Neural Associative Memories as Accelerators for Binary Vector Search

Chendi Yu,
Vincent Gripon
and Xiaoran Jiang

Email: name.surname@telecom-bretagne.eu
Telecom Bretagne, Electronics department
UMR CNRS Lab-STICC
Brest, France

Hervé Jégou

Email: name.surname@inria.fr
INRIA
IRISA, team Texmex
Rennes, France

***Abstract*—Associative memories aim at matching an input noisy vector with a stored one. The matched vector satisfies a minimum distance criterion with respect to the inner metric of the device. This problem of finding nearest neighbors in terms of Euclidean or Hamming distances is a very common operation in machine learning and pattern recognition. However, the inner metrics of associative memories are often misfitted to handle practical scenarios. In this paper, we adapt Willshaw networks in order to use them for accelerating nearest neighbor search with limited impact on accuracy. We provide a theoretical analysis of our method for binary sparse vectors. We also test our method using the MNIST handwritten digits database. Both our analysis for synthetic data and experiments with real-data evidence a significant gain in complexity with negligible loss in performance compared to exhaustive search.**

Keywords–Associative Memories; Binary Sparse Vector; Nearest Neighbors Search; Willshaw Networks.

I. INTRODUCTION

Associative memories are devices that store associations between multiple patterns. They are considered a good model for human memory for their ability to recall stored messages given part of them. For example consider the query of retrieving the word "neuron" from the partially erased query "n*uro*".

The literature on this subject is vast and many models have been proposed, amongst which Hopfield Neural Networks (HNNs) [1] play a prominent role. HNNs store the empirical covariance matrix associated with a set of d-dimensional binary ({-1,1}) vectors. This simple design is appealing. However, HNNs have a strong limitation on the number of vectors they can store. This quantity, referred to as *diversity*, is provably upper-bounded by $d/(2\log(d))$ [2]. Other neural-based methods [3] [4] focus on storing the co-occurrence matrix instead, under the assumption that stored vectors are c-sparse binary ({0,1}). With proper parameters, the diversity of such networks is in the order of the square of d [5].

In machine learning and pattern recognition, many applications aim at matching an input vector to a collection of other ones. In metric spaces, this operation is termed "nearest neighbor search". For high-dimensional vectors, nearest neighbor search complexity is linear in both the number of vectors in the collection and their dimension, as the naive strategy of computing all the distances is the best [6]. To overcome this issue and scale to larger collections, one has to resort to approximate neighbor search techniques, which trade accuracy against scalability [7] [8] [9].

This paper shows that binary neural networks can accelerate nearest neighbor search with limited impact on accuracy, in the case of sparse binary vectors. We support our claim by providing both (i) a theoretical analysis with simulation on synthetic data and (ii) experiments on real data carried out on the gold-standard MNIST handwritten digits database.

The rest of the paper is organized as follows. Section II introduces our method. In section III, we derive a theoretical analysis of the performance, while Section IV presents our real-data experiments. Section V concludes this paper.

II. METHODOLOGY

Consider a set $\mathscr{X}$ of n binary sparse nonzero $\{0,1\}$-vectors with dimension d. Given some input vector x_0, we want to retrieve the closest vector to the query with respect to the Hamming distance:

$$x \in \arg\min_{x' \in \mathscr{X}} d_H(x_0, x') \,,$$

where $d_H(x_0,x')$ denotes the Hamming distance (number of distinct values) between x_0 and x'. This problem is referred to as *nearest neighbor search*.

Since computing distances between two d-dimensional vectors is in $\mathscr{O}(d)$, the complexity of this problem is in $\mathscr{O}(dn)$ with the naive approach, which turns out to be the best choice for exact search: this complexity is tight in high-dimensional spaces. Yet, approximate solutions exist to reduce it [7] [8] [9]. In this case, beyond CPU and memory complexity, the accuracy of the algorithm is a key characteristic when comparing different approaches.

Since in our case, we focus on finding similar neighbors for the Hamming distance and not the Euclidean one, according to Andoni, "the best algorithm for the Hamming space remains the one described in [10]", that is a binary variant of Locality Sensitive Hashing, which requires a lot of memory to be effective in high-dimensional spaces.

