 Acta Univ. Sapientiae Informatica **14,** 2 (2022) 155–184

DOI: 10.2478/ausi-2022-0010

Experiments with holographic associative memory

Gábor Román
Eötvös Loránd University
Budapest, Hungary
email: rogpaai@inf.elte.hu

Abstract. We reiterate the theoretical basics of holographic associative memory, and conduct two experiments. During the first experiment, we teach the system many associations, while during the second experiment, we teach it only one association. In both cases, the recalling capability of the system is examined from different aspects.

1 Introduction

Holographic associative memory, or holographic neural network was presented by Sutherland [18, 19] as a new paradigm for artificial neural system design. As in the case of artificial neural networks, stimulus-response associations are taught to this system, and later the taught responses are recalled by presenting stimuli to the system. However, there are a few diversions from the conventional artificial neural systems.

First and foremost, information is represented by complex numbers, more precisely by phase angle orientation. Large number of associations are superimposed onto a single neural element, onto the same set of complex elements representing synaptic connections within a single neural cell. This representation allows us to manipulate the stored associations, enabling us to model

Computing Classification System 1998: I.5.1
Mathematics Subject Classification 2010: 68T07
Key words and phrases: holographic associative memory, artificial neural network, holographic neural technology, neurocomputing

short term-, long term-, or permanent memory. The learning of one association consists of one non-iterative transformation, without back-propagation or an iterative process. Recalling is also done by only one similar transformation.

Holographic associative memory has many applications, for example classification [15], facial expression analysis [3], large image database handling [7, 8], currency exchange rate prediction [13], data compression [11], land mine detection [20], and surface approximation [5], just to name a few areas. Note that we don't wish to present an extensive overview of neither the applications-, nor the bibliography of holographic neural networks.

We outline the theoretical basics of holographic associative memory in section 1.1. As there are many interfaces where the system can be fine tuned, we present our own setup in section 1.2. The first experiment is described in section 2. Here we examine how the system behaves when we teach it many associations. We conduct the experiment with different setups; the preparations for these, and their outcome can be found in sections 2.1, 2.2, 2.3, and 2.4. Our second experiment is described in section 3. Here we examine the system's behaviour when we only teach it one association. The results are presented in sections 3.1, and 3.2. The conclusions, and the possible areas of further research are presented in section 4. Finally, the results of the experiments are collected in appendix A.

1.1 Theory

The stimuli, and the corresponding responses are complex vectors $S \in \mathbb{C}^{1\times n}$ and $R \in \mathbb{C}^{1\times m}$ respectively; while the single neural element to which we teach the associations is a complex matrix $H \in \mathbb{C}^{n\times m}$. The standard way of teaching is

$$H \leftarrow H + S^*R \tag{1}$$

where the matrix A^* denotes the conjugate transpose of the matrix A. If the stimulus is given as $S := [\lambda_1 e^{i\theta_1}, \ldots, \lambda_n e^{i\theta_n}]$, with $\lambda_k \in \mathbb{R}_0^+$ and $\theta_k \in \mathbb{R}$ for $1 \le k \le n$; then the recalling happens as

$$R \leftarrow \frac{1}{\sum_{k=1}^{n} \lambda_k} SH \tag{2}$$

where the sum in the denominator is the normalisation coefficient. Note that we can batch together multiple stimuli into one matrix, then recall all of them with only one matrix-matrix multiplication, and some normalisations. Also note that the system can subsequently learn additional associations at any

given time, which is not true for traditional artificial neural networks, where one has to essentially restart the whole training in order to learn new information.

Let's look at the scenario, in which we have taught the associations (S_t, R_t) to the system for $1 \leq t \leq p$, and the magnitudes of the components were around 1. Imagine that we recall a new stimulus S'. We can write an arbitrary element of the response as

$$\gamma e^{i\phi} = \sum_{t=1}^{p} \Lambda_t e^{i\Psi_t} \tag{3}$$

with $\gamma, \Lambda_t \in \mathbb{R}_0^+$ and $\phi, \Psi_t \in \mathbb{R}$ for $1 \leq t \leq p$; where each component in the sum corresponds to one of the previously learned associations. If $S' \approx S_l$, where $1 \leq l \leq p$, then we can expect that $\Lambda_l \approx 1$, and $\Psi_l \approx \phi$; while $\Lambda_t \ll 1$ for $t \neq l$. We can also expect that these latter components will neutralise in a manner analogous to a random walk, see [18].

To teach the external data (being visual, audio, etc.) to the system, we need to convert it to the internal complex representation. This is done during the so called pre-processing step. During this step one can apply other transformations of course, for example one can prepare the data so that the above mentioned neutralisation becomes more prominent. One such method is called symmetrisation.

- One can compute the mean μ of the distribution of the phases in the stimulus, and the variance σ of the same distribution. Then the phases θ_k can be transformed using

$$\theta_k \leftarrow \frac{2\pi}{1 + e^{\frac{\mu - \theta_k}{\sigma}}} \tag{4}$$

  for $1 \leq k \leq n$. This is called the sigmoid transformation, and it maps the distribution, if it displays an approximate Gaussian form, to a fairly uniform state, see [14, 16, 18].

- If the distribution of the phases displays a different form, then one can still transform the phases θ_k as

$$\theta_k \leftarrow 2\pi F(\theta_k) \tag{5}$$

  for $1 \leq k \leq n$, where F is the cumulative function of the mentioned distribution, see [14, 16]. To apply this method, one has to have some knowledge about the nature of the stimuli, or has to approximate the distribution using the phases in the stimuli.

The counterpart of the pre-processing step is the post-processing step, which happens after the recall. During this step, one applies transformation to the response.

Next to the standard teaching scheme given by (1), there also exists a so called adjusted-, or enhanced teaching scheme. Assume that we want to associate a stimulus S with the response R, but instead of using (1), first we recall S using (2), gaining (say) R', and then we associate S with $R-R'$. In a nutshell, we teach only the difference to the system, thus we try to avoid the alteration of the already thought associations. So the adjusted teaching happens as

$$H \leftarrow H + S^* \left(R - \frac{1}{\sum_{k=1}^{n} \lambda_k} SH \right). \tag{6}$$

One can apply a given teaching scheme multiple times with the same associations, to reteach them to the system, effectively enforcing them. This is the so called iterative training, and with this latter adjusted teaching scheme it is quite powerful, as we will see.

As one teaches more and more associations to the system, the entries in the matrix H will change-, and most likely grow in magnitude. The average magnitude of the matrix entries approaches a theoretical saturation limit, which depends on the length of the stimuli n, see [18]. At this threshold, the storage capacity is reached, and fuzz becomes more dominant. Thus one has to apply some technique to prevent the saturation of memory.

- One can proportionally rescale the matrix entries, so that the average of all magnitudes remains at a predefined threshold, called the memory profile, see [12, 18]. This memory profile is expressed as the percentage of the saturation limit. A lower percentage results in a short-term memory, while a larger percentage results in a long-term memory.

- One can also "deactivate" those matrix entries, whose magnitude is below a predefined threshold, see [12]. This can be helpful if the matrix is sparsely stored, and we want to lower the required space.

These techniques can be applied either after each teaching, periodically, or when some condition is satisfied.

1.2 Our setup

During our experiments, we handled visual data, more specifically the red-green-blue (RGB) triplets coming from an OpenGL frame-buffer, where each

colour component is represented by a byte. The first task during pre-processing is to convert this external data into the internal complex representation. Let $\varepsilon_s := \exp(2\pi i/s)$, and $\mathbb{U}_s := \{\varepsilon_s^0, \ldots, \varepsilon_s^{s-1}\}$. We encoded each byte with the transformation $t : \{0, 1, \ldots, 255\} \to \mathbb{U}_{256}$ being given by $t(b) := \varepsilon_{256}^b$, so we encoded each of them as one of the 256th roots of unity. Note that this encoding could be used for other type of external data which we can represent as a sequence of bytes. Also note that the magnitudes of the vector components will be 1, thus the normalisation coefficient in expression (2) becomes simply n. This way, the information is carried by the phase angles of the complex elements, while the magnitudes can be treated as confidence levels.

During the symmetrisation of a stimulus, we computed the mean of the distribution of the phases θ_k as

$$\mu \leftarrow \frac{1}{n} \sum_{k=1}^{n} \theta_k \tag{7}$$

the variance of the mentioned distribution as

$$\sigma \leftarrow \sqrt{\frac{1}{n} \sum_{k=1}^{n} (\theta_k - \mu)^2} \tag{8}$$

and applied the sigmoid transformation (4), thus silently assuming that the distribution of the phases display a Gaussian form. As we will see, this was already beneficial, although it worths to investigate the distribution of the input more thoroughly in real world applications.

The linear algebraic computations can be sped up by delegating the work to the video card of the machine. To achieve this, we used the OpenCL BLAS package, see [23]. Both the standard teaching (1) and the recalling (2) can be implemented using either `clblasCgemm`, or `clblasZgemm` which are able to realise the computation

$$C \leftarrow \alpha A^* B + \beta C \tag{9}$$

on float-, or double complex elements respectively. (To implement the adjusted teaching (6), one has to combine the standard teaching and the recalling.) As the normalisation coefficient is n, it can be seen that we can precompute $1/n$, and pass it to the applied function as a constant multiplier.

2 Experiments with many associations

We examined the recalling capabilities of the holographic associative memory while teaching it many images. For this end, we used the so called CIFAR-10 dataset, see [9]. Every image in this dataset has a width of 32 pixel, and a height of 32 pixels. As every pixel is given by an RGB triplet, we have that $n = 3072$ in this case. (Next to its abundance, this is the second reason why this dataset has been chosen: n is small.)

