Complex Decision Making With Neural Networks: Learning Chess
Complete System Block Diagrams

Jack Sigan
Bradley University ECE Dept
Dr. Aleksander Malinowski, Advisor
November 23, 2004

System Description:

The system will operate in the learning mode prior to playing chess or advising. Figure 1 shows the two possible modes of operation, along with the interconnection between the modules included in each subsystem. The entire system is composed of software running on multiple PCs simultaneously, and no additional hardware is to be used. The details to follow in this paper assume that the geographical, or move based design paradigm is used.

No inputs or outputs are specified for the learning mode, which operates in an essentially automated process. The playing or advisory modes receive move inputs from the player and after evaluation is complete will return the best move(s) chosen by the neural network system. Prior to discussing the playing and training functions, it is necessary to briefly introduce the modules depicted in figure 1.

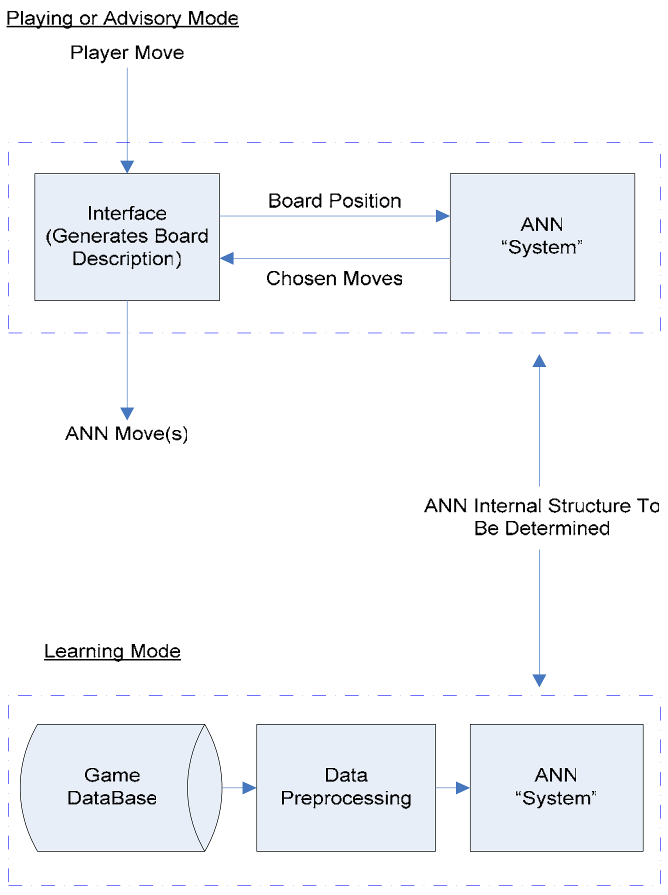

Fig. 1: System block diagrams for operating modes

Interface Module:

The interface module seen in figure 1 (playing or advisory mode) is an application the user interacts with on the PC. It displays a chess board and the piece layout for the game, and the user will make their moves on this board. In advisory mode, a list of returned neural network moves will be displayed in a text box for the user to examine. In playing mode, the computer will make its own moves directly on the board. The interface module must have bidirectional communication with the artificial neural network system (ANN system) to exchange board position data and move choices.

ANN System:

The ANN system is the central focus of the design. It will contain a series of neural networks (as many as 1856), operating in parallel, which are trained to play specific moves in the game as described in the functional description. Before use in the playing/advisory modes, the ANN system must be trained in the learning mode, where it starts as a randomly connected, randomly weighted network. The training takes place in an automated process utilizing millions of data samples representing "good" moves in chess. Once trained, the ANN system is used by the interface module to play the game. A board position may be passed to the system, and a list of suggested moves will be returned. The interface decides whether to make one move or to display a list of suggestions based on whether it is set to playing mode or advisory mode.

