Proceedings of the 2007 IEEE Symposium on Approximate
Dynamic Programming and Reinforcement Learning (ADPRL 2007)

Using ADP to Understand and Replicate Brain Intelligence: the Next Level Design

Paul J. Werbos[1]
Room 675, National Science Foundation
Arlington, VA 22203, US
pwerbos@nsf.gov

Abstract-Since the 1960's I proposed that we could understand and replicate the highest level of intelligence seen in the brain, by building ever more capable and general systems for adaptive dynamic programming (ADP) – like "reinforcement learning" but based on approximating the Bellman equation and allowing the controller to know its utility function. Growing empirical evidence on the brain supports this approach. Adaptive critic systems now meet tough engineering challenges and provide a kind of first-generation model of the brain. Lewis, Prokhorov and myself have early second-generation work. Mammal brains possess three core capabilities – creativity/imagination and ways to manage spatial and temporal complexity - even beyond the second generation. This paper reviews previous progress, and describes new tools and approaches to overcome the spatial complexity gap.

I. INTRODUCTION

No one on earth today can write down a complete set of equations, or software system, capable of learning to perform the complex range of tasks that the mammal brain can learn to perform. From an engineering viewpoint, this paper will provide an updated roadmap for how to reach that point. But actually, this paper is a revision or update of my earlier first and generation theories of how the brain actually works, as an engineering device. The implications for neuroscience, psychology and power grids are discussed further in an extended version of the paper, posted at http://arxiv.org , which is searchable by author.

Here I will not address the human mind as such. In nature, we see a series of *levels* of intelligence or consciousness [1]. Within the vertebrates, M. E. Bitterman [2] has shown that there are major qualitative jumps from the fish to the amphibian, from the amphibian to the reptile, and from the reptile to even the simplest mammal. 99% of the higher parts of the human brain consist of structures, like the 6-layer neocortex, which also exist in the mouse, and show similar general-purpose-learning abilities there. If we fully understand how learning and intelligence work in the mouse brain, this will help us understand the human mind, but the human mind is much more than that.

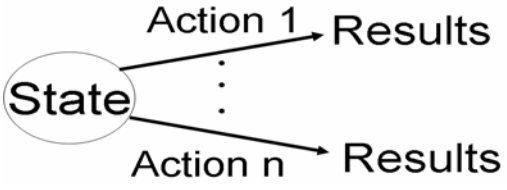
Section II discusses optimization and ADP in general. It gives a few highlights from the long literature on why these offer a central organizing principle both for understanding the brain and for improving what we can do in engineering. Section III reviews 1st and 2nd generation ADP designs, and their relevance to brain-style intelligence. Section IV will discuss how to move from second-generation designs to the level of intelligence we see in the brain of the smallest mouse – with a special emphasis on how to handle spatial complexity, by learning symmetry groups, and to incorporate that into an ADP design or larger brain.

II. WHY OPTIMALITY AND WHY ADP?

A. Optimality As An Organizing Principle for Understanding Brain Intelligence

For centuries, people have debated whether the idea of optimization can help us understand the human mind. Long ago, Aristotle proposed that all human efforts and thought are ultimately based on the pursuit of (maximizing) happiness – a kind of inborn "telos" or ultimate value. Utilitarians like John Stuart Mill and Jeremy Bentham carried this further. A more updated version of this debate can be found in [3]; here I will only review a few basic concepts.

To begin with [4], animal behavior is ultimately about choices as depicted here:.



[1]The views herein represent no one's official views, but the paper was written on US government time.

Functionality of the brain is about making choices which yield better results. *Intelligence* is about *learning* how to make better choices. The simplest types of animals may be born with fixed rules about what actions to take, as a function of the state of their environment as they see it. More advanced animals, instead, have an ability to select actions based on the *results* that the actions might have.

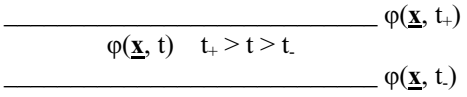
To put all this into mathematics, we must have a way to evaluate which results are "better" than which other results. Von Neumann's concept of Cardinal Utility function [5] provides that measure; it is the foundation of decision theory [6], risk analysis, modern investment analysis, and dynamic programming, among others. Usually, when we talk about discrete "goals" or "intentions," we are not talking about the long-term values of the organism. Rather, we are talking about subgoals or tactical values, which are intended to yield better results or outcomes. The utility function which defines what is "better" is the foundation of the system as a whole.

