Real-world optimal control problems such as optimal plant control, smart control in robotic systems, etc. need efficient and easily implemented tools for their design. One of the possible ways for implementation of optimal feedback control systems is recursive ANN.

Let  be input data sequence,  be internal system variable (control) describing reaction of the system on the influence  and  be a response of the system at the control  that can be described via some predefined smooth transfer function . Let  be some real-value positive quality functional. Note that for the correct functioning of the control system one can define initial value of , for example .

Then simple ANN control system design is graphically represented in Fig. 1.3.

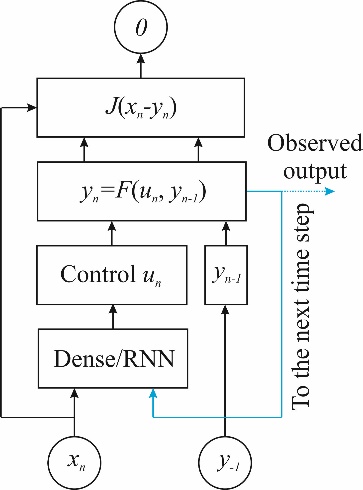


Fig. 1.3. Graphical representation simple ANN control system.

Here *Dense* or RNN (or mixed) trainable layer computes optimal control for the inputs  and , . In Fig. 1.3 blue arrow denotes recurrent cross-layer connection from the  to the next  time step via moving  to the -th time step. The unrolled (over time) version of the previous model has the form shown in Fig. 1.4.

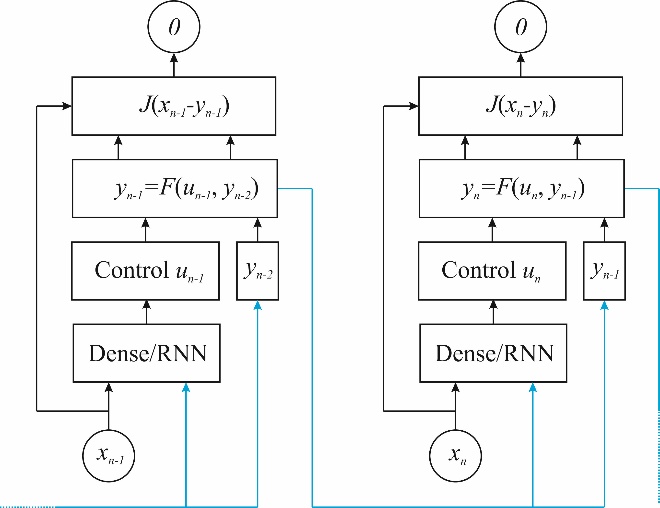


Fig. 1.4. Unrolling of ANN control system from Fig 1.3. over time steps.

This model has intuitive interpretation in terms of tracking problem: time series  is the target position, time series  is current tracker position (so that  its initial position) and the model tries to find optimal tracking strategy in order to “catch” the target. For example, let  (tracking with the limitation at displacement at the single time step) and - Euclidian distance. Then  and the optimization problem at the control ,  has the form:

.

Let  - the ANN solution of the optimization problem. Then:

.

It is clear, that in this simple case the solution has the form .

Let us note several important observations about the model from Fig.1.3. At first, note an absence of significant difference between train and inference regimes (no trainable output) – model can (and should) be in training regime all the time to continuously adaptation to changing conditions. At second, note that utilizing of RNN layers provides adding memory (about  as well as  up to the current time step) to the tracking strategy. Last observation reformulated at the language of control theory means that these strategies include PID (proportional-integral-differential) control technique and can be easily generalize to more advanced transfer functions (smart control).

Final note is connected with extension of previous approach. Training of the proposed ANN control supposes to have a target trajectory  versus time. Direct determining of this trajectory potentially has one important disadvantage – this trajectory can be too easy for tracking and does not account any information about tracker strategy. To avoid this problem consider the following game: let there are a fox (tracker) and a rabbit (runner). The fox haunts the rabbit and the problem is to synthesize optimal controls for both units. The ANN topology from the Fig. 1.5. (potentially) solve the problem.

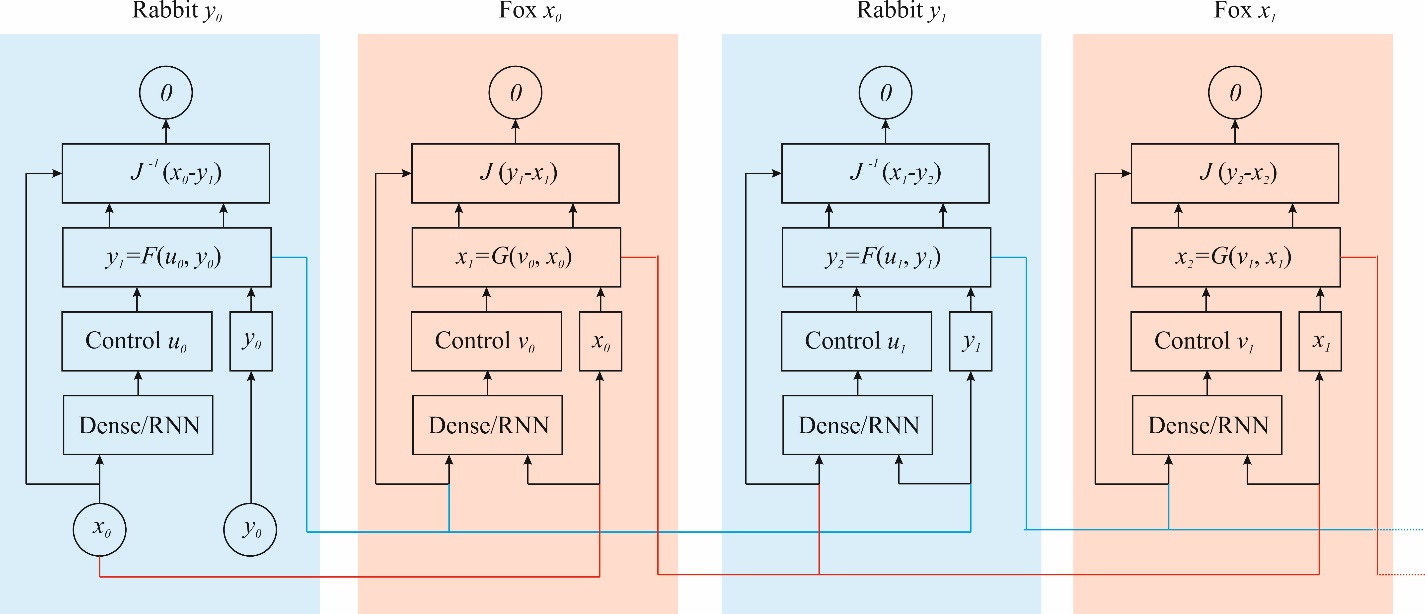


Fig. 1.5. Graphical representation of coupled, self-training control systems. Unrolling of two initial time steps is represented.

This example illustrates competition between different ANN strategy in real-time regime, competition between different ANN topology and units’ transfer functions  and . Another idea is to use combination of SNN layers and bProp algorithm to control/train units in order to approach to real biological systems.