Game Database:

The game database is a standalone application called ChessBase 9.0. This application allows a massive database of several million saved chess games to be sorted, filtered and searched by move. ChessBase 9.0 is used exclusively in the learning mode, at the start of the preprocessing stage. All game data must be preprocessed in order to convert it from the database format to a format usable for neural network training. Games from ChessBase 9.0 are extracted by searching for specific moves, which defines the first step in the preprocessing module to be described shortly. Many individual sets of games are made, which will then be passed to the remaining preprocessing stages.

Data Preprocessing:

Data preprocessing begins in the game database described above, and continues with a series of applications designed to convert the chosen game data into neural network training vectors. The data processing stage converts PGN files (portable game notation—a popular algebraic game recording standard) which are extracted from the database, to EPD strings (extended position description). EPD files contain a series of strings representing the full board position at each move recorded in the PGN file. Figure 2 shows a sample of PGN format, and figure 3 shows a sample of EPD format. At this time a detailed understanding of file content is not required.

```
1. e4 d6 2. d4 Nf6 3. Nc3 g6 4. Nf3 Bg7 5. Be2 O-O 6. O-O Bg4 7. Be3 Nc6
8. Qd2 e5 9. d5 Ne7 10. Rad1 Bd7 11. Ne1 Ng4 12. Bxg4 Bxg4 13. f3 Bd7 14. f4
Bg4 15. Rb1 c6 16. fxe5 dxe5 17. Bc5 cxd5 18. Qg5 dxe4 19. Bxe7 Qd4+ 20. Kh1
f5 21. Bxf8 Rxf8 22. h3 Bf6 23. Qh6 Bh5 24. Rxf5 gxf5 25. Qxh5 Qf2 26. Rd1
e3 27. Nd5 Bd8 28. Nd3 Qg3 29. Qf3 Qxf3 30. gxf3 e4 31. Rg1+ Kh8 32. fxe4
fxe4 33. N3f4 Bh4 34. Rg4 Bf2 35. Kg2 Rf5 36. Ne7 1-0
```

Fig. 2: Sample of the PGN file standard

```
rnbqkbnr/pppppppp/8/8/8/8/PPPPPPPP/RNBQKBNR w KQkq - pm d4;
rnbqkbnr/pppppppp/8/8/3P4/8/PPP1PPPP/RNBQKBNR b KQkq d3 pm Nf6;
rnbqkb1r/pppppppp/5n2/8/3P4/8/PPP1PPPP/RNBQKBNR w KQkq - pm Nf3;
rnbqkb1r/pppppppp/5n2/8/3P4/5N2/PPP1PPPP/RNBQKB1R b KQkq - pm b6;
rnbqkb1r/p1pppppp/1p3n2/8/3P4/5N2/PPP1PPPP/RNBQKB1R w KQkq - pm g3;
rnbqkb1r/p1pppppp/1p3n2/8/3P4/5NP1/PPP1PP1P/RNBQKB1R b KQkq - pm Bb7;
```

Fig. 3: Sample of the EPD file standard

Of course, neural networks require a numeric input vector for training. It is therefore required to convert the EPD stings into floating point input vectors by converting the EPD characters to floating point values. A sample training vector is shown in figure 4. The output pattern is 1 or -1, corresponding to whether or not to make the move specific to the training data set and the neural network. The hope is that after learning enough patterns, the neural network will be able to effectively make this move decision for data samples which are not contained in the original training set.

```
# Input pattern 1:
0.5 0.4 0.3 0.9 1 0.3 0.4 0.5
0.1 0.1 0.1 0.1 0.1 0.1 0.1 0.1
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0
-0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1 -0.1
-0.5 -0.4 -0.3 -0.9 -1 -0.3 -0.4 -0.5
# output pattern 1:
1
```

Fig. 4: Sample floating point training sample

Now that the main modules have been adequately introduced, it is now possible to describe the full functionality of specific modes of operation.