Next consider the analogy to physics.

In 1971-1972, when I proposed a first generation model of intelligence, based on ADP, to a famous neuroscientist, he objected: "The problem with this kind of model is that it creates an anthropomorphic view of how the brain works. I have spent my entire life teaching people how to overcome a bachtriomorphic view of the frog. Thinking about people by using empathy could be a disaster for science. Besides, even in physics, we know that the universe is maximizing a kind of utility function, and we don't think of the universe in anthropomorphic terms."

From a strictly objective viewpoint, his argument actually supports the idea of trying to use optimization as a central organizing principle in neuroscience. After all, if it works in physics, in a highly rigorous and concrete way, why not here? If we can unify our functional understanding of the brain not only with engineering, but with subjective experience and empathy, isn't this a source of strength, so long as we keep track of which is which?

But does it really work that way in physics? Partly so. According to classical physics, the universe really does solve the optimization problem depicted here:



The universe has a kind of "utility function," $\mathscr{L}(\mathbf{x}, t)$. It "chooses" the states φ of all particles and fields at all times t by choosing states which maximize the total sum of $\mathscr{L}$ across all of space time, between time t_- and time t_+, subject to the requirement that they provide a continuous path from the fixed state at some initial time t_- and some final time t_+. This elegant formalism, due to Lagrange, provides a very simple parsimonious description of the laws of physics; instead of specifying n dynamic laws for n types of particle or field, we can specify the "Lagrangian function $\mathscr{L}$" and derive all the predictions of physics from there. In order to perform that calculation, we can use an equation from classical physics, the Hamilton-Jacobi equation, which tells us how to solve deterministic optimization problems across time or space-time.

But that is not the whole story. Hamilton and Lagrange had many debates about whether the universe really maximizes $\mathscr{L}$– or does it minimize it or find a minmax solution? Does the physical universe find something that looks like the outcome of a two-person zerosum game? By the time of Einstein, it appeared so. Modern quantum theory gets rid of the deterministic assumption, but adds random disturbance in a very odd way. It actually turns out that we can recover something like Lagrange's original idea, which fits the tested predictions of modern quantum theory, by introducing a stochastic term whose statistics are symmetric both in space and in time; however, the details are beyond the scope of this paper. (See www.werbos.com/reality.htm.)

To describe the brain, it is not enough to use the old optimization rule of Hamilton and Jacobi. We need to consider the stochastic case, because animals, like us, cannot predict our environment in a deterministic way. The foundation for optimization over time in the stochastic case is the Bellman equation, a great breakthrough developed by Bellman in 1953, made possible by Von Neumann's concept of Cardinal Utility function.

The principles of optimality are important to fundamental physics – but also to thermodynamics, and to the physics of emergent phenomena in general. Those details are beyond the scope of this paper.

Finally, let me address two of the most common questions which people tend to ask when I talk about the brain as an "optimization machine."

First: If brains are so optimal, why do humans do so many stupid things? Answers: Brains are designed to *learn* approximate optimal policy, as effectively as possible with *bounded computational resources* (networks of neurons), starting from a less optimal start. They never learn to play a perfect game of chess (nor will our computers, nor will any other algorithm that can be implemented on a realistic computer) because of constraints on computational resources. *We just do the best we can*.

Also, when one human (a researcher) criticizes another, we are seeing a comparison between *two* highly intelligent systems. Some brains learn faster than others. In my view, humans themselves are an intermediate state towards an even higher/faster intelligence – beyond the scope of this paper.

Second question: if this optimization theory of the brain is correct, wouldn't brains get stuck in local minima, just like artificial optimization programs when confronted with a complex, nonlinear environment? Answers: they do indeed. Every person on earth is caught in a "local minimum," or rut, to some degree. In other words, we could all do a bit better if we had more creativity. But look at those hairy guys (chimpanzees) in the jungle, and the rut they are in!

The optimization theory of the brain implies that our brains *combine* an incremental learning ability with an ability to learn to be more creative – to do better and better "stochastic search" of the options available to us. There are a few researchers in evolutionary computing or stochastic search who tell us that their algorithms are guaranteed to find the global optimum, eventually; however, those kinds of guarantees are not very realistic because, for a system of realistic complexity, they require astronomical time to actually get to the optimum.