Our goal was to examine how long the system can retain an association while continuously fed with new associations. We taught associations in rounds. In each round, we taught a new image to the system; and after, we recalled all the previously taught images. (We've taught 128 images to the system from the first batch of the binary version of CIFAR 10, in 128 iterations.) The results of these recalls were compared with the taught responses.

To improve the performance of the system, we choose the responses as follows.

- We used the elements of $\mathbb{U}_2$ (that is 1 and -1) as the complex elements of the responses, to be able to associate the different stimuli with binary numbers. (Note that the magnitude of the elements is yet again 1.)

- These binary numbers were selected from a block code of length m, with a minimum Hamming distance d between the elements of the block code see for example [10].

When we taught a new association, we generated a new codeword as response R, and also stored it separately for later comparisons. (Our generation process was fairly simple: independently of m and d, the first codeword was always 0; and we've always searched for the next codeword sequentially, comparing with the previously generated codewords to maintain the minimum distance between the codewords.)

After a recall, the output can contain noise, so during the post-processing step we had to decide about the value of the complex elements in the recalled response R'. If the principal value of the phase fell into $(-\pi/2, \pi/2]$, then we've mapped the component to 1, otherwise to -1.

We would like to mention that we could have chosen the elements of $\mathbb{U}_3$, $\mathbb{U}_4$, etc. as the complex elements of the responses. This way we could have used ternary-, quaternary-, etc. numbers, so we could have encoded much larger set of numbers while keeping m constant. However, as we will see, the phase errors can be huge during recalling, so using even two digits is already challenging.

For example, using $\mathbb{U}_3$, phases falling in $(-\pi/3, \pi/3]$ would map to 1, the phases falling in $(\pi/3, \pi]$ would map to $(-1 + i\sqrt{3})/2$, and finally the phases falling in $(-\pi, -\pi/3]$ would map to $(-1 - i\sqrt{3})/2$. As the number of "digits" grow, the chance of misinterpretation grows too, because the "domains" of the roots get smaller and smaller.

Hypothetically, one could look at the magnitude of the complex elements really as confidence levels. If the confidence level is too low, we could switch to another digit. This is especially handy when we use binary code, because in this case we can only change to the other digit. It might happen that the interpretation of the confidence should depend on how close the phase is to a border of a "domain". For example when the phase is way inside the domain, we would need a low confidence level to switch digits, while near a border only a higher confidence level could suffice. We haven't conducted any experiments in this area.

After mapping the complex elements of R' to either 1 or -1, we got R'', which we compared with the elements of our block code using the following algorithm. It returns the evaluation of the recall, which can be either *perfect*, *confident*, *ambiguous*, or *faulty*.

1. Try to find a codeword c, whose distance from R'' is at most $\lfloor (d-1)/2 \rfloor$. If there doesn't exist such c, then go to step 3.

2. If $c = R$, then set the result to *perfect*, otherwise to *faulty*, and terminate the algorithm.

3. Let C be the set of those codewords which distance from R'' is smaller than d. If $C = \{R\}$, then set the result to *confident*, and terminate the algorithm. If $|C| > 1$, and $R \in C$, then set the result to *ambiguous*, and terminate the algorithm.

4. Set result of the algorithm as *faulty*, and terminate the algorithm.

As we will see, increasing the value of d will improve the performance of the system, while on the other hand, it will decrease the size of the block code. Denote with $A_q(m, d)$ the maximal size of a q-ary block code of length m, and minimum Hamming distance d between the elements of the block code. Based on the articles [4, 6], we have

$$\frac{q^m}{\mathrm{Vol}_q(m, d-1)} \leq A_q(m, d) \leq \frac{q^m}{\mathrm{Vol}_q(m, \lfloor (d-1)/2 \rfloor)} \tag{10}$$

where

$$\mathrm{Vol}_q(m, r) := \sum_{j=0}^{r} \binom{m}{j} (q-1)^j \tag{11}$$

denotes the volume of a Hamming ball of radius r in the space of the strings with length of m, composed from the digits $\{0, 1, \ldots, q-1\}$. When q is a prime power, then based on the article of Varshamov [22], we have

$$q^k \leq A_q(m, d) \tag{12}$$

where k is the greatest integer for which

$$q^k < \frac{q^m}{\mathrm{Vol}_q(m-1, d-2)} \tag{13}$$

holds. One can compute the required m using these inequalities, based on the number of associations which we plan to teach to the system and the distance d. We fixed the value $m = 32$ during our experiment. We examined the norms

$$\|H\|_{\max} := \max_{i,j} |H_{i,j}| \tag{14}$$

and

$$\|H\|_{\mathrm{avg}} := \frac{1}{nm} \sum_{i,j} |H_{i,j}| \tag{15}$$

to see how the magnitudes of the entries in H change. These norms can be calculated with the help of the functions `clblasiCamax` and `clblasScasum`, or `clblasiZamax` and `clblasDzasum` respectively for float-, or double complex elements.

We present our result in a specific form, see Figure 1. The iterations are displayed from left to right. In each column, we taught a new image to the system. The position at the ith row, and jth column correspond to the evaluation of the recall of the ith image in the jth iteration. The squares filled with the darkest grey correspond to *perfect* recalls, the ones filled with a lighter grey correspond to *confident* recalls, and the ones filled with the lightest grey correspond to *ambiguous* recalls. *Faulty* recalls are designated by the absence of a square at a given position. Although we've distinguished the different types of evaluations with different colours, what is more important is the shape of the diagram, and the transition of the colours.

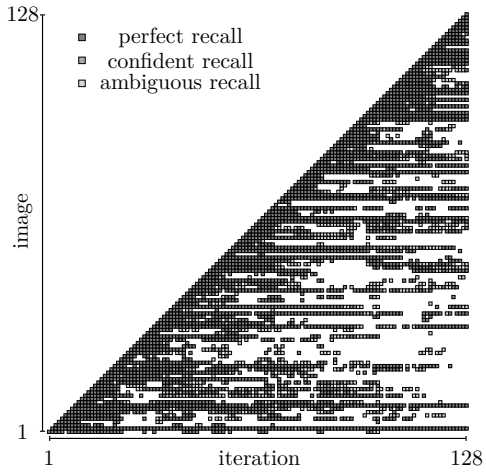


Figure 1: Example diagram of the iterations.

2.1 Basic case

In the basic case, we haven't applied any special transformations, only the method which we've described in section 2 with either the standard teaching, or the adjusted teaching.

The actual results can be seen in Figures 3, and 4. We conducted the experiment in both cases for the distance $d = 3$, see Figures 3(a) and 4(a); furthermore for the distance $d = 11$, see Figures 3(b) and 4(b).

Using the standard teaching method, it seems that the recalling is more successful for the images which we've taught earlier, see Figures 3(a) and 3(b). Observe that both norms grow somewhat linearly. The norm $\|H\|_{\max}$ grows in accordance with the number of taught images, or at least after a few iterations. This can pose a problem if one wants to teach many associations to the system, because of the saturation which we've explained in section 1.1. Now we give an explanation to the linear growth of $\|H\|_{\max}$.

Proposition 1 *Let* $n, m > 0$, $p, q \geq 2$, $S_1, S_2, \ldots \in \mathbb{U}_p^n$, *and* $R_1, R_2, \ldots \in \mathbb{U}_q^m$. *Furthermore, let* $H \in \mathbb{C}^{n \times m}$ *be the zero matrix. Then let* $H \leftarrow H + S_i^* R_i$ *first for* $i = 1$, *then for* $i = 2$, *and so on. After the* k*th such iteration we will have*

$$\|H\|_{\max} \leq k. \tag{16}$$

Proof. During matrix addition, the elements at different positions don't affect each other, so it is enough to assume that the phase of the elements at only *one*

given position in the products $S_1^* R_1, S_2^* R_2, \ldots$ have the same principal value. This way, the magnitude of the element at this position will be the greatest in H, and we can use it as an upper bound for $\|H\|_{\max}$. The elements of the stimuli S_i and the responses R_i are roots of unity, so the elements in their product $S_i^* R_i$ will have a magnitude of 1. Thus the magnitude of the element at the mentioned position will be k after teaching all the association up till the kth. □

Not surprisingly, the recalling capability becomes better as we use a larger distance in our block code. Nevertheless, we've conducted our experiments with both $d = 3$, and $d = 11$, to see the difference between the two distances.

We can see in Figures 4(a) and 4(b), that applying the adjusted teaching offers a clear advantage over the standard teaching, as the retaining becomes much better. Now the recalling becomes more successful for the more recently taught images. The norm $\|H\|_{\max}$ shows a hectic-, but also bounded behaviour. But the norm $\|H\|_{\text{avg}}$ still grows, so one still needs to handle the growth in the case when many images should be taught to the system, to avoid saturation. Note however that the growth is slower than in the previous case.

2.2 Preprocessing

During this part of the experiment, we repeated the same process, as in section 2.1, but this time we symmetrised the stimuli, as it is explained in section 1.2.

The results can be seen in Figures 5, and 6. Concerning the distance $d = 3$, see Figures 5(a) and 6(a); and for the distance $d = 11$, see Figures 5(b) and 6(b).

It can be seen that the application of the symmetrisation results in a slightly better recalling, so it worth spending time on it. Note however, that the characteristics of the different teaching methods remained the same. That is, when using the standard teaching method, the recalling is more successful for the images taught at the beginning; while when using the adjusted teaching method, this holds true for the more recently taught images. The norm $\|H\|_{\max}$ grows a bit slower when using the standard teaching method, however it shows a higher variance when using the adjusted teaching method. As for the norm $\|H\|_{\text{avg}}$, it grows similarly to the basic case, that is when the symmetrisation was not applied.

