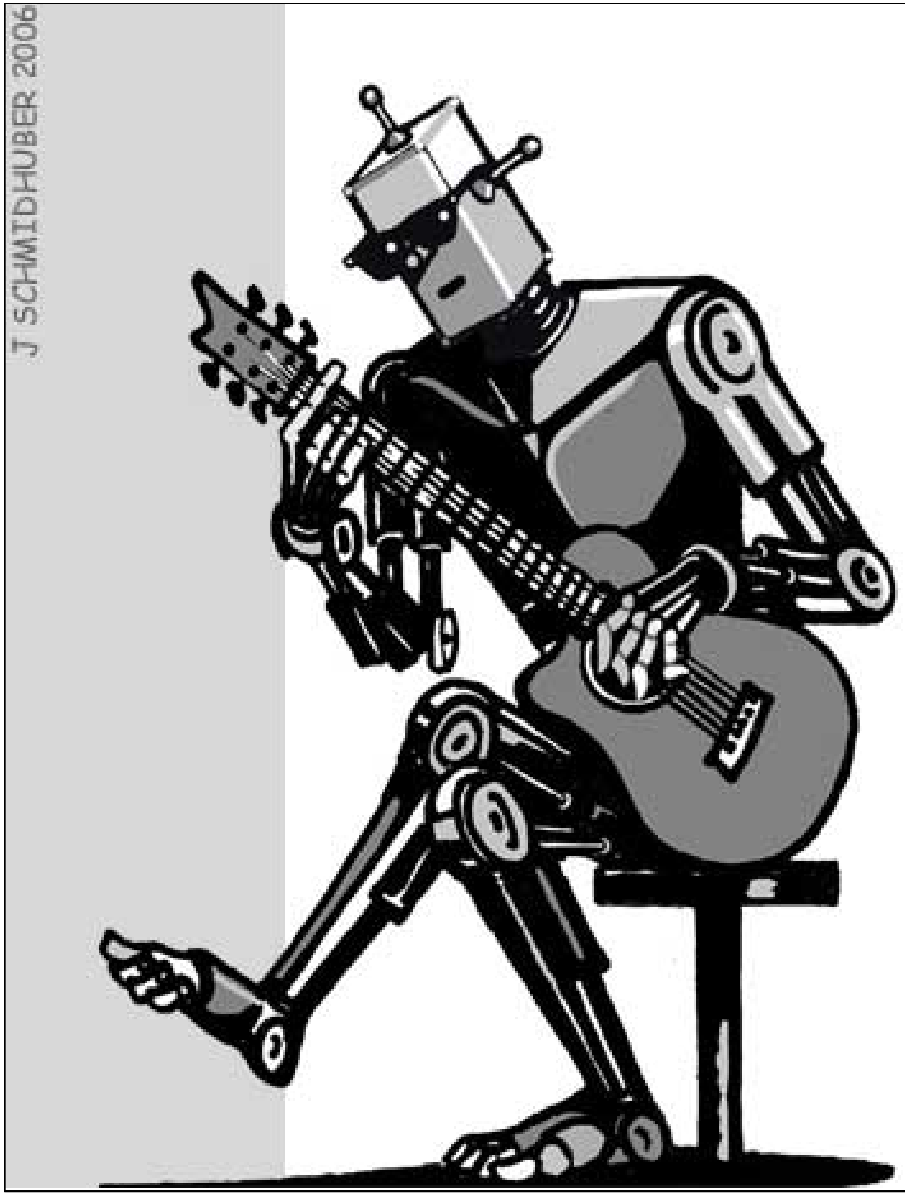


Continual Lifelong Meta-Learning & Artificial Curiosity

Jürgen Schmidhuber
The Swiss AI Lab IDSIA
Univ. Lugano & SUPSI
http://www.idsia.ch/~juergen

NNAISENSE

Jürgen Schmidhuber
You_again Shmidhoobuh

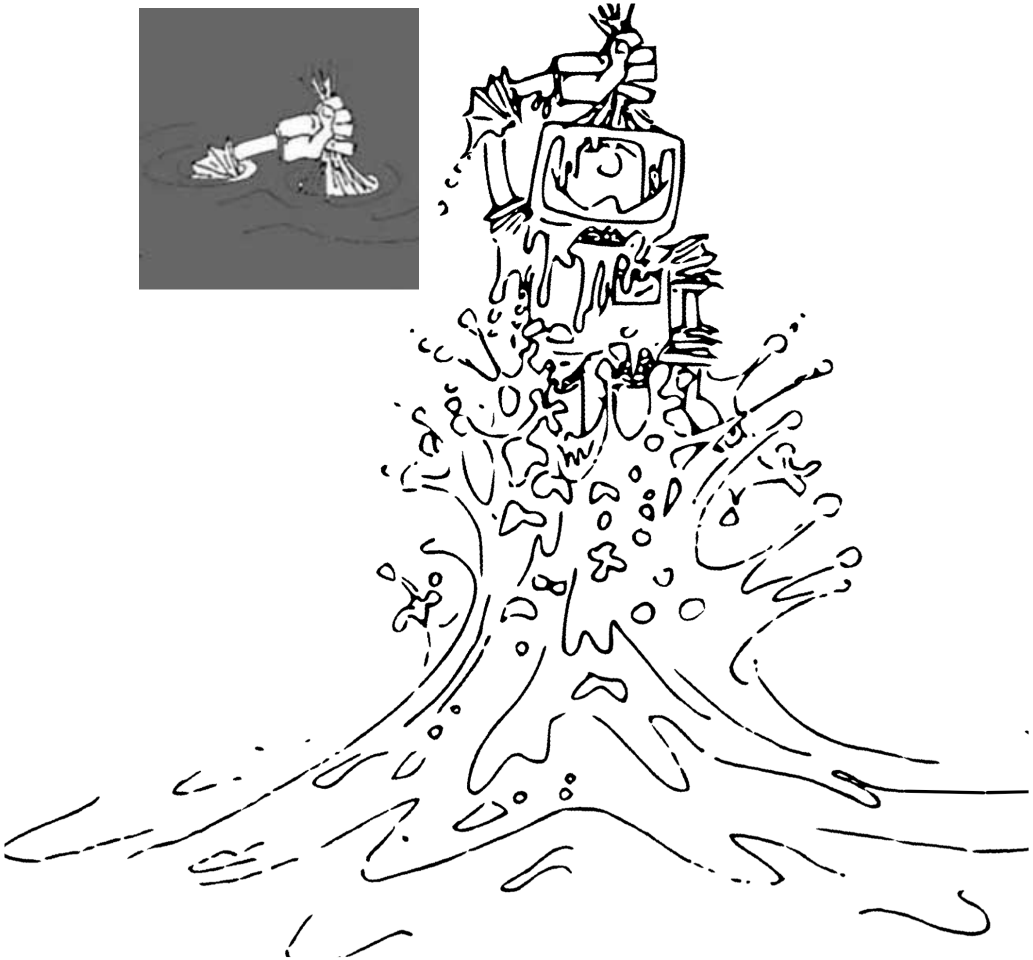
Continual Learning (Mark Ring, 1994-)
Meta-Learning (JS, 1987-)
Artificial Curiosity (JS, 1990-)

“True” Learning to Learn (L2L) is not just transfer learning! Even a simple feedforward NN can transfer-learn to learn new images faster through pre-training on other image sets

True L2L is not just about learning to adjust a few hyper-parameters such as mutation rates in evolution strategies (e.g., Rechenberg & Schwefel, 1960s)

Radical L2L is about encoding the initial learning algorithm in a universal language (e.g., on an RNN), with primitives that allow to modify the code itself in arbitrary computable fashion

Then surround this self-referential, self-modifying code by a recursive framework that ensures that only “useful” self-modifications are executed or survive (Recursive Self-Improvement)

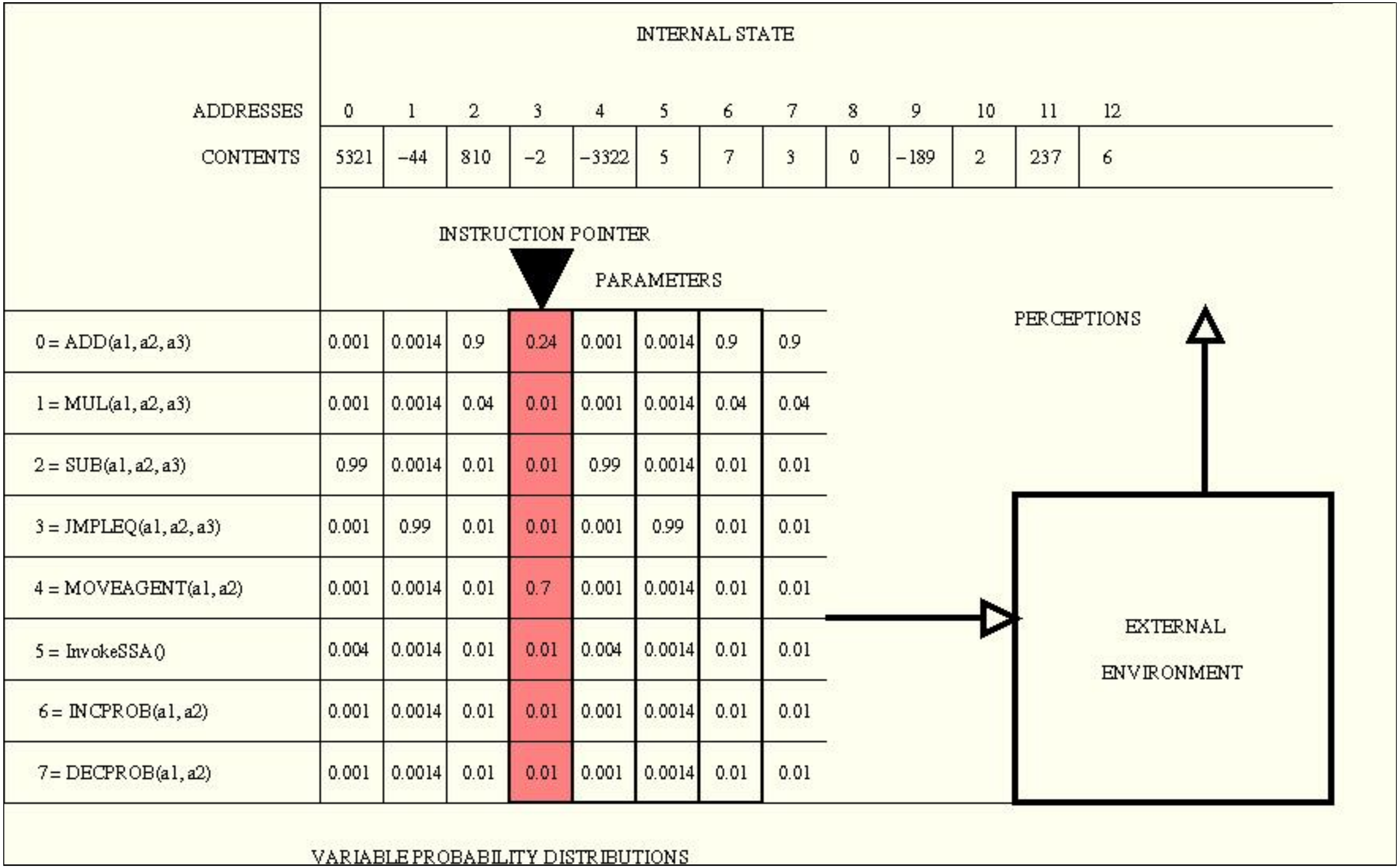


Lifelong R-Learning to Learn Learning Algorithms (1994)

Jürgen Schmidhuber
The Swiss AI Lab IDSIA
Univ. Lugano & SUPSI
http://www.idsia.ch/~juergen

NNAISENSE

1987: Diploma thesis on meta-learning how to learn how to learn & recursive self-improvement