B. Optimality and ADP In Technology

The benefits of adaptive dynamic programming (ADP) to technology have been discussed by many other authors in the past[7] and at this conference, with specific examples. Again, I will review only a few highlights.

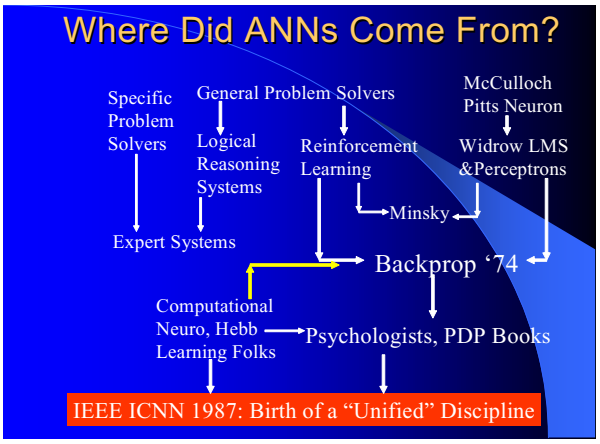
Many control engineers ask: "Why try to find the optimal controller out of all possible controllers? It is hard enough just to keep things from blowing up – to stabilize them at a fixed point." In fact – the most truly stable controllers now known are nonlinear feedback controllers, based on "solving" the "Hamilton-Jacobi-Bellman" equation. But in order to implement that kind of control, we need mechanisms to "numerically solve" (approximate) the Bellman equation as accurately as possible. ADP is the machinery to do that.

Furthermore – there are times when it is impossible to give a truly honest absolute guarantee of stability, under accurate assumptions. Certainly, a mouse running through the field has no way to guarantee its survival – nor does the human species as a whole, in the face of the challenges now confronting us. (See www.werbos.com.) In that kind of real-world situation, the challenge is to *maximize the probability* of survival; that, in turn, is a stochastic optimization problem, suitable for ADP, and not for deterministic methods. (Recent work by Gosavi has explored that family of ADP applications.) Verification and validation for real complex systems in the real world is heavily based on empirical tests and statistics already.

Finally, in order to address nonlinear optimization problems in the general case, we absolutely must use universal nonlinear function approximators. Those could be Taylor series – but Barron showed years ago that the simplest form of neural networks offer more accurate nonlinear approximation that Taylor series or other linear basis function approximators, in the general case, when there is a large number of state variables. Use of more powerful and accurate approximators (compatible with distributed hardware, like emerging multicore chips) is essential to more accurate approximations and better results.

III. FIRST AND SECOND GENERATION ADP MODELS OF BRAIN INTELLIGENCE

A. Origins and Basics of the First Generation Model



Backpropagation and the first true ADP design both originated in my work in the 1970's. The flow chart above gives a simplified view of the flow of ideas.

In essence, the founders of artificial intelligence (AI) – Newell, Shaw and Simon, and Minsky [8] – proposed that we could build brain-like intelligent systems by building powerful reinforcement learning systems. However, after a great deal of experimentation and intuition and heuristic thinking, they could not design systems which could optimize more than a few variables. Knowing that the brain can handled many thousands of variables, they simply gave up – just as they gave up on training simplified neural models (multilayer perceptrons). Amari, at about the same time, wrote that perhaps derivatives might be used somehow to train multilayer perceptrons – but suggested that it would be unlikely to work, and did not provide any algorithm for actually calculating the required derivatives in a distributed, local manner.