The approach we propose below relies on a rather different mechanism (and less memory-demanding) than the hashing-based approach of LSH: neuro-inspired associative memories. It consists of two steps: *selection* and *check*.

 ISBN: 978-1-61208-390-2

a) Selection: We partition $\mathscr{X}$ into subsets $\mathscr{X}_1, \ldots, \mathscr{X}_q$, as

- $\mathscr{X} = \bigcup_{j=1}^{q} \mathscr{X}_j$,
- $\forall j, j', j \neq j', \mathscr{X}_j \cap \mathscr{X}_{j'} = \emptyset$.

In the following, we conveniently consider that the split is regular: $|\mathscr{X}_1| = \cdots = |\mathscr{X}_q| = k$, such that $n = kq$.

We then represent each subset as an associative memory, and search for the input query x_0 in each of them. Formally, we use Willshaw networks. The Willshaw network associated with $\mathscr{X}_j$ is defined as:

$$W(\mathscr{X}_j) = \max_{x \in \mathscr{X}_j} x x^\top .$$

In order to have an estimation of whether some input vector x_0 resembles the content of $\mathscr{X}_j$ or not, we use the following formula:

$$s(x_0, \mathscr{X}_j) \triangleq \frac{x_0^\top W(\mathscr{X}_j) x_0}{(x_0^\top x_0)^2} ,$$

that we shall refer to as the score of x_0 in $\mathscr{X}_j$.

By construction of $W(\mathscr{X}_j)$, the score of x_0 in $\mathscr{X}_j$ is maximal if $x_0 \in \mathscr{X}_j$ and equals one. Conversely, it may happen that the maximal score is achieved even if $x_0 \notin \mathscr{X}_j$, typically when the matrix $W(\mathscr{X}_j)$ is overfilled.

b) Check: Having selected all the associative memories for which the score x_0 is high, we exhaustively compare the query to all the vectors stored in the corresponding subsets. We therefore find the nearest neighbor in a restricted subset of the whole dataset. As we will see theoretically and experimentally, it is likely to contain the desired neighbors to the query.

The complexity of computing the score of x_0 in $\mathscr{X}_j$ is in $\mathscr{O}(d^2)$, and is independent of k. This is the reason that motivates us for using such devices as accelerators for nearest neighbor search. More formally, if the number of subsets that are selected for fine-grain search is N, the overall complexity for performing nearest neighbor search is:

$$C = \mathscr{O}(qd^2 + Nkd) . \tag{1}$$

III. Analysis for synthetic data

A. Theoretical analysis

We consider n binary column vectors x_i of dimension d and L_0 norm $\|x_i\|_0 = c$, c being a constant. The vectors are generated independently and randomly according to a uniform distribution. They are then regularly split by group of k and stored in q Willshaw networks in the way defined in the previous section.

Given an input vector x_0, which is not necessarily contained in the training set, we use this model to perform an approximate search of its nearest neighbor. For simplifying the demonstration, we assume that it has the same norm as the training vectors, $\|x_0\|_0 = c$. Let us denote by x its nearest neighbor and $W(\mathscr{X}_z)$ the matrix it is stored into. We have

$$W(\mathscr{X}_z) = x^\top x + R(\mathscr{X}_z, x)$$

with $R(\mathscr{X}_z, x) = W(\mathscr{X}_z) - x^\top x$.

The score of x_0 in $\mathscr{X}_z$ is written as:

$$\begin{aligned} s(x_0, \mathscr{X}_z) &= \frac{x_0{}^\top W(\mathscr{X}_z) x_0}{(x_0{}^\top x_0)^2} \\ &= \frac{x_0{}^\top x^\top x x_0 + x_0{}^\top R(\mathscr{X}_z, x) x_0}{c^2} \\ &= \frac{\left(c - \frac{d_H(x,x_0)}{2}\right)^2 + x_0{}^\top R(\mathscr{X}_z, x) x_0}{c^2} \\ &\geq \frac{\left(c - \frac{d_H(x,x_0)}{2}\right)^2}{c^2} . \end{aligned}$$

If $x_0 = x$, we have $d_H(x, x_0) = 0$ and $x_0{}^\top R(\mathscr{X}_z, x) x_0 = 0$, thus we rediscover the maximum score of 1.

