An architecture for pattern recognition and decision-making

Amy de Buitléir, Ronan Flynn, Michael Russell, and Mark Daly

Faculty of Engineering and Informatics,
Athlone Institute of Technology, Athlone, Ireland
amy@nualeargais.ie,{rflynn,mrussell,mdaly}@ait.ie

Abstract. We present a simple brain architecture that allows agents to recognise patterns and make decisions based on those patterns. It takes into account not only the type of situation the agent thinks it is facing, but also how confident the agent is in its assessment, and possible alternatives. An agent using this brain was applied to two classification tasks: handwritten numeral recognition and spoken numeral recognition. In both cases, its accuracy was comparable to more traditional classifiers. This suggests that the new architecture could be useful as a general-purpose brain, for agents in a variety of domains.

Keywords: artificial life, decision-making, handwriting recognition, automatic speech recognition

1 Introduction

Decision-making is the ability of an animal to choose an action from a set of possible actions. In *goal-directed* decision-making, the animal weighs the anticipated reward (e.g., food) against the cost (e.g., energy expenditure) [1]. Thus, a good decision-making process, and the ability to learn from previous decisions, improve the animal's chance of survival. The same logic can be used by artificial (software) agents as well as animals.

Wains are an artificial life species created for data mining. For wains, data mining is a survival problem. In order to stay alive, they must discover patterns in the data, build a model of the data, classify new data based on the model, decide how to respond to data, and adapt to changes in the patternicity of the data [2].

De Buitléir et al.[2] demonstrated that a population of wains can indeed discover patterns, make survival decisions based on those patterns, and adapt to changes in the patternicity. Individual wains learned to make better decisions during their lifetimes, and evolution optimised the (genetic) operating parameters of their brains over a few generations. Several directions for future research were identified, including improving the wain's decision-making process, and implementing cultural transmission (allowing children to learn by observing the actions of their parents, and adults to learn by observing their peers).

The original brain design used a self-organising map (SOM) as a classifier, with a neural network to make decisions [2]. A SOM is a technique for representing high-dimensional data in fewer dimensions (typically two), while preserving the topology of the input data[3]. As part of the work to improve the wain's decision-making process, the SOM algorithm was modified for artificial life; we call the modified version a *self-generating model* (SGM). The SGM has been presented previously [4]. In this paper, we build on that work by redesigning the brain to use one SGM to classify inputs, and a second to predict the outcome of possible actions. The process is explained in Section 2.2.

It has already been demonstrated that wains can *learn* through trial and error; [2]; we now wish to show that they can be *taught*. The focus of this paper is on the individual wain rather than a population, but the wain will trained using the same mechanism that allows wains to learn from one another. In addition to demonstrating the decision-making ability of the wain, we hope to show that a wain can be a useful general-purpose classifier (one that might be used when specialised classifiers are not available).

To demonstrate how the new design could be used as a general-purpose brain, for agents in a variety of domains, we will demonstrate that it can classify two very different types of data: images and audio. Common classification tasks such as handwriting recognition and Automatic Speech Recognition (ASR) have benefitted from years of research, resulting in classifiers that are designed and fine-tuned for specific types of data. Since the wain is intended as a general-purpose data miner, we do not expect it to outperform a domain-specific classifier. Instead, we hope to demonstrate that wains can provide comparable accuracy.

The image data consists of handwritten numerals; the wain will attempt to identify the numeral. To evaluate the performance of the brain at handwriting recognition, we compare it with a traditional classifier. Other classification techniques can achieve better accuracy at handwriting recognition than the SOM, for example, support vector machines [5] and traditional neural networks [6]. However, the wain's new brain design is partly based on modified SOMs (see Section 2.2). For this reason, we chose the SOM as the benchmark.

The audio data consists of spoken numerals; again the wain will attempt to identify the numeral. One widely used ASR technique is hidden Markov models (HMM) [7]. The hidden Markov model toolkit (HTK) provides the ability to construct and manipulate HMMs [8]. HTK is widely used for speech recognition research, making an HMM-based classifier implemented using HTK a suitable benchmark.