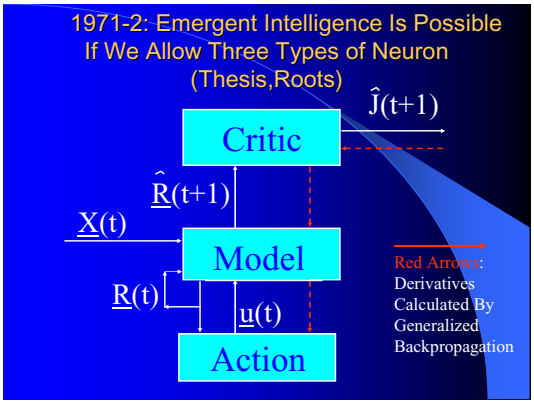
. In 1964, I – like many others – was deeply inspired by Hebb's classic book on intelligence [9]. Inspired by the empirical work on mass action and learning in the brain (by Lashley, Freeman, Pribram and others), he proposed that we would not really need a highly complex model in order to explain or reproduce brain-like intelligence. Perhaps we could generate intelligence as the emergent result of learning; we could simply construct billions of model neurons, each following a kind of universal neuron learning rule, and then intelligence could emerge strictly as a result of learning. I tried very hard to make that work in the 1960's, and failed. The key problem was that Hebb's approach to a universal learning rule is essentially calculating correlation coefficients; those are good enough to construct useful associative memories, as Grossberg showed, but not to make good statistical predictions or optimal control. They are simply not enough by themselves to allow construction of an effective general-purpose reinforcement learning machine.

By 1971-1972, I realized that Hebb's vision could be achieved, if we relax it only very slightly. It is possible to design a general purpose reinforcement learning machine, if we allow just three types of neuron and three general neuron learning rules, instead of just one.

Actually, the key insight here came in 1967. In 1967 (in a paper published in 1968 [4]), I proposed that we could overcome the problems with reinforcement learning by going back to basic mathematical principles – by building systems which learn to approximate the Bellman equation. Use of the Bellman equation is still the only exact and efficient method to compute an optimal strategy or policy of action, for a general nonlinear decision problem over time, subject to noise. The equation is:

$$J(\mathbf{x}(t)) = \max_{\mathbf{u}(t)} \langle U(\mathbf{x}(t), \mathbf{u}(t)) + J(\mathbf{x}(t+1)) \rangle / (1+r),$$

where $\mathbf{x}(t)$ is the state of the environment at time t, $\mathbf{u}(t)$ is the choice of actions, U is the cardinal utility function,. R is the interest or discount rate (exactly as defined by economics and by Von Neumann), where the angle brackets denote expectation value, and where J is the function we must solve for in order to derive the optimal strategy of action. In any state $\mathbf{x}$, the optimal $\mathbf{u}$ is the one which solves the optimization problem in this equation. A learning system can learn to approximate this policy by using a neural network (or other universal approximator) to approximate the J function and other key parts of the Bellman equation, as shown in the next figure, from my 1971-1972 thesis proposal to Harvard:



In that design, I needed a *generalized* form of backpropagation as a tool to calculate the essential derivatives or sensitivity coefficients needed to allow correct incremental learning of all three parts. I formulated and proved a new chain rule for "ordered derivatives" which makes it possible to compute the required derivatives exactly through any kind of large nonlinear system, not just neural networks.

For my PhD thesis (reprinted in entirety in [10]), I included the proof, and many applications of backpropagation to systems other than neural networks. In [11,12], I described how *generalized backpropagation* can be used in a wide variety of applications, including ADP with components that could be neural networks or *any other* nonlinear differentiable system.

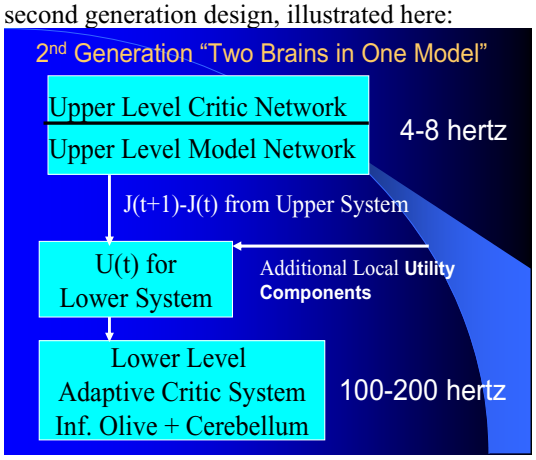
The method which I proposed to adapt the Critic network in 1971-1972 I called "Heuristic

Dynamic Programming" (HDP). It is essentially the same as what was later called "the Temporal Difference Method." But I learned very early that the method does not scale very well, when applied to systems of even moderate complexity. It learns too slowly. To solve this problem, I developed the core ideas of two new methods – dual heuristic programming (DHP) and Globalized DHP (GDHP) – published in a series of papers from 1977 to 1981 [13-15]. To prove convergence, in [12], I made small but important changes in DHP; thus [12] is the definitive source for DHP proper. For more robust extensions, see the final sections of [16].