INTERNAL STATE
ADDRESSES 0 1 2 3 4 5 6 7 8 9 10 11 12
CONTENTS 5321 −44 810 −2 −3322 5 7 3 0 −189 2 237 6
INSTRUCTION POINTER
PARAMETERS
0 = ADD(a1, a2, a3) 0.001 0.0014 0.9 0.24 0.001 0.0014 0.9 0.9
1 = MUL(a1, a2, a3) 0.001 0.0014 0.04 0.01 0.001 0.0014 0.04 0.04
2 = SUB(a1, a2, a3) 0.99 0.0014 0.01 0.01 0.99 0.0014 0.01 0.01
3 = JMPLEQ(a1, a2, a3) 0.001 0.99 0.01 0.01 0.001 0.99 0.01 0.01
4 = MOVEAGENT(a1, a2) 0.001 0.0014 0.01 0.7 0.001 0.0014 0.01 0.01
5 = InvokeSSA() 0.004 0.0014 0.01 0.01 0.004 0.0014 0.01 0.01
6 = INCPROB(a1, a2) 0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01
7 = DECPROB(a1, a2) 0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01
PERCEPTIONS
EXTERNAL
ENVIRONMENT
VARIABLE PROBABILITY DISTRIBUTIONS

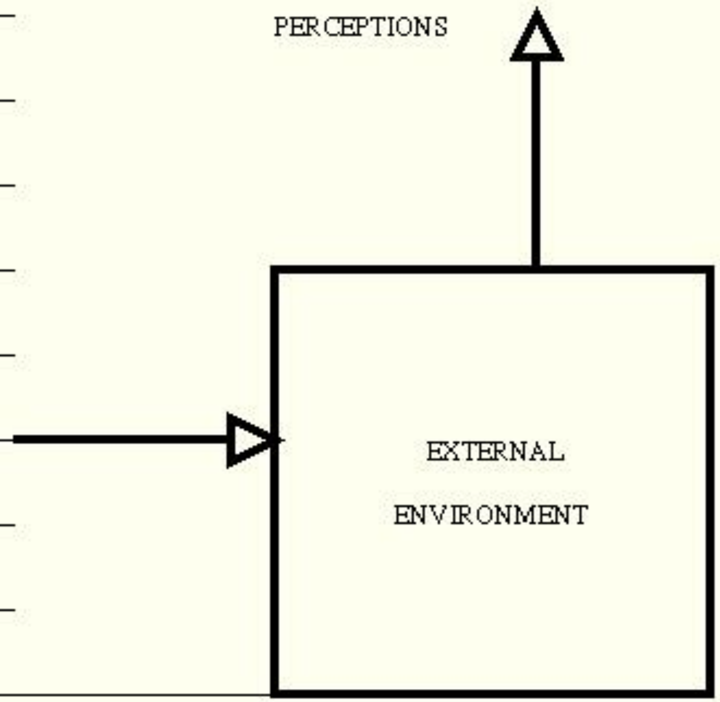
INTERNAL STATE

ADDRESSES	0	1	2	3	4	5	6	7	8	9	10	11	12
CONTENTS	5321	−44	810	−2	−3322	5	7	3	0	−189	2	237	6

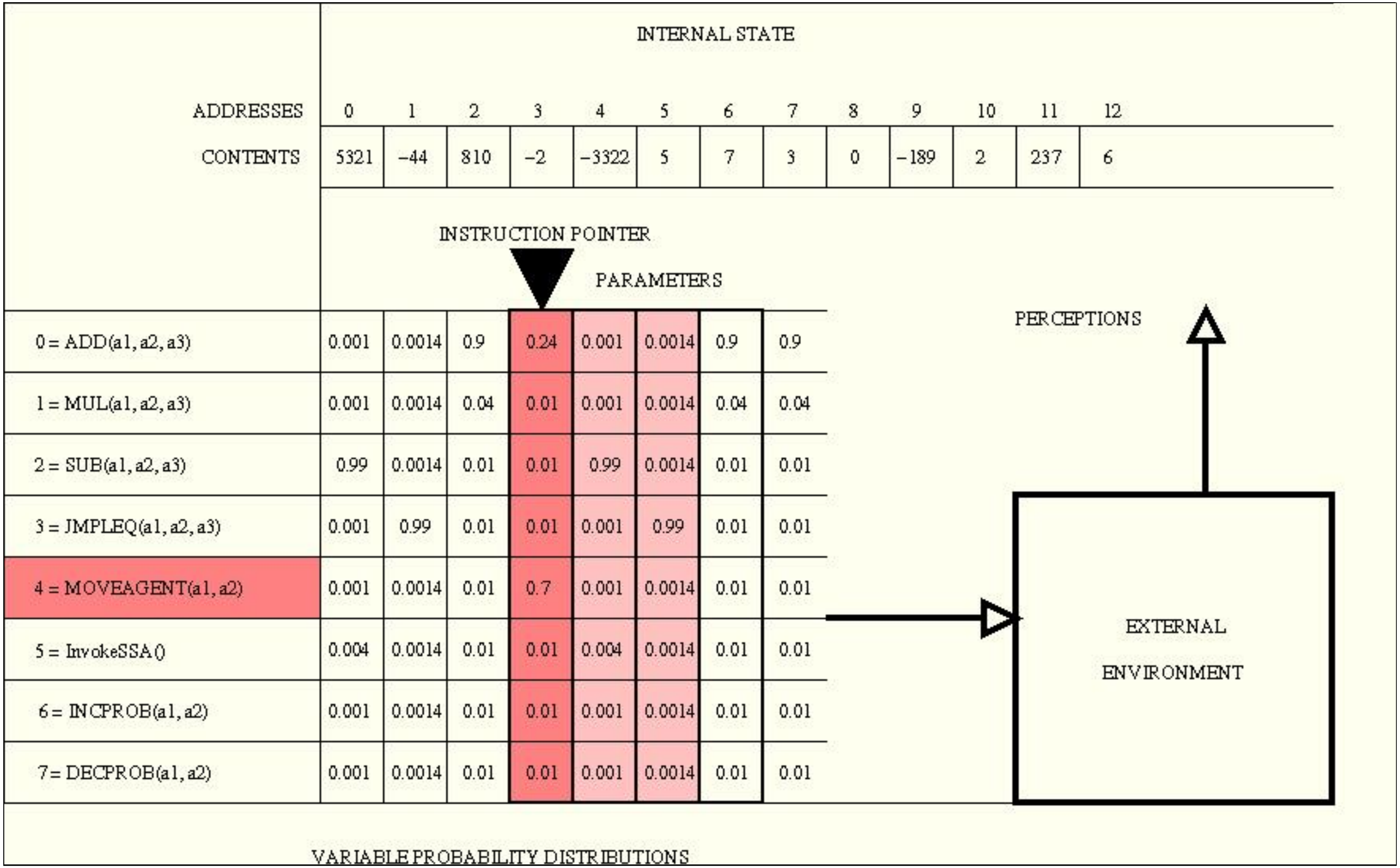
INSTRUCTION POINTER

PARAMETERS

0 = ADD(a1, a2, a3)	0.001	0.0014	0.9	0.24	0.001	0.0014	0.9	0.9
1 = MUL(a1, a2, a3)	0.001	0.0014	0.04	0.01	0.001	0.0014	0.04	0.04
2 = SUB(a1, a2, a3)	0.99	0.0014	0.01	0.01	0.99	0.0014	0.01	0.01
3 = JMPLEQ(a1, a2, a3)	0.001	0.99	0.01	0.01	0.001	0.99	0.01	0.01
4 = MOVEAGENT(a1, a2)	0.001	0.0014	0.01	0.7	0.001	0.0014	0.01	0.01
5 = InvokeSSA()	0.004	0.0014	0.01	0.01	0.004	0.0014	0.01	0.01
6 = INCPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01
7 = DECPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01



VARIABLE PROBABILITY DISTRIBUTIONS



ADDRESSES	0	1	2	3	4	5	6	7	8	9	10	11	12
CONTENTS	5321	−44	810	−2	−3322	5	7	3	0	−189	2	237	6

0 = ADD(a1, a2, a3)	0.001	0.0014	0.9	0.24	0.001	0.0014	0.9	0.9
1 = MUL(a1, a2, a3)	0.001	0.0014	0.04	0.01	0.001	0.0014	0.04	0.04
2 = SUB(a1, a2, a3)	0.99	0.0014	0.01	0.01	0.99	0.0014	0.01	0.01
3 = JMPLEQ(a1, a2, a3)	0.001	0.99	0.01	0.01	0.001	0.99	0.01	0.01
4 = MOVEAGENT(a1, a2)	0.001	0.0014	0.01	0.7	0.001	0.0014	0.01	0.01
5 = InvokeSSA()	0.004	0.0014	0.01	0.01	0.004	0.0014	0.01	0.01
6 = INCPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01
7 = DECPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01

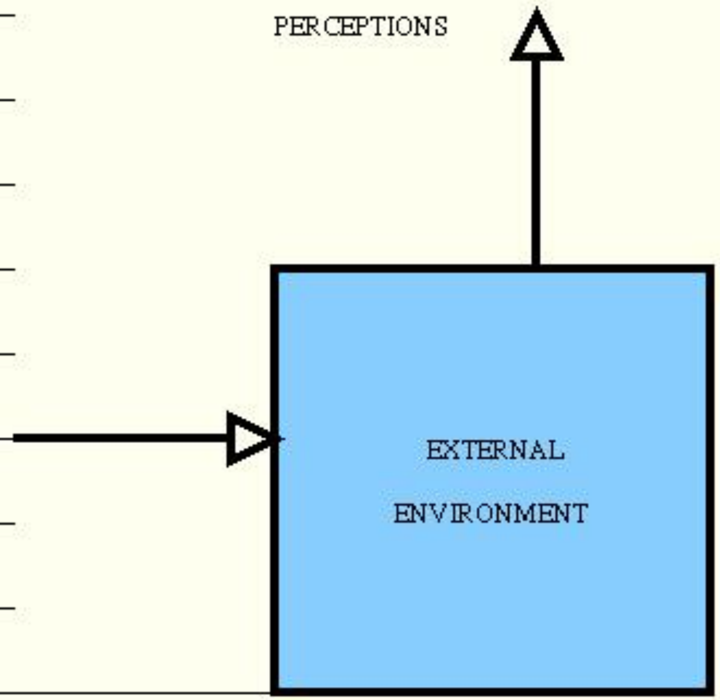
INTERNAL STATE

	0	1	2	3	4	5	6	7	8	9	10	11	12
ADDRESSES	0	1	2	3	4	5	6	7	8	9	10	11	12
CONTENTS	5321	−44	810	−2	−3322	5	7	3	0	−189	2	237	6

INSTRUCTION POINTER

PARAMETERS

0 = ADD(a1, a2, a3)	0.001	0.0014	0.9	0.24	0.001	0.0014	0.9	0.9
1 = MUL(a1, a2, a3)	0.001	0.0014	0.04	0.01	0.001	0.0014	0.04	0.04
2 = SUB(a1, a2, a3)	0.99	0.0014	0.01	0.01	0.99	0.0014	0.01	0.01
3 = JMPLEQ(a1, a2, a3)	0.001	0.99	0.01	0.01	0.001	0.99	0.01	0.01
4 = MOVEAGENT(a1, a2)	0.001	0.0014	0.01	0.7	0.001	0.0014	0.01	0.01
5 = InvokeSSA()	0.004	0.0014	0.01	0.01	0.004	0.0014	0.01	0.01
6 = INCPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01
7 = DECPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01