2.3 Rescaling

In this case, we repeated the process described in section 2.1, but we multiplied our matrix H with a decay term λ every time when we learned a new association. (Except after the first learning of course.) To multiply the entries in the matrix, one can use the functions `clblasCsscal`, or `clblasZdscal` respectively for float-, or double complex elements. Of course one can economise, and rescale the entries during teaching by incorporating this λ into α and β, see operation (9). We conducted this part of the experiment with three different decay terms, namely with $\lambda = 0.9$, $\lambda = 0.95$, and finally $\lambda = 0.975$.

The results can be seen in Figures 7 and 8. For the case of standard teaching, and distance $d = 3$, see 7(a), 7(d) ($\lambda = 0.9$), 7(b), 7(e) ($\lambda = 0.95$), and 7(c), 7(f) ($\lambda = 0.975$); for distance $d = 11$, see 7(g), 7(j) ($\lambda = 0.9$), 7(h), 7(k) ($\lambda = 0.95$), and 7(i), 7(l) ($\lambda = 0.975$). As for the case of adjusted teaching, and distance $d = 3$, see 8(a), 8(d) ($\lambda = 0.9$), 8(b), 8(e) ($\lambda = 0.95$), and 8(c), 8(f) ($\lambda = 0.975$); for distance $d = 11$, see 8(g), 8(j) ($\lambda = 0.9$), 8(h), 8(k) ($\lambda = 0.95$), and 8(i), 8(l) ($\lambda = 0.975$).

As expected, the recalling is only successful for the images which we've taught recently, and the time of retention grows as λ approaches 1. When applying standard teaching, both the norm $\|H\|_{\max}$ and $\|H\|_{\text{avg}}$ seem to stop growing after reaching a certain limit. As for the adjusted teaching, the norm $\|H\|_{\max}$ shows a hectic-, but bounded behaviour again, and the norm $\|H\|_{\text{avg}}$ seems to stop growing over a certain limit in this case too. Thus one can prevent the saturation of memory by applying a decay term, however its value should be chosen based on the required retention time and the threshold which we are willing to reach with the presented norms.

Proposition 2 *Let* $\lambda \in (0,1)$, $n, m > 0$, $p, q \geq 2$, $S_1, S_2, \ldots \in \mathbb{U}_p^n$, *and* $R_1, R_2, \ldots \in \mathbb{U}_q^m$. *Furthermore, let* $H \leftarrow S_1^* R_1$. *After this, let* $H \leftarrow \lambda(H + S_i^* R_i)$ *first for* $i = 2$, *then for* $i = 3$, *and so on. During this process, we have*

$$\|H\|_{\max} < \frac{1}{1-\lambda} - 1. \tag{17}$$

Proof. The proof will be similar to the one given for proposition 1. We assume that the phase of the elements at only *one* given position in the products $S_1^* R_1, S_2^* R_2, \ldots$ have the same principal value. By defining $f(x) := \lambda(x+1)$, one can see that

$$\|H\|_{\max} \leq f^k(x)\Big|_{x=1} \tag{18}$$

will hold after the kth step, where f^k is the kth iterate of f. Indeed, the magnitude of the element at the mentioned position will be 1 at the beginning, $\lambda(1+1)$ after the first step, $\lambda(\lambda(1+1)+1)$ after the second step, and so on. The right side of the above inequality has a simple form for $k > 1$, namely

$$2\lambda^k + \lambda^{k-1} + \ldots + \lambda = 2\lambda^k + \frac{1-\lambda^k}{1-\lambda} - 1 \xrightarrow[k\to\infty]{} \frac{1}{1-\lambda} - 1 \tag{19}$$

as $\lambda \in (0,1)$. □

For $\lambda = 0.9$, we get $\|H\|_{\max} < 9$; for $\lambda = 0.95$, we get $\|H\|_{\max} < 19$; and for $\lambda = 0.975$, we get $\|H\|_{\max} < 39$. Observe that these values give a good bound for the norms $\|H\|_{\max}$ which can be observed in Figures 7(d), 7(e), and 7(f) respectively.

During recalling, one has to sum up n elements with magnitudes which can reach this limit, to get one element of the recalled response. Using this result, one can pose a limit on the magnitude of the elements, and effectively avoid floating point miscalculations. Note that by using symmetrisation and adjusted teaching, the situation becomes better, and we can chose a decay term even closer to 1.

2.4 Iterative training

Although the teaching itself is not an iterative process, we can extend it to be one, namely by reteaching those associations which we've recalled perfectly. This iterative training only makes sense when one uses the adjusted teaching, otherwise the system would perform poorly using the standard teaching. During this part of the experiment, we've essentially repeated what we did during the basic case, see section 2.1, but after the teaching part, when we checked how well the system could recall the already stored associations, we stored the indexes of those associations, which the system recalled perfectly. After the recalling step, we applied the adjusted teaching for these indexes, but with the original response of the given association, so we readjusted the original response, but to the current state of the system.

The results can be seen in Figure 9. We examined this iterative training with distance $d = 3$, see Figure 9(a), and with distance $d = 11$, see 9(b).

As we can see, the recalling capability of the system became quite good, however we have to keep in mind that we've only checked 128 associations. It can happen that the performance of the system starts degrading as we try to maintain more and more associations in this iterative manner. The norms $\|H\|_{\max}$ and $\|H\|_{\text{avg}}$ show similar behaviour as when we've applied the

adjusted teaching in the basic case, however the variance of $\|\mathsf{H}\|_{\max}$ is greater, and $\|\mathsf{H}\|_{\text{avg}}$ grows a bit faster.

3 Experiments with one association

In our second experiment, we worked with OpenGL to render a three dimensional model, capture the scene, and teach it to our system after encoding. The response which we taught to the system was always 1. (So this time $m = 1$.) The model was rendered in grayscale, lit by one single light source positioned at $x = 1$, $y = 1$, $z = 1$. The background of the scene was coloured with a non-grayscale colour. When the teaching was done, we rotated the model around different axes; or moved the camera closer to-, or farther from the model; and checked how well the system can recall the transformed model. Our choice fell onto the famous teapot model, see Figure 2.

The width of the scene was 320 pixels, and its height was 240 pixels, which means that in this case $n = 230400$. (Using the capabilities of the video card, the system still performed well.) The encoding happened as we've explained in section 1.2, except that the bytes of the background colour were mapped to 0. We repeated the experiment while using symmetrisation.

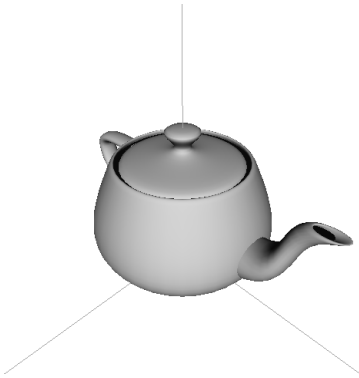


Figure 2: The teapot model which we used during our experiment, rendered at the position $x = 0$, $y = 0$, and $z = 0$ without any rotations. The lines represent the beginning of the axes' positive portion. The line following the spout of the teapot corresponds to the x axis, the line going upward corresponds to the y axis, and the third remaining line corresponds to the z axis.

The renderings which we taught to the system can be seen in Figure 10. As the response which we taught to the system was always 1, we only checked the

difference of the phase angle of the recalled response from 0. So on the figures corresponding to this experiment, we marked with $|\delta|$ the absolute value of the measured phase difference.

3.1 Rotation

During this part of the experiment, the transformation which we applied to our model was rotation around two axes with separate angles. The results can be seen of Figures 11, 12, 13, 14, 15, and 16. The axes are listed below the figures. The value of α corresponds to the rotation around the first axis, and the value of β corresponds to the rotation around the second axis. The subfigures in the second line of the figures correspond to the cases when symmetrisation was applied.

For example, Figure 11 corresponds to the case when the model was rendered as in Figure 10(a); in Figure 11(a) α corresponds to the rotation around axis x, and β corresponds to the rotation around axis y. In Figure 11(d) we can see the results which we gained while repeating the same experiment, but while using symmetrisation. The experiments for the other axis combinations can be seen in Figures 11(b), 11(e), and 11(c), 11(f). We can see in Figures 11(a) and 11(d), that as the value of β changes, the graph stays nearly the same, so the rotation around the y axis doesn't change the outcome of the recalling. This can be explained by the quasi rotational symmetry of the teapot model with respect to this axis. However, when the value of α change, the phase difference starts to change drastically. This is due to the fact that the teapot model doesn't show any symmetry with respect to the x axis.

The results for the well-lit renderings (see Figures 10(a), 10(b), and 10(c)) seem to reflect the quasi rotational symmetry of the teapot model. This is quite prominent on Figures 11(a), 12(a), and 13(a). The results in Figures 11(b), 11(c), 12(b), 12(c), 13(b), and 13(c) show similar behaviour, but they are more complex.

However, the results for the silhouette-like renderings (see Figures 10(d), 10(e), and 10(f)) show no particular pattern, see Figures 14(a), 14(b), 14(c), 15(a), 15(b), 15(c), 16(a), 16(b), and 16(c). This could be because of the simpler structure of H, and because the body of the teapot keeps nearly the same silhouette under rotation.

As for the symmetrisation, in the case of well-lit renderings, it makes the graphs "smoother", see Figures 12(d), 12(e), 12(f), 13(d), 13(e), and 13(f); while for the silhouette-like renderings, it introduces a "fuzziness", see Figures 14(d), 14(e), 14(f), 15(d), 15(e), 15(f), 16(d), 16(e), and 16(f).