See [7], [12] and many papers in this conference for reviews of practical applications of HDP, DHP, GDHP and related adaptive critic systems.

B. A Second Generation Model/Design for Brain-Style Intelligence

By 1987, I realized that the brain has certain capabilities beyond what any of these first-generation design offer. Thus I proposed [17] a second generation design, illustrated here:



The key point is that truly powerful foresight, in an ADP system, requires the use of Critic Networks and Model Networks which are far more powerful than feedforward neural networks (or Hebbian networks or Taylor series or linear basis function networks). It requires the use of recurrent networks, including networks which "settle down" over many cycles of an inner loop calculation before emitting a calculation. That, in turn, requires a relatively low sampling rate for calculation; about 4-8 calculations per second is the rate observed for the higher centers of the mammal brain. However, smooth muscle control requires a much higher bandwidth of control; to achieve that, I proposed that the brain is actually a kind of master-slave system. I published some papers joint with Pellionisz on the details, and how to look for them in neuroscience data. (Some are posted on my web page.) In chapter 13 of [12], I provided equations for an "Error Critic for motor control" which provide one possible design for a fast model-free "slave" neural network, matching this model.

Danil Prokhorov, in various IJCNN papers, showed how that kind of fast design (and some variations he developed) works well, by certain measures, in computational tests. Recent formal work in Frank Lewis's group at the University of Texas (ARRI) has shown strong stability results for continuous-time model free ADP designs which *require* an external value input, exactly like what this master-slave arrangement would provide.

Intuitively… the "master" is like the coach within you, and the "slave" is like the inner football player. The football player has very fast reflexes, and is essential to the game, but he needs to strive to go where the more far-seeing coach sends him. The coach can learn more complex stuff faster than the football player, and responds to a more complex strategic picture. Lower-level stability is mainly provided by the football player.

In 1987, Richard Sutton read [17], and arranged for us to discuss it at great length in Massachusetts. This was the event which injected the idea of ADP into the reinforcement learning school of AI. The paper is cited in Sutton's chapter in [18], which includes an implementation of the idea of "dreaming as simulation" discussed in [17].

C. Engineering Roadmap and Neuroscience Evidence for Second Generation Theory/Design

In 1992, I believed that we could probably replicate the level of intelligence we see in the basic mammal brain, simply by refining and filling in these first and second generation theories of how the brain works. In fact, the first and second generation design already offer potential new general-purpose adaptive capabilities far beyond what we now have in engineering. It is still essential that we continue the program of refining and understanding and improving these classes of designs as far as we can go – both for the sake of engineering, and as a prerequisite to set the stage for even more powerful designs.

I have suggested that half of the funding aimed at reverse engineering the brain should still go towards the first and second generation program – half towards the ADP aspects, and half towards the critical subsystems for

prediction, memory and so on. (See www.eas.asu.edu/~nsfadp .) Because those are complicated issues, and I have written about them elsewhere, I will not elaborate here.

More and more evidence has accumulated suggesting that optimization (with a predictive or "Model" component) is the right way to understand the brain. For example, Nicolelis and Chapin, in *Science*, reported that certain cells in the thalamus act as advance predictors of other cells. More important, when they cut the existing connections, the thalamo-cortical system would adapt in exactly the right way to relearn how to predict. This is clear evidence that the thalamo-cortical system – the biggest part of the brain – is in great part an adaptive "Model" network, a general-purpose system for doing adaptive "system identification" (as we say in control theory). Barry Richmond has observed windows of forwards and backwards waves of information in this circuit, fully consistent with our TLRN model of how such a Model network can be constructed and adapted.

Papez and James Olds senior showed decades ago how cells in the "limbic system" convey "secondary reinforcement signals," exactly as we would predict for an adaptive Critic component of the brain. More recent work on the dopamine system in the basal ganglia suggests even more detailed relations between reinforcement learning and actual learning in neural circuits.

