entropy

MDPI

Article

Improving the Accuracy of Nearest-Neighbor Classification Using Principled Construction and Stochastic Sampling of Training-Set Centroids

Stephen Whitelam

Molecular Foundry, Lawrence Berkeley National Laboratory, 1 Cyclotron Road, Berkeley, CA 94720, USA; swhitelam@lbl.gov

Abstract: A conceptually simple way to classify 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 cover configuration space. Here we show that this coverage can be substantially increased using coarse-graining (replacing groups of images by their centroids) and stochastic sampling (using distinct sets of centroids in combination). We use the MNIST and Fashion-MNIST data sets to show that a principled coarse-graining algorithm can convert training images into fewer image centroids without loss of accuracy of classification of test-set images by nearest-neighbor classification. Distinct batches of centroids can be used in combination as a means of stochastically sampling configuration space, and can classify test-set data more accurately than can the unaltered training set. On the MNIST and Fashion-MNIST data sets this approach converts nearest-neighbor classification from a mid-ranking- to an upper-ranking member of the set of classical machine-learning techniques.

Keywords: image recognition; nearest-neighbor classification; stochastic sampling

Citation: Whitelam, S. Improving the Accuracy of Nearest-Neighbor Classification Using Principled Construction and Stochastic Sampling of Training-Set Centroids. *Entropy* **2021**, *23*, 149. https://doi.org/10.3390/e23020149

Received: 3 January 2021
Accepted: 21 January 2021
Published: 26 January 2021

1. Introduction

Machine learning, used 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 and guide the outcome of nonequilibrium processes such as self-assembly [13,14]. It is also natural to expect physical concepts, and methods developed in the physical sciences, to be used increasingly to study and improve machine-learning tools [15]. Here we show that physics-inspired ideas of coarse-graining and sampling can be used to improve the efficiency and accuracy of nearest-neighbor image classification.

A conceptually simple method of image recognition is to assign to a test-set image the type of the most similar training-set image [16–20]. The accuracy of such an approach is limited by the method of image comparison, and by the extent to which the training-set data "sample" configuration space. In Figure 1a we show 36 examples of 3 s and 5 s taken from the MNIST data set, which consists of hand-drawn digits [21], to emphasize 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 translations or elastic distortions of digits [21]. One drawback of such approaches is that they require prior knowledge of the concept to be classified.

Here we use the MNIST and Fashion-MNIST [22] data sets to show that principled coarse-graining and sampling can be used to substantially reduce error rates of memory-based, nearest-neighbor classifier schemes without requiring information beyond that contained in the training set.

We show how to identify a set of coarse-grained symbols or *memories* able to correctly classify training-set data by capturing the diversity of the set of symbols while omitting redundant objects [see Figure 1b]. In this context, "coarse-graining" means combining symbols into groups and identifying their centroids, to produce a sparse description of configuration space; it does not mean sampling images at lower resolution. The process of coarse-graining is similar in spirit to the process enacted by particle-clustering algorithms [23–25]; by a supervised k-means algorithm [26]; and by learning vector quantization classifiers [19] such as the growing neural gas [18] or growing self-organizing network [20]. For MNIST or Fashion-MNIST data, a set of N training-set symbols can be described by a set of about $0.15N$ or $0.2N$ memories, respectively. 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 [21].