3.2 Scaling

In this part of the experiment, as we've described earlier, we moved the camera closer to-, or farther from the model. To avoid the situation where some part of the model gets rendered outside the frame-buffer, we adjusted the position of the model before teaching. The new camera positions were $x = 4$, $y = 0$, $z = 0$ for 10(a); $x = 0$, $y = 4$, $z = 0$ for 10(b); $x = 0$, $y = 0$, $z = 4$ for 10(c); $x = -4$, $y = 0$, $z = 0$ for 10(d); $x = 0$, $y = -4$, $z = 0$ for 10(e); and $x = 0$, $y = 0$, $z = -4$ for 10(f). During the experiment, we multiplied the camera's position with a factor γ, which we continuously incremented with a small delta from $1/2$ to 4.

The results can be seen in Figure 17. The results for the well-lit renderings can be seen in Figures 17(a), 17(b), and 17(c); while the results for the silhouette-like renderings can be seen in Figures 17(g), 17(h), and 17(i). The results for the same renderings repeated with symmetrisation can be seen in Figures 17(d), 17(e), 17(f), 17(j), 17(k), and 17(l) respectively.

As expected, the graphs have their minimum at 1, except for the results 17(a) and 17(d), where it seems that the graphs approach 0 at multiple values of γ. Most probably, the results are only "correct" when $\gamma \in (1 - \varepsilon, 1 + \varepsilon)$, where ε is small, say around 0.2, or 0.3. Outside this area, the results seem to be merely noise, saturated at a given level. (Interesting however that the error stayed below $\pi/2$, but this can be due to the fact that we taught only one rendering to the system.) Observe that in both the well-lit cases, and in the silhouette-like cases, symmetrisation slows down the growth of the error in this small area around 1. It seems that the system is able to identify a scaled object when it is close to the original rendering.

4 Conclusions and further research

When one teaches many associations to the system (section 2) it worths combining the adjusted teaching with symmetrisation (section 2.2). (Note that the standard teaching is an easily invertible operation, so it would be convenient if we would want to remove associations; however the adjusted teaching performs much better.) If the recalling is perfect, then it also worths applying iterative training (section 2.4). To avoid the saturation of the memory, one must use rescaling (section 2.3). Our proposition 2 gives a starting point for the choice of the decay term λ. However the behaviour of the norms $\|H\|_{\max}$ and $\|H\|_{\text{avg}}$ could be investigated deeper using the theory of random matrices,

see for example [1, 2, 17, 21]. It would worth observing how the system behaves when one teaches it more associations.

Based on the results of section 3, we can expect that system can recall three dimensional objects if they show some (quasi) rotational symmetry, and they are rotated; or they are scaled a bit. It would be interesting to examine how the system performs when one teaches it renderings of many three dimensional objects. (Either when performing rotations, or scaling.)

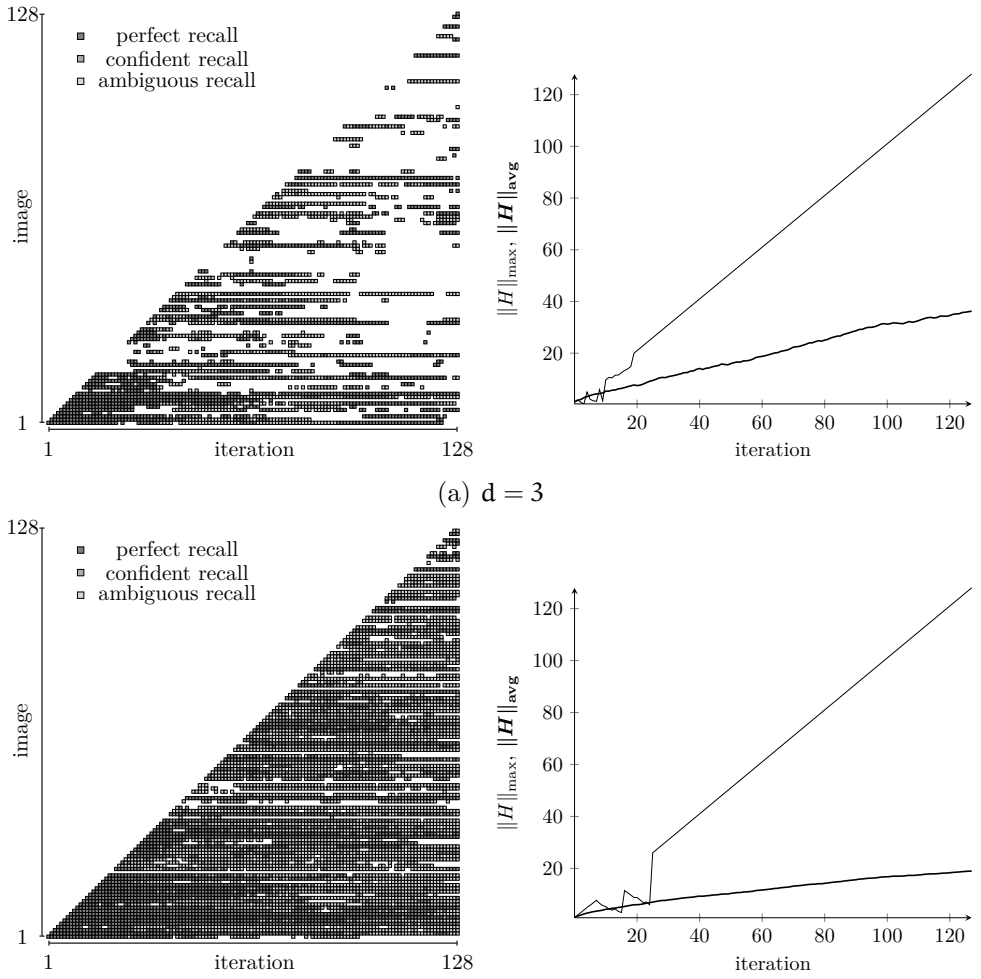
One could also teach the system renderings of the same object but from different angles. (Let's call this "sampling" of the same object.) We would expect that if the teaching happens from many different angles, which essentially cover the whole object, then the error could be kept under a certain level despite how the object is rotated during recalling. One could examine how the error levels change with the different samplings, and what would be the optimal sampling, if there is such.

5 Acknowledgments

The author wishes to thank Robert Manger for their correspondence, and the reviewer for the helpful remarks.

A Figures

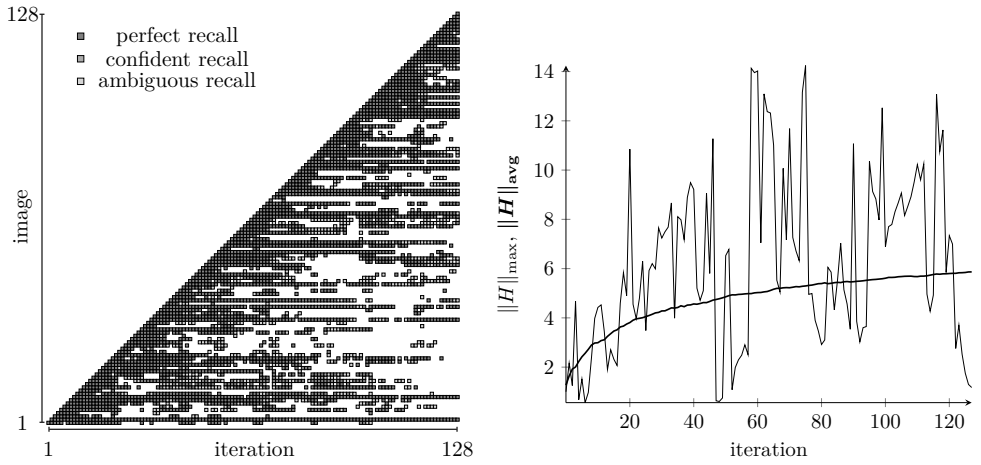
We collected here the figures corresponding to the results of our experiments, so they won't break the flow of the text of the article.



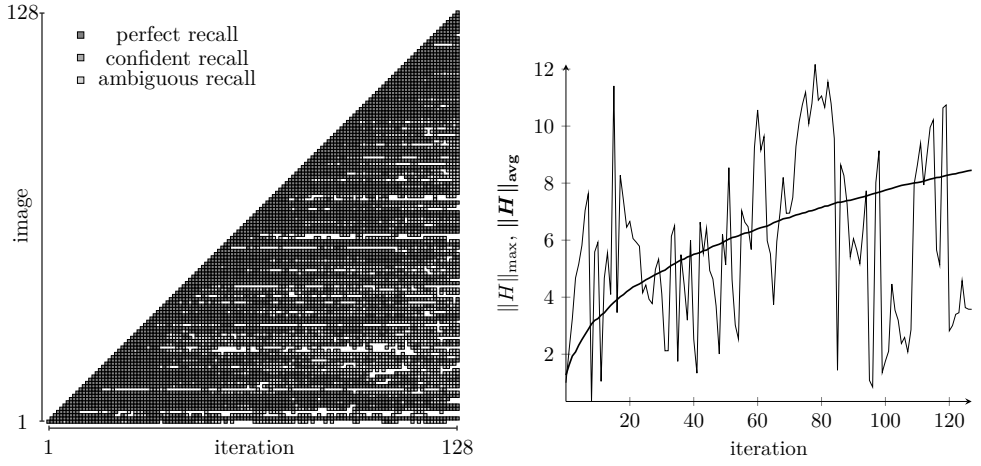
(a) $\mathsf{d} = 3$

(b) $\mathsf{d} = 11$

Figure 3: Basic case, standard teaching. See end of section 2 for the detailed description about the interpretation of the figures on the left hand side.

