

# Practical Architectural Limits on Complex Learning Systems

L. Andrew Coward  
Department of Computer Science  
Australian National University  
Canberra, ACT 0200  
E-mail: andrew.coward@anu.edu.au

**Abstract** - The primary reason for the inability of most neural network models to support learning of large scale problems is lack of attention to the general architectural constraints which result from a range of practical considerations including the need to limit use of information handling resources but to add and modify features without side effects on existing features. The resultant constraints can be clearly seen in the architectures of complex real time control systems, from the relatively simple flight simulators to central office telecommunications switches which are currently the most complex such systems. The constraints on systems in which features are defined under external intellectual control can be a guide for understanding analogous constraints on learning systems, although in detail the architectural forms are radically different. Research on systems which learn to solve large scale problems must be guided by knowledge of the appropriate architectural constraints.

## I. PROBLEM STATEMENT

The design of learning systems has rarely been approached from a system architectural perspective. The main focus in system architecture is upon efficient use of physical resources including information storage, information processing and information exchange (connectivity) resources. Other important issues are the need to maintain adequate synchronization between information processing in different parts of the system and to maintain adequate meanings for information shared between different parts of the system. The design problem for a system which must perform many different operational (or behavioural) features is relatively simple if separate physical information handling resources are available for each feature. The problem is much more difficult if resources must be shared between different features.

The source of the difficulty is that when resources must be shared any changes to resources required to modify a feature or add a new feature will tend to have undesirable side effects on other features using the same resources. The design of electronic systems that perform complex combinations of operational features demonstrates this problem most sharply [see e.g. Bass, Clements and Kazman, 1998]. These electronic systems are real time systems controlling extensive physical equipment. A relatively simple example is the software to control a flight simulator [Chastek and Brownsword, 1996]. Perhaps the most complex example is the central office switch that provides on demand

telecommunications services to 100 thousand people or computers with no human intervention [Nortel Networks, 2000]. Such a switch has tens of millions of lines of software and many billions of transistors packaged into custom integrated circuits. Switch design requires tens of thousands of man-years of engineer effort.

The architecture of such systems must be an adequate compromise between resource economy and modifiability. This compromise results in severe constraints on system architectures, including a requirement for modular hierarchies with careful control over the information exchange between modules [see Kamal, 1987]. Such modules do not correspond with operational features, resulting in considerable differences between system architecture descriptions and user manual descriptions.

For systems which must learn a complex combination of behavioural features with resources that are not unlimited, it can be demonstrated theoretically [Coward 2001, 2004] that the practical needs result in some analogous architectural constraints, although in detail they are quite different from those on systems designed under external intellectual control. The remarkably specific constraints on complex learning systems are called the recommendation architecture.

The recommendation architecture limits require a complex learning system to separate into two major subsystems: clustering and competition. Clustering defines and detects conditions within the input space available to the system. Most of the condition detections are used to determine when and where additional information will be recorded, and small proportion of condition detections are released to competition. Each condition detection released to competition is interpreted as a range of different behavioural recommendations, each with a different weight. Competition determines and implements the most strongly recommended behaviours. Such behaviours include both externally directed system actions and indirect activations of additional information within clustering. Consequence feedback following a behaviour can change recently activated recommendation weights but cannot change the definition of conditions. In fact, under most circumstances, once a condition has been defined by clustering it cannot be changed, resulting in device algorithms which permanently record portfolios of similar

conditions, with the portfolio being allowed to expand but not contract [Coward 2001]. The perceptron type device algorithm widely used in artificial neural networks is essential within the competition subsystem but cannot be used in clustering.

A high proportion of the information handling resources within clustering are required to manage when and where conditions are recorded, and to avoid excessive portfolio expansions. These requirements result in clustering devices being organized into a modular hierarchy of layers, columns, areas and subsystems; and competition into a component hierarchy with components controlling general behaviour types, more specific behaviour types, and so on down to individual behaviours.