Let us now consider any other matrix $W(\mathscr{X}_j), j = 1, ..., q, j \neq z$. The score of x_0 in $W(\mathscr{X}_j)$ is calculated as:

$$s(x_0, \mathscr{X}_j) = \frac{\sum_{(l,m) \in M(x_0)} w_{lm}}{c^2}$$

with $M(x_0) = \{(l,m) | x_0(l) = 1, x_0(m) = 1\}$. Obviously, $|M(x_0)| = c^2$.

Let us make the assumption that the random variables w_{lm} are independent. Although this statement is false as a stored vector adds multiple ones in the matrix, it is a fair approximation [4] [5], considering c to be small compared to d. According to the law of large numbers, this normalized sum approaches the expectation $\mathbb{E}(w_{lm})$ when c is large enough. Since w_{lm} is taken within discrete values in {0, 1}, this is nothing else than the probability that any coefficient w_{lm} is equal to 1. We denote this probability $d(W(\mathscr{X}_j))$, and term it the density of the matrix. It can be expressed as:

$$\begin{aligned} d(W(\mathscr{X}_j)) &= \Pr(w_{lm} = 1) \\ &= \Pr(\exists x' \in \mathscr{X}_j, x'(l) x'(m) = 1) \\ &= 1 - \Pr(\forall x' \in \mathscr{X}_j, x'(l) x'(m) = 0) \\ &= 1 - \left(1 - \Pr(x'(l) = 1 \wedge x'(m) = 1)\right)^k \\ &= 1 - \left(1 - \frac{c^2}{d^2}\right)^k . \end{aligned}$$

For sake of simplicity, the matrix that obtains the greatest score is chosen as the winner of this approximate search: $N = 1$. Then, an exhaustive search will be performed for all the vectors that compose the winner matrix. In order to be sure that this matrix contains the nearest neighbor x, the score $s(x_0, \mathscr{X}_z)$ needs to be greater than $s(x_0, \mathscr{X}_j)$ for all $\mathscr{X}_j$. We have then

$$\left(c - \frac{d_H(x,x_0)}{2}\right)^2 \geq c^2 d(W(\mathscr{X}_j)) .$$

If we suppose that $c \ll d$ (the vectors are sparse) and $d_H(x, x_0) \ll c$ (distance from x_0 to its nearest neighbor is small), we obtain:

$$k \leq -\frac{d^2}{c^2} \ln \frac{d_H(x,x_0)}{c} . \tag{2}$$

Let us now analyze the complexity of this method. As given by Equation (1), the complexity when $N = 1$ is expressed as:

$$C = \mathscr{O}(\frac{n}{k} d^2 + kd) .$$

On the other hand, the complexity of an exhaustive nearest neighbor search is written as:

$$C_{\text{knn}} = \mathcal{O}(nd) \, .$$

An interesting value of k should satisfy:

$$nd \geq \frac{n}{k} d^2 + kd$$

We obtain:

$$\begin{aligned} k &\geq \frac{n - \sqrt{n^2 - 4nd}}{2} \\ &\approx d, \text{if } d \ll n. \end{aligned} \tag{3}$$

Equations (2) and (3) gives the interval of k such that the proposed model finds in expectation the nearest neighbor without error and with a reduced complexity compared to exhaustive nearest neighbor search. The existence of such a value of k implies:

$$-\frac{d^2}{c^2} \ln \frac{d_H(x,x_0)}{c} \geq d$$

We conclude:

$$d_H(x,x_0) \leq \frac{c}{e^{c^2/d}} \, .$$

For $c = \sqrt{\alpha d}$ with $\alpha > 0$, we have $d_H(x,x_0) \leq \frac{c}{e^{\alpha}}$. Thus, for sparse binary vectors of norm c that grows with the square root of the vector dimension d, if a given vector is sufficiently near from its nearest neighbor, the Willshaw networks can always accelerate the nearest neighbor search without compromising the performance in terms of error rate.