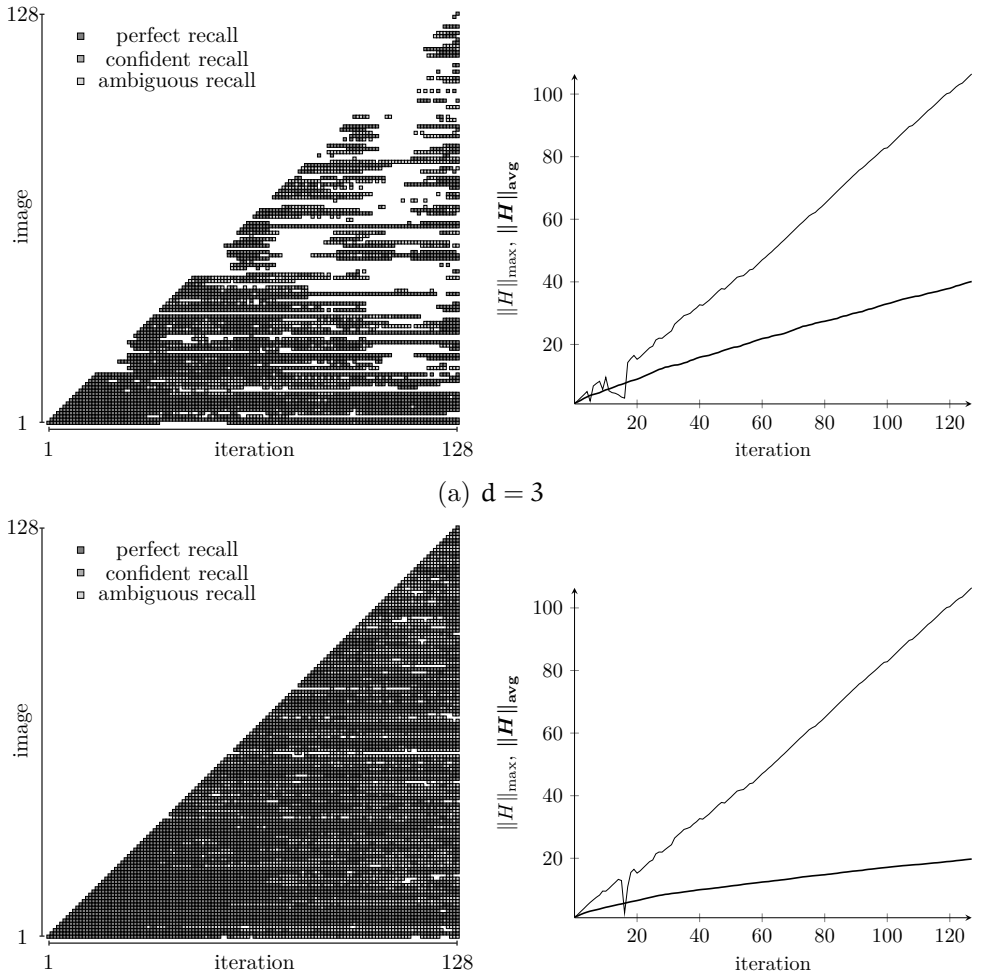


(a) $\mathsf{d} = 3$



(b) $\mathsf{d} = 11$

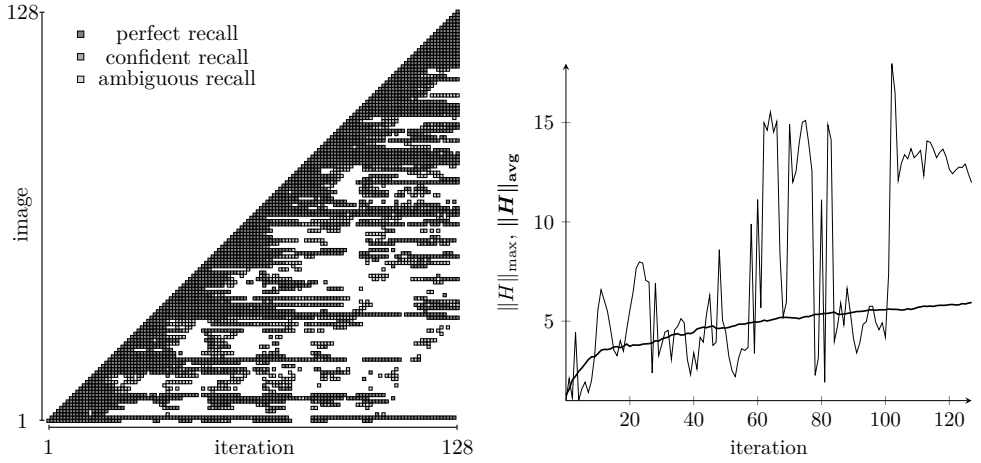
Figure 4: Basic case, adjusted teaching. See end of section 2 for the detailed description about the interpretation of the figures on the left hand side.



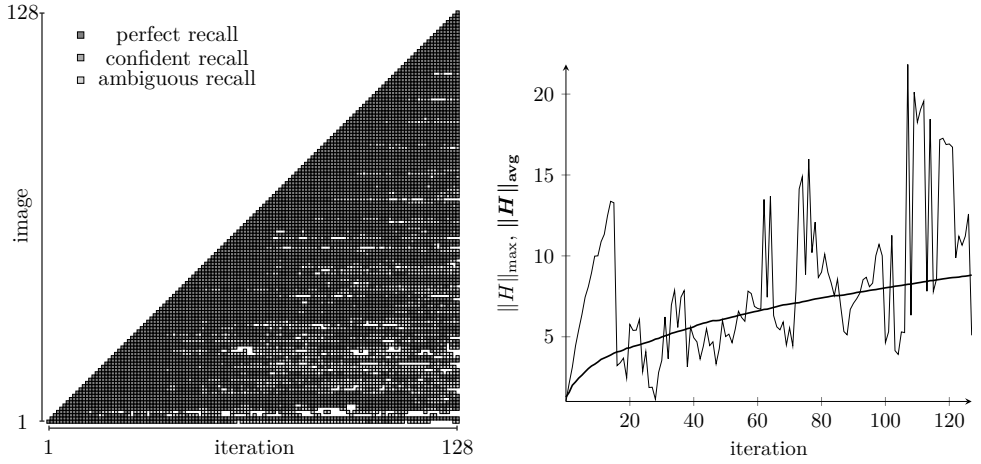
(a) d = 3

(b) d = 11

Figure 5: Preprocessing, standard teaching. See end of section 2 for the detailed description about the interpretation of the figures on the left hand side.

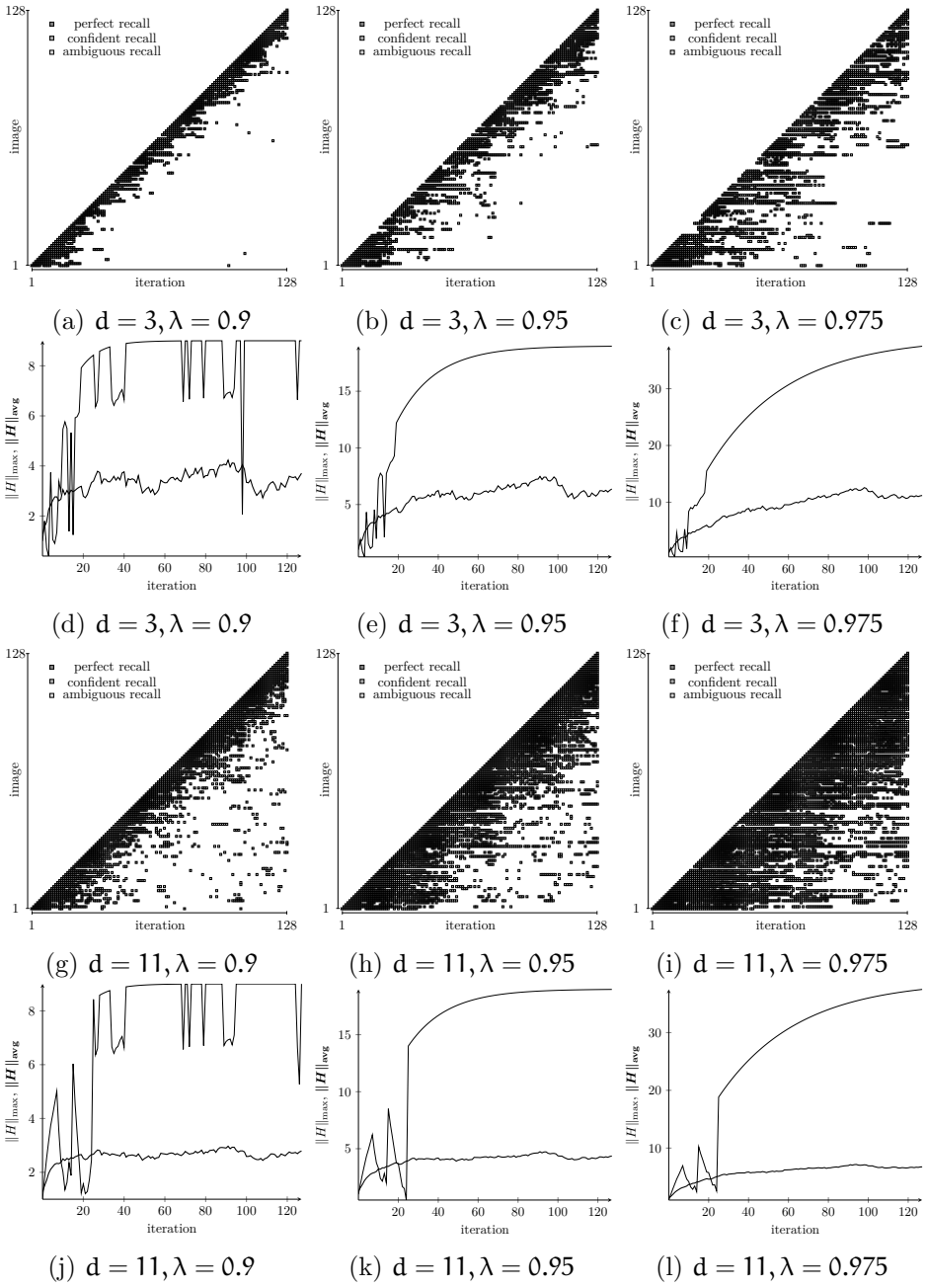


(a) d = 3



(b) d = 11

Figure 6: Preprocessing, adjusted teaching. See end of section 2 for the detailed description about the interpretation of the figures on the left hand side.