As the ratio of behaviours to resources increases, the system will be more and more tightly constrained into the recommendation architecture form.

## II. CURRENT RESEARCH

Electronic implementations of systems within the recommendation architecture bounds have demonstrated how such systems can avoid interference between early and later learning even when many different features are learned [Coward, Gedeon and Ratanayake, 2004]. Research continues on increasingly complex problem spaces. It is also of interest that the mammal brain strongly resembles a system within the recommendation architecture bounds both in physical form and in cognitive processes [Coward 2005]. Research to test experimental predictions resulting from these resemblances is under way. Higher memory and cognitive processes in human beings strongly resemble processes by which a system within the recommendation architecture bounds determines appropriate behaviour. Research continues on building systems with large numbers of simulated but physiologically realistic neurons within the recommendation architecture limits to establish more quantitative correlations with cognitive behaviour.

## III. KEY AVENUES

The recommendation architecture forms are required for systems which perform large numbers of different behavioural with limited resources. If the number of features is relatively small and the need for ongoing modification is limited (e.g. if learning only occurs in an initial training period) then other architectural forms may be more effective for the specific problem.

It is therefore inappropriate to benchmark systems within the recommendation architecture bounds against other learning approaches using limited problems, because it will in general be possible to solve such simpler problems more effectively without the recommendation architecture constraints.

An appropriate benchmark would be learning to discriminate between hundreds or

thousands of different categories, where categories were introduced gradually, and accurate performance required immediately after a category was introduced. A further aspect of such a test would be a large input space (10,000+ bits) and a requirement that input states (i.e. category instances) were only experienced once: there would be no repetitive learning on exactly the same input state.

One direction of research is investigation of different device algorithms and modular arrangements within the recommendation architecture bounds against some such benchmark problem.

Modeling of human cognitive capabilities, including bootstrapping of such capabilities with minimal a priori knowledge, is feasible for systems within the recommendation architecture bounds. Such modeling has been carried out at an intermediate design level and the necessary processes confirmed by electronic simulations [Coward 2005] but a further important research direction is more detailed modeling of such capabilities.

## REFERENCES

- Bass, L., Clements, P. and Kazman, R. (1998). *Software Architecture in Practice*. Addison-Wesley.
- Chastek, G. and Brownsword, L. (1996). *A Case Study in Structural Modeling*. Carnegie Mellon University Technical Report CMU/SEI-96-TR-035.  
[www.sei.cmu.edu/pub/documents/96.reports/pdf/tr035.96.pdf](http://www.sei.cmu.edu/pub/documents/96.reports/pdf/tr035.96.pdf)
- Coward, L.A. (2001). *The Recommendation Architecture: lessons from the design of large scale electronic systems for cognitive science*. *Journal of Cognitive Systems Research* 2(2), 111-156.
- Coward, L. A. (2004). *The Recommendation Architecture Model for Human Cognition*. *Brain Inspired Cognitive Systems 2004*, L. S. Smith, A. Hussain and I. Aleksander, (editors), University of Stirling.
- Coward, L. A., Gedeon, T. D. and Ratanayake, U. (2004). *Managing Interference between Prior and Later learning*. ICONIP 2004, Calcutta.
- Coward, L. A. (2005). *A System Architecture Approach to the Brain: from Neurons to Consciousness*. New York: Nova Science Publishers
- Coward, L. A. (2005). *Accounting for episodic, semantic and procedural memory in the recommendation architecture cognitive model*. *Proceedings of the Ninth Neural Computation and Psychology Workshop: Modelling Language, Cognition, and Action*.
- Kamel, R. (1987). *Effect of Modularity on System Evolution*. *IEEE Software*, January, 48 – 54.
- Nortel Networks (2000). *DMS-100/DMS-500 Systems Feature Planning Guide*.  
<http://a2032.g.akamai.net/7/2032/5107/20030925230957/www.nortelnetworks.com/products/library/collateral/50004.11-02-00.pdf>