DOI: http://dx.doi.org/10.7551/978-0-262-31709-2-ch160

Evolution of an artificial visual cortex for image recognition

Samuel D. Chapman[1,3], David B. Knoester[2,3], Arend Hintze[2,3], and Christoph Adami[2,3]

[1]Dept. of Computer Science and Engineering
[2]Dept. of Microbiology and Molecular Genetics
[3]BEACON Center for the Study of Evolution in Action
Michigan State University, East Lansing, MI 48824
Email: {chapm236, dk, hintze, adami}@msu.edu

Abstract

Evolutionary robotics has been successful in creating agents that successfully link perception with appropriate action. However, the visual fields utilized by such agents is usually extremely small compared to the retinas linked to the visual cortex of animals. Evolving a cortex that processes larger fields of view in a selective and robust manner is challenging because fitness landscapes that are sensitive to this level of details are difficult to design. Here, we decouple the perception from the action part of evolutionary robotics, and present a new way to evolve logic circuits to perform image recognition on the well-known MNIST data set, which comprises 60,000 training and 10,000 testing handwritten numerals. The logic circuits are encoded in a genome that is evolved using a fitness function based on the true positive and true negative classification rates of the numerals. Following evolution, individual circuits achieve in excess of 80% recognition accuracy on the testing data. By pooling highly evolved individual circuits from multiple evolutionary histories into a committee, testing accuracy is increased to 93.5%.

This work demonstrates that evolving logic circuits to solve a classification task is feasible. We also found the evolved circuits to be much smaller in scale compared to other machine learning methods that are conventionally used on such problems. To our knowledge, this represents the first time that relatively small logic circuits have been evolved to reach this level of performance on the automated recognition of handwriting, and promises new approaches to the integration of evolutionary algorithms and intelligent systems.

Introduction

The primary visual cortex of mammals is a marvel of complexity and function, and has been studied for decades both by those wishing to *understand* it, as well as those wishing to *duplicate* it. Object recognition by the visual cortex is fast (Thorpe et al., 1996), specific, and invariant: we can recognize faces, for example, in different orientations, under a variety of lighting conditions, and bearing different expressions. Significant efforts have been expended on modeling the hierarchical organization of the visual cortex, with the aim of both elucidating the algorithm behind fast and invariant object recognition (Riesenhuber and Poggio, 1999), as well as harnessing it for computer vision (Serre et al., 2007). The usefulness of such a system for the organism bearing it is almost too obvious to point out: it is clear that the selective advantage of a fast, accurate, and robust sensing system must be enormous. Indeed, the visual cortex evolved to be the largest system in the human brain.

While significant progress has been made on our understanding of the visual cortex, the development of a visual cortex suitable for embodied robotics (Pfeifer and Bongard, 2006) remains a challenging problem. Typically, embodied mobile robots need to carry lightweight, fast, and energy-efficient components, and typically such systems have a very limited bandwidth. Most (if not all) of the implementations of automatic computer vision fall short in one or more of these requirements. To be sure, advanced systems such as Google's self-driving car (Ibañez-Guzmán et al., 2012) have achieved extremely robust object recognition and navigation skills, however at the cost of processing almost a Gigabyte/s of sensorial data (Chau et al., 2013). Less advanced systems (such as, say, an ant) navigate robustly with arguably nine orders of magnitude less sensorial data, using a far more compact circuitry, and minimal energy requirements. How can we understand the structure and function of such visual systems?

Evolutionary robotics

Within the field of evolutionary robotics (Nolfi and Floreano, 2000; Floreano and Keller, 2010), it is becoming more and more clear that algorithms for perception should not be separated from those for action, that instead perception and action should be considered as an integrated system computing appropriate behavior from contextual information. Evolving the neural control structures for navigation (Floreano and Mondada, 1998; Edlund et al., 2011) has the advantage that it is free of any preconceived notions of what is designable and the added bonus of robustness, because the prime directive of an evolutionary fitness landscape is the survival of the agent. Advances have been made specifically in the area of *active categorical perception*, where an agent actively moves to better categorize the object it sees (Beer, 1996, 2003; van Dartel et al., 2005; Marstaller et al., 2013).

In this type of work, the visual system of the agent is necessarily primitive, as the emphasis lies squarely on the evolution of the perception-action loop. One approach to extend this work could be to simply extend the size of an agent's retina, with the hope that the added detail within the visual field will prove useful for robot behavior. The drawback of this approach is familiar to most that have worked in this field: designing a fitness landscape (Nelson et al., 2009) that makes exquisitely accurate and robust image recognition necessary is extremely challenging, because in most artificial worlds an agent can achieve success with relatively few cues. Floreano et al. (2005) were able to evolve robots that navigate using an evolved retina with a resolution of 5×5 visual neurons, and showed that selection tried to exploit a subset of salient visual features, rather than to process the whole available image.