The complexity C should also be compared to that of the Mount Approximate Nearest Neighbor (the Mount ANN) search [7]:

$$C_{ANN} = \mathcal{O}(d(1 + \frac{6d}{\varepsilon})^d \log(n)),$$

where ε is a constant.

In practical scenarios where the dimension is not smaller than 10, it is reasonable to consider $n \ll d^d$. Adding the fact that $d \leq k \leq n$, we easily show that $C = o(C_{ANN})$. Note that in this case, the complexity of the Mount ANN is even larger than that of exhaustive search.

B. Synthetic simulations

We consider a vector set $\mathcal{X}$ that contains $n = 20000$ randomly generated sparse binary vectors of dimension $d = 400$ and $c = 10$ of them are 1s. The test vector is generated as a modified version of any vector in the set $\mathcal{X}$. To generate a test vector x_0, we randomly pick up a vector $x \in \mathcal{X}$ and permute s 1s and 0s, s being smaller than c. Formally, we have

$$\|x \wedge x_0\|_0 = c - s \, ,$$

and

$$\|x\|_0 = \|x_0\|_0 = c \, .$$

If s is small enough, x is likely the nearest neighbor of x_0 with Hamming distance $d_H(x,x_0) = 2s$. We take $s = 4$ for the simulations.

Note that we still have not exploited the fact that vectors are sparse in our estimation of algorithm complexity. As a matter of fact, sparsity of vectors helps reducing complexity: we only need to check the positions where are 1s, the number of which is c, rather than looking at every coefficients in a vector of dimension d. Thus, we can update Equation (1) as follows:

$$C = \mathcal{O}(\frac{n}{k} c^2 + \alpha kc) \, ,$$

and

$$C_{\text{knn}} = \mathcal{O}(\alpha nc)$$

where α is some constant between 1 and 2. In fact, calculating the Hamming distance between two sparse binary vectors of norm c needs at most $2c$ operations if there is not any common position that has coefficient 1, and at least c operations if these two vectors are perfectly identical. We then define the *relative complexity* R_C as the ratio between C and C_{knn}:

$$R_C = \frac{C}{C_{\text{knn}}} = \frac{c}{\alpha k} + \frac{k}{n} \geq \frac{c}{2k} + \frac{k}{n}$$

Therefore, we obtain $R_{C\min} = \sqrt{\frac{2c}{n}} \approx 3.2\%$ in condition that $k = \sqrt{\frac{nc}{2}} \approx 316$ if $n = 20000$ and $c = 10$. Nevertheless, in terms of the error rate, k needs to be small enough to avoid overfitting.

Simulated results are illustrated in Figure 1 with parameters listed as follows: $d = 400$, $c = 10$, $n = 20000$, $s = 4$, $N = 1$. Three measures are assessed in function of k, the number of vectors that contains in each matrix: the red curve represents the Hamming distance gap from the vector that obtains the highest score in our method to the *de facto* nearest vector obtained by the exhaustive search while the blue one indicates the error rate and the black one signifies the relative complexity R_C. For example, when $k = 200$, we obtain a negligible error rate of 0.5% and a Hamming distance gap of 0.003 in consuming only about 3.5% of the complexity of a classical nearest neighbor search.

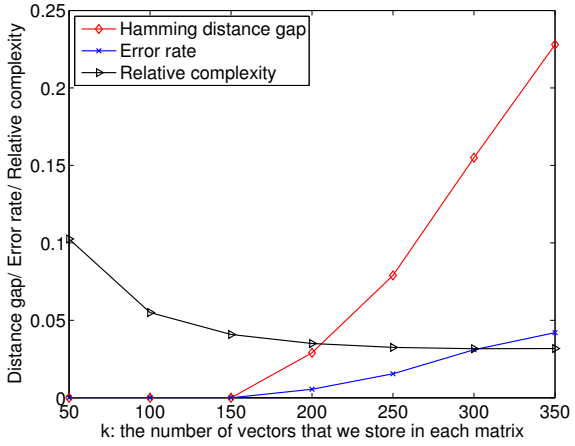

Figure 1. Performance evaluation for the use of associative memory to accelerate nearest neighbor research.