(a) $d = 3, \lambda = 0.9$ (b) $d = 3, \lambda = 0.95$ (c) $d = 3, \lambda = 0.975$

(d) $d = 3, \lambda = 0.9$ (e) $d = 3, \lambda = 0.95$ (f) $d = 3, \lambda = 0.975$

(g) $d = 11, \lambda = 0.9$ (h) $d = 11, \lambda = 0.95$ (i) $d = 11, \lambda = 0.975$

(j) $d = 11, \lambda = 0.9$ (k) $d = 11, \lambda = 0.95$ (l) $d = 11, \lambda = 0.975$

Figure 7: Rescaling with standard teaching. See end of section 2 for the detailed description about the interpretation of the figures in the odd rows.

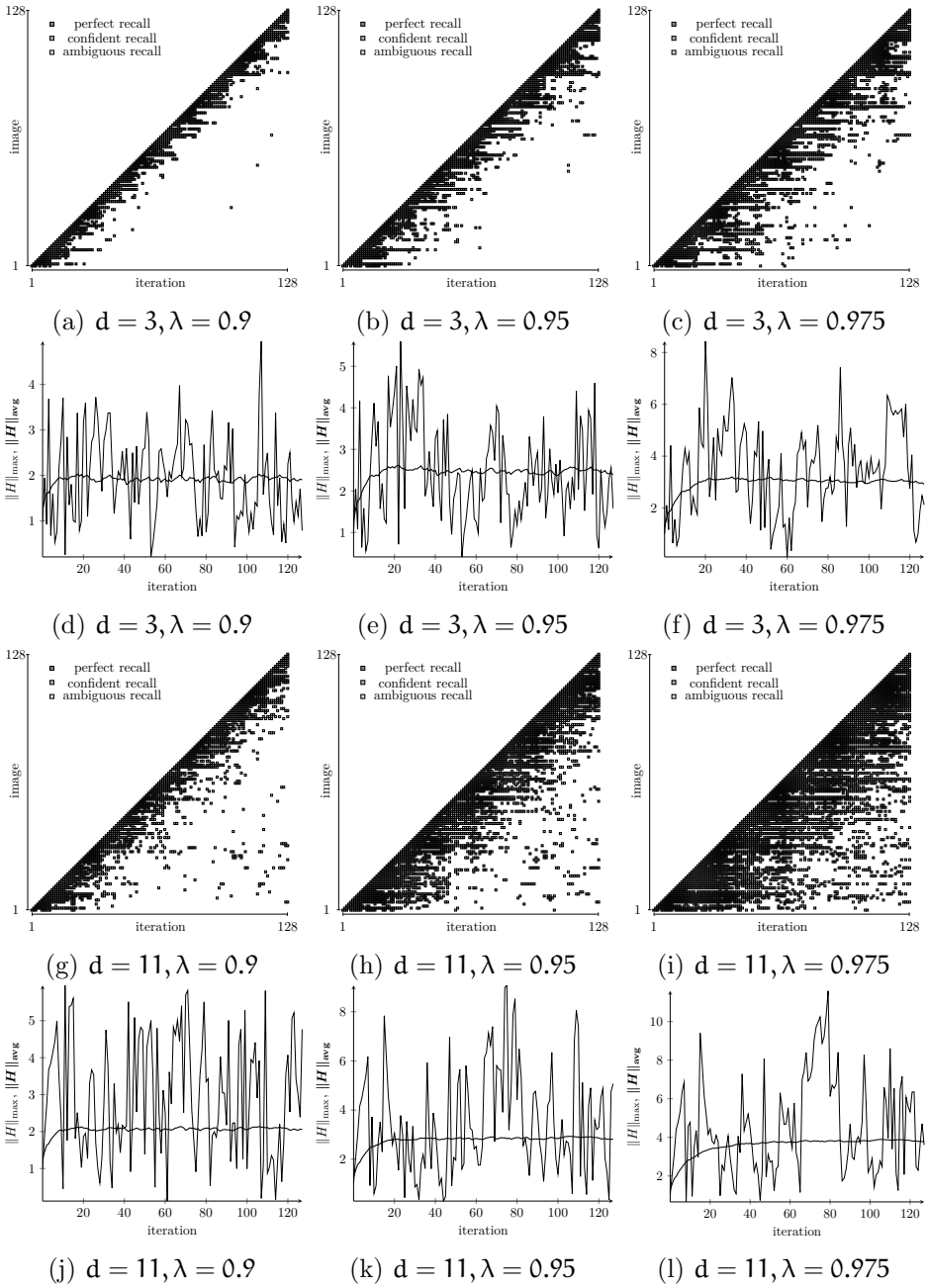
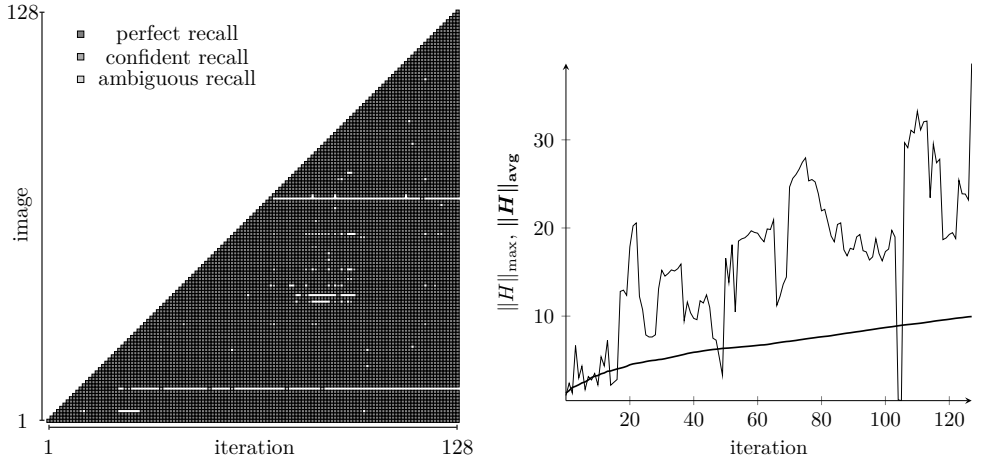
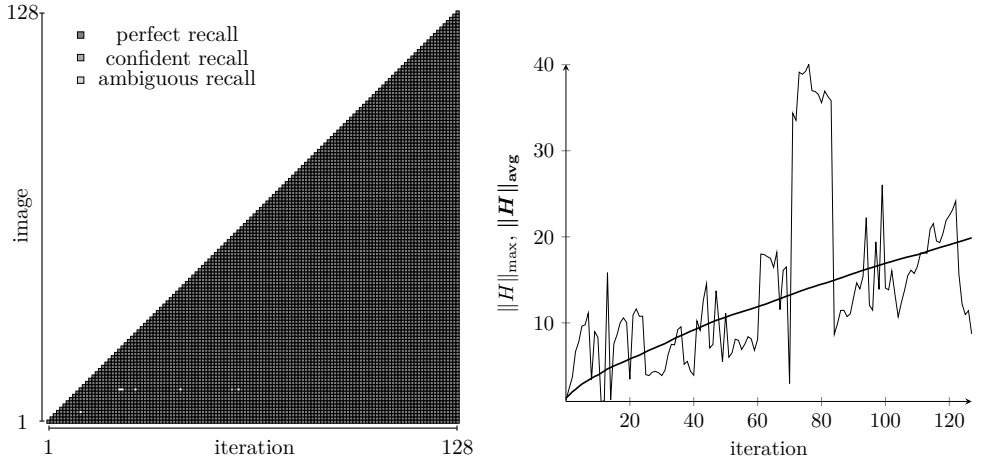


Figure 8: Rescaling with adjusted teaching. See end of section 2 for the detailed description about the interpretation of the figures in the odd rows.



(a) d = 3



(b) d = 11

Figure 9: Iterative training. See end of section 2 for the detailed description about the interpretation of the figures on the left hand side.