Playing/Advisory System:

The interface module and the ANN system are used in the playing/advisory system. Figure 5 describes the operation of this entire system (or functional mode).

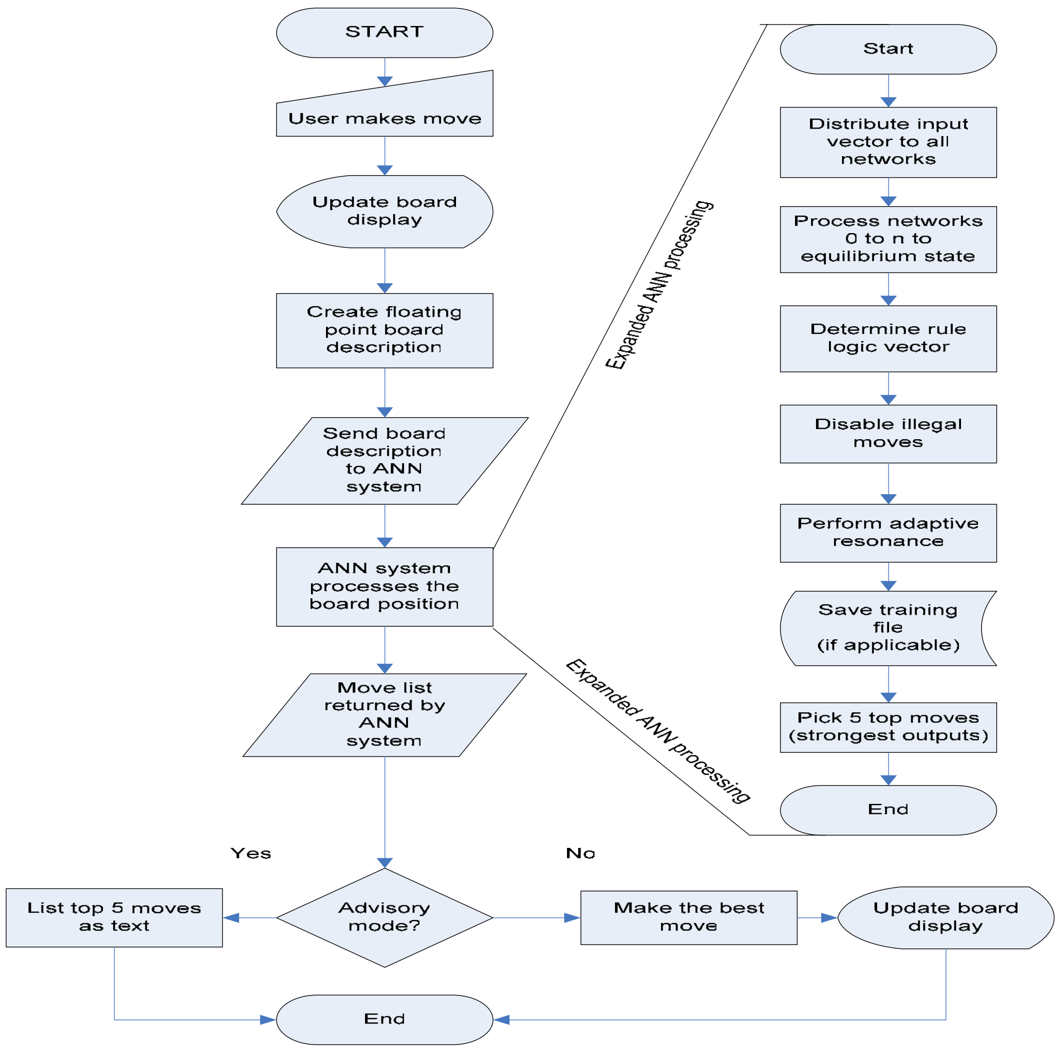
START
User makes move
Update board display
Create floating point board description
Send board description to ANN system
ANN system processes the board position
Move list returned by ANN system
Yes
No
Advisory mode?
List top 5 moves as text
Make the best move
Update board display
End
Expanded ANN processing
Expanded ANN processing
Start
Distribute input vector to all networks
Process networks 0 to n to equilibrium state
Determine rule logic vector
Disable illegal moves
Perform adaptive resonance
Save training file (if applicable)
Pick 5 top moves (strongest outputs)
End