One may notice that, the error rate and the Hamming distance gap grow as k, the number of vectors stored in each matrix increases, which is especially true when k passes

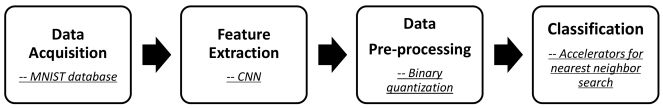

Figure 2. Procedure of the simulation.

a certain threshold (150 in this case). This degradation of performance can be mainly attributed to the saturation of 1s in matrices, which aggravates the noise interference in the selection phase. Since the interference issue is closely related to the matrix density, the performance depends on the length of vectors and the number of vectors contained in each matrix.

IV. Real Data

To evaluate our method on real data, we have carried out experiments on the MNIST handwritten digits database [11]. Created by Yann Lecun, Corinna Cortes and Christopher J.C. Burges, the MNIST database of handwritten digits consists of a training set of 60,000 examples, and a test set of 10,000 examples. All the digits in the database are real-world data.

For the purpose of being more representative, the simulation follows the steps of typical image recognition as is shown in Figure 2.

In the simulation, Convolutional Neural Networks (CNNs) are exploited to extract the proper features followed by a binary quantization. These binary features will be subsequently used as inputs for the proposed method. The nearest neighbor as identified by the proposed method decides which class a test input belongs to. The complexity and the corresponding error rate are then calculated.

For the computation using CNNs, we adapted the code of Rasmus Berg Palm [12]. A synthetic representation of the network considered here is shown in Figure 4 (this representation is inspired by [13]).

More precisely, given some 28×28 image input taken from the MNIST database, we use a 5-layer deep network. The first and the third layers are convolutional layers while the second and the forth layer are sub-sampling layers. The size of kernel for the convolution layers is 5 and the scale of average pooling for sum-sampling layers is 2. Six feature maps are applied for the first two layers and there are 12 for the third and the forth layer. In the fifth layer, we put all the feature maps already present at the fourth layer $(12 \times (4 \times 4))$ into a vector whose dimension is 192×1. The parameter *training rate* is set as 1 and the parameter *training times* as 10.

For the binary quantization, we choose a pre-defined threshold $t = 0.3$. We observe that with this choice the proportion of 1s reaches approximately 0.42 on average in the output vectors. This threshold was tuned to obtain the best performance when using nearest neighbor classification afterwards.

Then we use the proposed method. Let us denote by $\mathscr{X}^D$ the set of transformed training vectors corresponding to digit D, $D = 0,\ldots,9$. For each digit D, we split $\mathscr{X}^D$ into $6000/k$ subsets $\mathscr{X}_j^D$, each of them containing k vectors, and such that $\mathscr{X}^D = \bigcup_j \mathscr{X}_j^D$.

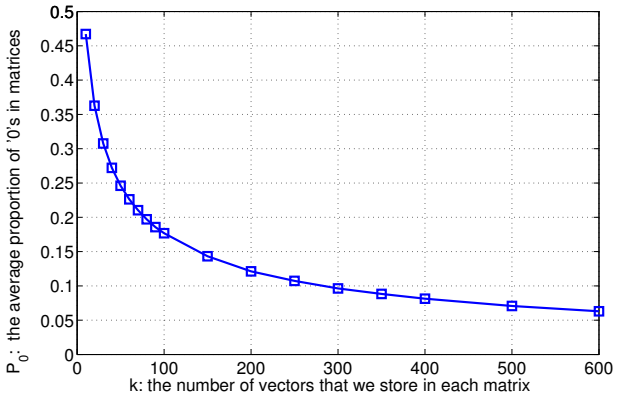

Figure 3. The relation between k and the proportion p_0 of 0s in the matrices.