In order to test whether image recognition algorithms can be evolved in the absence of actions–and to circumvent the problem of having to design a fitness landscape where fast, robust and selective image recognition is necessary for appropriate behavior–we decided to study whether evolution can re-create an object-recognition system suitable for embodied robotics, by evolving an artificial visual cortex that recognizes hand-written numerals. This task (a sub-problem of what is commonly referred to as "optical character recognition", or OCR) is a well-known staple of the machine learning community, and as a consequence several benchmarks are available to compare performance. In our work (as opposed to the standard machine learning application), we have more goals besides trying to achieve maximum accuracy in image classification. We also require the resulting network to be fast, small, and easily transferrable from one computing platform to another. But most importantly, the solution must be evolvable. We can imagine, for example, that the evolved visual cortex would be connected to the neural controlling machinery of an embodied robot, so that both can then evolve together.

In what follows, we use evolutionary algorithms to evolve logic circuits that classify the numerals from the MNIST data set (LeCun et al., 1998), a well-known benchmark in OCR. Evolutionary algorithms employ the principles of evolution by natural selection in a computational context to solve complex problems (Bäck, 1996). Our approach makes use of a framework to evolve *Markov networks* (MN), a development from recent work (Edlund et al., 2011). MNs comprise a series of state variables that interact with one another through a collection of evolved "computational gates." These gates specify how state variables change in response to one another. Gates can be either probabilistic, much like fuzzy logic, or deterministic, as in traditional logic circuits. The Markov networks themselves are represented by a circular list of integers (the "genome") that describes each gate and the state variables with which they interact. Mutations within our evolutionary algorithm operate directly on these genomes, and we employ a fitness function that is based on true positive and true negative recognition rates. In this work, we make use of deterministic gates only, and thus evolve logic circuits that are essentially complex boolean functions that classify numerals from the MNIST data set.

Previous techniques applied to OCR on handwritten characters include K-nearest-neighbors (LeCun et al., 1998), support vector machines (Decoste and Schölkopf, 2002), and learning classifiers such as artificial neural nets and convolutional nets (LeCun et al., 1998; Ciresan et al., 2012; Salakhutdinov and Hinton, 2007). In the case of neural network-based approaches, edge weights are typically optimized based on an error-minimization algorithm such as backpropagation. However, the number of network layers, along with the number of computational units per layer, remains fixed during training. Over time, these approaches have improved in performance on the MNIST data set. For example, in 1998 the best neural network *error rate*, the rate of misclassification on the test images, was 1.6%, achieved using a 2-layer neural network with 300 hidden states (LeCun et al., 1998). In 2007, Salakhutdinov and Hinton achieved an error rate of 1.0%, using a "deep learning" neural network architecture (Salakhutdinov and Hinton, 2007). Deep learning involves the use of a higher-than-normal number of layers; in this case, five neural network layers were used, with all but the last layer using hundreds of nodes. In 2010, Ciresan *et al.* were able to decrease the error rate to 0.35%, using six layers and an even greater number of nodes per layer (Ciresan et al., 2010). The current world record error rate for the MNIST data set is 0.23%, again set by Ciresan *et al.* in 2012, where they used a 35-member committee (7,700 total neurons) of convolutional networks trained via back-propagation for 490 hours on GPU (Ciresan et al., 2012).

Here we use evolution to discover logic circuits that classify the numerals from the MNIST data set. The evolved circuits are of larger scale than other evolutionary approaches, having 784 inputs and 20 outputs, and unlike ANNs and convolutional networks, evolve to a highly concise representation of about 100 computational gates. While individual evolved logic circuits have remarkable accuracy given their relatively small size, the diversity inherent to evolutionary algorithms enables us to make effective use of *committees*. Using committees of 30 circuits, we achieve an accuracy of approximately 93.5% on the MNIST data set. This level of scalability of logic circuit evolution has not been seen in other studies (Stomeo et al., 2005; Vassilev and Miller, 2000; Torresen, 2001), and demonstrates that the evolution of large logic circuits to perform a practical application is feasible. Finally, because Markov networks represent logic circuits, they are capable of being rendered on physical hardware such as FPGAs.

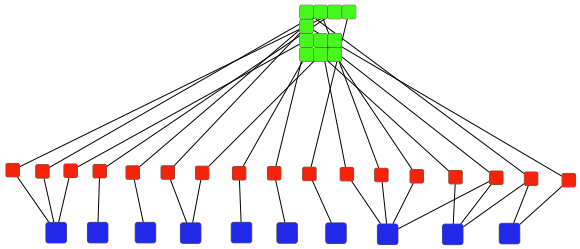


Figure 1: A schematic showing how a Markov network operates. Inputs (green) in a 4x4 image field connect to logic gates (red), which produce the 10 outputs (blue).