Fig. 5: Playing/advisory system flowchart

In figure 5, the ANN board processing stage is expanded on the far right to give more insight into the ANN system. It is possible to treat the ANN processing as a single function as it has a clearly defined input and output, and its inner workings are of no consequence to the interface module so long as data is passed back and forth. The expanded ANN processing function can be better understood by figure 6, which shows the block diagram of the geographical ANN system design. Here, "adaptive resonance" is simply a specialized training session held after the initial training to "fine tune" networks so they make fewer illegal moves over time. The true form of adaptive resonance would make changes to the network during training, while in this case, although the data is saved within the playing/advisory mode, it is required to go back to the training mode to complete the "adaptive resonance." Thus, while not adhering to the true definition of adaptive resonance, the end result is the same; adjustment of weights to differentiate between similar inputs based on observation of a logical error.

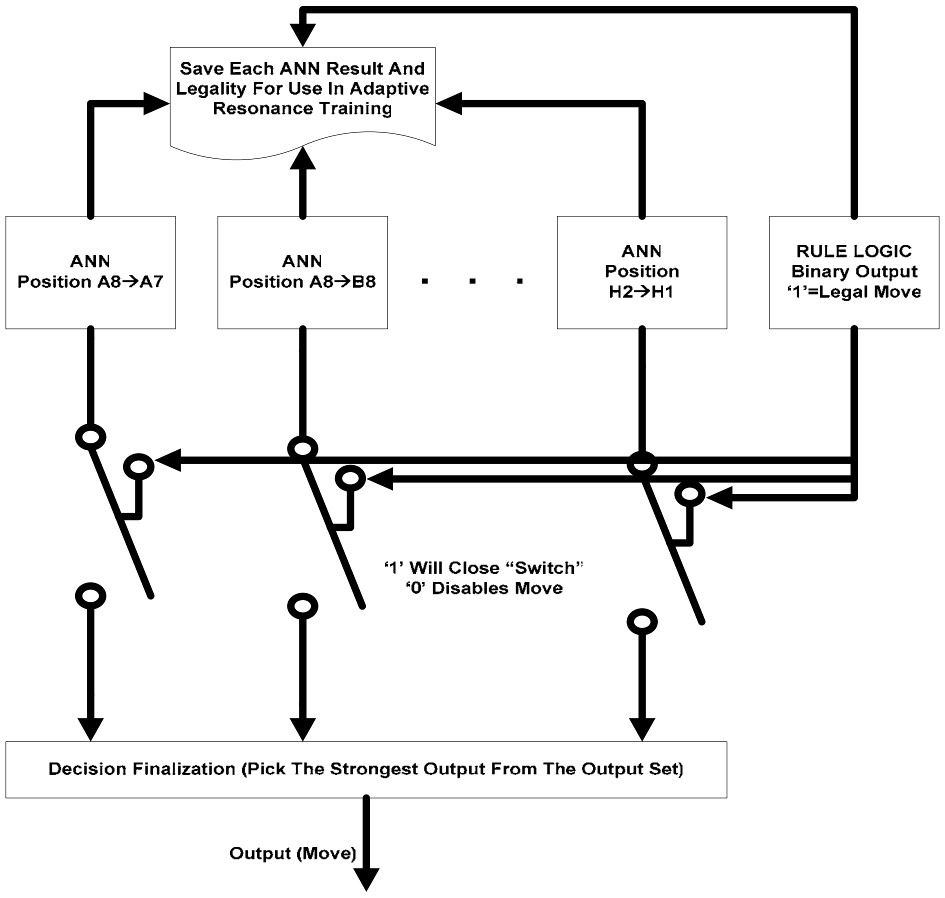
Save Each ANN Result And Legality For Use In Adaptive Resonance Training
ANN Position A8→A7
ANN Position A8→B8
. . .
ANN Position H2→H1
RULE LOGIC Binary Output '1'=Legal Move
'1' Will Close "Switch"
'0' Disables Move
Decision Finalization (Pick The Strongest Output From The Output Set)
Output (Move)