Therefore, after the training phase, we have $q = \frac{60000}{k}$ matrices in total. Then, at classification time, given a vector x_0 belonging to the test set, we calculate the score of this vector in each set $\mathscr{X}_j^D$, $s(x_0, \mathscr{X}_j^D)$ in the way defined in Section II. The N matrices with the highest scores will be chosen. If there is a tie in matrices scores, we perform a random choice of which to select. The vectors stored in these chosen matrices will be then exhaustively searched to find the nearest neighbor. The label of the matrix that contains the nearest neighbor is returned as the result of the classification process.

The general expression of the overall complexity given by Equation (1) should be adapted in this real data scenario. In fact, the matrix $W(\mathscr{X}_j^D)$ contains much more 1s than 0s. Clearly, as k grows, the number of 0s in the matrix decreases. However, due to the non-uniformity of the vectors in MNIST database, the speed at which 0s decreases is not as fast as for synthetic data (see Section III). Figure 3 depicts the proportion of 0s p_0 in the matrices as function of k.

Therefore, we can exploit this fact to adjust the computation of complexity. In fact,

$$s(x_0, \mathscr{X}_j^D) = \frac{x_0^\top W(\mathscr{X}_j^D) x_0}{\left(x_0^\top x_0\right)^2} = 1 - \frac{\sum_l \sum_m x_{0l} x_{0m}}{\|x_0\|^2}$$

the values of l and m satisfying $w_{lm} = 0$.

In this way, we only have to look at the positions where are 0s in the matrix $W(\mathscr{X}_j^D)$. The overall complexity becomes:

$$C = \mathscr{O}(p_0 q d^2 + N k d)$$

Simulations are performed using a wide range of values of k and N. In Figure 5, the error rate is represented in function of the relative complexity compared to exhaustive nearest neighbor search. We observe that the complexity can be greatly reduced (up to a factor of 0.3 compared to the exhaustive search) for almost the same error rate when parameters are well chosen. The results we obtain in terms of error rate are similar to that obtained using the Mount ANN [7].

 ISBN: 978-1-61208-390-2

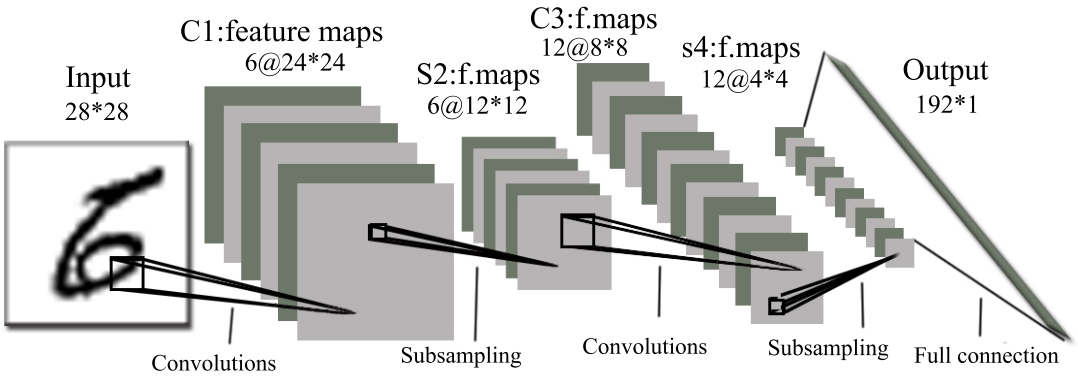

Figure 4. Representation of the CNNs layers used to extract features from MNIST raw images.

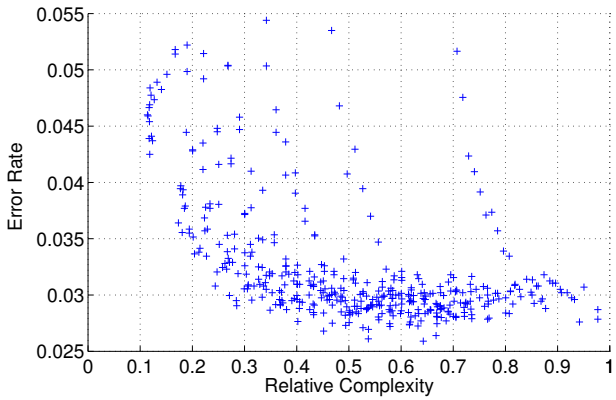

Figure 5. Error rate in function of the relative complexity. Simulations are performed for the MNIST handwritten digits database.

