Simple coarse graining and sampling strategies for image recognition

Stephen Whitelam*

Molecular Foundry, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA

A conceptually simple way to recognize images is to directly compare test-set data and training-set data. The accuracy of this approach is limited by the method of comparison used, and by the extent to which the training-set data covers the required configuration space. Here we show that this coverage can be substantially increased using simple strategies of coarse graining (replacing groups of images by their centroids) and sampling (using distinct sets of centroids in combination). We use the MNIST data set to show that coarse graining can be used to convert a subset of training images into about an order of magnitude fewer image centroids, with no loss of accuracy of classification of test-set images by direct (nearest-neighbor) classification. Distinct batches of centroids can be used in combination as a means of sampling configuration space, and can classify test-set data more accurately than can the unaltered training set. The approach works most naturally with multiple processors in parallel.

I. INTRODUCTION

Machine learning, used successfully for many years in fields such as image recognition [1–4] and game playing [5, 6], is becoming established in the physical sciences [7]. Machine learning has been used to calculate thermodynamic quantities of molecular systems, both classical [8, 9] and quantum [10–12], and to predict the outcome of nonequilibrium processes such as self-assembly [13]. Certain ideas, such as sampling by Monte Carlo simulation, are widely used in both machine learning [14, 15] and the physical sciences [16], and it is natural to expect physical concepts, and methods developed in the physical sciences, to be used increasingly to study and improve machine-learning tools [17]. Here we show that simple physics-inspired ideas of coarse graining and sampling can be used to improve the efficiency and accuracy of a simple memory-based classification algorithm.

A conceptually simple method of image recognition is to assign to a test-set image the type of the most similar training-set image [18–22]. The accuracy of such an approach is limited by the method of image comparison, and by the extent to which the training-set data "samples" configuration space. In Fig. 1(a) we show 36 examples of 3s and 5s taken from the MNIST training set [23, 24], which makes clear that a single concept can be represented by many different configurations. One way to better sample configuration space is to synthetically expand the set of training data, e.g. by effecting elastic distortions of digits [23]. One drawback of such approaches is that they require prior knowledge of the concept to be classified.

Here we use the MNIST data set, which contains hand-drawn digits or symbols of type 0 to 9, to show that simple strategies of coarse graining and sampling can be used to substantially reduce error rates of direct comparison (memory-based, nearest-neighbor classifier) schemes without requiring information beyond that contained in

the test set. Combining symbols into groups allows us to identify a set of coarse-grained symbols or *memories* [see Fig. 1(b)]; these objects correctly classify the training-set data by capturing the diversity of the set of symbols while omitting redundant objects. In this context "coarse graining" means combining symbols into groups and identifying their centroids, in order to produce a sparse description of configuration space; it does not mean sampling images at lower resolution.

The process of coarse graining described here is similar in spirit to the process enacted by particle-clustering algorithms [25–27]; by a supervised k-means algorithm [28]; and by learning vector quantization classifiers [21] such as the growing neural gas [20] or growing self-organizing network [22]. For MNIST data, using the algorithm described here, a set of N training-set symbols can be described by a set of about $0.15N$ memories. Memories classify unseen test-set symbols, via nearest-neighbor classification, as accurately or slightly more accurately than do the symbols from which they are derived; a natural expectation is the opposite [23]. Moreover, memory sets created by drawing batches of symbols stochastically from the training set can be used in parallel to classify test-set data substantially more accurately than can the original training set. In effect, memories can be used to "sample" configuration space. The idea is similar in principle to combining nearest-neighbor classifiers into voting blocs [29]; however, here we perform sampling using objects not present in the original test set, and use the process of sampling to identify new symbols to better describe unseen data.