As will be explained in Section 2.2, a wain maintains a set of internal models for the range of objects that it has encountered. These internal models need not (and usually do not) map directly to human categories. Based on the resemblance between a stimulus and its internal models, the wain chooses, from a predefined set, the response that it predicts will lead to the greatest happiness. Then how can we get a wain to perform classification? By making the set of available responses be classifications! By using the wain as a classifier, we are also demonstrating its ability to make good decisions.

2 Implementation

This project uses a computational ecosystem called Créatúr[1]. Créatúr is both a software framework for automating experiments with artificial life, and a library of modules that can be used (with or without the framework) to implement agents. The system architecture is illustrated in Figure 1. The package `creatur` provides the ecosystem. The package `creatur-wains` provides a general-purpose implementation of a wain, `creatur-image-wains` contains tools for working with images, and `creatur-audio-wains` contains tools for working with audio feature files.

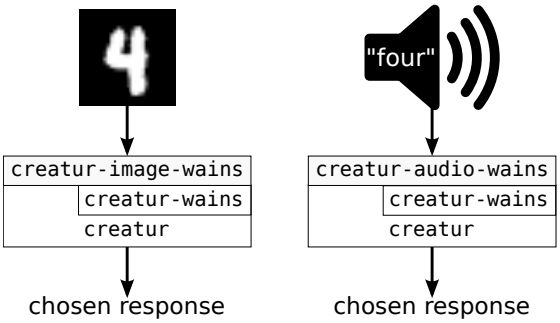

Fig. 1. System architecture for working with MNIST images (left) and TIDIGITS feature files (right). The rectangles represent software packages. A horizontal boundary between two packages indicates that the upper package calls functions provided by the lower package. For example, `creatur-wains` calls functions provided by `creatur`, and `creatur-image-wains` calls functions provided by `creatur-wains`, but also makes calls directly to functions provided by `creatur`.

The wain implementation has been described in detail elsewhere [2]; a summary is provided below, which focuses on the features used in the experiments presented in this paper and highlights changes that have been made to the implementation.

2.1 Condition

Wains have an *energy level* from 0 to 1. They gain or lose energy as a result of the reward system, which is unique to the type of experiment. For example, a wain might be rewarded for accurately identifying a pattern. If a wain's energy falls below 0, it dies. Wains also have a *boredom level* and a *passion level*, each from 0 to 1. Depending on the reward system, boredom might be decreased as a result of novelty-seeking behaviour. The wain's passion level is set to 0 at birth

[1] *Créatúr* (pronounced kray-toor) is an Irish word meaning animal, creature, or unfortunate person.

or as a result of mating, and increases at a genetically determined rate until the next mating. (A "genetically determined" value is one that is specified by an agent's genes, can be different for each agent, is inherited by offspring, and is subject to evolutionary pressures.) Collectively, the wain's energy level, passion level, boredom level, and whether or not it is currently rearing a child, are called its *condition*.

Wains seek to maximise their *happiness*, which is given by

$$happiness = w_e e + w_p(1-p) + w_b(1-b) + w_l l, \tag{1}$$

where e is the wain's energy level; p its passion level; b its boredom level; l is 1 if the wain is currently rearing a child, 0 otherwise; and w_e, w_p, w_b, w_l are genetically-determined weights. The weights are normalised so that the happiness is from 0 to 1.

2.2 The brain

The brain has three components: a *classifier*, a *muser*, and a *predictor*. This structure is fixed; however, evolution can fine-tune operating parameters such as the learning rate. The classifier maintains a model of the space of patterns encountered, the muser generates possible responses to situations, and the predictor maintains a model of the space of responses selected.

Both the classifier and predictor use a modified SOM called a *self-generating model* (SGM) [4]. In a SOM, the models are arranged on a two-dimensional grid; in an SGM the models form an unconnected set. Unlike a SOM, the SGM does not preserve the topology of the input space. Another difference is that the SOM has a fixed number of models, but the number of models in the SGM is variable. When the difference (according to a chosen metric) between the input pattern and the closest matching model exceeds a predefined threshold, and the SGM is not at capacity, the SGM creates a new model based on the input pattern. Therefore, while the SOM must be initialised with a set of models (possibly random data), the SGM can begin empty, adding new models as needed to reflect the diversity of the input data.

The process by which the brain makes decisions is illustrated in Figure 2. When one or more patterns are presented to the agent, the classifier produces a *signature*, a vector whose elements indicate how similar each input pattern is to each classifier model, and reports this to the brain.