A key prediction of the engineering approach has always been the existence of subcircuits to compute the derivatives – the generalized backpropagation – required here. When we first predicted backwards synapses, to make this possible, many ridiculed the engineering approach. But later, in Science, Bliss et al reported a "myserious" but strong reverse NMDA synapse flow. Spruston and collaborators have reported backpropagation flows (totally consistent with the mathematics of generalized backpropagation) in cell membranes. The synchronized clock signals implied in these designs are also well-known at present to "wet," empirical neuroscientists.

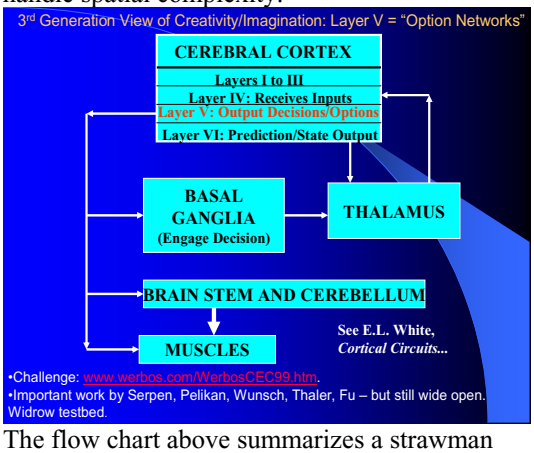
More details – and the empirical implications which cry out for follow-on work – are discussed in some of the papers on my web page, such as papers for books edited by Pribram.

One interesting recent thought. From engineering work, we have learned that the complexity of the *learning* system needed to train a simple input-output system or learned policy is far greater than the complexity of the input-output system itself. A simple example comes from Kalman filtering, where the "scaffolding" matrices (P, etc.) needed for consistent filtering are n times as large as the actual state estimates themselves; n is the number of state variables. Could it be that "junk DNA" includes a large system whose purpose is to tune the adaptation of the "coding DNA," which are after all only a small portion of our genetic system? Could it be that individual neurons do contain very complex molecular memories after all – memories invisible to our conscious mind, but essential to more efficient learning (such as the matrices for DEKF learning)? These are important empirical issues to explore.

IV. BRIDGING THE GAP TO THE MAMMAL-BRAIN LEVEL

AI researchers like Albus [19] have long assumed that brains must have very complex, explicit, hard-wired hierarchies of systems to handle a high degree of complexity in space and in time. By 1997, I became convinced that they are partly right, because I was able to formulate modified Bellman equations which allow much faster learning in cases where a state space can be sensibly partitioned in a (learnable) hierarchical way[20,21]. Nature would not neglect such an opportunity – and it fit well with emerging new knowledge about the basal ganglia, and ideas from Pribram.

Recent biological data does not support the older hierarchy ideas form AI, but it clearly call out for some kind of specific mechanisms in three core areas: (1) a "creativity/imagination" mechanism, to address the nonconvex nature of complicated optimization problems; (2) a mechanism to exploit modified Bellman equations, in effect; and (3) a mechanism to handle spatial complexity.

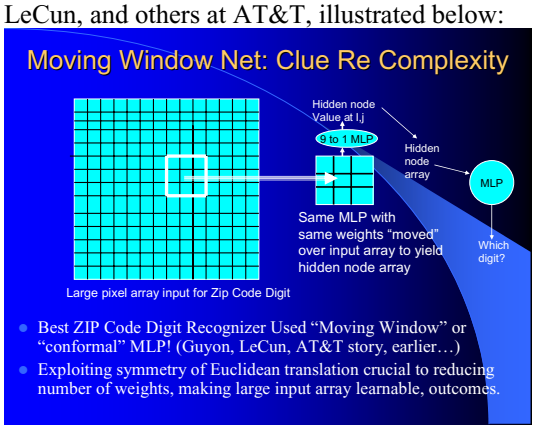


The flow chart above summarizes a strawman

model of the creativity mechanism which I proposed in 1997 [20]. I hoped to stimulate broad research into "brain-like stochastic search." (See my web page for a CEC plenary talk on that challenge). Wunsch, Serpen, Thaler, Pelikan and Fu's group at Maryland have all done important early work relevant to this task, but it hasn't really come together. Likewise, work in the last few years on temporal complexity has not done full justice to the modified Bellman equations, and has not shown as much progress as hoped for; part of the problem is that temporal complexity is usually associated with spatial complexity as well. Also, there is new evidence from neuroscience which has not yet been assimilated on the technology side.