Using a rudimentary method of image comparison, a nearest-neighbor classifier using the normalized vector dot product applied to raw pixels, we show that coarse graining and sampling can classify the MNIST test set more accurately than can the unaltered training set. This classification scheme is of only modest accuracy ($\approx 1\%$ error rate) by present standards [30], but can likely be improved in several ways (using a better method of image comparison, or additional methods of combining symbols to produce memories), thereby improving its accuracy. Furthermore, while memory-based classification meth-
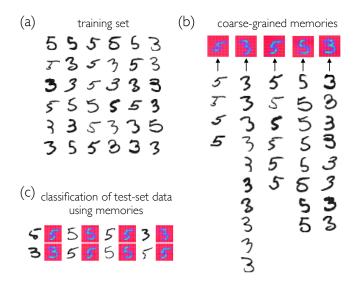
FIG. 1. (a) A set of 36 examples taken from the MNIST training set gives rise to 5 memories [panel (b)] upon coarse graining according to the procedure described in this paper. (b) Each of the 36 symbols is "stored" in a memory; memories are the centroids of their constituent symbols. Memories are colored so that red and green areas correspond to white and black portions of symbols, respectively. These 5 memories correctly classify the 36 symbols by direct (nearest-neighbor) comparison: the coarse-graining algorithm "learns" to partition configuration. (c) Memories can be used to classify unseen test-set symbols as accurately as can the symbols from which they are derived: here we show 8 correctly classified test-set digits next to the memory they most closely resemble. As we show in this paper, combinations of memories derived from different symbol sets can be used to "sample" configuration space and achieve more accurate classification, of unseen symbols from the test set, than can the original training-set symbols.

ods are generally much more computer memory-intensive than e.g. neural networks [23], the present approach is most naturally implemented using parallel processors. The more processors used, the faster is training and classification, and (on the scales we have studied) the more accurate is classification. In what follows we describe the method and apply it to the MNIST data set.

II. IMAGE RECOGNITION VIA DIRECT COMPARISON

A. Unaltered training set

The MNIST data set consists of $M = 7 \times 10^4$ handwritten digits, of types 0 to 9 inclusive, divided into a training set of size 6×10^4 and a test set of size 10^4. Digits (hereafter called symbols) are grayscale images displayed on a 28×28 pixel grid, and are represented as $P = 784$-dimensional vectors, each entry of which is a scalar between 0 and 255. We divided each entry by 255 to produce a real number between 0 and 1. Let S^θ be symbol

number θ of the MNIST set, where $\theta \in \{1, \ldots, M\}$, and let S_i^θ be its ith component, where $i \in \{1, \ldots, P\}$. Let $T(S^\theta) \in \{0, \ldots, 9\}$ be the *type* of symbol S^θ.

In Fig. 2(a) we show the error rate, the number of incorrect assignments divided by the test-set size of 10^4, that results from direct comparison (nearest-neighbor classification): we compare each test-set symbol with each of the first N symbols of the training set, and assign to each test set symbol the type of the most similar training set symbol. A standard measure of similarity is the Euclidean distance between vectors A and B,

$$D(A, B) = P^{-1} \sum_{i=1}^{P} (A_i - B_i)^2, \qquad (1)$$

with smaller distances representing greater similarity. This measure results in the red line shown in Fig. 2(a). Another measure of similarity is the normalized vector dot product, the cosine of the angle between vectors A and B,

$$\hat{A} \cdot \hat{B} = \frac{\sum_{i=1}^{P} A_i B_i}{\sqrt{\sum_{i=1}^{P} A_i^2} \sqrt{\sum_{i=1}^{K} B_i^2}}, \qquad (2)$$

with larger values indicating greater similarity. This measure results in the blue line shown in the figure. There exist much more discriminating measures of the similarity of two images [31, 32]. Here our aim is to show how the accuracy of a classifier, using a simple measure of similarity, can be improved by sampling. We shall refer to the normalized vector dot product as the *overlap* of vectors A and B,

$$\mathcal{O}(A, B) = \hat{A} \cdot \hat{B}. \qquad (3)$$

The error rate resulting from direct comparison, using the normalized vector dot product as a measure of similarity, is about 2.8%. This is not close to the error rates produced by the most accurate methods [33] ($\approx 0.2\%$), but neither is it terrible: it is comparable to the rates produced by some of the simpler neural networks of the late 1990s [23]. However, the error rate shown in Fig. 2(a) decreases approximately algebraically with N (the plot is log-log), and so it is clear that substantial additional reduction in error rate would require orders of magnitude more symbols in the training set.