Fig. 6: Move based ANN system design (geographical design paradigm)

Learning System:

The learning system is comprised of the ChessBase 9.0 database, the preprocessing software, and the ANN system. It is possible to divide learning into two distinct functions: data preprocessing and training.

The data preprocessing function is described in figure 7. The database specific operations and selections can be considered a unique function, and an expanded view is provided at the bottom of figure 7. It is vital that the final output of the preprocessing stage is a set of training vectors specifically created for each possible legal move in chess!

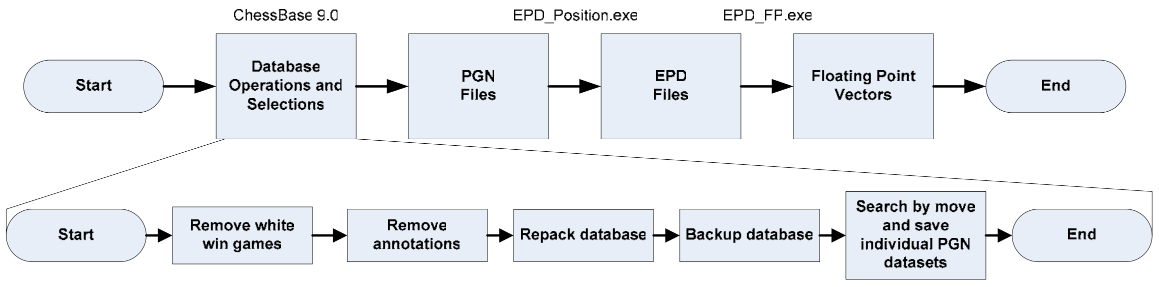

Fig. 7: Data preprocessing subsystem flowchart

Training takes place after the data preprocessing is complete. The end result of preprocessing is a set of “pattern files” which are loaded by the Stuttgart Neural Network Simulator (SNNS) to perform the training. Each network has its own pattern set, and training takes place one network at a time. Each training pattern is once again in the format described in figure 4. Training is performed entirely by SNNS, so the mathematical details of learning will not be described in this document. It is possible to instead treat the actual learning as a self contained process. Figure 8 describes the training process.

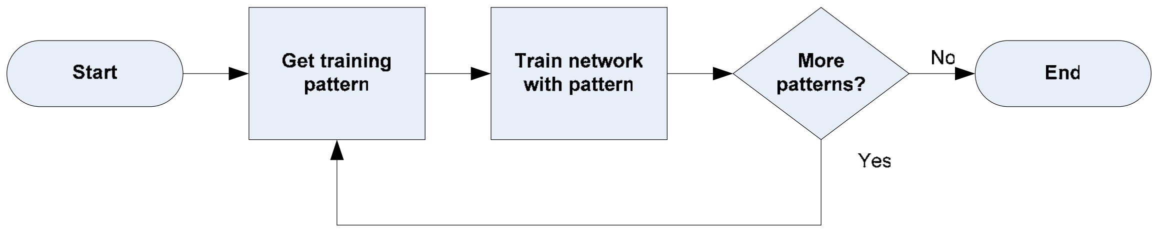

Fig. 8: Training process flowchart, using Stuttgart Neural Network Simulator

Training will actually be repeated numerous times for each network. Each time the entire pattern set is trained is known as a cycle. Approximately 100 cycles will be processed for each network, which is subject to change based on observed training effectiveness.