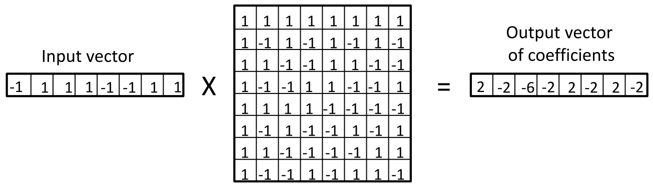

Fig. 1. Walsh–Hadamard transform on a $\{-1, 1\}^8$ vector.

3.1. *Graceful degradation in case of memory corruption*

If a coefficient is corrupted in the saved transformed vector, the resulting vector after the reconstruction of the input vector from all the coefficients will contain minor errors (small changes in the values) in all the entries, rather than major errors (large changes in the values) in some of the entries. This is due to the fact that the reconstruction is done by multiplying the saved vector by the same matrix and dividing by the number of rows of that matrix, meaning that each entry in the reconstructed vector is the result of adding and subtracting certain entries of the saved vector. If, for example, an entry is changed from x to x', then all the entries in the reconstructed vector will differ from their counterparts in the original vector exactly by $(x' - x)/n$, where n is the length of both vectors (as well as the number of rows and columns of the matrix).

The reconstructed vector may be regarded as a gracefully degraded version of the original vector. It does not contain one meaningful error; instead, the error is distributed over the whole vector. This can be best understood by thinking of an image. If the original vector is a vector of integers, or elements of some other discrete domain, the entries of the reconstructed vector should be rounded to the "nearest" legal value. In this case, we also get an error correction feature due to the rounding. For example, if after a corruption of the saved vector it is multiplied by the matrix, and each entry is then divided by n, then a value of an entry may change from 9 to 8.8. In this case it is rounded to 9 and the error has been corrected.

This example of a meaningful error being split into many small errors works for an input vector with integer or real values. For a binary input vector, there is no distinction between a minor and a major error because the only possible error is a bit flip. In this case we understand blurriness to mean that the error probabilities for all entries of the reconstructed vector are the same.

3.2. *Randomization for better coping with memory erasure*

The fact that the Walsh–Hadamard transform is holographic works for both errors and erasures. Since the work is inspired by the holographic brain theory and its implications on brain damage, and it is natural to assume that lesions are more similar to erasures than to errors, we focus in the sequel on coefficient erasures.

In case of a coefficient erasure, i.e., the situation where the coefficient is unknown, we treat the coefficient as if it vanishes. The characteristics of the Walsh–Hadamard matrix may, in some cases, cause a "poor distribution" issue. Namely, an input vector in $\{-1, 1\}^n$ that has a high correlation (in absolute value) with several columns of the Walsh–Hadamard matrix will yield a coefficient vector with a few relatively large coefficients and a lot of zeros or values close to zero. Due to the orthogonality of the columns in the Walsh–Hadamard matrix, the higher the correlation with one column is, the closer to zero the product with the other columns tends to be. In this case, an erasure of one of the few large coefficients will result in a major change in the reconstructed vector. In order to avoid such a scenario, we randomize a binary input vector by xoring it with a random input vector, and thus decrease the probability of a high correlation. In case of an input vector in $\{-1, 1\}^n$, the randomization can be done by multiplying