Here we show that reduction in error rate by direct comparison can be achieved instead by stochastic sampling of the existing training set. The heart of this approach is the coarse-graining algorithm described below. This algorithm turns a subset of training-set symbols into a lesser number of coarse-grained *memories*, in such a way that each member of the training subset is correctly classified, via direct comparison, by the set of coarse-grained memories.
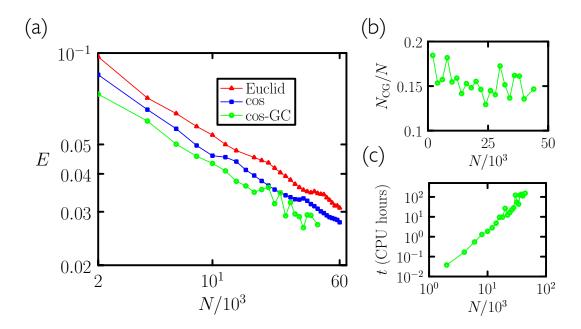
FIG. 2. (a) Error rate E for classification of the 10^4-digit MNIST test set, by direct (nearest-neighbor) comparison with N symbols from the training set (log-log plot). For red and blue lines the measure of similarity is the Euclidean distance (1) and normalized vector dot product (2), respectively. The green line was obtained by coarse graining N symbols of the training set into $N_{\rm CG} < N$ memories. Classification accuracy is no worse and is often better upon coarse graining (compare blue and green lines). (b) $N_{\rm CG}/N$ is typically about 15%. (c) The CPU time required for coarse graining scales roughly algebraically with N.

B. Coarse graining training-set symbols

We aim to *coarse grain* N symbols S^θ to produce $N_{\rm CG} \leq N$ *memories* M^α, $\alpha \in \{1, \ldots N_{\rm CG}\}$. Each memory has entries M_i^α, $i \in \{1, \ldots, P\}$, and is of type $T(M^\alpha) \in \{0, 1, \ldots, 9\}$. Memories are linear combinations (centroids) of symbols. To facilitate the addition and removal of symbols from memories it is convenient to write

$$M^\alpha = \hat{M}^\alpha / N^\alpha, \qquad (4)$$

where $\hat{M}^\alpha$ is the un-normalized version of memory α, and N^α the number of symbols comprising memory α. Our coarse-graining strategy proceeds as follows.

1. Consider a *batch*, an ordered collection of B symbols S^θ, $\theta \in \{1, \ldots, B\}$, taken from the training set. Create a new memory M^α that is equal to the first symbol $\theta = 1$ from this batch and is of the same type:

$$\hat{M}_i^1 = S_i^1 \, (\forall i); \quad N^1 = 1; \quad T(M^1) = T(S^1). \quad (5)$$

Record the fact that symbol 1 is now stored in memory 1. Create one memory for each additional type of symbol in the batch (with each memory consisting of one symbol), giving up to ten initial memories. Record the memory in which each symbol is stored (here and subsequently).

2. Return to the start of the batch of symbols and pass through the batch in order. For each symbol, compute its *virtual overlap* with all existing memories. If the symbol S^θ and memory M^α are of the same type, and if the symbol is *not* currently stored in that memory, then the virtual overlap is $\mathcal{O}(S^\theta, M^{\alpha+\theta})$ [see Eq. (3)]. Here $M^{\alpha+\theta}$ is the vector that results if symbol S^θ is added to memory M^α; it has components $M_i^{\alpha+\theta} = (\hat{M}_i^\alpha + S_i^\theta)/(N^\alpha + 1)$. Otherwise (i.e. if the symbol is currently stored in the memory, or if symbol and memory are of distinct type) the virtual overlap is $\mathcal{O}(S^\theta, M^\alpha)$.

3. Let the largest virtual overlap between symbol S^θ and any memory be with memory M^β.

   (a) If S^θ is currently stored in M^β then no action is necessary.

   (b) If S^θ is not currently stored in M^β, and M^β is of the same type as S^θ, then add S^θ to M^β:

   $$\hat{M}_i^\beta \to \hat{M}_i^\beta + S_i^\theta; \quad N^\beta \to N^\beta + 1. \quad (6)$$

   If S^θ is currently stored in a different memory M^α then remove S^θ from M^α:

   $$\hat{M}_i^\alpha \to \hat{M}_i^\alpha - S_i^\theta; \quad N^\alpha \to N^\alpha - 1. \quad (7)$$