Methods

Evolving logic circuits

Defined as a set of probabilistically interacting state variables (Koller and Friedman, 2009), Markov networks (MNs)–which are frequently used to model stochastic processes–can also encode a wide variety of behaviors. The state variables (SVs) represent inputs to the MN, outputs from the MN, and "hidden state variables," that is, SVs that are internal to the MN. A proof-of-principle that MNs can successfully evolved to control behavior (including the usage of memory) was provided in a study of animat navigation in a maze (Edlund et al., 2011). There, some SVs act as sensors of the external environment, and other SVs act as motor outputs and memory. To evolve logic circuits, we use that framework for the evolution of MNs, however, here we use only deterministic gates, as we will discus in more detail below. Figure 1 depicts an example Markov network. Here, inputs to the MN are pixels from a small 4x4 image (green). These inputs are provided to a series of logic gates (red), which in turn produce outputs (blue).

State variables are connected to each other in a directed fashion in a MN, and this connection is mediated through logic gates. Each gate is connected to a number of SVs as input and produces a specified number of SVs as output. During each network update, the inputs to each gate are used to calculate the values that will be written into the gate's output SVs. The outputs produced by a gate thus depend on the value of the input SVs and the specific logic of the gate. To illustrate how these gates work, Table 1 shows a truth table for a 2-input, 2-output logic gate. Here, this gate is connected to input SVs a and b and output SVs c and d. If a and b take the binary values 1 and 0 respectively during one *tick* (a tick represents one execution of all logic gates in a MN), then this gate will write a 1 into both output SVs c and d. If more than one gate writes into the same SV, the values are combined via a logical OR.

To evolve logic circuits, we encode the circuit within a digital genome: a circular list of integers. Within this genome, each gate is encoded by a gene. The beginning

Table 1: A sample logic table for a deterministic gate with 2 inputs and 2 outputs.

Inputs	Outputs
a b	c d
1 1	0 1
0 1	0 0
1 0	1 1
0 0	0 1

of each gene is identified by a specific marker (the "start codon"), and the gene defines the entire functionality of its associated gate. Specifically, genes specify the identity of input and output state variables for each gate, as well as the logic table defining the operation of the gate. In this study, the genome is limited to 40,000 integers, which has an overall capacity of a few thousand genes.

In the evolutionary process, the genome is subject to point mutation, duplication, or deletion. A *point mutation* in the genome randomly changes the value of an integer to another value, whereas a *duplication* or *deletion* inserts random integers into or removes a small section from the genome, respectively. Aside from selecting the range from which random integers are drawn, there are no constraints placed on the values of the integers within the genome. For this reason, not all genes will necessarily be useful toward the problem at hand. Of course, as with natural genomes, nonfunctional genes can serve as a "bank" of genetic diversity that evolution can make use of. Finally, while the MN framework allows the input and output dimension of gates to change via mutation, for simplicity here we fix gates to 4 inputs and 4 outputs. A detailed exposition of the encoding of gates within the circular list of integers is provided in the supplementary information of Edlund et al. (2011).

Recognizing characters

Each image in the MNIST data set is rendered as a 28×28 pixel grayscale image, where each pixel has a grayscale value from 0 to 255. A sample of these images is shown in Fig. 2, along with a sample of diccicult to classify images on the right. In the literature, numerous pre-processing steps are often taken in order to render the images more conducive to particular machine-learning methods, but here we use a simple binary transform on the raw images as delivered in a previous study (LeCun et al., 1998). Specifically, we transform each image such that all pixels with a gray level of 0 are treated as a binary "0," and all nonzero pixels are treated as a binary "1." Each of the 784 binary pixel values are then provided to MNs as inputs. Each MN produces 20 outputs, which represent the classification produced by the MN. Each pair of output bits represents a yes/no answer for one class. Specifically, $c_i = b_{0,i} \wedge \neg b_{1,i}$, where $b_{0,i}$ is the first bit for class i, $b_{1,i}$ is the second bit for class i, and

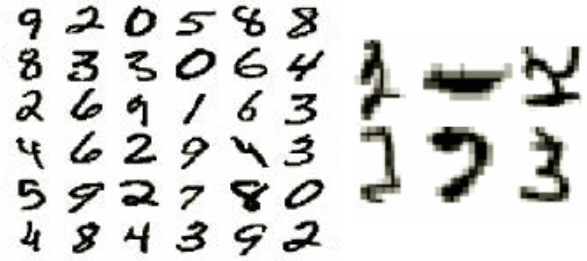


Figure 2: A sample of grey-scale images of hand-written numerals from the MNIST database (left), along with a set of difficult to classify images from the same set.

c_i is the MNs boolean decision for whether the image under consideration is thought to belong to class i. $b_{0,i}$ can be thought of as the activating bit for a class decision, whereas $b_{1,i}$ serves an inhibitory function such that it negates a positive class decision by $b_{0,i}$. We allow the MNs to produce multiple answers for each image. This may seem counter-intuitive, but with the appropriate choice of fitness function, the MNs can evolve to only guess the correct answer, while producing negative answers for the other classes.