The most exciting opportunity before us now is to follow up on more substantial progress and new ideas related to spatial complexity.

An important early clue towards spatial complexity came from the work of Guyon, LeCun, and others at AT&T, illustrated below:



The most accurate ZIP code digit recognizer then came from a simple MLP network, *modified* to exploit symmetry with respect to spatial translation. Instead of independently training hidden neurons to process pieces of an image, they would train a *single* hidden neuron, and *re-use* it in different locations, by moving it around the image. Le Cun later called this a "conformal neural network." He had excellent results training it by backpropagation in many image processing tasks. Nevertheless, these feedforward networks could still not learn the more complex kinds of mappings, like the connectedness mapping described long ago by Minsky[22]; it is not surprising that a network which could not handle connectedness, or learn to emulate Hough relaxation of image data, could not learn how to segment an entire ZIP code.

In 1994, Pang and I demonstrated a network that could solve these problems – a "Cellular SRN," (CSRN), which combines the key capabilities of a Simultaneous Recurrent Network [12] and a "conformal" network. This immediately allows prediction and control and navigation through complex two-dimensional scenes (and images) far more complex than an MLP could ever learn. That did not become immediately popular, in part because the learning was slow and tools were not available to make it easy for people to take advantage of the great brain-like power of such networks. This year, however, Ilin, Kozma and myself, at IJCNN06, reported a new learning tool which dramatically speeds up learning, and is available from Kozma as a MatLab tool. This by itself opens the door for neural networks to solve complex problems they could never really handle in the past. (There is great room for research to speed it up even more, but it is ready for practical use already.)

In 1997 [20] and in subsequent tutorials (and a patent), I proposed a more general approach to exploiting symmetry, which I called the ObjectNet. Instead of mapping a complex input field into M rectangular cells, all governed by a common "inner loop" neural network, one may map it into a network of k types of "Objects," with k different types of "inner loop" neural networks. This has great potential in areas like electric power and image processing, for example. A conventional MLP or recurrent network can learn to manage perhaps a few dozen variables in a highly nonlinear system – but how can one design a neural network which inputs the thousands of variables of an entire electric power grid and predict the system as a whole? Object nets provide a way of doing that.

This year, Venayagamoorthy has published preliminary results showing how an ADP system based on a simple, feedforward version of ObjectNet can handle power systems more complex than the earlier first-generation brain-like systems (which already outperformed more conventional control methods). More astonishing – David Fogel used a simple ObjectNet as the Critic in a system adapted to play chess. This was the world's first computer system to achieve master-class performance in chess *without* using a supercomputer and without using detailed clues and advice from a human; it *learned* how to play the game at that level.

But all of this is just a beginning. At www.face-rec.org, a series of reviews basically show that two of the three top working systems today rely on neural network concepts (vonderMalsburg and Wechsler). The key to face recognition turns out to be the ability to *learn* new "invariants" or transformations, more

complex than simple two-dimensional translation. This offers some easy short-term possibilities: to exploit CSRNs to learn the relevant mappings, which could never be learned before. But it also poses a very fundamental question: how can the *brain* learn such transformations?

Here is a new, more formal way to think about what is going on here. The first challenge here is to learn "symmetries of the universe." More concretely, the challenge to the brain is to learn a family of vector maps f_α such that: $Pr(f_\alpha(\underline{\mathbf{x}}(t+1)|f_\alpha(\underline{\mathbf{x}}(t))=Pr(\underline{\mathbf{x}}(t+1)|\ \underline{\mathbf{x}}(t))$ for all α and the same conditional probability distribution Pr. This new concept may be called stochastic invariance.

Once a brain learns these symmetries, it may exploit them in one or more of three ways:
*"reverberatory generalization": after observing or remembering a pair of data $\{\underline{\mathbf{x}}(t+1), \underline{\mathbf{x}}(t)\}$, also train on $\{f_\alpha(\underline{\mathbf{x}}(t+1)),f_\alpha(\underline{\mathbf{x}}(t))\}$;
*"multiple gating": after inputting $\underline{\mathbf{x}}(t)$, pick α so as to use f_α to map $\underline{\mathbf{x}}(t)$ into some canonical form, and learn a universal predictor form canonical forms. (This is analogous to the Olshausen model, which is *very* different in principle from neuroscience models of spontaneous or affective gating and attention.)
*"multimodular gating": like multiple gating, except that multiple parallel copies of the canonical mapping are used in parallel to process more than one subimage at a time in a powerful way.