(a) $x = 2$, $y = 0$, $z = 0$

(b) $x = 0$, $y = 2$, $z = 0$

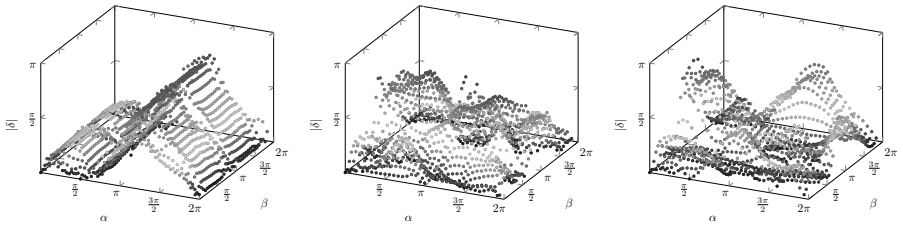
(c) $x = 0$, $y = 0$, $z = 2$

(d) $x = -2$, $y = 0$, $z = 0$

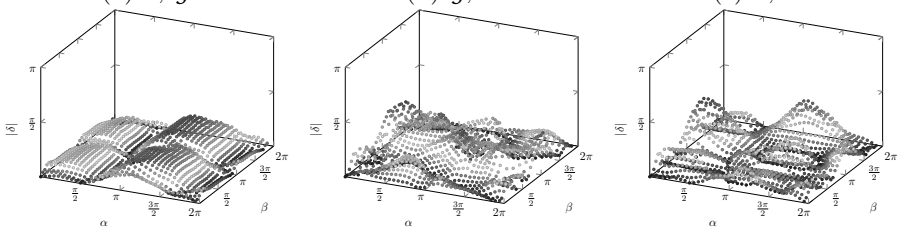
(e) $x = 0$, $y = -2$, $z = 0$

(f) $x = 0$, $y = 0$, $z = -2$

Figure 10: The different renderings of the teapot model, which renderings we taught to the system. The coordinates designate the position of the camera.

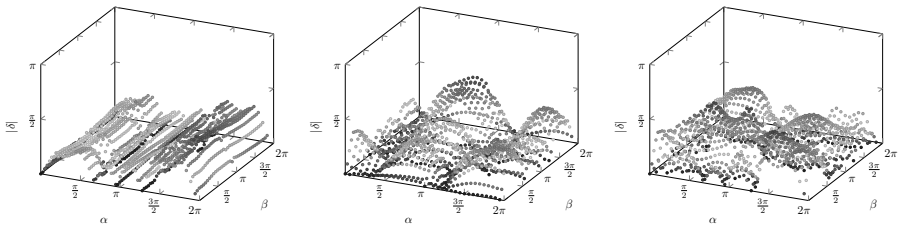


(a) x, y (b) y, z (c) x, z

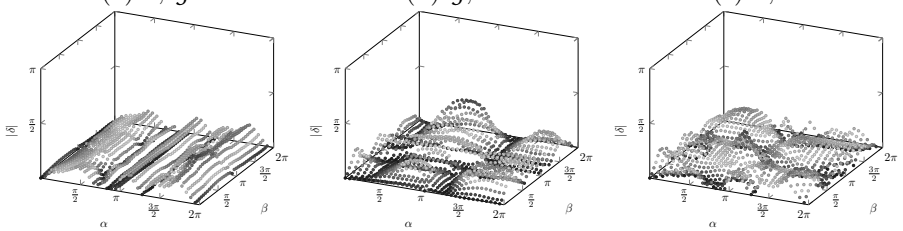


(d) x, y (e) y, z (f) x, z

Figure 11: Rotating, after teaching the rendering 10(a). The rotation happened around the noted axes separately with angles α and β. The third axis marks the absolute value of the measured phase difference. For further explanation about the graphs see section 3.1.

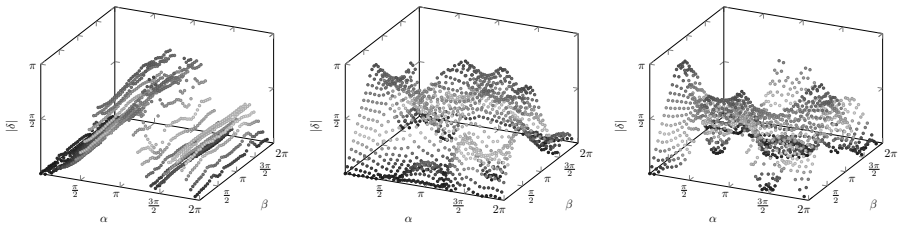


(a) x, y (b) y, z (c) x, z

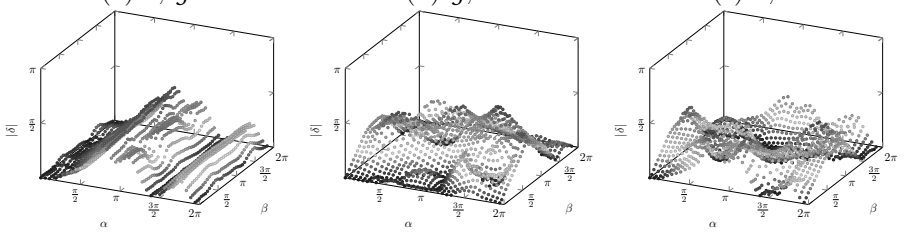


(d) x, y (e) y, z (f) x, z

Figure 12: Rotating, after teaching the rendering 10(b). For further explanation about the graphs see section 3.1 or Figure 11.

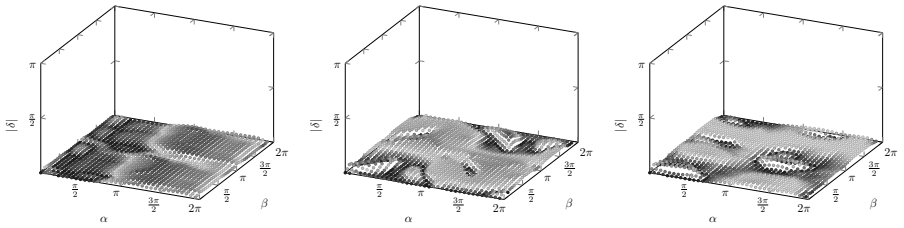


(a) x, y (b) y, z (c) x, z

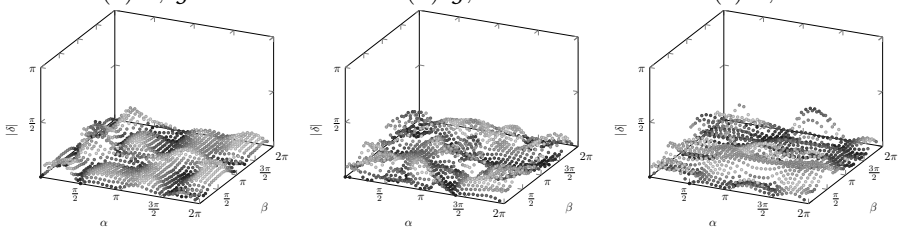


(d) x, y (e) y, z (f) x, z

Figure 13: Rotating, after teaching the rendering 10(c). For further explanation about the graphs see section 3.1 or Figure 11.

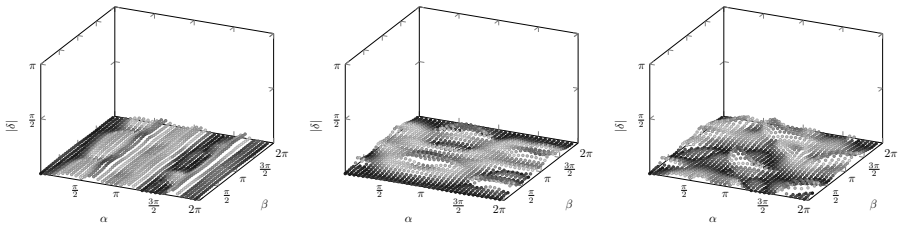


(a) x, y (b) y, z (c) x, z

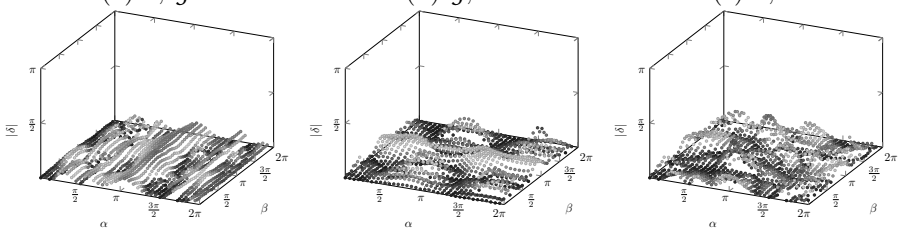


(d) x, y (e) y, z (f) x, z

Figure 14: Rotating, after teaching the rendering 10(d). For further explanation about the graphs see section 3.1 or Figure 11.

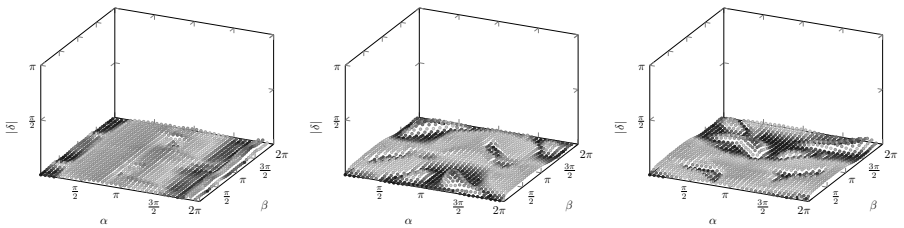


(a) x, y (b) y, z (c) x, z

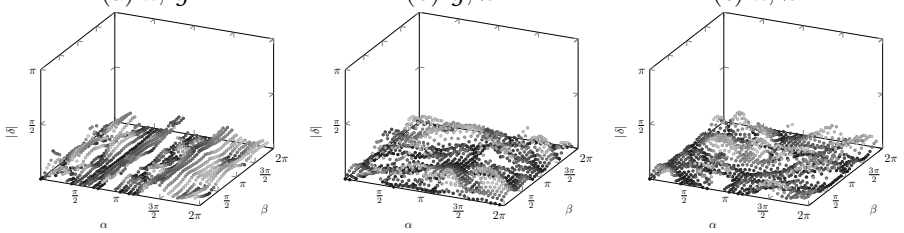


(d) x, y (e) y, z (f) x, z

Figure 15: Rotating, after teaching the rendering 10(e). For further explanation about the graphs see section 3.1 or Figure 11.