For each object and model, the brain estimates the probability that the object actually belongs to the category represented by the model. This is a simple calculation,

$$\mathbf{p_i} = \frac{1 - \mathbf{d_i}}{|1 - \mathbf{d_i}|},$$

where $\mathbf{p_i}$ is a vector where each element p_{ij} is the estimated probability that object i belongs to the category represented by model j, and $\mathbf{d}$ is a vector where each element d_{ij} is the difference between object i's pattern and model j.

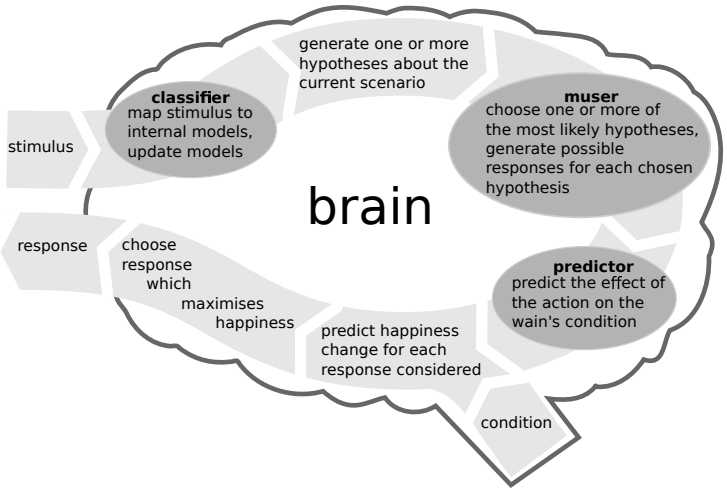

Fig. 2. The decision-making process

The brain generates *hypotheses* by considering each possible combination of object and model. The estimated probability for each hypothesis is the product of the individual object-model probabilities. Next, the muser chooses one or more of the most likely hypotheses (the number of hypotheses chosen is genetically determined), and generates a set of responses to evaluate.

The predictor then estimates how each proposed response will affect each aspect of the agent's condition (energy, passion, boredom, litter size). It does this by selecting the response model that best matches the proposed response, and returning the condition changes predicted by that model, adjusted according to the probability that the hypothesis is true. If no response model is sufficiently similar, a new model may be created.

The brain combines the agent's current condition with the predicted changes, and calculates the resulting happiness change, according to Equation 1. The brain chooses the action that is predicted to have the most favourable (most positive or least negative) effect on happiness. After the agent has received any rewards or penalties as a result of that action, the predictor adjusts its models according to the actual change in happiness.

By considering more than one hypothesis, the agent can employ more subtle reasoning. It can base its actions not only on what scenario it thinks it is facing, but also on how confident it is, and what is likely to happen if the agent is wrong. For example, suppose the agent considers two hypotheses, where the estimated payoff (happiness increase) is given by Table 1. If the agent is reasonably confident that the more likely hypothesis is actually true, the best response is action #1. Otherwise, it may be worth the gamble to go for action #2, in hope of the large payoff.

Table 1. Sample payoff matrix

	payoff if more likely hypothesis is true	payoff if less likely hypothesis is true
action #1	medium	small
action #2	small	large

The brain can also learn as a result of *imprinting*, which is a shortcut where the agent is shown one or more patterns and an action, and concludes that taking the action in a similar situation would optimise its condition, maximising its happiness. This can be used to allow children to learn by observing their parents, or for adults to learn by observing other adults. Although this feature was originally intended to allow wains to learn from each other, it can also be used by the operator to train wains.

3 Experimental Setup

The methods used for the experiments presented in this paper are described below. These experiments use an individual wain rather than a population, but the wain is trained using the same mechanism that allows wains to learn from one another.

3.1 Images

The MNIST database is a collection of images of hand-written numerals that is a useful benchmark for comparing classification methods [9]. The training set contains 60,000 images, while the test set contains 10,000 images. All images are 28x28 pixels, and are grey-scale. The numerals are centred within the image. The centre of pixel mass of the numeral has been placed in the centre of the image. A sample image is shown in Figure 3.

Fig. 3. Sample MNIST image of a handwritten "2" [9].