   If S^θ is not yet stored in a memory then (7) is not necessary.
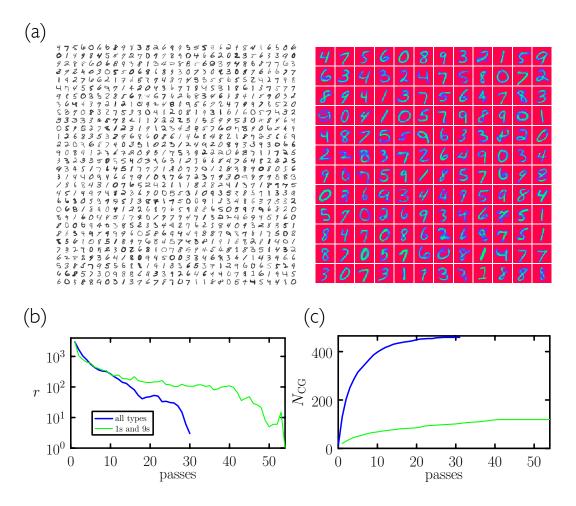
(a)

(b)

(c)

FIG. 3. Coarse-graining algorithm. (a) This 900-symbol batch (left) taken from the MNIST training set yields 144 memories (right) under coarse graining. Both the symbols and the memories achieve a 10% direct-comparison error rate on the MNIST test set. (b) Number of rearrangements r required upon each pass through a batch of 3000 symbols taken from the MNIST training set, versus number of passes through the batch. The blue line refers to a batch composed of all types of symbol; the green line refers to a batch composed of 1s and 9s. (c) For the same batches, we show the number of coarse-grained memories produced by the algorithm as a function of the number of passes.

(c) If S^θ and M^β are of different type then S^θ has been misclassified. Create a new memory M^γ equal to the symbol S^θ and of the same type:

$$\hat{M}_i^\gamma = S_i^\theta; \quad N^\gamma = 1; \quad T(M^\gamma) = T(S^\theta). \tag{8}$$

If S^θ is currently stored in a different memory M^α then remove S^θ from M^α [per Eq. (7)]. If S^θ is not yet stored in a memory then step (7) is not necessary.

4. Continue until we have considered all symbols in the batch. Return to 2 and pass through the batch in order again. Note the number of memory increases or symbol relocations that occur on each pass. If the number of each is zero then the algorithm is finished; if not, return to 2 and pass through the batch in order again.

This algorithm produces a set of memories that correctly classifies, by direct (nearest-neighbor) comparison,

each digit of the batch of symbols from which it is made. All symbols from the batch are stored in a memory, and some memories contain many symbols. For a wide range of batch types and sizes we observed the algorithm to converge [34]. With batches made from the MNIST training set we observed a compression rate of almost an order of magnitude: the number of coarse-grained memories N_{CG} produced from a batch of N symbols is typically $\approx 0.15N$ [Fig. 2(b)]. The CPU time taken for the algorithm to run scales roughly algebraically with N [Fig. 2(c)].

Unless otherwise stated we made batches by drawing symbols without replacement from the training set, with frequencies designed to produce, on average, equal numbers of symbol types within the batch (symbol types in the training set are not equally numerous). We picked at random a symbol S^θ from the training set. With probability M_-/M_α we moved that symbol to the batch. Here $\alpha = T(S^\theta)$ is the type of symbol S^θ; M_α is the number of symbols of type $\alpha \in \{0,\ldots,9\}$ in the training set
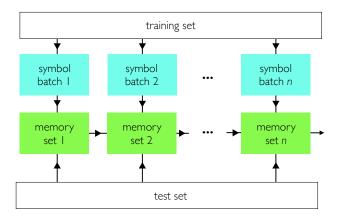
FIG. 4. Coarse graining and sampling. Batches of symbols are constructed by drawing symbols from the training set. Each batch is coarse grained to produce a set of memories. Each set of memories is used to classify the entire test set, and results from all batches are compared.

prior to the move; and $M_- \equiv \min_\alpha M_\alpha$. If the move was successful then we reduced M_α by 1 and removed S^θ from the training set. We then chose another symbol from the training set, and repeated the procedure until a given batch size was obtained. (When constructing many batches for use in parallel we allow each batch to draw from the entire training set, without replacement.)

Significantly, sets of coarse-grained memories are no less accurate than their constituent symbols in classifying the MNIST test set [via direct comparison, using Eq. (2)]: compare the blue and green lines in Fig. 2. Indeed, in most cases the memories are slightly more accurate than their constituent symbols. In a simple sense the coarse-graining algorithm "learns" to partition configuration space, respecting the diversity of the symbol batch and at the same time combining similar-looking symbols. The result is an efficient sampling of that space.