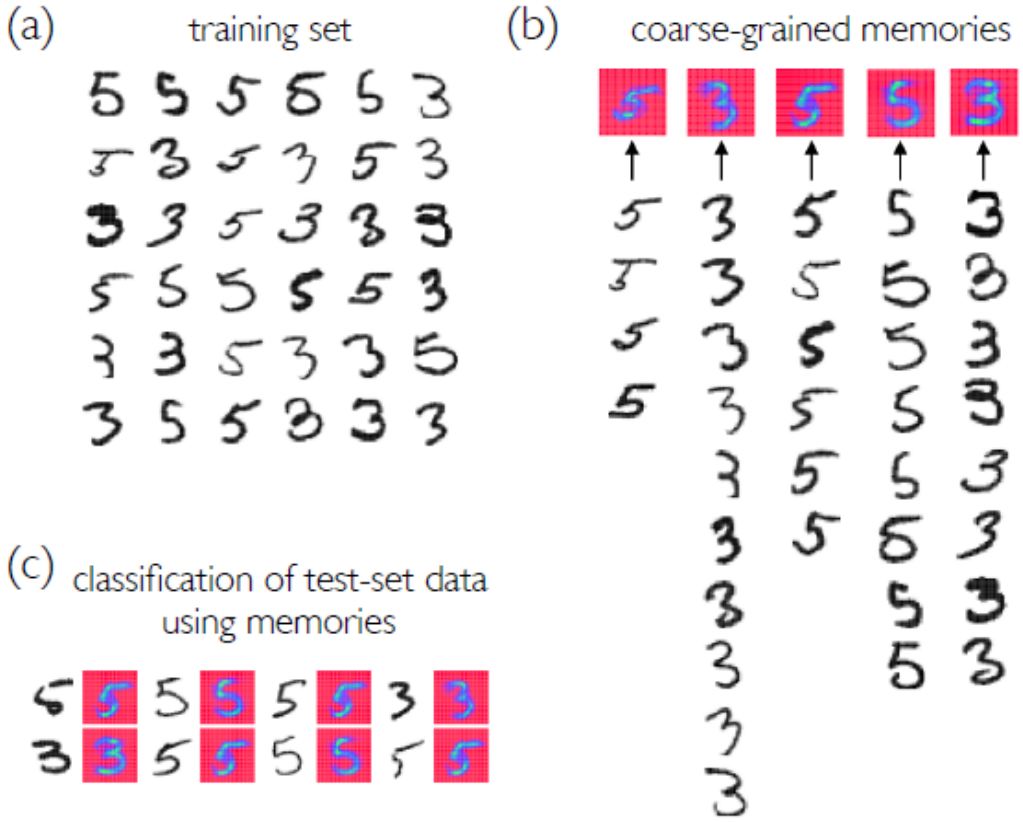

Figure 1. (**a**) A set of 36 examples taken from the MNIST training set gives rise to 5 memories [panel (**b**)] upon coarse-graining according to the procedure described in this paper. (**b**) Each of the 36 symbols is "stored" in a memory; memories are the centroids of their constituent symbols. Memories 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 nearest-neighbor classification: the coarse-graining algorithm "learns" to partition configuration space accurately. (**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.

Moreover, memory sets created by drawing batches of symbols stochastically from the training set can be used 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 [27]; however, here we perform classification using objects not present in the original

training set, and use the process of sampling to identify new symbols that better describe unseen data.

Nearest-neighbor classification using the normalized vector dot product applied to raw pixels classifies the MNIST and Fashion-MNIST data sets at an error rate of about 2.8% and 15% respectively, making them mid-ranking members of the set of classical machine-learning methods [22]. We show here that under coarse-graining and sampling these error rates decrease to about 1.6% and 10% respectively, comparable with the best classical machine-learning methods surveyed in Ref. [22]. Deep-learning methods are generally more accurate than classical ones [28], but the point of this paper is to show the improvement possible using principled combination and sampling of training-set data, without knowledge of the concept being classified and without knowing that the data correspond to images (and so not allowing expansion of the training set using e.g., elastic distortions or translations).

In what follows we describe the method using the MNIST data set as an illustration, and apply it to MNIST and Fashion-MNIST.

2. Image Recognition via Nearest-Neighbor Classification

2.1. Unaltered Training Set

The MNIST data set consists of $L = 7 \times 10^4$ 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 the value of 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 $\theta \in \{1, \ldots, L\}$, 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 Figure 2a we show the error rate, the number of incorrect assignments divided by the test-set size of 10^4, that results from 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,

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

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

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

with larger values indicating greater similarity. This measure results in the blue line shown in the figure. We shall refer to the normalized vector dot product as the *overlap* of vectors A and B,

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

There exist more discriminating measures of the similarity of two images [29,30]. Here our aim is to show how the accuracy of a classifier, using a simple measure of similarity, can be improved by sampling.