VARIABLE PROBABILITY DISTRIBUTIONS

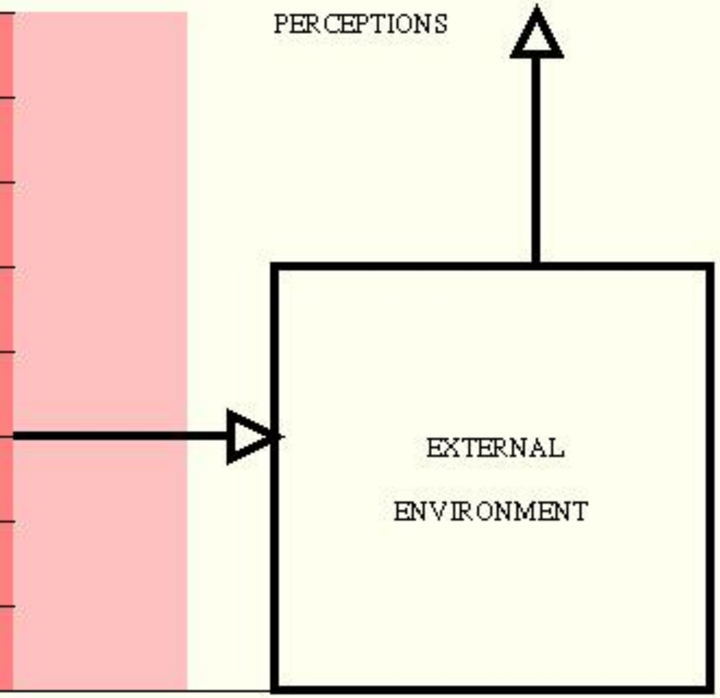
INTERNAL STATE

ADDRESSES	0	1	2	3	4	5	6	7	8	9	10	11	12
CONTENTS	5321	−44	810	−2	−3322	5	7	3	0	−189	2	237	6

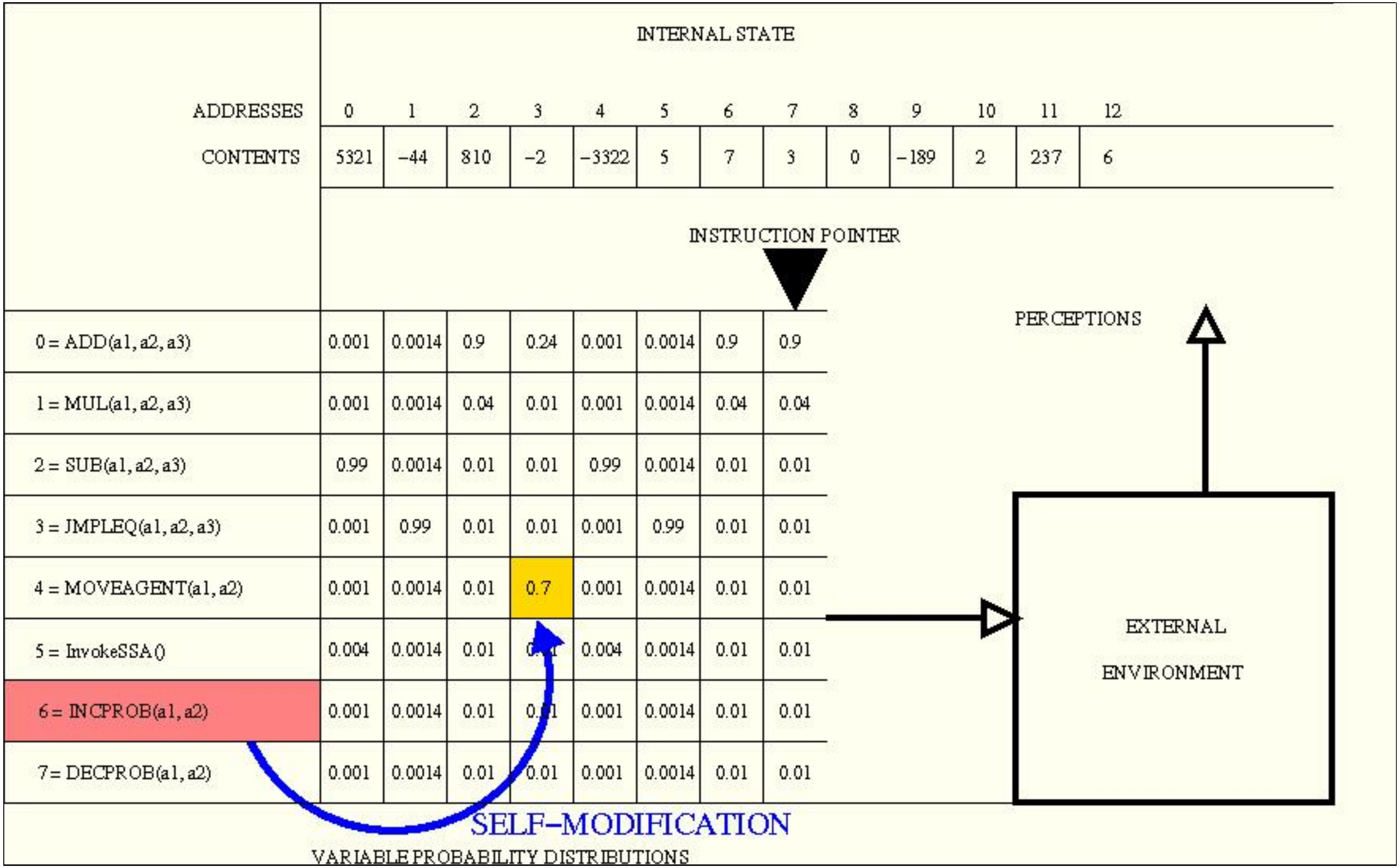
INSTRUCTION POINTER

PARAMETERS

0 = ADD(a1, a2, a3)	0.001	0.0014	0.9	0.24	0.001	0.0014	0.9	0.9
1 = MUL(a1, a2, a3)	0.001	0.0014	0.04	0.01	0.001	0.0014	0.04	0.04
2 = SUB(a1, a2, a3)	0.99	0.0014	0.01	0.01	0.99	0.0014	0.01	0.01
3 = JMPLEQ(a1, a2, a3)	0.001	0.99	0.01	0.01	0.001	0.99	0.01	0.01
4 = MOVEAGENT(a1, a2)	0.001	0.0014	0.01	0.7	0.001	0.0014	0.01	0.01
5 = InvokeSSA()	0.004	0.0014	0.01	0.01	0.004	0.0014	0.01	0.01
6 = INCPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01
7 = DECPROB(a1, a2)	0.001	0.0014	0.01	0.01	0.001	0.0014	0.01	0.01



VARIABLE PROBABILITY DISTRIBUTIONS

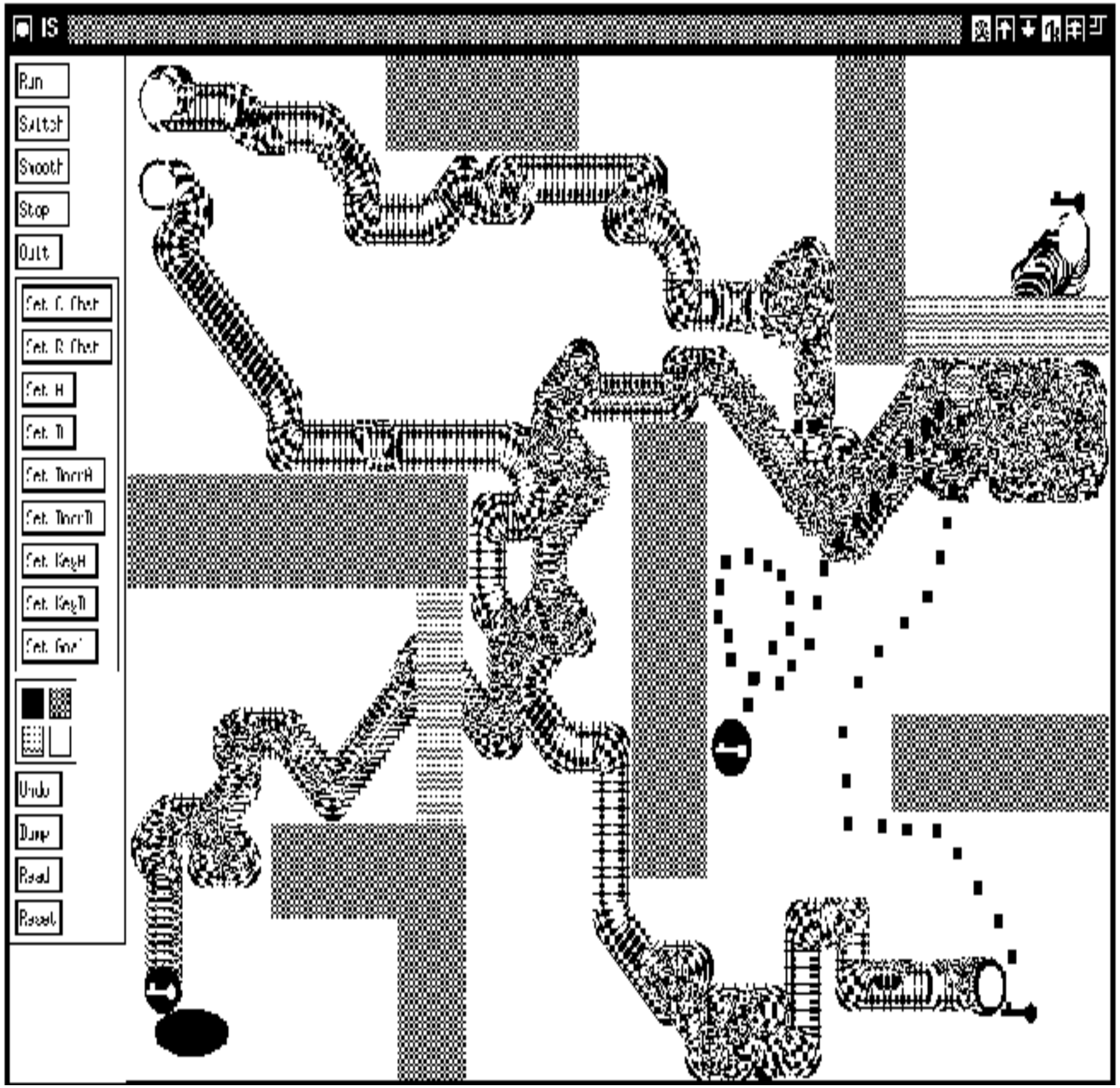

INTERNAL STATE
ADDRESSES 0 1 2 3 4 5 6 7 8 9 10 11 12
CONTENTS 5321 −44 810 −2 −3322 5 7 3 0 −189 2 237 6
INSTRUCTION POINTER
0 = ADD(a1, a2, a3) 0.001 0.0014 0.9 0.24 0.001 0.0014 0.9 0.9
1 = MUL(a1, a2, a3) 0.001 0.0014 0.04 0.01 0.001 0.0014 0.04 0.04
2 = SUB(a1, a2, a3) 0.99 0.0014 0.01 0.01 0.99 0.0014 0.01 0.01
3 = JMPLEQ(a1, a2, a3) 0.001 0.99 0.01 0.01 0.001 0.99 0.01 0.01
4 = MOVEAGENT(a1, a2) 0.001 0.0014 0.01 0.7 0.001 0.0014 0.01 0.01
5 = InvokeSSA() 0.004 0.0014 0.01 0.01 0.004 0.0014 0.01 0.01
6 = INCPROB(a1, a2) 0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01
7 = DECPROB(a1, a2) 0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01
PERCEPTIONS
EXTERNAL
ENVIRONMENT
SELF−MODIFICATION
VARIABLE PROBABILITY DISTRIBUTIONS

Success-story algorithm (SSA) for self-modifying code (since 1994)

J. Schmidhuber. On learning how to learn learning strategies. TR FKI-198-94, 1994.

R(t): Reward until time t. Stack of past check points $v_1 v_2 v_3$ … with self-mods in between. SSA undoes selfmods after v_i that are not followed by long-term reward acceleration up until t (now):