In Fig. 3(a) we show 1000 symbols that yield 144 memories upon coarse graining; both classify the MNIST test set at 10% error rate [35]. In panels (b) and (c) we show the number of rearrangements made by the coarse-graining algorithm during each pass through a batch of 3000 images, and the total number of memories as a function of the number of passes.

C. Sampling

Thus coarse graining achieves a significant compression of information with no loss (and often some slight gain) of direct-comparison classification accuracy with unseen data. Used in this way the algorithm can be regarded as a computer memory-saving measure, similar to prototype-identification methods [36] such as the condensed nearest-neighbor approach [37, 38], or to adaptive versions of learning vector quantization [20–22]. Because memories are objects not present in the original test set,

memory sets can be used in combination to classify unseen data more accurately than can the original training set. If batches are made by drawing stochastically from the training set then each batch is in general different, and will produce different memories upon coarse graining. Such variation can be regarded as a simple means of "sampling", in that memories in each set cover different portions of configuration space. This idea is related to that described in Ref. [29], in which combinations of sub-sampled data are used to improve the accuracy of a nearest-neighbor classifier by constructing voting blocs. Here, coarse graining provides a means of accessing regions of configuration space not present in the original test set.

Fig. 4 shows a simple way in which memory sets can be combined. n batches are built by drawing symbols stochastically from the training set, and each set is coarse grained to produce a set of memories. Each set of memories is used to classify the entire test set by direct (nearest-neighbor) comparison. By comparing all memory sets we assign to each test-set symbol the type of memory with which it has largest overlap. This scheme is naturally carried out using n processors in parallel.

In Fig. 5(a) we show the error rate achieved on the MNIST test set using n memory sets in parallel, per Fig. 4. n batches of 5000 symbols are drawn from the training set, in the manner described in Section II B. Each batch is coarse grained, producing n sets of about 750 memories (coarse graining 5000 memories takes about 5 minutes on single 3.1 GHz Intel Core i7 processor). No *single* memory set does better than about 5% error rate on the MNIST test set (blue line). However, used in combination (taking the closest match between a test-set symbol and any memory from the n batches) they are much more accurate (green line). 10 memory sets (about 7500 memories in total) scores 2.8%, equal to the rate achieved using all 6×10^4 unaltered training-set symbols (see Fig. 2). With about 200 memory sets the error rate falls to 1.6%. Further improvement should be possible using larger n or a better choice of similarity measure: in panel (b) we show data in which memories, once obtained, were compared with test-set symbols by translating memories up to ± 3 lattice sites in either direction. The lowest error rate shown is 1.17%. When the memory sets used to produce the data in panel (b) are combined, they classify the MNIST test set at 1.14% error rate. Panel (c) shows the misclassified symbols: some are clearly recognizable, and might be correctly classified if e.g. symbol rotation was accounted for, while others are hard to interpret and would be "correctly" classified only if the training set contained a similar symbol of that type.

The error rate is not a strictly decreasing function of the number of batches n: adding more batches can increase the error rate if the new batches contain a memory of the wrong type that resembles a test-set symbol more closely than any memory of the correct type. However, the overall trend is that increasing n reduces the error
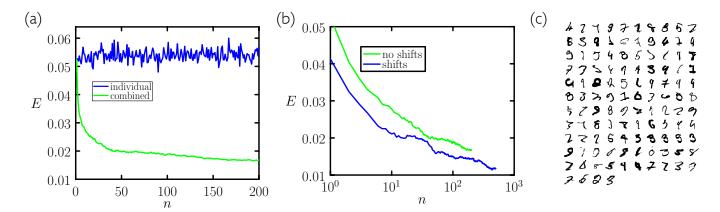
FIG. 5. Sampling. (a) Error rate E achieved on the MNIST test set using n memory sets in parallel (see Fig. 4). Each set (which contains about 750 memories) was obtained by coarse graining batches of 5000 symbols drawn from the MNIST training set. No single memory set does better than about 5% error rate on the MNIST test set (blue line). However, in combination they are much more potent (green line). 10 sets (about 7500 memories in total) scores 2.8%, equal to the rate achieved using all 6×10^4 unaltered training-set symbols (see Fig. 2). With about 200 sets the error rate falls to 1.6%. (b) Further improvement is possible with better measures of image similarity: the green line is reproduced from panel (a), while the blue line was obtained using simple linear shifts of symbols. (c) The memory sets used to produce panel (b) together classify the MNIST test at 1.14% error rate; shown are the 114 misclassified symbols.

rate.