The error rate resulting from nearest-neighbor classification, 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 [31] ($\approx$0.2%), but neither is it terrible: it is comparable to the rates produced by some of the neural networks of the late 1990s [21]. However, the error rate shown in Figure 2a 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 nearest-neighbor classification 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 nearest-neighbor classification, by the set of coarse-grained memories.

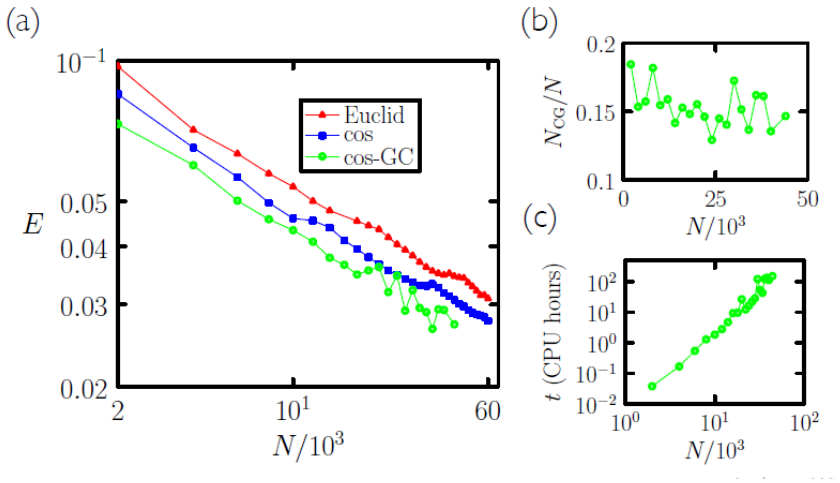

Figure 2. (**a**) Error rate E for classification of the 10^4-digit MNIST test set by nearest-neighbor classification using N symbols from the training set (log-log plot). For red and blue lines, the measure of similarity is the Euclidean distance (1) and normalized vector dot product (2), respectively. The green line was obtained by coarse-graining N symbols of the training set into $N_{CG} < N$ memories. Classification accuracy is no worse and is often better upon coarse-graining (compare blue and green lines). (**b**) N_{CG}/N is typically about 15%. (**c**) The CPU time required for coarse-graining scales roughly algebraically with N.

2.2. Coarse-Graining Training-Set Symbols

We aim to *coarse grain* N symbols S^θ to produce $N_{CG} \leq N$ *memories* M^α, $\alpha \in \{1, \ldots N_{CG}\}$. Each memory has entries M^α_i, $i \in \{1, \ldots, P\}$, 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

$$M^\alpha = \hat{M}^\alpha / N^\alpha, \tag{4}$$

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:

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

   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 Equation (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:
   $$\hat{M}_i^\beta \to \hat{M}_i^\beta + S_i^\theta; \quad N^\beta \to N^\beta + 1. \tag{6}$$
   If S^θ is currently stored in a different memory M^α then remove S^θ from M^α:
   $$\hat{M}_i^\alpha \to \hat{M}_i^\alpha - S_i^\theta; \quad N^\alpha \to N^\alpha - 1. \tag{7}$$
   If S^θ is not yet stored in a memory then (7) is not necessary.
   (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 Equation (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 nearest-neighbor classification, 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 (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.). With batches made from the MNIST training set we observed a compression rate of about 6 or 7: the number of coarse-grained memories N_{CG} produced from a batch of N symbols is typically $\approx 0.15N$ (Figure 2b). The CPU time taken for the algorithm to run scales roughly algebraically with N (Figure 2c).

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 x_- / x_α we moved that symbol to the batch. Here $\alpha = T(S^\theta)$ is the type of symbol S^θ; x_α is the number of symbols of type $\alpha \in \{0, \ldots, 9\}$ in the training set prior to the move; and $x_- \equiv \min_\alpha x_\alpha$. If the move was successful then we reduced x_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 independently, without replacement.)