The fitness function used to evolve MNs is based on the true positive and true negative rates (TPR and TNR, respectively) of a network's decisions for each class. In essence, a network is rewarded for identifying which class an image belonged to, and also for identifying which classes it did *not* belong to. If even one of these details was correct, the network receives some fitness for that guess. By allowing such "partial credit," we provide a more smooth fitness landscape than if the exact answers for an image were to be required. The specific fitness function we use is:

$$f = \sqrt{\sum_{i=0}^{9} (\mathrm{tpr}_i + \mathrm{tnr}_i + \mathrm{out}_i)^2}, \qquad (1)$$

where tpr_i is the measured true positive rate on class i, tnr_i is the true negative rate on class i, and f is the resulting fitness. If the network outputs a positive decision for *any* image of class i, out_i is set to 1.0, thus encouraging the networks to evolve to classify *all* numerals (as opposed to simply focusing on the more easily classified numerals). Ties (outputting multiple classifications for a single image), are broken randomly to produce an accuracy score. We did not use the accuracy score of the network as fitness, as the resulting landscape proved too difficult to adapt to (data not shown). Note that when reporting fitness, we show the percentage of the maximum fitness achievable via Eq. (1).

The MNIST data set is divided into two parts: A set of 60,000 training images, and a set of 10,000 testing images. The Markov networks were only evolved on the training images. After evolution on the training set, the most fit individual network was then tested on the testing set. The accuracy

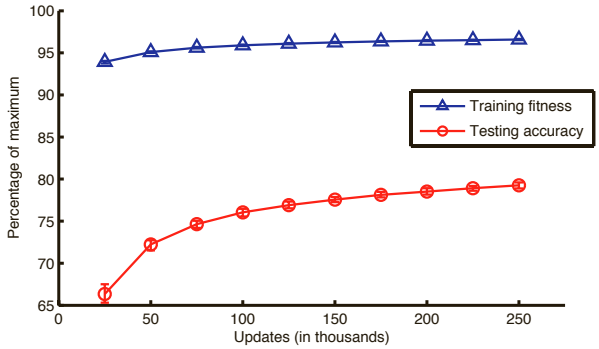


Figure 3: Mean training fitness and test accuracy of dominant individuals over time, over 30 independent runs. Using 200-fold bootstrapping, (small) error bars are constructed depicting 95% confidence intervals around the mean.

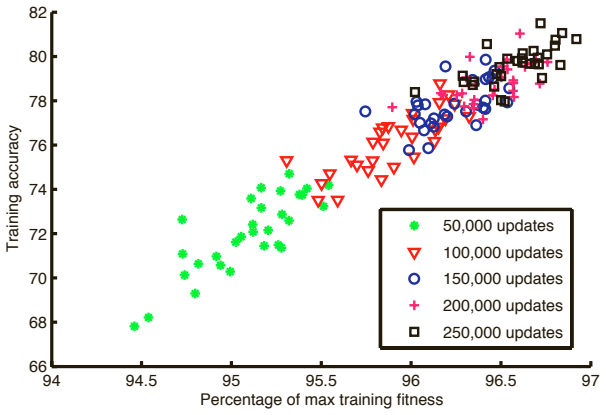
score of this individual was calculated as the fraction of test images correctly identified by the circuit. The parameters of our evolutionary algorithm are summarized in Table 2. In all, 30 replicate populations were evolved on the training set. After every 25,000 updates (up to 250,000 updates), we tested the highest-fitness individual from each replicate and recorded its accuracy.

Table 2: Parameters for the evolutionary runs.

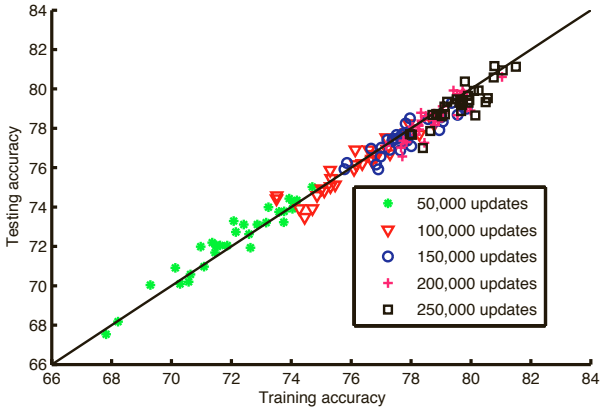
Parameter	Value
Updates	250,000
Population size	500
Starting gates	100
Max inputs	784
Max outputs	20
Inputs per gate	4
Outputs per gate	4
Gene duplication rate per update	0.05
Gene deletion rate per update	0.05
Site mutation rate per update	0.001