Images from the MNIST database were used without modification. We presented the images to the agent as a sequence of integers. Each element of the sequence was a number from 0 to 255, indicating the intensity of the pixel. The

agent was not given any information about the geometry of the image. For example, it did not know that in a 28x28 image, the 29th pixel is immediately below the first pixel.

The brain was configured to use the mean of absolute differences (MAD) as a measure of difference between an input image and the classifier models. This is calculated by taking the absolute difference between each pair of corresponding pixels, and taking the mean to obtain a number from 0 (identical) to 1 (maximally dissimilar). All the images in the MNIST database have the same size, viewing direction (normal to the plane of the image, from above), and comparable intensity, so the MAD is an appropriate difference metric.

The learning function of the SOM is given by Equation 2,

$$f(d,t) = re^{-\frac{d^2}{2w^2}}, \tag{2}$$

where

$$r \equiv r_0 \left(\frac{r_f}{r_0}\right)^a, \quad w \equiv w_0 \left(\frac{w_f}{w_0}\right)^a, \quad \text{and} \quad a \equiv \frac{t}{t_f}.$$

The function input d is the distance between the node being updated and the winning node; t is the "time", a counter of the number of patterns learned so far. The parameter r_0 is the initial learning rate, r_f is the learning rate at time t_f, w_0 is the radius of the initial neighbourhood, w_f is the radius of the neighbourhood at time t_f, and a indicates the brain's "age".

For the winning node, $d = 0$, and Equation 2 reduces to Equation 3, which is the learning function for the SGM.

$$f(t) = r = r_0 \left(\frac{r_f}{r_0}\right)^a. \tag{3}$$

Note that at all times the learning rate of the SGM matches the learning rate of the winning node in the SOM. This permits a fairer comparison of the SOM and the SGM.

Table 2 shows the configuration of the two classifiers. The values r_0 and r_f were chosen so that the learning rate would start at maximum and be near zero by the end of training. The values w_0 and w_f were determined empirically. The value of t_f is the number of training images. To determine the difference threshold, we tried a range of values near the mean difference between images of the same numeral, and chose the one that resulted in the best accuracy.

3.2 Audio samples

The TI46 speech database is a corpus of 46 isolated spoken words recorded for both male and female speakers. The corpus is intended for the evaluation of ASR products [10]. The words in the corpus include the numerals "zero" through "nine". In the experiments presented in this paper, only numerals are used. The training set contains 1,594 samples of spoken numerals; the test set contains 2,541 samples.

Table 2. Configuration for working with MNIST images

variable	SOM	brain
final node count	1024	956
grid type	rectangular	unconnected nodes
classifier r_0	1	1
classifier r_f	1×10^{-15}	1×10^{-15}
classifier w_0	3	not applicable
classifier w_f	1×10^{-7}	not applicable
classifier t_f	60000	60000
classifier threshold	not applicable	0.12
predictor r_0	not applicable	1×10^{-9}
predictor r_f	not applicable	1×10^{-10}
predictor t_f	60000	60000
predictor threshold	not applicable	0.1

We extracted the MFCC feature vectors from the samples in the TI46 corpus using the HCopy tool provided as part of HTK [8]. Each frame had 13 static coefficients (cepstral coefficients C1-C12 and energy). The corresponding velocity and acceleration coefficients were also calculated to give 39 coefficients per frame. First order pre-emphasis was applied using a coefficient of 0.97. There were 23 filterbank channels and 22 cepstral liftering coefficients. The frame rate used was 10 ms with a 25ms Hamming window. The feature vectors for each audio sample were concatenated, in time order, and presented to the brain as a sequence of double-precision floats.

The HMM-based classifier is implemented using the HTK Speech Recognition Toolkit[8]. There are ten whole word HMMs, one for each numeral, each of which has three states, with each state having three Gaussian mixtures. For working with non-endpointed samples, two additional models are defined to represent pauses in speech, sil and sp. The sil model has three states and each state has six mixtures. The sp model has a single state.

End-pointing is the process of removing silence from the beginning and end of an audio sample, in order to simplify the classification task. The short-term energy for each frame is calculated as the sum of the absolute values of the sample amplitudes in the frame. End-pointing is performed by determining whether or not the short-term energy of successive frames is above a defined threshold (to determine the start of the utterance) or below a defined threshold (to determine the end of the utterance). For example, to get the start point, look for three consecutive frames with energy exceeding the threshold; the first frame of the three is assumed to be the start of the utterance.