Significantly, sets of coarse-grained memories are no less accurate than their constituent symbols in classifying the MNIST test set (via nearest-neighbor classification, using Equation (2)): compare the blue and green lines in Figure 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 Figure 3a we show 1000 symbols that yield 144 memories upon coarse-graining; both classify the MNIST test set at 10% error rate (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%.). 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.

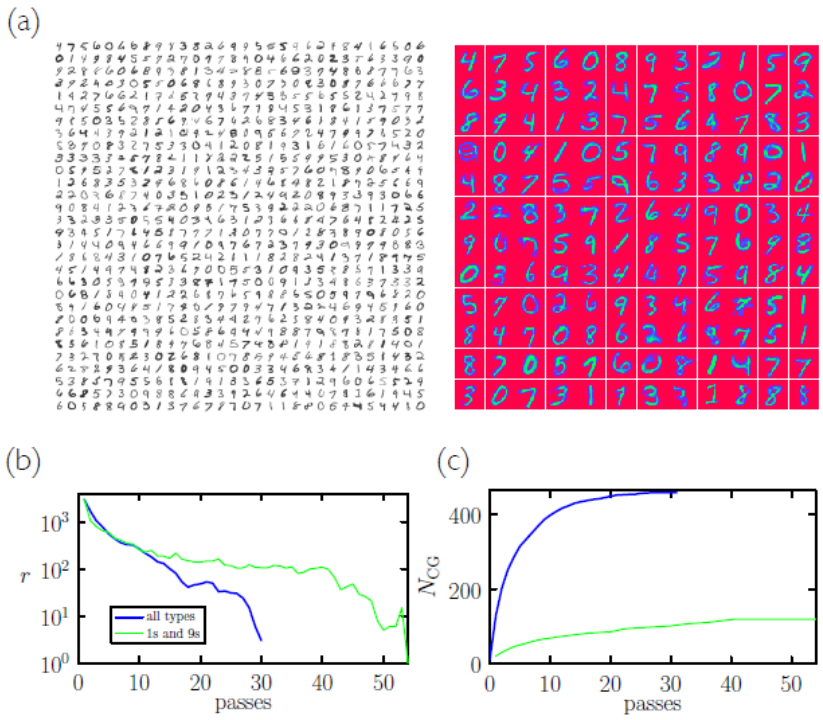

Figure 3. Coarse-graining algorithm. (**a**) This 900-symbol batch (left) taken from the MNIST training set yields 144 memories (right) under coarse-graining. Both the symbols and the memories achieve a 10% nearest-neighbor classification error rate on the MNIST test set. (**b**) Number of rearrangements 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 1 s and 9 s. (**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.

2.3. *Sampling*

Thus, coarse-graining achieves a significant compression of information with no loss (and often some slight gain) of nearest-neighbor 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 [32] such as the condensed nearest-neighbor approach [33,34], or to adaptive versions of learning vector quantization [18–20]. 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. [27], 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.

Figure 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 nearest-neighbor classification. 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.