Results

Figure 3 shows the mean training fitness and testing accuracy of the dominant (most fit) individuals from 30 different replicates over time. Fitness and accuracy rise to approximately 96% and 79% over 250,000 updates, respectively. The single best individual accuracy (across the 30 replicate runs) is 81%. High fitness is indicative of high accuracy, although the fitnesses of the individuals are much closer to maximum than their accuracies. The two measures are also strongly correlated. Figure 4(a) demonstrates this in a scatter plot of the relationship between training fitness and training accuracy over time ($\rho = 0.9255$). The correlation is not perfect, however, because our fitness function did not reward



(a)



(b)

Figure 4: (a): Relationship between training fitness and training accuracy of the 30 replicates over time ($\rho = 0.9255$). Note that training fitness reaches much closer to its maximum than training accuracy. (b): Relationship between training accuracy and testing accuracy of the 30 replicates over time ($\rho = 0.9757$). The y=x line is shown for clarity, and shows how most of the time training accuracy is higher than testing accuracy, although there are many cases where the reverse is true.

accuracy *per se*, but rather the true positive and true negative rates of the Markov network class decisions. Similarly, as the data in Figure 4(b) shows, accuracy on the training set correlates well with accuracy on the testing set ($\rho = 0.9757$). While training accuracy is often higher than testing accuracy, it can also be lower, and the two accuracies are never different by more than a slight amount. Using the same reasoning as with Figure 4(a), one can say that the discrepancy between training and testing accuracies is due to the fact that the networks were not specifically evolved using accuracy as a fitness function. The results also demonstrate that the networks were capable of generalizing features learned from the training set to the images of the testing set.

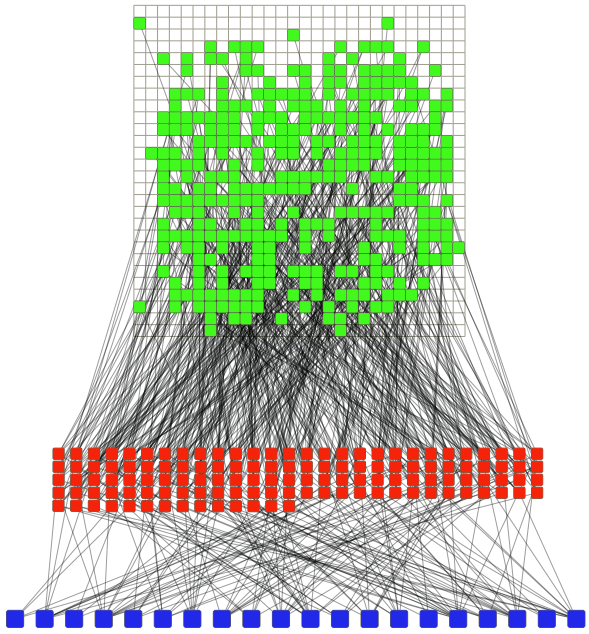


Figure 5: The structure of the Markov network from 250,000 updates with the highest individual testing accuracy. The green squares are the inputs corresponding to the pixels of the 28x28 image field. The red squares are the gates. The blue squares are the output nodes, ordered from $\{b_{0,0}, b_{1,0} ... b_{0,9}, b_{1,9}\}$.

An example evolved Markov network is depicted in Figure 5, with the inputs shown according to their location on the 28x28 image field. The network shown is the individual at 250,000 updates that has the highest testing accuracy, (81.16%). Notably, the network is able to achieve a high relative testing accuracy with a sparse number of inputs. Because of the diversity of the evolutionary process, the structures of the Markov networks that evolved were not all the same as in Figure 5. However, common features of the networks tended to appear. Figure 6(b-d) shows the probabilities of pixels from the 28x28 image field being used as inputs by the 30 dominant individual networks from the replicates at 25,000, 100,000, and 250,000 updates, respectively. The colors range from dark blue to dark red, with a dark blue pixel meaning that no networks had an input at that location, and a dark red pixel meaning that all 30 networks had an input there. It is clear from this progression that certain areas of the image field are favored by the networks more than others. This demonstrates how the Markov networks improve their fitness over time by focusing on certain areas of the image field, and clearly shows the sparsity of the inputs to the networks.