$R(t)/t <$
$[R(t)-R(v_1)] / (t-v1) <$
$[R(t)-R(v_2)] / (t-v_2) <$…



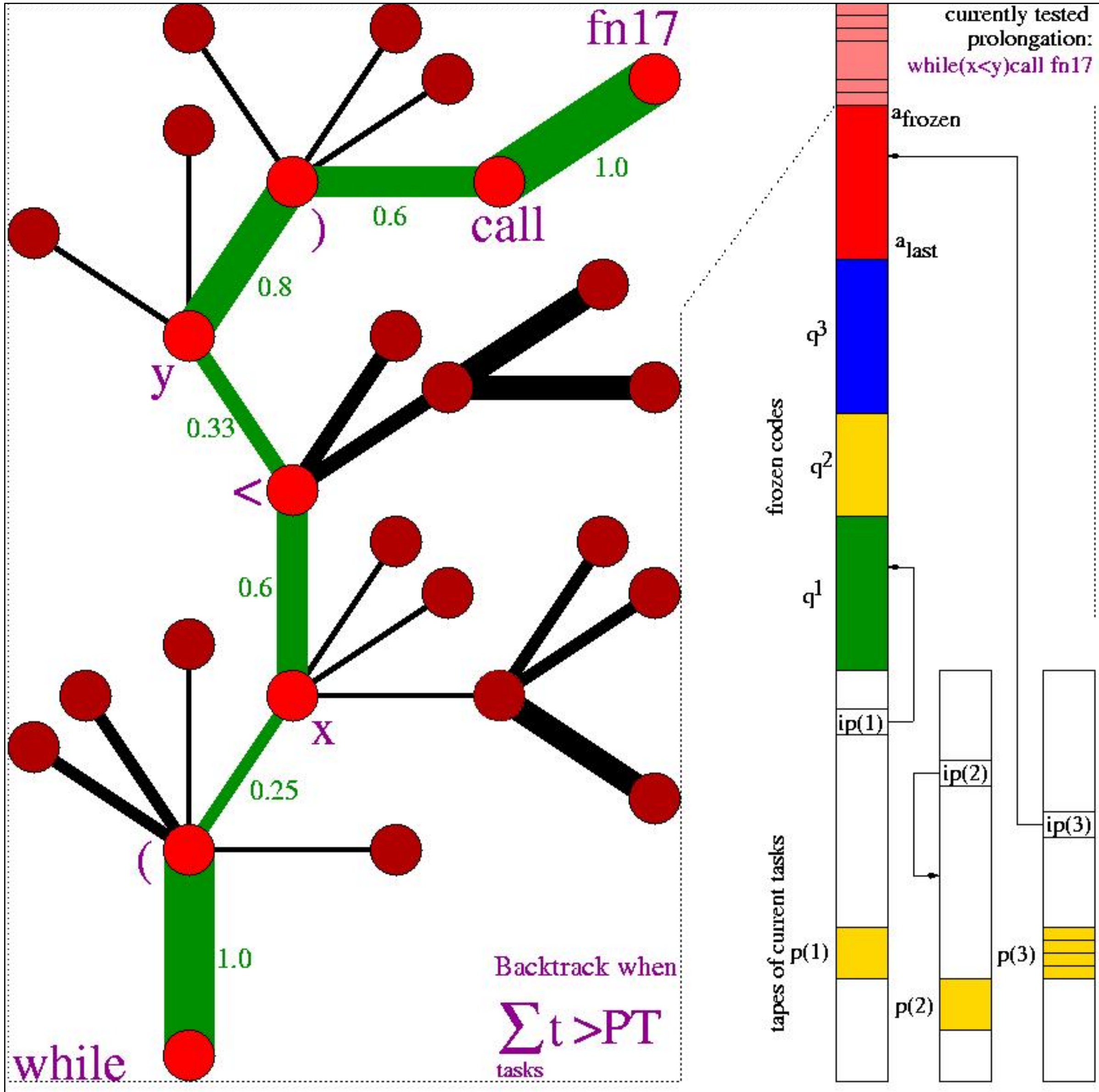
1997: Lifelong meta-RL with self-modifying policies and success-story algorithm: 2 agents, 2 doors, 2 keys. 1st southeast wins 5, the other 3. Through recursive self-modifications only: from 300,000 steps per trial down to 5,000.

Universal Search: run all programs until one of them finds and verifies a solution, where a program of k bits gets 2^{-k} of total search time

Leonid Levin 1973

Fastest solver for given problem class, save for constant factor

$O(n^3)10^{100} = O(n^3)$

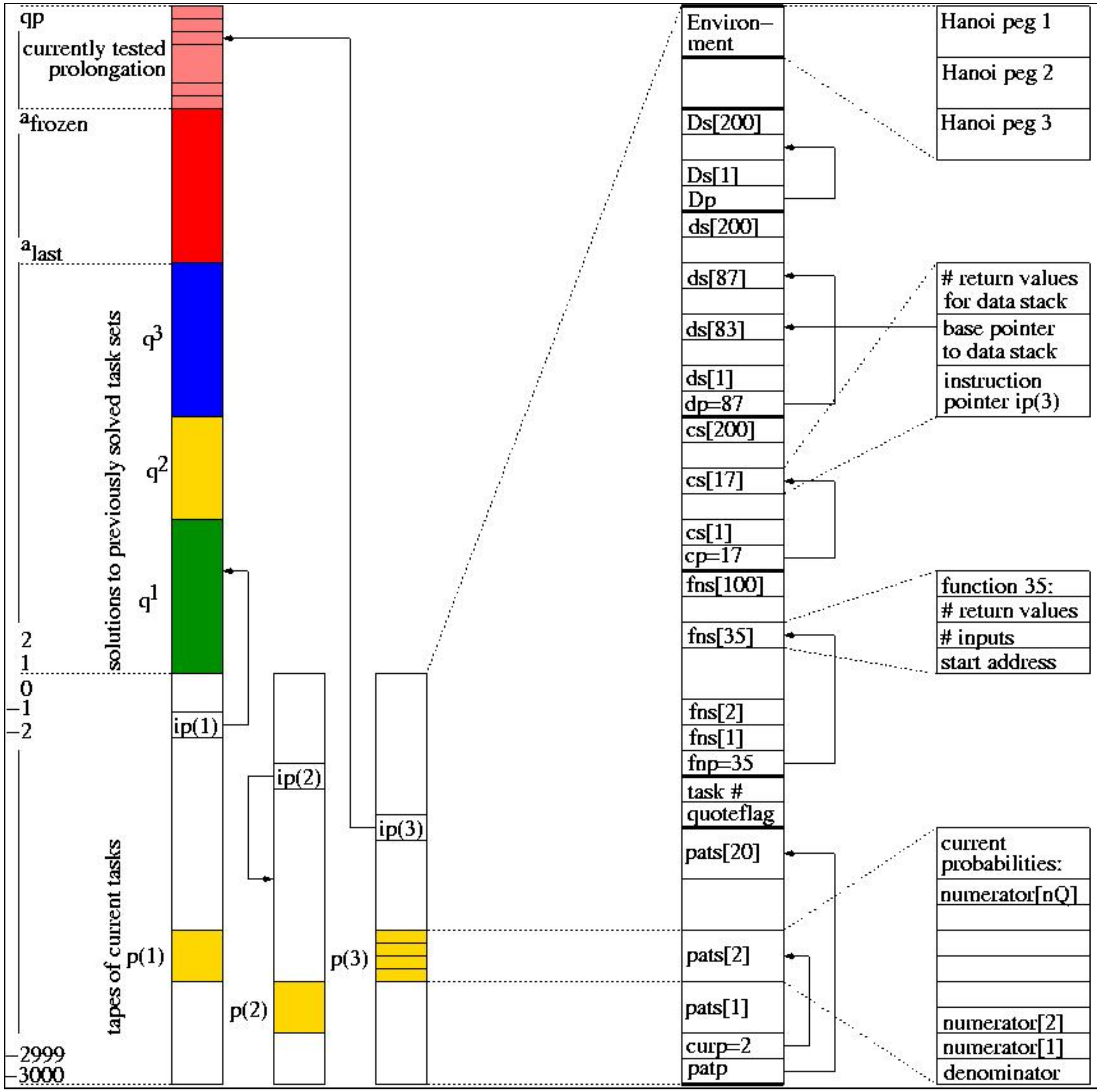


Asymptotically optimal curriculum learner: Optimal Ordered Problem Solver OOPS (Schmidhuber, MLJ, 2004, extending Levin's universal search, 1973)

Time-optimal incremental search and algorithmic transfer learning in program space

Branches of search tree are program prefixes

Node-oriented backtracking restores partially solved task sets & modified memory components on error or when $\sum t > PT$



61 primitive instructions operating on stack-like and other internal data structures. For example:

push1(), not(x), inc(x), add(x,y), div(x,y), or(x,y), exch_stack(m,n), push_prog(n), movstring(a,b,n), delete(a,n), find(x), define function(m,n), callfun(fn), jumpif(val,address), quote(), unquote(), boost_probability(n,val) ….

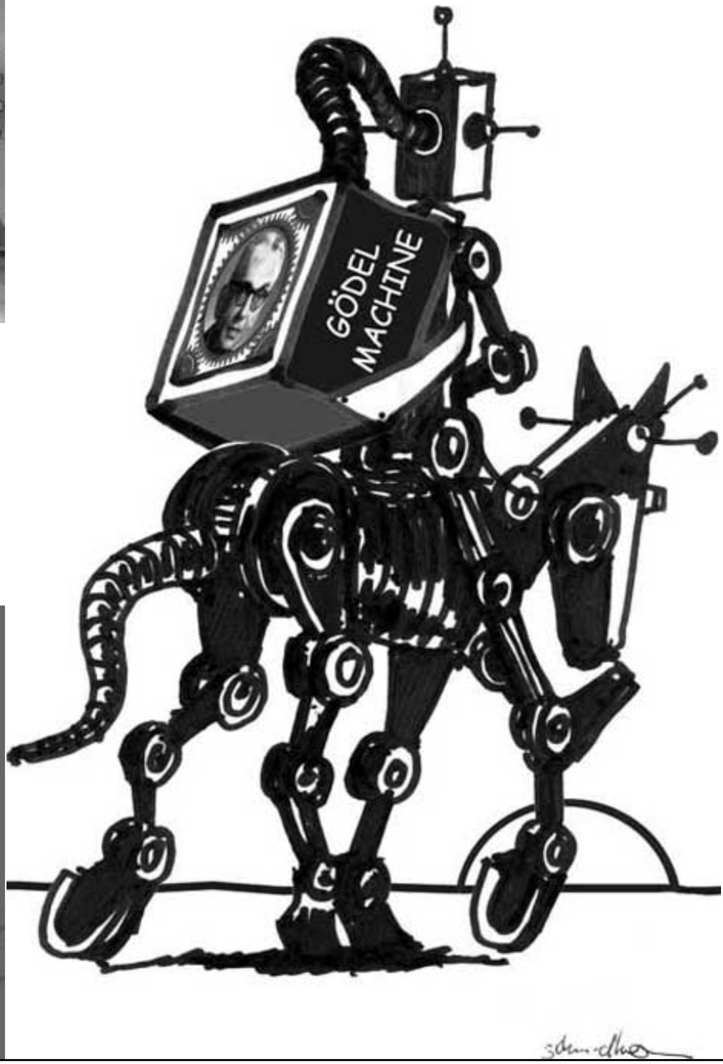
Programs are integer sequences; data and code look the same; makes functional programming easy

Towers of Hanoi: incremental solutions

- +1ms, n=1: *(movdisk)*
- 1 day, n=1,2: *(c4 c3 cpn c4 by2 c3 by2 exec)*
- 3 days, n=1,2,3: *(c3 dec boostq defnp c4 calltp c3 c5 calltp endnp)*
- 4 days: n=4, n=5, …, n=30: *by same double-recursive program*
- Profits from 30 earlier context-free language tasks (1^n2^n)*: transfer learning*
- 93,994,568,009 prefixes tested
- 345,450,362,522 instructions
- 678,634,413,962 time steps
- longest single run: 33 billion steps (5% of total time)! Much deeper than recent memory-based "deep learners" …
- top stack size for restoring storage: < 20,000

What the found Towers of Hanoi solver does:

- *(c3 dec boostq defnp c4 calltp c3 c5 calltp endnp)*
- Prefix increases P of double-recursive procedure:
  Hanoi(Source,Aux,Dest,n): IF n=0 exit; ELSE BEGIN
  Hanoi(Source,Dest,Aux,n-1); move top disk from Aux to Dest;
  Hanoi(Aux,Source,Dest,n-1); END
- Prefix boosts instructions of previoulsy frozen program, which happens to be a previously learned solver of a context-free language (1^n2^n). This rewrites search procedure itself: Benefits of metalearning!
- Prefix probability 0.003; suffix probability $3*10^{-8}$; total probability $9*10^{-11}$
- Suffix probability without prefix execution: $4*10^{-14}$
- That is, Hanoi does profit from 1^n2^n experience and incremental learning (OOPS excels at algorithmic transfer learning): speedup factor 1000

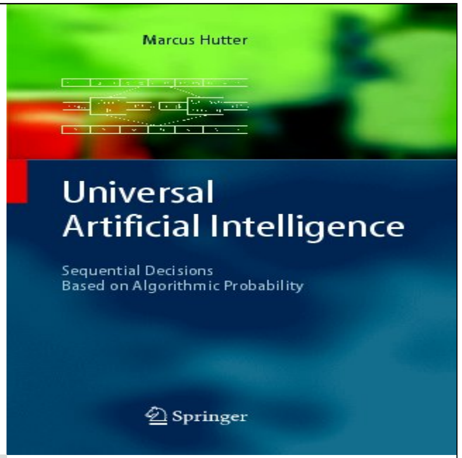


Gödel Machine (2003): agent-controlling program that speaks about itself, ready to rewrite itself in arbitrary fashion once it has found a proof that the rewrite is useful, given a user-defined utility function

Theoretically optimal self-improver!