The brain was configured to use the square of the Euclidean distance as a measure of difference between an input sample and the classifier models. The length of samples differs, so the resulting number of vectors in each sample differs as well. However, brains require that all input patterns have the same length.

Therefore, the agent was configured to "stretch" or "compress" the samples as needed so they all have the same number of vectors. Stretching is achieved by duplicating vectors; the duplications were distributed as evenly throughout the pattern as possible.

The algorithm for compressing samples is straightforward. First, calculate the differences between each consecutive pair of vectors. Second, find the vector with the smallest change from the previous one, and drop it. These two steps are repeated until the sample is of the desired length.

Table 3 shows the configuration of the brain. The values r_0 and r_f were chosen so that the learning rate would start at maximum and be near zero by the end of training. The values w_0 and w_f, and the number of vectors, were determined empirically. The value of t_f is the number of training images. To determine the difference threshold, we tried a range of values near the mean difference between samples of the same numeral, and chose the one that resulted in the best HMM accuracy.

Table 3. Configuration of brain for working with audio samples

variable	as-is samples	end-pointed samples
classifier r_0	0.1	0.1
classifier r_f	0.001	0.001
classifier t_f	1594	1594
difference threshold	0.00018	0.00018
predictor r_0	0.1	0.1
predictor r_f	0.001	0.001
predictor t_f	1594	1594
num. vectors	159	154

3.3 Training and testing

The general procedure for working with either images or audio samples is the same. In both cases, the training data set and the test data set are distinct; we used the standard training and test sets for both the MNIST and TI46 data. First, we presented the training patterns in random order to the agent, along with the correct identification. This was done using imprinting, as described at the end of Section 2.2.

Next, we presented the test patterns to the agent, again in random order. As each pattern was presented, the agent responded with an identification. For a fair comparison with the SOM or HMM, we needed to prevent learning during the testing phase. To achieve this, each time the wain responded, we restored it to the state it had at the end of the training (imprinting) phase. Although

the wain's condition never actually changes, it continues to expect an increase in happiness, and to take that into account when making decisions.

4 Results and interpretation

Table 4 compares the image classification performance of the brain with that of the SOM. The accuracy of both methods is comparable. Training and testing the brain required less than half the time of the SOM. The reduction in processing time occurs primarily because the SGM only updates one model during training, while the SOM updates the models in the neighbourhood of the winning node.

Table 4. Comparison of image classification results

classifier	SOM	brain
no. models	1024	941
numeral	accuracy	
0	0.952	0.9408
1	0.970	0.9736
2	0.837	0.9109
3	0.835	0.8634
4	0.725	0.6609
5	0.739	0.8341
6	0.967	0.9415
7	0.873	0.7772
8	0.753	0.7956
9	0.834	0.7929
all	0.853	0.8508
time	6273s	2514s

Table 5 compares the audio classification performance of the brain with that of the HMM. The accuracy of both methods is comparable, however, the brain is significantly slower. The brain was slightly more accurate when working with the as-is data than with the end-pointed data. The compression algorithm has the side-effect of removing some of the silence from the beginning and end of the sample, thus an extra end-pointing step is not required.

The code and results for the experiments presented in this paper are open access [11–13]. A tutorial for Créatúr is available [14].

5 Conclusion

The wain was applied to two classification tasks: handwritten numeral recognition and spoken numeral recognition. In both cases, its accuracy was compara-

Table 5. Comparison of audio classification results

data type	as-is		end-pointed	
classifier	HMM	brain	HMM	brain
word	accuracy			
"zero"	1.0000	1.0000	1.0000	0.9840
"one"	1.0000	0.9882	1.0000	0.9922
"two"	1.0000	1.0000	1.0000	1.0000
"three"	1.0000	0.9881	1.0000	0.9961
"four"	1.0000	1.0000	1.0000	0.9961
"five"	1.0000	1.0000	1.0000	0.9961
"six"	0.9961	0.9961	1.0000	1.0000
"seven"	1.0000	0.9922	1.0000	0.9961
"eight"	1.0000	1.0000	1.0000	0.9883
"nine"	0.9881	0.9763	1.0000	0.9802
all	**0.9984**	**0.9941**	**1.0000**	**0.9929**
time	<1m	14m	<1m	12m

ble to more traditional classifiers. This suggests that wains could be useful as a general-purpose classifier, applied to a variety of domains.