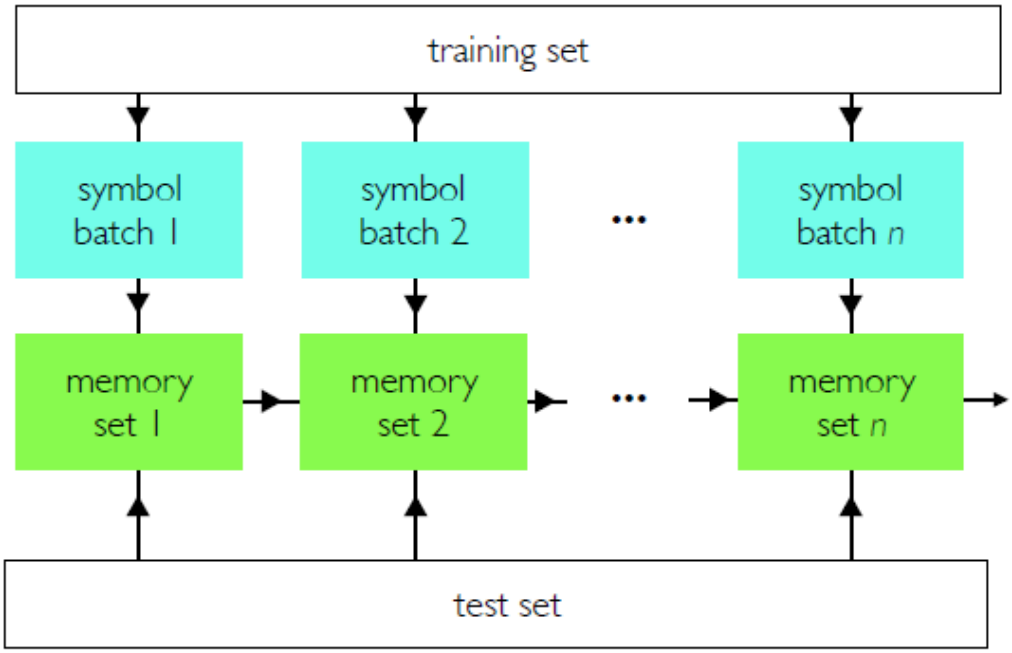

Figure 4. Coarse-graining and sampling. Batches of symbols are constructed by drawing symbols from the training set. Each batch is coarse-grained to produce a set of memories. Each set of memories is used to classify the entire test set, and results from all batches are compared.

In Figure 5a we show the error rate achieved on the MNIST test set using n memory sets in parallel, per Figure 4. n batches of 5000 symbols are drawn from the training set, in the manner described in Section 2.2. Each batch is coarse-grained, producing n sets of about 750 memories (coarse-graining 5000 memories takes about 5 min 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 Figure 2). With about 200 memory sets the error rate falls to 1.6%.

Further improvement is possible using larger n or by invoking additional knowledge about the concept being classified: in panel (b) we show data for 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.

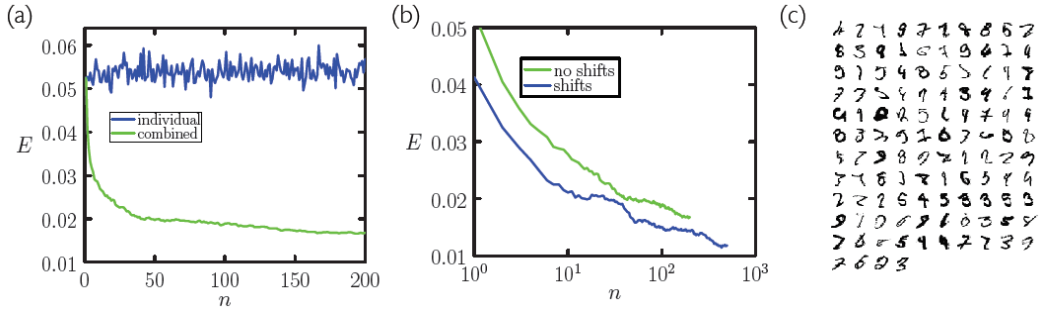

Figure 5. Sampling. (**a**) Error rate E achieved on the MNIST test set using n memory sets in parallel (see Figure 4). Each set (which contains about 750 memories) was obtained by coarse-graining batches of 5000 symbols drawn from the MNIST training set. No single memory set does better than about 5% 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 Figure 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 linear shifts of symbols. (**c**) The memory sets used to produce panel (**b**) together classify the MNIST test at 1.14% error rate; we show here the 114 misclassified symbols.

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 rate.

In Figure 6 we repeat the calculation of Figure 5a, but now using the Fashion-MNIST dataset [22]. Fashion-MNIST is a more challenging variant of MNIST consisting of items of clothing of 10 types. Coarse-graining results in a symbols-to-memories compression rate of about 4 or 5 (as opposed to 6 or 7 for MNIST), and 1000 batches of memories each derived from 5000 symbols yields a test-set error rate of 10.5%. This result compares favorably with the error rates quoted in Ref. [22], all of which lie above 10%: for instance, K-nearest-neighbor classifiers using the unaltered training set achieve 14–16% error rate, showing the advantage accrued by statistical sampling of the training set.