Coarse graining and sampling provide a way of improving the efficiency and accuracy of direct-comparison classification over that offered by unaltered training-set data. This approach works best with processors used in parallel, with each used to sample and coarse grain a single batch of symbols. The load on any single processor is then relatively light; in the example described, each processor needs only to store 5000 symbols drawn from the test set, and subsequently store about 750 memories. Training can be done at the same time on all processors, as can classification, with a final step being a comparison between processors of their results.

D. Coarse graining as a genetic algorithm

The coarse-graining algorithm used here is a simple method of clustering, resembling a particle-clustering algorithm [25–27] or a supervised version of the k-means algorithm [28]. It can also be considered to be a simple genetic algorithm: two parents (a symbol and a memory) produce offspring (a memory) that is retained only if it passes a fitness test (recognizing the parent symbol better than does any other memory). This latter feature suggests that memories can be specialized (made "fitter") for particular tasks by varying the environment in which memories are made. In simulations described thus far we have used batches that contain, on average, equal numbers of symbols of all types. In "harsher" environments, i.e. batches that contain only symbol types that are easily confused, we speculate that memories must be "fitter" in order to survive. Some evidence in support of this speculation is shown in Fig. 3(b): a batch of 3000 symbols containing only 1s and 9s requires more passes of

the coarse-graining algorithm than does a similarly-sized batch containing all symbol types, suggesting that the coarse-graining algorithm has to work "harder" to partition configuration space accurately when only similar symbol types are present [39].

In Fig. 6 we show that memories produced by coarse graining training-set batches that contain only two symbol types can tell apart those two types with reasonable accuracy. Two sets of memories, each coarse grained using 5000 symbols of types 0 and 1 or 0 and 6, achieve an error rate of 1 in 2115 or 8 in 1938 on the (0,1)- or (0,6)-MNIST test set, respectively.
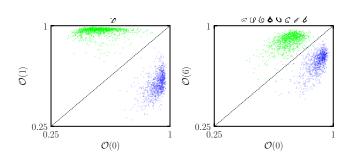

FIG. 6. 2 batches of 5000 symbols, of type 0 and 1 or 0 and 6, are coarse grained and used to classify the $(0, 1)$- or $(0, 6)$-MNIST test set, respectively. Horizontal and vertical axes show the largest overlap between a test-set symbol and the relevant memory type (symbols of type 0 are shown blue). Misclassified symbols are shown at the top of each plot.

III. CONCLUSIONS

Direct comparison of test-set data with training-set data is a conceptually simple method of classification [18–22, 37, 38]. Given a measure of the similarity of two images, we have shown that simple methods of coarse graining and sampling can be used to achieve a more efficient and more accurate direct classification of test-set data than can the unaltered training set. This process creates new symbols from a training set that are a better match for the test set than any of the original training-set symbols. The approach described here, similar to other sampling strategies [29], works naturally on parallel processors: the more processors, the faster is training and classification, and the lighter is the memory load on each processor. The approach applied to the MNIST data set, using nearest-neighbor classification and the simple vector dot product, is not as accurate as state-of-the-art methods [33], but can be improved upon in at least two ways, leading (presumably) to a more rapid fall-off of error rate with batch number than is shown in Fig. 5.

First, a better measure of image similarity [31, 32] could be used, or a better distance metric could be learned [40]. Second, the coarse-graining algorithm constructs the centroids of symbols and memories, but other constructions (e.g. splicing together pieces of symbols) are possible; some of these may enable better sampling of configuration space. More generally, memory-based methods are used in e.g. reinforcement learning [41], and coarse graining and sampling schemes might find application there.