V. Conclusion

We proposed a method to perform approximate nearest neighbor search using neural-based associative memories. This method advantageously takes benefit from the fact that the neural network dimensions of the representation associated with a set of k vectors is independent of k.

As a result, we prove that for synthetic data, one can expect dramatically lower complexity with no reduction on accuracy. We also evaluate our method using real data and prove that it is possible to achieve a reduction of 70% of complexity with a limited impact on the error rate.

In future work, we consider looking at alternative ways to use Willshaw networks to really benefit from their associative capabilities [14], as well as perform experiments on more challenging datasets.

Acknowledgment

This work was founded in part by the European Research Council under Grant ERC-AdG2011 290901 NEUCOD.

References

[1] John J Hopfield, "Neural networks and physical systems with emergent collective computational abilities," *Proceedings of the national academy of sciences*, vol. 79, no. 8, pp. 2554–2558, 1982.

[2] Robert J McEliece, Edward C Posner, Eugene R Rodemich, and Santosh S Venkatesh, "The capacity of the hopfield associative memory," *Information Theory, IEEE Transactions on*, vol. 33, no. 4, pp. 461–482, 1987.

[3] David J Willshaw, O Peter Buneman, and Hugh Christopher Longuet-Higgins, "Non-holographic associative memory.," *Nature*, vol. 222, pp. 960–962, 1969.

[4] Vincent Gripon and Claude Berrou, "Sparse neural networks with large learning diversity," *IEEE Transactions on Neural Networks*, vol. 22, no. 7, pp. 1087–1096, July 2011.

[5] Judith Heusel, Matthias Löwe, and Franck Vermet, "On the capacity of a new model of associative memory based on neural cliques," *arxiv preprint*.

[6] Roger Weber, Hans-J. Schek, and Stephan Blott, "A quantitative analysis and performance study for similarity-search methods in high-dimensional spaces," in *Proceedings of the International Conference of Very Large Databases*, 1998, pp. 194–205.

[7] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Y Wu, "An optimal algorithm for approximate nearest neighbor searching fixed dimensions," *Journal of the ACM (JACM)*, vol. 45, no. 6, pp. 891–923, 1998.

[8] Sunil Arya, David M Mount, Nathan S Netanyahu, Ruth Silverman, and Angela Wu, "An optimal algorithm for approximate nearest neighbor searching," in *Proceedings of the fifth annual ACM-SIAM symposium on Discrete algorithms*. Society for Industrial and Applied Mathematics, 1994, pp. 573–582.

[9] Piotr Indyk and Rajeev Motwani, "Approximate nearest neighbors: towards removing the curse of dimensionality," in *Proceedings of the thirtieth annual ACM symposium on Theory of computing*. ACM, 1998, pp. 604–613.

[10] Aristides Gionis, Piotr Indyk, and Rajeev Motwani, "Similarity search in high dimensions via hashing," in *Proceedings of the International Conference of Very Large Databases*, 1999, pp. 518–529.

[11] "Mnist database," 2014, URL: http://yann.lecun.com/exdb/mnist/ [accessed: 2014-01-02].

[12] "Deeplearntoolbox," 2014, URL: https://github.com/rasmusbergpalm/DeepLearnToolbox [accessed: 2014-01-02].

[13] "Net5," 2014, URL: http://eblearn.sourceforge.net [accessed: 2014-01-02].

[14] Ala Aboudib, Vincent Gripon, and Xiaoran Jiang, "A study of retrieval algorithms of sparse messages in networks of neural cliques," in *Proceedings of Cognitive 2014*, May 2014, pp. 140–146.

 ISBN: 978-1-61208-390-2