Human brains seem to rely on the first two, or the second. Perhaps higher levels of intelligence could be designed here. But this begs the question: how could these maps be learned? How could the brain learn to map complex fields into a condensed, canonical form for which prediction is much easier to learn? How can the "Objects" in an ObjectNet be learned?

This suggests an immediate and astonishingly simple extension of the ObjectNet theory. In 1992, I proved basic consistency results for a new architecture called the "Stochastic Encoder/ Decoder Predictor" (SEDP).[12, chapter 13]. SEDP *directly* learns condensed mappings. It is an adaptive nonlinear generalization of Kalman filtering, explicit enough to allow the learning of symmetry relations. As with the earlier HDP and CSRN architectures, it will require many specific tricks to improve its learning speed. (e.g., exploitation of nearest neighbor relation in the learning, and salience flows?). It provides a principled way to *learn* the symmetry groups which are the foundation for a principled approach to spatial complexity.

REFERENCES

[1] P.Werbos, What do neural nets and quantum theory tell us about mind and reality? In K. Yasue et al, eds, *No Matter, Never Mind : Proc. of Toward a Science of Consciousness.* John Benjamins 2002

[2] M. Bitterman, The evolution of intelligence *Scientific American* Jan.1965

[3] D.Levine & Elsberry (eds) *Optimality in Biological and Artificial Networks?*, Erlbaum, 1997

[4] P.Werbos, The elements of intelligence. Cybernetica (Namur), No.3, 1968.

[5] J.Von Neumann and O.Morgenstern, *The Theory of Games and Economic Behavior*, Princeton NJ: Princeton U. Press, 1953.

[6] H.Raiffa*, Decision Analysis* Addison-Wesley 1968

[7] J. Si et al (eds) *Handbook of Learning & Approximate Dynamic Programming* Wiley/IEEE 2004.

[8] E.A.Feigenbaum and J.Feldman, *Computers and Thought*, McGraw-Hill, 1963.

[9] D.O.Hebb, *Organization of Behavior*, Wiley 1949

[10] P.Werbos, The Roots of Backpropagation, Wiley, 1994

[11] P. Werbos, Backwards differentiation in AD and Neural Nets. M. Bucker et al (eds), *Automatic Differentiation: Applications, Theory & Implementations*, Springer (LNCS), New York, 2005.

[12] White & D.Sofge, eds, *Handbook of Intelligent Control*, Van Nostrand, 1992.

[13] P.Werbos, Advanced forecasting for global crisis warning and models of intelligence, *General Systems Yearbook*, 1977 issue.

[14]--, Changes in global policy analysis procedures suggested by new methods of optimization, *Policy Analysis & Info. Systems*, Vol.3, No.1, June 1979.

[15]--, Applications of advances in nonlinear sensitivity analysis, in R.Drenick & Kozin (eds), *System Modeling and Optimization,* Springer 1981

[16]see adap-org 9810001 at arXiv.org, 1998

[17] P.Werbos, Building & understanding adaptive systems: A statistical/numerical approach to factory automation & brain research, *IEEE Trans. SMC*, Jan./Feb. 1987.

[18] Miller et al eds, *Neural Networks for Control, MIT Press 1990*

[19] J.Albus, Outline of Intelligence, *IEEE Trans. Systems, Man and Cybernetics*, Vol.21, No.2, 1991.

[20] Sutton TD R.S.Sutton, Learning to predict by the methods of temporal differences, *Machine Learning*, Vol. 3, p.9-44, 1988

[21] P.Werbos, Brain-Like Design To Learn Optimal Decision Strategies in Complex Environments, in M.Karny et al eds, *Dealing with Complexity: A Neural Networks Approach*. Springer, London, 1998.

[22] P.Werbos, Multiple Models for Approximate Dynamic Programming ... In K. Narendra, ed., *Proc. 10th Yale Conf. on Learning and Adaptive Systems*. New Haven: K.Narendra, EE Dept., Yale U., 1998.

[23] M. Minsky & S. Papert, *Perceptrons: An Introduction to Computational Geometry*, MIT Press, 1969