[1] Y. LeCun, Y. Bengio, and G. Hinton, Nature **521**, 436 (2015).
[2] N. M. Nasrabadi, Journal of Electronic Imaging **16**, 049901 (2007).
[3] Y. LeCun, B. E. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. E. Hubbard, and L. D. Jackel, in *Advances in neural information processing systems* (1990) pp. 396–404.
[4] G. E. Hinton, P. Dayan, and M. Revow, IEEE transactions on Neural Networks **8**, 65 (1997).
[5] J. R. Quinlan, in *Machine learning* (Springer, 1983) pp. 463–482.
[6] A. L. Samuel, IBM Journal of Research and Development **3**, 210 (1959).
[7] A. L. Ferguson and J. Hachmann, Molecular Systems Design & Engineering (2018).
[8] C. Desgranges and J. Delhommelle, The Journal of Chemical Physics **149**, 044118 (2018).
[9] N. Portman and I. Tamblyn, Journal of Computational Physics **350**, 871 (2017).
[10] K. Mills, M. Spanner, and I. Tamblyn, Physical Review A **96**, 042113 (2017).
[11] N. Artrith, A. Urban, and G. Ceder, The Journal of Chemical Physics **148**, 241711 (2018).
[12] A. Singraber, T. Morawietz, J. Behler, and C. Dellago, Journal of Physics: Condensed Matter **30**, 254005 (2018).
[13] B. Thurston and A. Ferguson, Molecular Simulation , 1 (2018).
[14] C. Andrieu, N. De Freitas, A. Doucet, and M. I. Jordan, Machine learning **50**, 5 (2003).
[15] D. Levy, M. D. Hoffman, and J. Sohl-Dickstein, arXiv preprint arXiv:1711.09268 (2017).
[16] D. Frenkel and B. Smit, *Understanding molecular simulation: from algorithms to applications*, Vol. 1 (Elsevier, 2001).
[17] F. Y. K. Kossio, S. Goedeke, B. van den Akker, B. Ibarz, and R.-M. Memmesheimer, Physical Review Letters **121**, 058301 (2018).
[18] M.-L. Zhang and Z.-H. Zhou, Pattern recognition **40**, 2038 (2007).
[19] N. Bhatia, arXiv preprint arXiv:1007.0085 (2010).
[20] B. Fritzke, in *Advances in neural information processing systems* (1995) pp. 625–632.
[21] D. Nova and P. A. Estévez, Neural Computing and Applications **25**, 511 (2014).
[22] S. Marsland, J. Shapiro, and U. Nehmzow, Neural networks **15**, 1041 (2002).
[23] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Proceedings of the IEEE **86**, 2278 (1998).
[24] http://yann.lecun.com/exdb/mnist/.
[25] U. Wolff, Physical Review Letters **62**, 361 (1989).
[26] J. Liu and E. Luijten, Physical Review Letters **92**, 035504 (2004).
[27] S. Whitelam, Molecular Simulation **37**, 606 (2011).
[28] K. Wagstaff, C. Cardie, S. Rogers, S. Schrödl, *et al.*, in *ICML*, Vol. 1 (2001) pp. 577–584.
[29] D. B. Skalak, *Prototype selection for composite nearest neighbor classifiers*, Ph.D. thesis, University of Massachusetts at Amherst (1997).
[30] https://en.wikipedia.org/wiki/MNIST_database.
[31] P. Simard, Y. LeCun, and J. S. Denker, in *Advances in neural information processing systems* (1993) pp. 50–58.
[32] S. Belongie, IEEE Trans. Pattern Analysis and Machine Intelligence **24**, 509 (2002).
[33] http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html.
[34] Occasionally we observed two symbols of the same type to shuffle repeatedly between two memories of the same type; at this point the algorithm can be terminated with no loss of accuracy.

[35] Coarse-graining the memories themselves results in a set of 36 new memories that achieves a 14% classification error rate. Repeated coarse-graining eventually produces 10 memories (1 per symbol type) and a classification error rate of about 50%.

[36] S. Garcia, J. Derrac, J. Cano, and F. Herrera, IEEE transactions on pattern analysis and machine intelligence **34**, 417 (2012).

[37] P. Hart, IEEE transactions on information theory **14**, 515 (1968).

[38] F. Angiulli, in *Proceedings of the 22nd international conference on Machine learning* (ACM, 2005) pp. 25–32.

[39] This batch of 3000 symbols was coarse-grained to produce 119 memories (in 23 seconds on a 3.1 GHz Intel Core i7 processor); this set of memories achieves a direct-classification error rate of 0.3% on the $(1, 9)$-MNIST test set, misclassifying 7 out of 2144 symbols.

[40] K. Q. Weinberger, J. Blitzer, and L. K. Saul, in *Advances in neural information processing systems* (2006) pp. 1473–1480.

[41] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, arXiv preprint arXiv:1511.05952 (2015).