Initialize Gödel Machine by Marcus Hutter's asymptotically fastest method for all well-defined problems

IDSIA 2002 on my SNF grant



Given $f: X \rightarrow Y$ and $x \in X$, search proofs to find program q that provably computes f(z) for all $z \in X$ within time bound $t_q(z)$; spend most time on f(x)-computing q with best current bound

$n^3+10^{1000}=n^3+O(1)$

As fast as fastest f-computer, save for factor $1+\varepsilon$ and f-specific const. independent of x!

Separation of Storage and Control (Zuse 1936) for NNs: End-to-End-Differentiable Neural Stack Machines (Das, Giles, Mike Mozer, 1992), NTM & DNC (Graves et al 2014-16) & Memory Nets (Weston et al 2014)

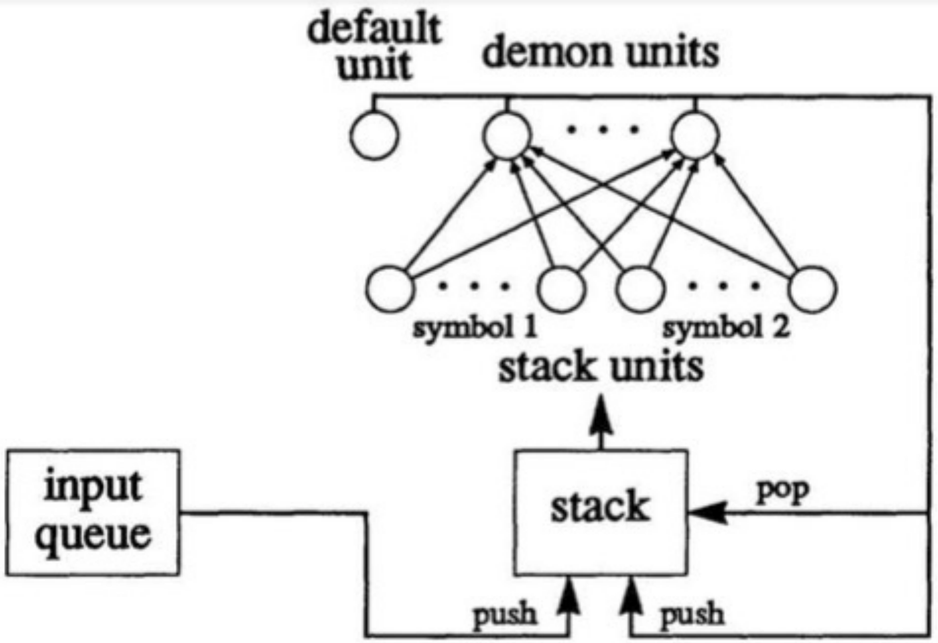


Figure 1: The demon model.

Neural stack machine of 1992-1993

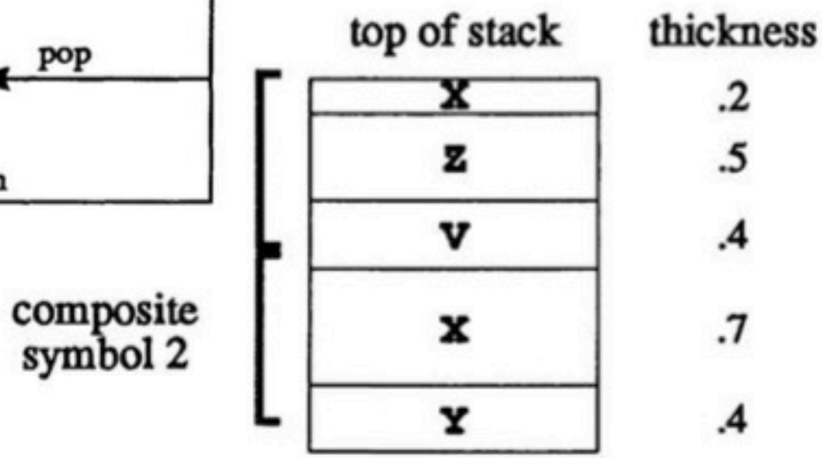
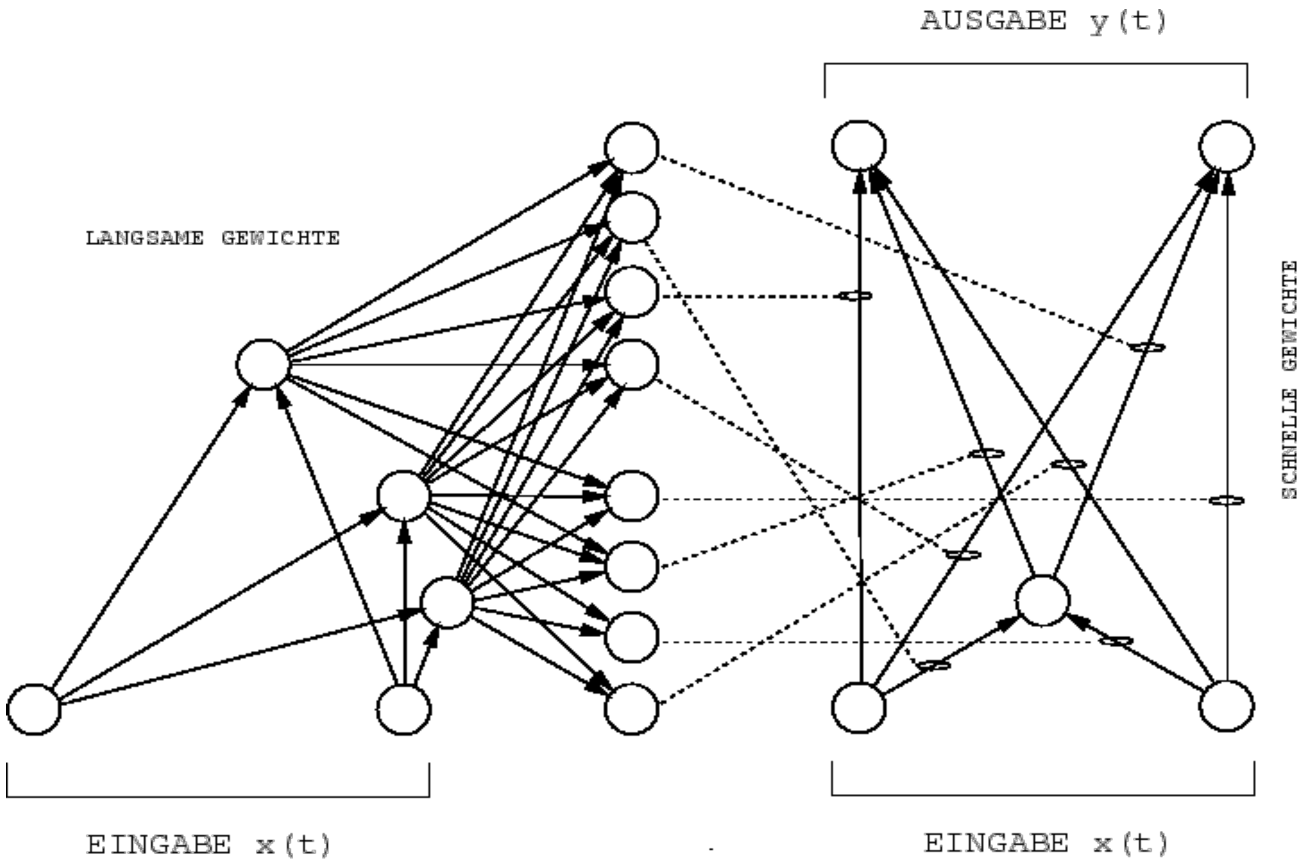
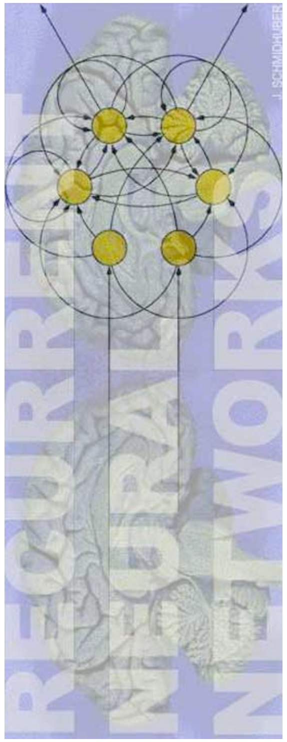
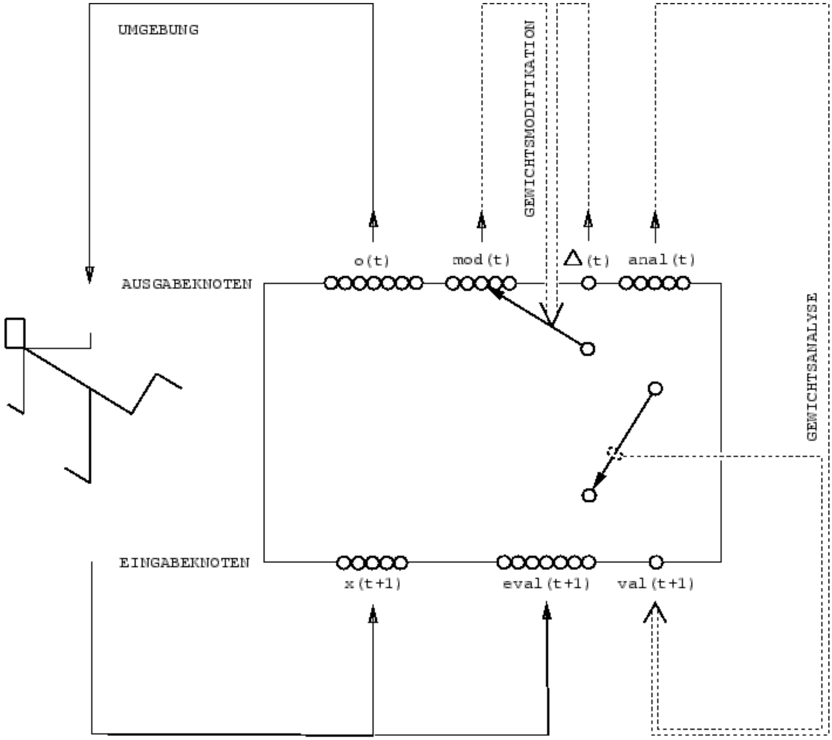


Figure 3: A continuous stack. The symbols indicate the contents; the height of a stack entry indicates its thickness, also given by the number to the right. The top composite symbol on the stack is a combination of the items forming a total thickness of 1.0; the next composite symbol is a combination of the items making up the next 1.0 units of thickness.