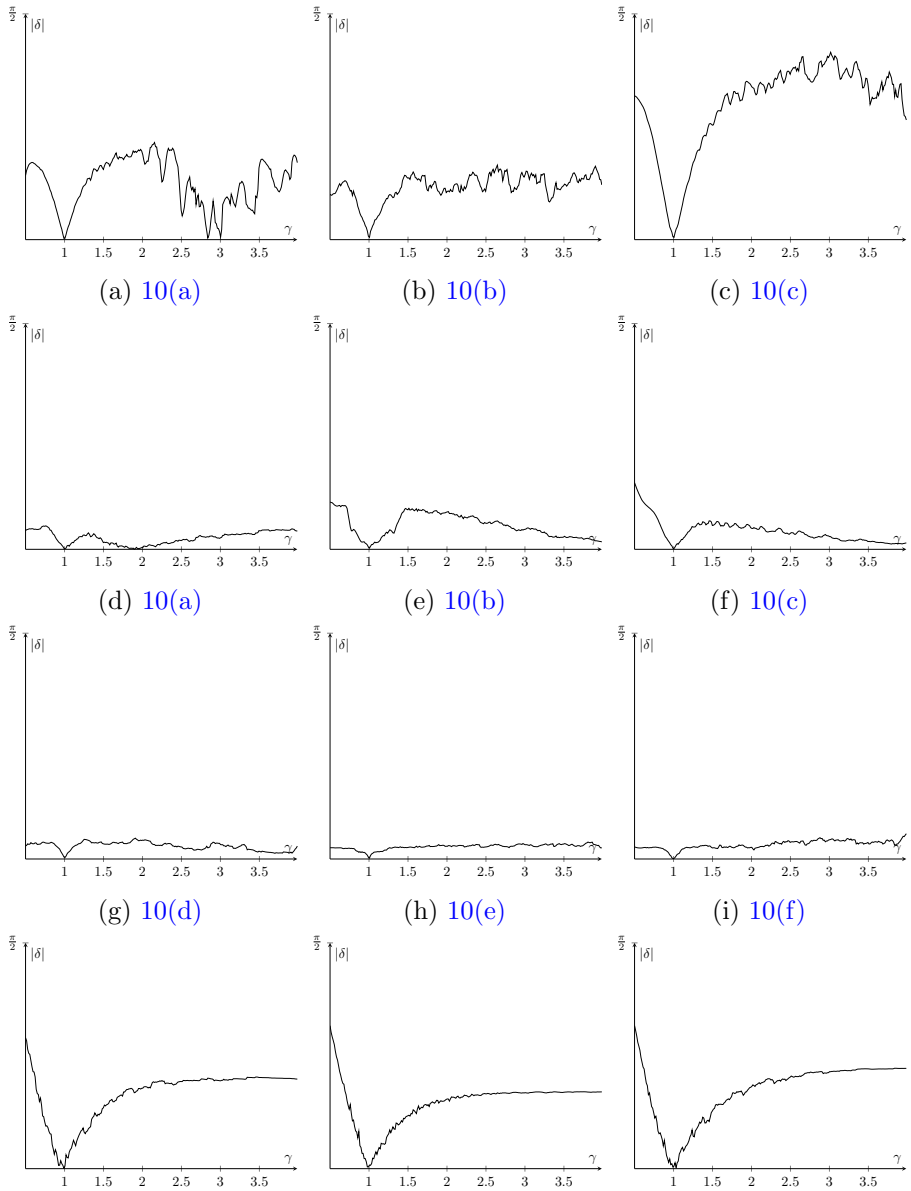


(a) x, y (b) y, z (c) x, z



(d) x, y (e) y, z (f) x, z

Figure 16: Rotating, after teaching the rendering 10(f). For further explanation about the graphs see section 3.1 or Figure 11.



(a) 10(a) (b) 10(b) (c) 10(c)

(d) 10(a) (e) 10(b) (f) 10(c)

(g) 10(d) (h) 10(e) (i) 10(f)

(j) 10(d) (k) 10(e) (l) 10(f)

Figure 17: Scaling. We've multiplied the camera position with the value γ, effectively moving the camera closer-, and farther from the object. The vertical axis marks the absolute value of the measured phase difference. For further explanation about the graphs see section 3.2

References

[1] G. Akemann, J. Baik, P. Di Francesco, The Oxford Handbook of Random Matrix Theory, *Oxford, Oxford University Press* (2011). ⇒170

[2] G. W. Anderson, A. Guionnet, O. Zeitouni, An introduction to random matrices, *Cambridge, Cambridge University Press* (2010). ⇒170

[3] L. A. Diago, T. Kitaoka, I. Hagiwara, Development of a System for Automatic Facial Expression Analysis, *Journal of Computational Science and Technology* **2**, 4 (2008) 401–412. ⇒156

[4] E. N. Gilbert, A comparison of signalling alphabets, *Bell System Technical Journal* **31** (1952) 504–522. ⇒161

[5] I. Hagiwara, Global Optimization Method to Multiple Local Optimals with the Surface Approximation Methodology and Its Application for Industry Problems, in: *Response Surface Methodology in Engineering Science* (ed. K. Palanikumar), IntechOpen, (2021). ⇒156

[6] R. W. Hamming, Error Detecting and Error Correcting Codes, *The Bell System Technical Journal* **29**, 2 (1950) 147–160. ⇒161

[7] Y. Hendra, R. P. Gopalan, M. G. Nair, A method for dynamic indexing of large image databases, *IEEE SMC'99 Conference Proceedings, 1999 IEEE International Conference on Systems, Man, and Cybernetics* **1** (1999) 302–307. ⇒156

[8] J. I. Khan, D. Y. Y. Yun, H. Garcia, Visual query in medical image archive, *Proceedings of the Fourth International Conference on Image Management and Communication (IMAC 95)* (1995) 21–31. ⇒156

[9] A. Krizhevsky, Learning Multiple Layers of Features from Tiny Images, *Technical Report, Computer Science Department, University of Toronto* (2009). ⇒160

[10] J. H. Lint, Introduction to Coding Theory, *Graduate Texts in Mathematics 86* (1999). ⇒160

[11] R. Manger, Holographic Neural Networks and Data Compression, *Informatica* **21** (1997) 665–673. ⇒156

[12] R. Manger, Memory reduction techniques for holographic neurons, *Proceedings of the 4th International Symposium on Operational Research in Slovenia (SOR '97 - Preddvor, Slovenia, October 1-3, 1997)* (ed. V. Rupnik, L. Zadnik Stirn, S. Drobne), Slovenian Society Informatika, Ljubljana, Slovenia (1997) 195–200. ⇒158

[13] R. Manger, M. Mauher, Using holographic neural networks for currency exchange rates prediction, *Proceedings of the 16th International Conference on Information Technology Interfaces (ITI '94 - Pula, June 14-17, 1994)* (ed. V. Čerić, V. Hljuz Dobrić), University Computing Centre, Zagreb, Croatia (1994) 143–150. ⇒156

[14] R. Manger, V. L. Plantamura, B. Souček, Stimulus preprocessing for holographic neural networks, *Frontier Decision Support Concepts* (ed. V. L. Plantamura, B. Souček and G. Visaggio), John Wiley and Sons, New York, USA (1994) 79–90. ⇒157

[15] R. Manger, V. L. Plantamura, B. Souček, Classification with holographic neural networks, *Frontier Decision Support Concepts* (ed. V. L. Plantamura, B. Souček and G. Visaggio), John Wiley and Sons, New York, USA (1994) 91–106. ⇒156

[16] R. Manger, B. Souček, New Preprocessing Methods for Holographic Neural Networks, *Albrecht, R.F., Reeves, C.R., Steele, N.C. (eds) Artificial Neural Nets and Genetic Algorithms. Springer* (1993) 190–197. ⇒157

[17] M. L. Mehta, Random Matrices, *Amsterdam, Elsevier/Academic Press* (2004). ⇒170

[18] J. G. Sutherland, A Holographic Model of Memory, Learning and Expression, *International Journal of Neural Systems* **1**, 3 (1990) 259–267. ⇒155, 157, 158

[19] J. G. Sutherland, The Holographic Neural Method, in: *Fuzzy, Holographic and Parallel Intelligence* (ed. B. Souček), (1992). ⇒155

[20] J. G. Sutherland, W. C. Radzelovage, Land Mine Detection Applying Holographic Neural Technology (HNet), *Proceedings of SPIE* **6553**, 1 (2007). ⇒156

[21] T. Tao, *Topics in random matrix theory*, Graduate Studies in Mathematics, vol. 132, American Mathematical Society (2012). ⇒170

[22] R. R. Varshamov, Estimate of the number of signals in error correcting codes, *Doklady Akademii Nauk SSSR* **117** (1957) 739–741. English Translation in I. F. Blake, *Algebraic Coding Theory: History and Development*, Dowden, Hutchinson & Ross, 1973, pp. 68–71. ⇒162

[23] ∗ ∗ ∗, clBLAS, https://github.com/clMathLibraries/clBLAS ⇒159

Received: September 21, 2022 • Revised: October 10, 2022