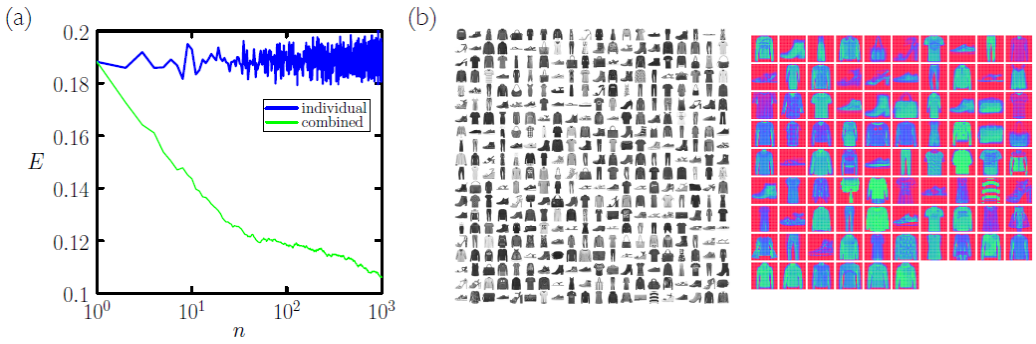

Figure 6. (**a**) Similar to Figure 5a, but for the Fashion-MNIST dataset [22]. We use n memory sets each derived from 5000 symbols. For $n = 1000$, classification accuracy is 10.5%. (**b**) For this data set the symbols-to-memories compression rate is about 4 or 5, rather than 6 or 7 for MNIST, indicating a slightly greater variability per symbol type. This batch of 400 Fashion-MNIST symbols yields 86 memories under coarse-graining.

Coarse-graining and sampling provide a way of improving the efficiency and accuracy of nearest-neighbor 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 examples described, each processor needs only to store 5000 symbols drawn from the test set, and subsequently store about 750 (for MNIST) or 1000 (for Fashion-MNIST) 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.

2.4. Coarse-Graining as a Genetic Algorithm

The coarse-graining algorithm used here is a method of clustering, resembling a particle-clustering algorithm [23–25] or a supervised version of the *k*-means algorithm [26]. It can also be considered to be a type of 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 to survive. Some evidence in support of this speculation is shown in Figure 3b: a batch of 3000 symbols containing only 1 s and 9 s 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 (This batch of 3000 symbols was coarse-grained to produce 119 memories (in 23 s on a 3.1 GHz Intel Core i7 processor); this set of memories achieves a nearest-neighbor classification error rate of 0.3% on the $(1,9)$-MNIST test set, misclassifying 7 out of 2144 symbols.).

In Figure 7 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.

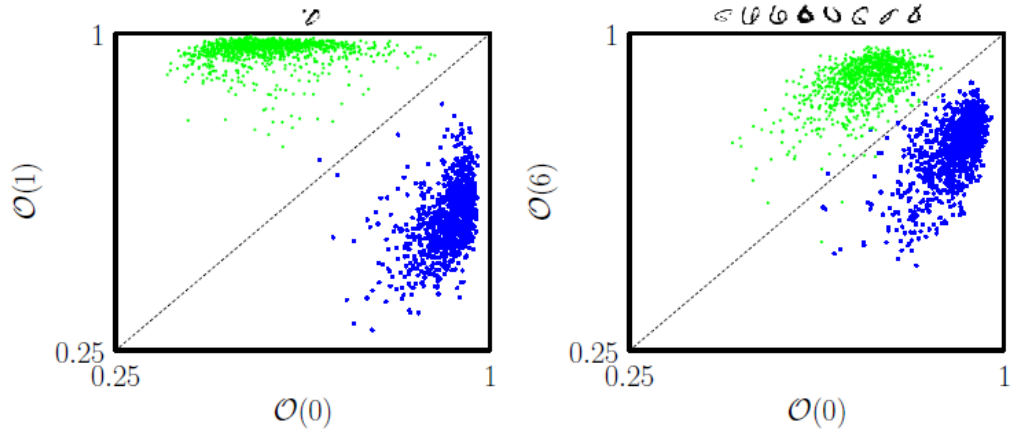

Figure 7. 2 batches of 5000 symbols, of type 0 and 1 or 0 and 6, are coarse-grained and used to classify the $(0,1)$- or $(0,6)$-MNIST test set, respectively. Horizontal and vertical axes show the largest overlap between a test-set symbol and the relevant memory type (symbols of type 0 are shown blue). Misclassified symbols are shown at the top of each plot.