Looks a bit like supervised L2L but is not yet: Separation of Storage and Control for NNs: End-to-End Differentiable Fast Weights (Schmidhuber, 1992) extending v.d. Malsburg's non-differentiable dynamic links (1981)



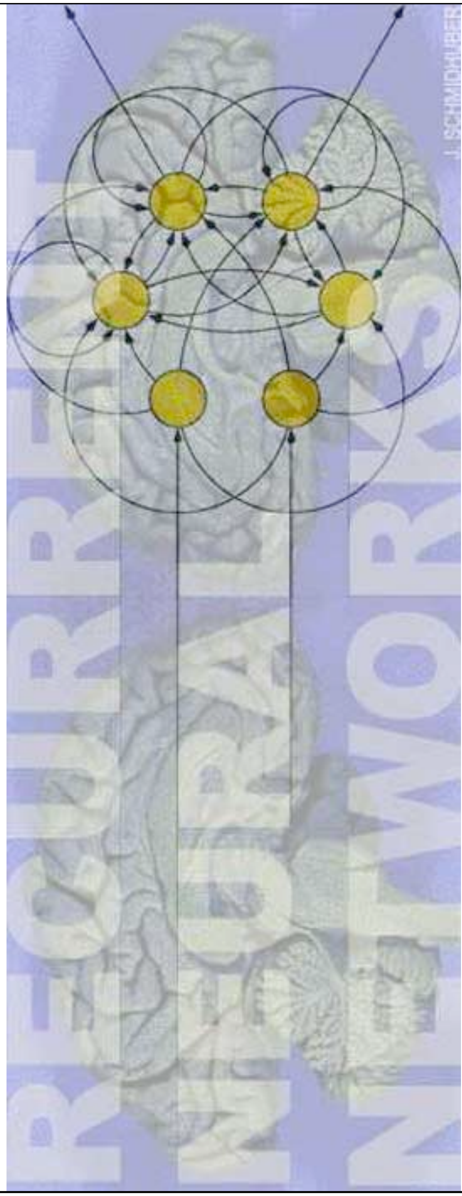




1992-1993: Gradient-based meta-RNNs that can learn to run their own weight change algorithm, e.g.: J. Schmidhuber. A self-referential weight matrix. ICANN 1993. Based on TR at U Colorado, 1992.

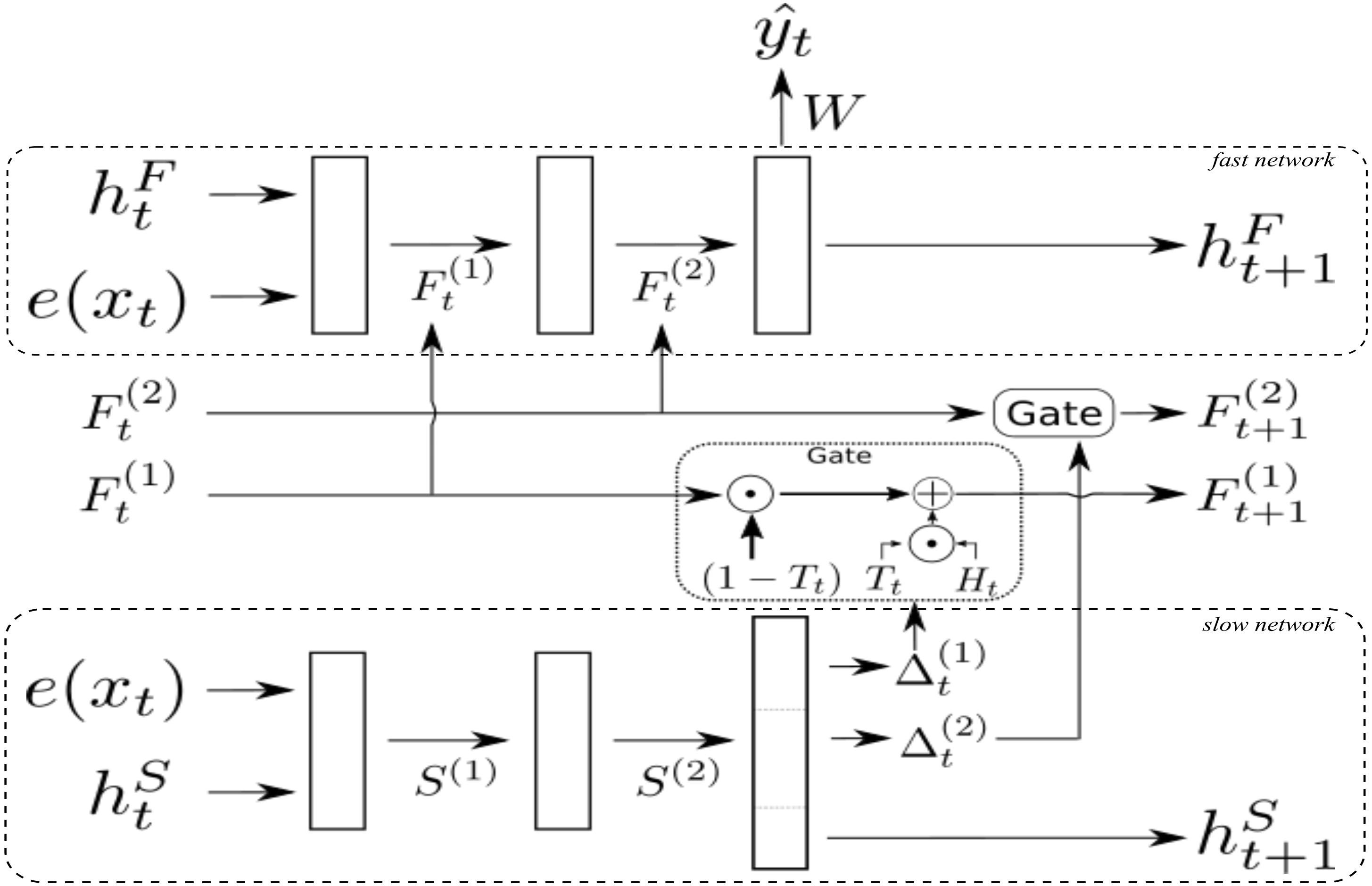
An RNN, but no LSTM yet. In 2001, however, Sepp Hochreiter taught a meta-LSTM to learn a learning algorithm for quadratic functions that was faster than backprop

1993: More elegant Hebb-inspired addressing to go from (#hidden) to $(\#\text{hidden})^2$ temporal variables: gradient-based RNN learns to control internal end-to-end differentiable spotlights of attention for fast differentiable memory rewrites – again fast weights



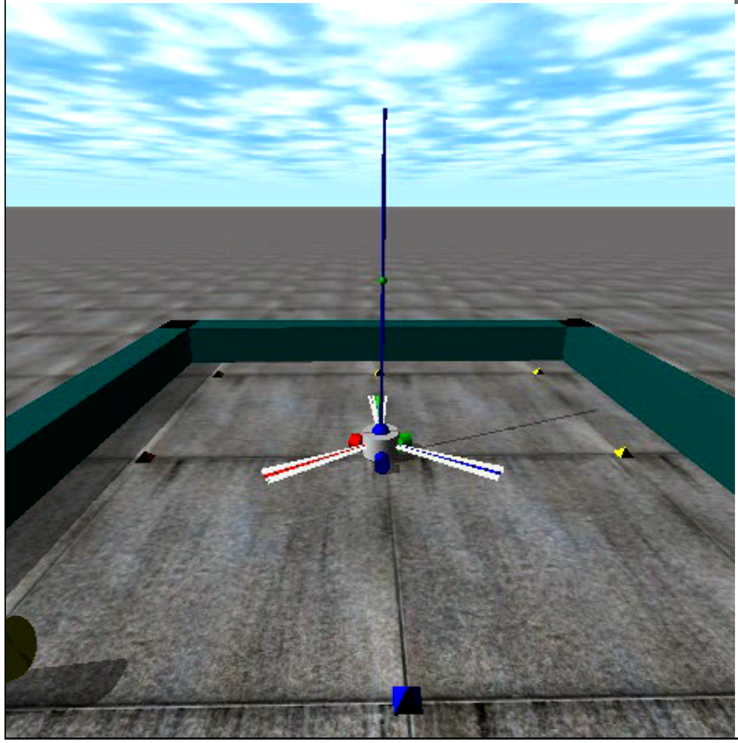
Schmidhuber, ICANN 1993: Reducing the ratio between learning complexity and number of time-varying variables in fully recurrent nets.

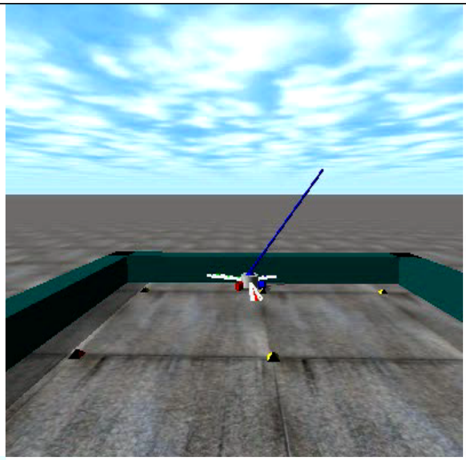
Similar NIPS 2016 paper by Ba et al. See I. Schlag at NIPS Metalearning Symposium 2017!



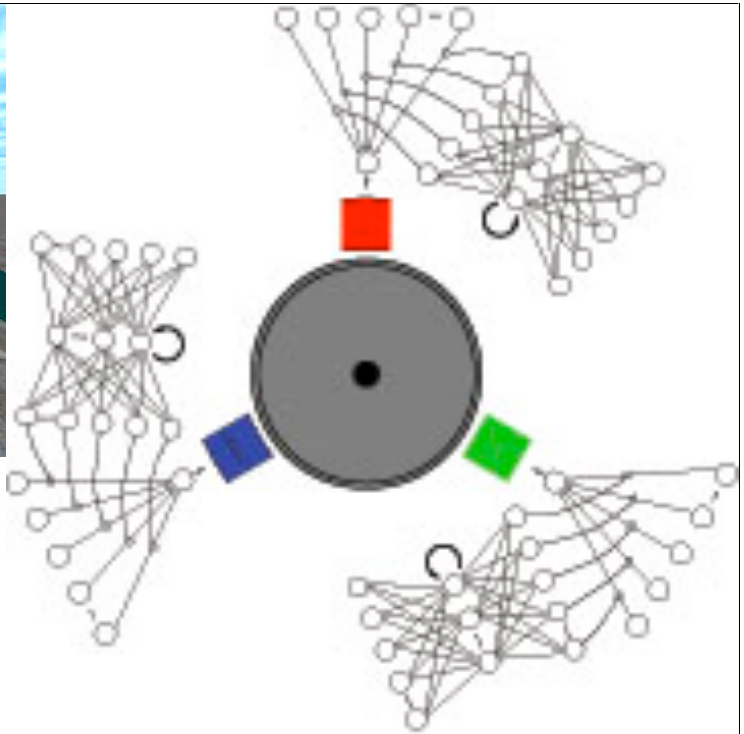
New fast weight addressing scheme: Imanol Schlag @ NIPS Meta-learning Workshop 2017