In addition, the entropy of the pixels of the 60,000 training images is shown in Figure 6(a), with dark blue representing zero bits of entropy and dark red representing the max of

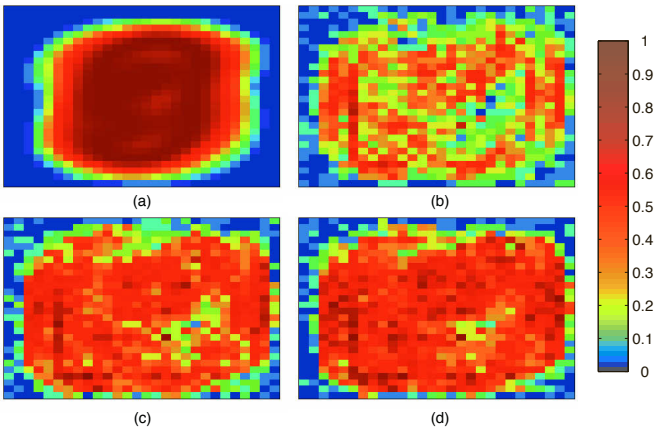


Figure 6: (a): Training image entropy per pixel Eq. (2). Dark blue represents zero bits of entropy and dark red represents the maximum entropy of 1 bit (see colorbar) (b-c): Network input probabilities at 25,000, 100,000, and 250,000 updates, respectively. Colors follow the colorbar.

1 bit. Entropy was calculated in order to compare it to the input probabilities of the networks. Entropy of each of the pixels was calculated using the probability p_i that a pixel i was a binary 1 ("turned on") over all images:

$$H(x) = -\sum_{i=1}^{784} p_i * \log_2(p_i), \qquad (2)$$

where pixel_i is the probability that a pixel in the set of training images is represented. As shown in Equation 2, entropy of a pixel is higher the closer its probability of being turned on is to 50%. If a pixel was turned on either none or all of the time, its entropy was zero. It was hypothesized that pixels with higher entropy would be more informative about the data set in general, and thus the networks would preferentially connect to these high-entropy pixels. When comparing Figure 6(a) to Figure 6(b-d), this indeed appears to be the case, especially on the border areas.

While the Markov networks focus on certain areas of the image field for input, Figure 6(b-d) also shows how the mean number of inputs of the networks increases over time. Figure 7 depicts a curve showing the mean number of inputs of the networks over time, along with the mean number of gates. Both the mean number of inputs and number of gates increases over time, although after approximately 75,000 updates the increase of both values occurs at a lower rate. Note that because gates can share inputs, there is not a simple linear relationship between the number of gates and the number of inputs. The mean number of outputs per gate also tends to increase, as depicted in Figure 8 (the number of inputs per gate is almost always 4 and is not shown here). It is not a requirement of Markov networks for there to be this concomitant increase in the number of outputs per gate alongside an

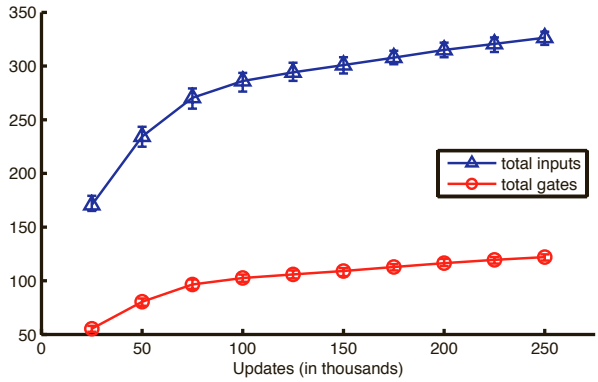


Figure 7: Mean number of inputs (neurons that connect to image pixels) averaged over 30 independent runs (blue, triangles) and mean number of logic gates (red, circles), as a function of evolutionary time. Error bars are constructed using 200-fold bootstrapping with 95% confidence intervals.

increase in the number of inputs and gates. Therefore, this phenomenon in the evolution of the networks suggests that one of the ways that the networks get better at recognizing the images is by having the gates increase the number of outputs that they cover as opposed to focusing on only one output.

Following the evolution of individual Markov networks, we next constructed committees from the dominant individuals from all 30 replicates. A committee is a method for combining the answers of multiple individuals to improve performance, a technique that has been shown to be effective on the MNIST data set (Ciresan et al., 2012). The committee decisions were formed by summing the decisions from each individual. Interestingly, committee results were strongest when individual decisions were allowed to contain votes for multiple classes, i.e., individual votes were allowed to contain ties. At the committee level, the single classification