3. Conclusions

Nearest-neighbor classification of test-set data by training-set data is a conceptually simple method of classification [16–20,33,34]. 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 nearest-neighbor 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 [27], works naturally on parallel processors: the more processors, the more accurate is classification. The approach applied to the MNIST and Fashion-MNIST data sets, using nearest-neighbor classification and the simple vector dot product, compares favorably with other forms of classical machine-learning method [22]. Although not as accurate as deep-learning methods [28,31], the present method could be improved upon in at least two ways. First, a better measure of image similarity could be used [29,30], or a better distance metric could be learned [35]. 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, coarse-graining and sampling schemes might find application in other settings, such as neural networks that make use of memory-based structures.

Funding: This work was performed at the Molecular Foundry, Lawrence Berkeley National Laboratory, supported by the Office of Science, Office of Basic Energy Sciences, of the U.S. Department of Energy under Contract No. DE-AC02–05CH11231.

Data Availability Statement: Code used to produce the results in this paper can be shared upon request.

Acknowledgments: I thank Marc Pons Whitelam for many discussions about digit recognition, and thank Isaac Tamblyn and Tess Smidt for comments on the paper.

Conflicts of Interest: The author declares no conflict of interest.

References

1. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436. [CrossRef]
2. Nasrabadi, N.M. Pattern recognition and machine learning. *J. Electron. Imaging* **2007**, *16*, 049901.
3. LeCun, Y.; Boser, B.E.; Denker, J.S.; Henderson, D.; Howard, R.E.; Hubbard, W.E.; Jackel, L.D. Handwritten digit recognition with a back-propagation network. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 26–29 November 1990; pp. 396–404.
4. Hinton, G.E.; Dayan, P.; Revow, M. Modeling the manifolds of images of handwritten digits. *IEEE Trans. Neural Netw.* **1997**, *8*, 65–74. [CrossRef] [PubMed]
5. Quinlan, J.R. Learning efficient classification procedures and their application to chess end games. In *Machine Learning*; Springer: Berlin/Heidelberg, Germany, 1983; pp. 463–482.
6. Samuel, A.L. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.* **1959**, *3*, 210–229. [CrossRef]
7. Ferguson, A.L.; Hachmann, J. Machine learning and data science in materials design: A themed collection. *Mol. Syst. Des. Eng.* **2018**, *3*, 429–430. [CrossRef]
8. Desgranges, C.; Delhommelle, J. A new approach for the prediction of partition functions using machine learning techniques. *J. Chem. Phys.* **2018**, *149*, 044118. [CrossRef]
9. Portman, N.; Tamblyn, I. Sampling algorithms for validation of supervised learning models for Ising-like systems. *J. Comput. Phys.* **2017**, *350*, 871–890. [CrossRef]
10. Mills, K.; Spanner, M.; Tamblyn, I. Deep learning and the Schrödinger equation. *Phys. Rev. A* **2017**, *96*, 042113. [CrossRef]
11. Artrith, N.; Urban, A.; Ceder, G. Constructing first-principles phase diagrams of amorphous Li x Si using machine-learning-assisted sampling with an evolutionary algorithm. *J. Chem. Phys.* **2018**, *148*, 241711. [CrossRef]
12. Singraber, A.; Morawietz, T.; Behler, J.; Dellago, C. Density anomaly of water at negative pressures from first principles. *J. Phys. Condens. Matter* **2018**, *30*, 254005. [CrossRef]
13. Thurston, B.; Ferguson, A. Machine learning and molecular design of self-assembling-conjugated oligopeptides. *Mol. Simul.* **2018**, *44*, 930–945. [CrossRef]
14. Whitelam, S.; Tamblyn, I. Learning to grow: Control of material self-assembly using evolutionary reinforcement learning. *Phys. Rev. E* **2020**, *101*, 052604. [CrossRef] [PubMed]
15. Kossio, F.Y.K.; Goedeke, S.; van den Akker, B.; Ibarz, B.; Memmesheimer, R.M. Growing Critical: Self-Organized Criticality in a Developing Neural System. *Phys. Rev. Lett.* **2018**, *121*, 058301. [CrossRef] [PubMed]
16. Zhang, M.L.; Zhou, Z.H. ML-KNN: A lazy learning approach to multi-label learning. *Pattern Recognit.* **2007**, *40*, 2038–2048. [CrossRef]
17. Bhatia, N. Survey of nearest neighbor techniques. *arXiv* **2010**, arXiv:1007.0085.
18. Fritzke, B. A growing neural gas network learns topologies. In Proceedings of the Advances in Neural Information Processing Systems, Denver, CO, USA, 27–30 November 1995; pp. 625–632.
19. Nova, D.; Estévez, P.A. A review of learning vector quantization classifiers. *Neural Comput. Appl.* **2014**, *25*, 511–524. [CrossRef]