2005: Reinforcement-Learning or Evolving RNNs with Fast Weights





Robot learns to balance 1 or 2 poles through 3D joint

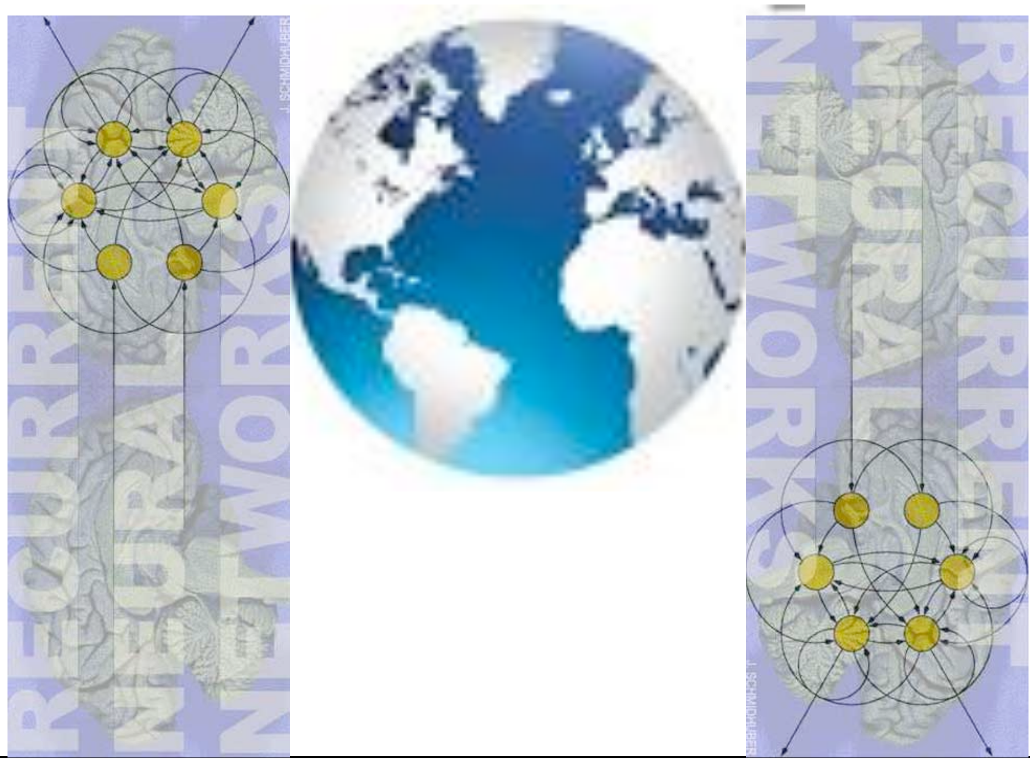


Gomez & Schmidhuber: Co-evolving recurrent neurons learn deep memory POMDPs. GECCO 2005

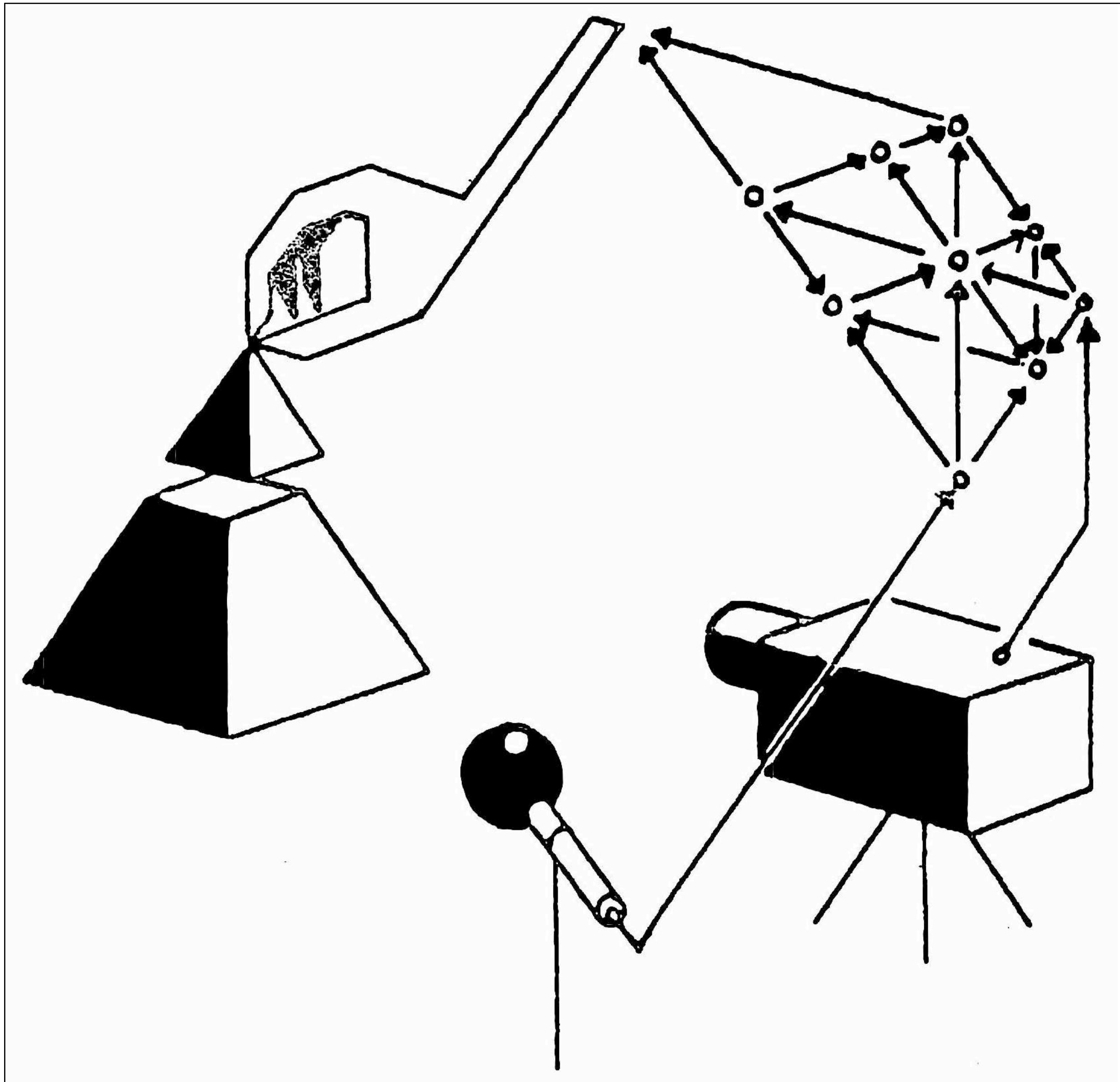
http://www.idsia.ch/~juergen/evolution.html

1. Schmidhuber. Evolutionary principles in self-referential learning, or on learning how to learn: The meta-meta-... hook. Diploma thesis, TUM, 1987. (First concrete RSI.)
2. Schmidhuber. A self-referential weight matrix. ICANN 1993. Based on TR CU-CS-627-92, Univ. Colorado, 1992. (Supervised gradient-based RSI.)
3. Schmidhuber. On learning how to learn learning strategies. TR FKI-198-94, 1994. (RL)
4. Schmidhuber and J. Zhao and M. Wiering. Simple principles of metalearning. TR IDSIA-69-96, 1996. (Meta-RL and RSI based on 3.)
5. Schmidhuber, J. Zhao, N. Schraudolph. Reinforcement learning with self-modifying policies. In *Learning to learn*, Kluwer, pages 293-309, 1997. (Meta-RL based on 3.)
6. Schmidhuber, J. Zhao, and M. Wiering. Shifting inductive bias with success-story algorithm, adaptive Levin search, and incremental self-improvement. Machine Learning 28:105-130, 1997. (Partially based on 3.)
7. Schmidhuber. Gödel machines: Fully Self-Referential Optimal Universal Self-Improvers. In *Artificial General Intelligence*, p. 119-226, 2006. (Based on TR of 2003.)
8. T. Schaul and Schmidhuber. Metalearning. Scholarpedia, 5(6):4650, 2010.
9. More under http://people.idsia.ch/~juergen/metalearner.html

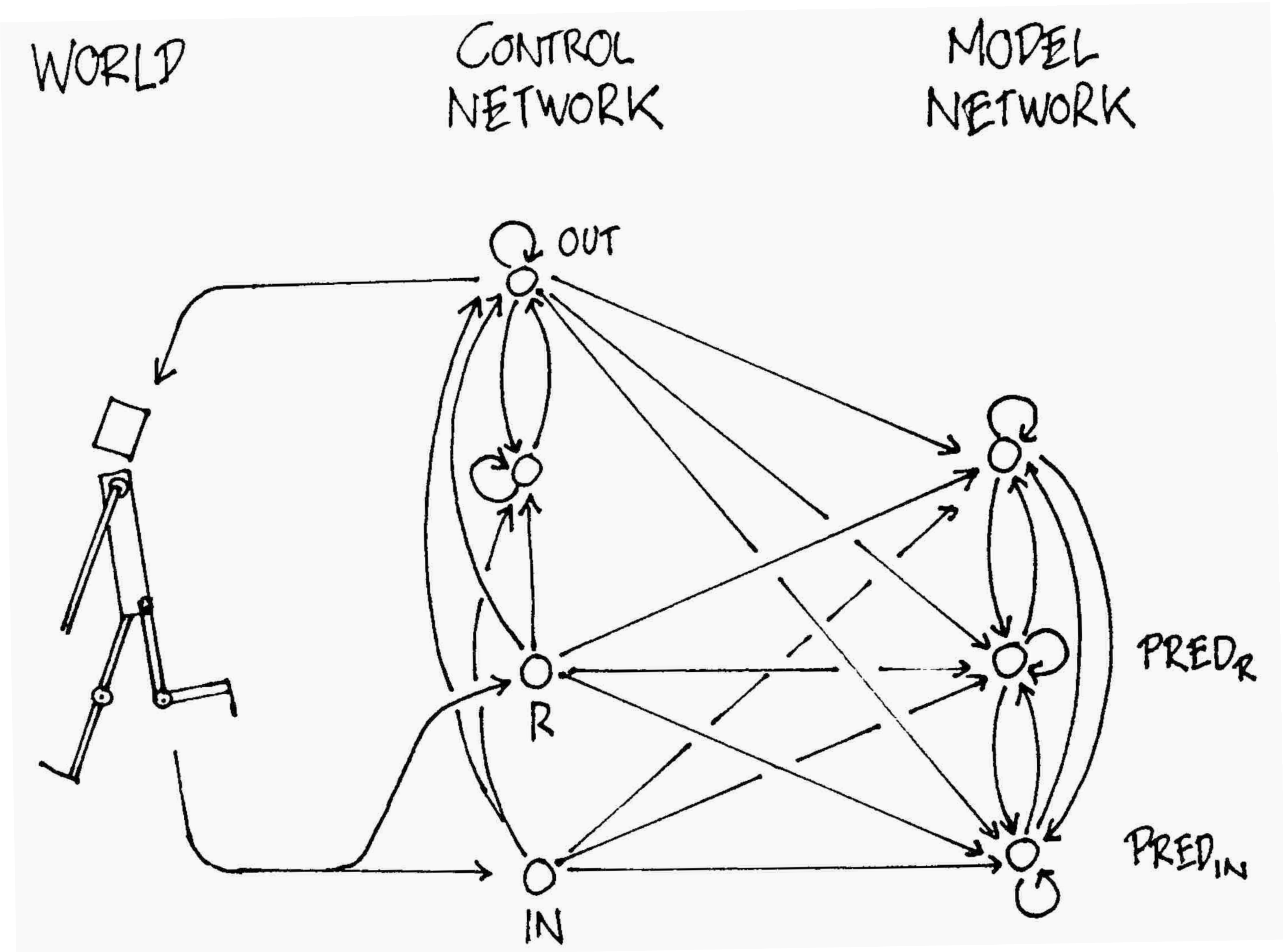
IJCNN 1990, NIPS 1991: Reinforcement Learning & Planning with RNN Controller & RNN World Model



A bit like universal AIXI, but with feasible local search



My old drawings from: Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments. J. Schmidhuber, 1990.



From: Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments. J. Schmidhuber, 1990.

IJNS 1991: R-Learning of Visual Attention on 100,000 times slower computers

http://people.idsia.ch/~juergen/attentive.html

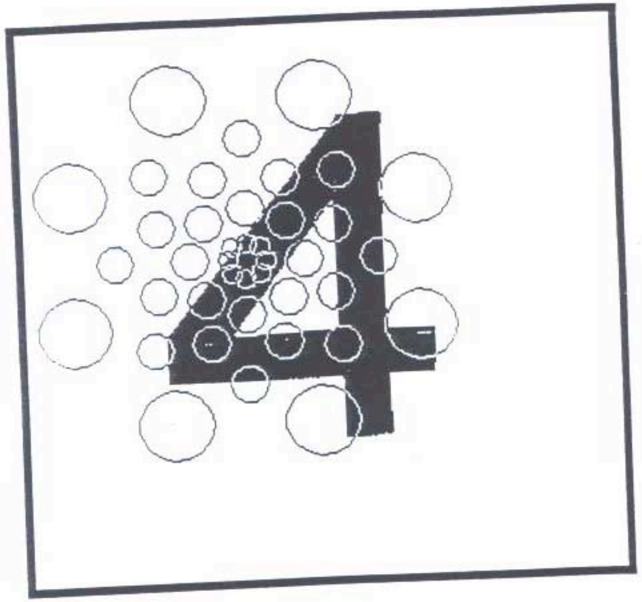


Fig. 1. A typical visual scene. The diameters of the receptive fields of the retina's input units are indicated by circles.