Why should anyone be interested in a new classifier that is no more accurate than traditional classifiers, and for audio, is significantly slower? One advantage is that the new brain design is not just a classifier; it also *makes decisions* by choosing the action that leads to the best predicted outcome. In the experiments described in this paper, the only available actions were to choose a classification; however, other types of actions could also be performed. Another advantage to the new design is its generality; it could be used in domains where custom classifiers have not yet been developed.

As this is a new approach to pattern recognition and decision-making, there is scope for improvement. Accuracy could be improved by choosing more sophisticated distance metrics. For images, the MAD could be replaced with a metric that takes into account a pixel's neighbours. This might allow it to cope better with writing that is heavily slanted, or is thinner or thicker than typical writing. For audio samples, a variable frame rate analysis such as that suggested by Le Cerf and Van Compernolle[15] could be used. The run-time of the software is dominated by the comparisons between models, so performance could also be improved by choosing a different distance metric.

Although a single wain was used in these experiments, wains were designed to be used in a population. The configuration parameters are genetic, so it is possible to have a population of wains with varying configurations. Awarding energy for accurate classifications would encourage evolution to find a range of suitable configurations. Wains have the ability to teach their young, as well as other adults, so each generation can augment the species' knowledge. A popula-

tion of wains with slightly different configurations, and different life experiences, could give independent opinions on a classification.

References

1. Rangel, A., Hare, T.: Neural computations associated with goal-directed choice. Current opinion in neurobiology 20(2), 262–270 (2010)
2. de Buitléir, A., Russell, M., Daly, M.: Wains: A pattern-seeking artificial life species. Artificial Life 18(4), 399–423 (2012)
3. Kohonen, T.: Self-organizing maps. Springer series in information sciences, 30, Springer, Berlin, 3rd edn. (December 2001), http://www.worldcat.org/isbn/3540679219
4. de Buitléir, A., Daly, M., Russell, M.: The self-generating model: an adaptation of the self-organizing map for intelligent agents and data mining. Accepted for Proceedings of the Artificial Life and Intelligent Agents Symposium, Birmingham, 2016 (2016)
5. Decoste, D., Schölkopf, B.: Training invariant support vector machines. Machine learning 46(1-3), 161–190 (2002)
6. Cireşan, D.C., Meier, U., Gambardella, L.M., Schmidhuber, J.: Deep, big, simple neural nets for handwritten digit recognition. Neural Computation 22(12), 3207–3220 (Dec 2010), http://dx.doi.org/10.1162/NECO{_}a{_}00052
7. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77(2), 257–286 (1989)
8. Young, S., Evermann, G., Gales, M., Hain, T., Kershaw, D., Liu, X.A., Moore, G., Odell, J., Ollason, D., Povey, D., Valtchev, V., Woodland, P.: The HTK Book. Cambridge University Engineering Department, htk version 3.4 edn. (2006)
9. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
10. Liberman, M., Amsler, R., Church, K., Fox, E., Hafner, C., Klavans, J., Marcus, M., Mercer, B., Pedersen, J., Roossin, P., Walker, D., Warwick, S., Zampolli, A.: Ti 46-word ldc93s9 (1991), https://catalog.ldc.upenn.edu/docs/LDC93S9/ti46.readme.html
11. de Buitléir, A.: Software release: exp-image-id-wains v2.18 (Mar 2016), http://dx.doi.org/10.5281/zenodo.46981
12. de Buitléir, A.: Software release: exp-audio-id-wains v2.17 (Mar 2016), http://dx.doi.org/10.5281/zenodo.46980
13. de Buitléir, A.: Software release: som v9.0 (Jan 2016), http://dx.doi.org/10.5281/zenodo.45039, http://dx.doi.org/10.5281/zenodo.45039
14. de Buitléir, A.: Créatúr Tutorial (2014), https://github.com/mhwombat/creatur-examples/raw/master/Tutorial.pdf
15. Le Cerf, P., Van Compernolle, D.: A new variable frame analysis method for speech recognition. Signal Processing Letters, IEEE 1(12), 185–187 (1994)