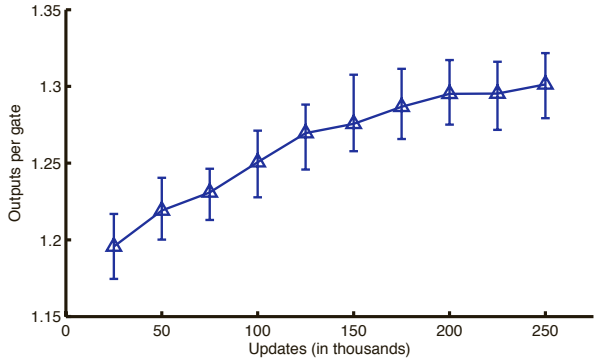
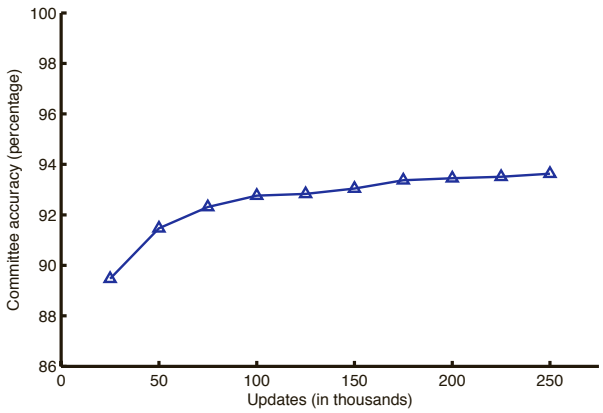
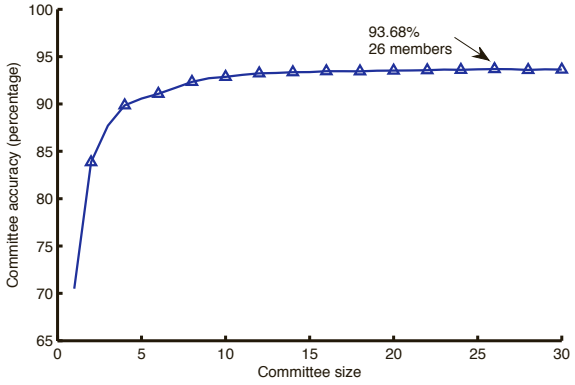


Figure 8: Mean number of outgoing edges per logic gate as a function of evolutionary time. Error bars are constructed using 200-fold bootstrapping with 95% confidence intervals.



(a)



(b)

Figure 9: (a): Accuracy of a 30-member committee over evolutionary time. (b): Committee accuracy as a function of committee size, showing diminishing returns.The arrow indicates the most accurate committee.

with the most votes was selected, and ties were again broken randomly. Figure 9(a) shows the 30-member committee accuracy over time on the testing set. The accuracy gradually increases, reaching 93.63% after 250,000 updates.

To examine the effects of committee size on accuracy on the test set, the most-fit individuals from each of the 30 replicates at 250,000 updates were ranked according to how well they did on the training set in terms of fitness, and the highest-ranked n members assigned to a committee of size n. We can then determine the dependence of committee accuracy as a function of committee size. Figure 9(b) shows that committee accuracy dramatically increases as committee members are added. We note that a committee of 30 members does not provide the absolute best accuracy for this update (93.63%), although it is very close to the maximum committee accuracy achieved (93.68% with 26 members). These results clearly show the utility of increasing the size of a committee. Although the benefit from adding new committee members faces diminishing returns, it is remarkable how only a few committee members can dramatically improve accuracy. This suggests that the diversity of the evolutionary process can produce Markov networks that are able to recognize different aspects of the OCR problem, such that pooling their answers in a committee plays on their strengths.

Conclusion

We have demonstrated an evolutionary approach to image classification that is capable of high accuracy on the MNIST benchmark, while having a small computational footprint. Because the network is essentially a digital circuit, it is easily transferred from one computational environment to another, for example, we have used it on a standard tablet computer to recognize digits drawn by the user on the screen. While the accuracy is probably sufficient for usage in mobile agent navigation, we believe that the image recognitiomn accuracy can be improved significantly. For example, we have only explored a limited set of parameters and fitness functions, and a number of unexplored avenues to improve accuracy remain. For example, there undoubtedly exists a fitness function that shows a better correlation with accuracy than the one used here. Also, pre-processing images to extract salient features (such as lines and corners) could reduce the dependence of class assignments on the locations of specific pixels. Increasing the number of images in the training set by modification of the current set could also help. Indeed, the work by Ciresan et al. (2012) utilized both of these methods.

Perhpas the most promising path to more accurate and invariant image recognition, however, is to create an evolutionary landscape and framework where a more hierarchical image processing algorithm can evolve. In this work, we have limitied the processing time (from reading the input image to writing the image classification into the outputs) to exactly one time step. As a consequence, the resulting algorithm cannot possibly use any interediate Markov variables to build up representations Marstaller et al. (2013) of the concepts. In contrast, a complex visual cortex is hierarchical and processes the simple elements of the image into more and more complex “concepts”, which are then finally used to categorize. We have seen in preliminary work that such a hierarchical organization can emerge, but at the price of a significant slowdown in evolution.