20. Marsland, S.; Shapiro, J.; Nehmzow, U. A self-organising network that grows when required. *Neural Netw.* **2002**, *15*, 1041–1058. [CrossRef]
21. LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. Gradient-based learning applied to document recognition. *Proc. IEEE* **1998**, *86*, 2278–2324. [CrossRef]
22. Xiao, H.; Rasul, K.; Vollgraf, R. Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms. *arXiv* **2017**, arXiv:1708.07747.
23. Wolff, U. Collective Monte Carlo updating for spin systems. *Phys. Rev. Lett.* **1989**, *62*, 361. [CrossRef]
24. Liu, J.; Luijten, E. Rejection-free geometric cluster algorithm for complex fluids. *Phys. Rev. Lett.* **2004**, *92*, 035504. [CrossRef] [PubMed]
25. Whitelam, S. Approximating the dynamical evolution of systems of strongly interacting overdamped particles. *Mol. Simul.* **2011**, *37*, 606–612. [CrossRef]
26. Wagstaff, K.; Cardie, C.; Rogers, S.; Schrödl, S. Constrained k-means clustering with background knowledge. In Proceedings of the ICML, Williamstown, MA, USA, 28 June–1 July 2001; Volume 1, pp. 577–584.
27. Skalak, D.B. Prototype Selection for Composite Nearest Neighbor Classifiers. Ph.D. Thesis, University of Massachusetts at Amherst, Amherst, MA, USA, 1997.
28. A MNIST-Like Fashion Product Database. Benchmark. Available online: https://github.com/zalandoresearch/fashion-mnist (accessed on 15 January 2020).
29. Simard, P.; LeCun, Y.; Denker, J.S. Efficient pattern recognition using a new transformation distance. *Adv. Neural. Inform. Process Syst.* **1993**, *1*, 50.
30. Belongie, S. Shape Matching and Object Recognition Using Shape Contexts. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *24*, 509–522. [CrossRef]
31. What Is the Class of This Image? Available online: http://rodrigob.github.io/are_we_there_yet/build/classification_datasets_results.html (accessed on 15 January 2020).
32. Garcia, S.; Derrac, J.; Cano, J.; Herrera, F. Prototype selection for nearest neighbor classification: Taxonomy and empirical study. *IEEE Trans. Pattern Anal. Mach. Intell.* **2012**, *34*, 417–435. [CrossRef]
33. Hart, P. The condensed nearest neighbor rule (Corresp.). *IEEE Trans. Inf. Theory* **1968**, *14*, 515–516. [CrossRef]
34. Angiulli, F. Fast condensed nearest neighbor rule. In Proceedings of the 22nd International Conference on Machine Learning, Bonn, Germany, 7–11 August 2005; pp. 25–32.
35. Weinberger, K.Q.; Blitzer, J.; Saul, L.K. Distance metric learning for large margin nearest neighbor classification. *J. Mach. Learn. Res.* **2006**, *10*, 1473–1480.