Acknowledgements

We thank Randal Olson, Fred Dyer, and Rob Pennock for discussions. This work was supported in part by the Paul G. Allen Family Foundation and the National Science Foundation BEACON Center for the Study of Evolution in Action under Cooperative Agreement DBI-0939454. We wish to acknowledge the support of the Michigan State University

High Performance Computing Center and the Institute for Cyber Enabled Research (iCER).

References

Bäck, T. (1996). *Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms*. Oxford University Press, Oxford, UK.

Beer, R. (1996). Toward the evolution of dynamical neural networks for minimally cognitive behavior. In P. Maes et al., editor, *Proc. 4th International Conference on Simulation of Adaptive Behavior*, pages 421–429, Cambridge, MA. MIT Press.

Beer, R. (2003). The dynamics of active categorical perception in an evolved model agent. *Adaptive Behavior*, 11:209–243.

Chau, S., Lafon, S., Shao, J., Szybalski, A. T., and Vincent, L. (2013). Panoramic images within driving directions. U.S. Patent 8,428,873.

Ciresan, D., Meier, U., Gambardella, L., and Schmidhuber, J. (2010). Deep, big, simple neural nets for handwritten digit recognition. *Neural Computation*, 22:3207–3220.

Ciresan, D., Meier, U., and Schmidhuber, J. (2012). Multi-column deep neural networks for image classification. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3642–3649. IEEE Press.

Decoste, D. and Schölkopf, B. (2002). Training invariant support vector machines. *Machine Learning*, 46:161–190.

Edlund, J., Chaumont, N., Hintze, A., Koch, C., Tononi, G., and Adami, C. (2011). Integrated information increases with fitness in the evolution of animats. *PLoS Computational Biology*, 7(10):e1002236.

Floreano, D. and Keller, L. (2010). Evolution of adaptive behaviour in robots by means of Darwinian selection. *PLoS Biol*, 8(1):e1000292.

Floreano, D. and Mondada, F. (1998). Evolutionary neurocontrollers for autonomous mobile robots. *Neural Networks*, 11:1461–1478.

Floreano, D., Suzuki, M., and Mattiussi, D. (2005). Active vision and receptive field development in evolutionary robots. *Evol Comput*, 13(4):527–44.

Ibañez-Guzmán, J., Laugier, C., Yoder, J.-D., and Thrun, S. (2012). Autonomous driving: Context and state-of-the-art. In Eskandarian, A., editor, *Handbook of Intelligent Systems*, chapter 50, pages 1273–1310. Speinger-Verlag London Ltd.

Koller, D. and Friedman, N. (2009). *Probabilistic graphical models: principles and techniques*. MIT Press, Cambridge. MA.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86:2278–2324.

Marstaller, L., Hintze, A., and Adami, C. (2013). The evolution of representation in simple cognitive networks. *Neural Computation*, 25:to appear.

Nelson, A. L., Barlow, G. J., and Doitsidis, L. (2009). Fitness functions in evolutionary robotics: A survey and analysis. *Robotics and Autonomous Systems*, 57:345–370.

Nolfi, S. and Floreano, D. (2000). *Evolutionary Robotics: The Biology, Intelligence and Technology of Self-Organizing Machines*. MIT Press, Cambridge, MA.

Pfeifer, R. and Bongard, J. (2006). *How The Body Shapes The Way We Think*. MIT Press, Cambridge, MA.

Riesenhuber, M. and Poggio, T. (1999). Hierarchical models of object recognition in cortex. *Nat Neurosci*, 2(11):1019–25.

Salakhutdinov, R. and Hinton, G. (2007). Learning a nonlinear embedding by preserving class neighbourhood structure. In *AI and Statistics*, volume 3, page 5.

Serre, T., Oliva, A., and Poggio, T. (2007). A feedforward architecture accounts for rapid categorization. *Proc Natl Acad Sci U S A*, 104(15):6424–9.

Stomeo, E., Kalganova, T., Lambert, C., Lipnitsakya, N., and Yatskevich, Y. (2005). On evolution of relatively large combinational logic circuits. In *Proc. NASA/DoD Conference on Evolvable Hardware*, pages 59 – 66.

Thorpe, S., Fize, D., and Marlot, C. (1996). Speed of processing in the human visual system. *Nature*, 381(6582):520–2.

Torresen, J. (2001). Two-step incremental evolution of a prosthetic hand controller based on digital logic gates. *Lecture Notes in Computer Science*, 2210:1–13.

van Dartel, M., Sprinkhuizen-Kuyper, I., Postma, E., and van den Herik, J. (2005). Reactive agents and perceptual ambiguity. *Adaptive Behavior*, 13:227–42.

Vassilev, V. and Miller, J. (2000). Scalability problems of digital circuit evolution evolvability and efficient designs. In *Proceedings of the Second NASA/DoD Workshop on Evolvable Hardware*, pages 55–64. IEEE.