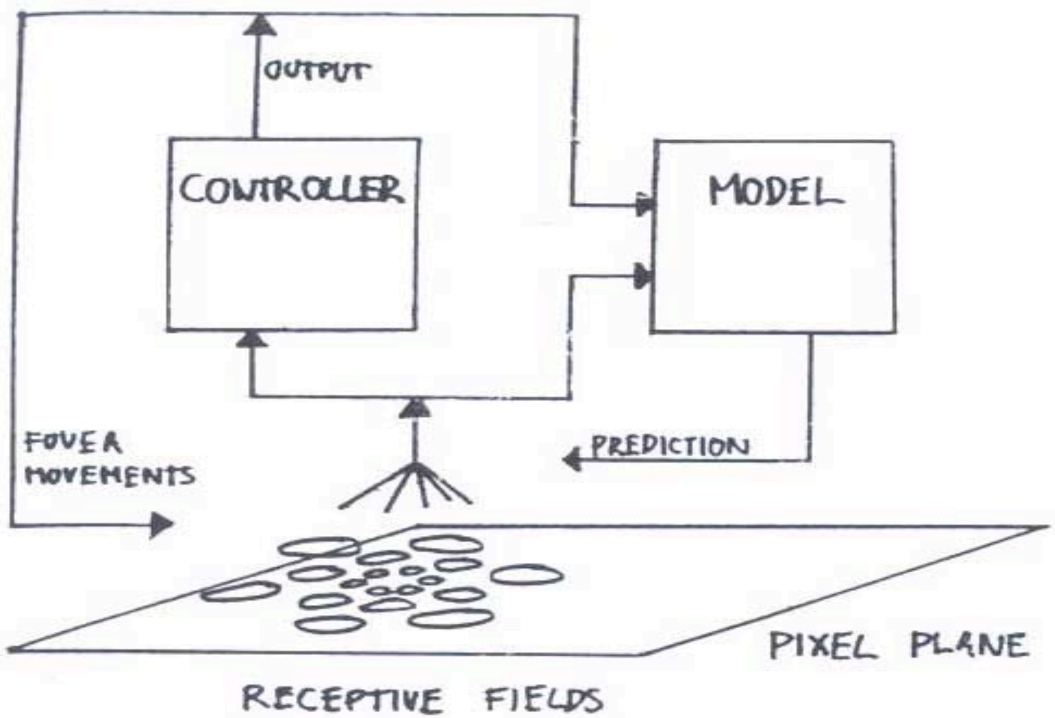


Fig. 2. An artificial fovea provides inputs for a control network which is able to move the fovea around. A model network is trained to predict the next input from the current input and the current controller action.

1991: current goal=extra fixed input
2018: all of this is coming back!

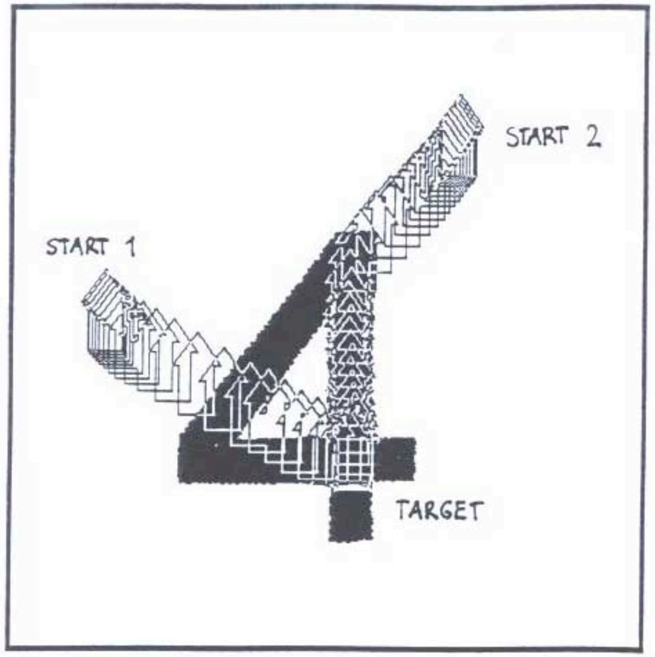


Fig. 4. Translations: Examples of fovea trajectories leading

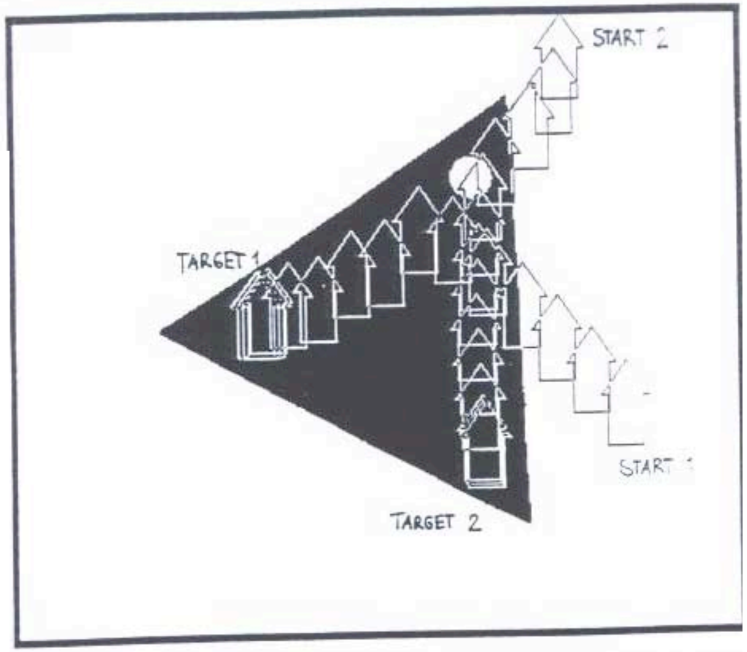
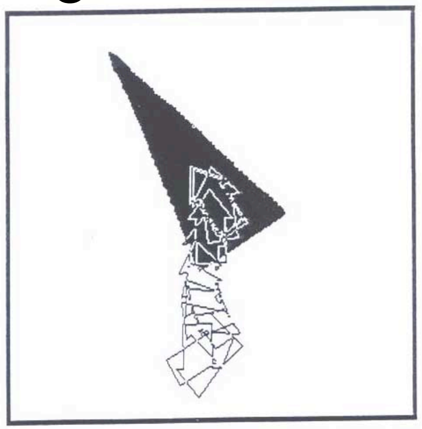
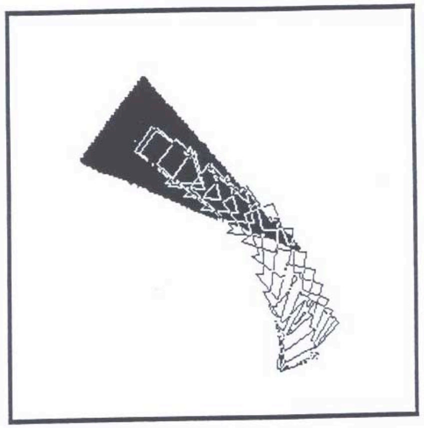


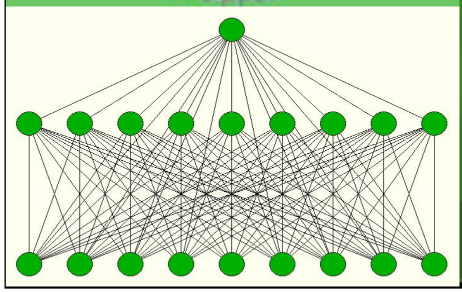
Fig. 5. One controller for various targets specified by an additional constant input: Examples of fovea trajectories leading from various start positions to different targets. The first target is near the left corner of the triangle. The second target is near the lower corner.

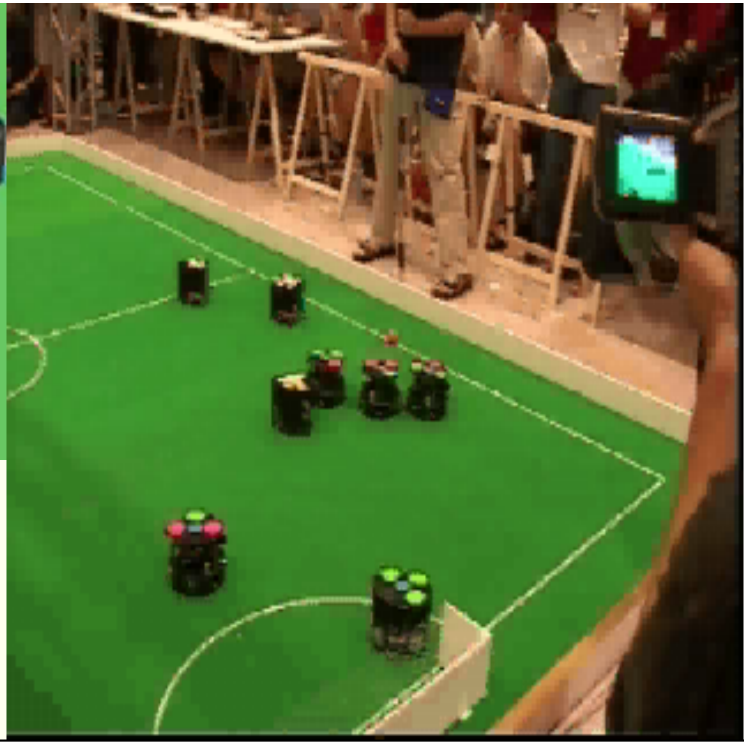




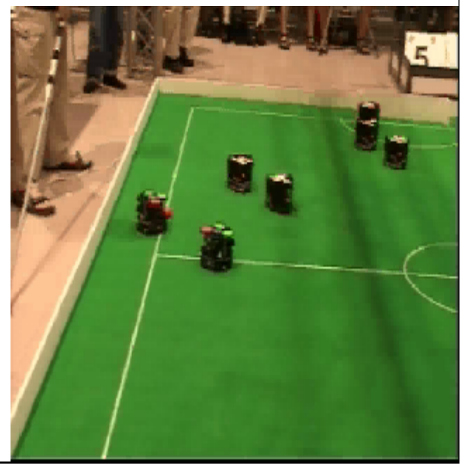
RoboCup World Champion 2004, Fastest League, 5m/s

Lookahead expectation & planning with neural networks (Schmidhuber, IEEE INNS 1990): successfully used for RoboCup by Alexander Gloye-Förster (went to IDSIA)
http://www.idsia.ch/~juergen/learningrobots.html

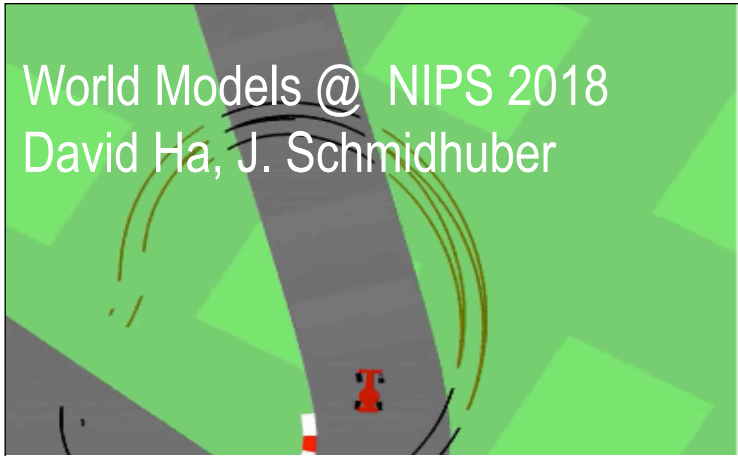




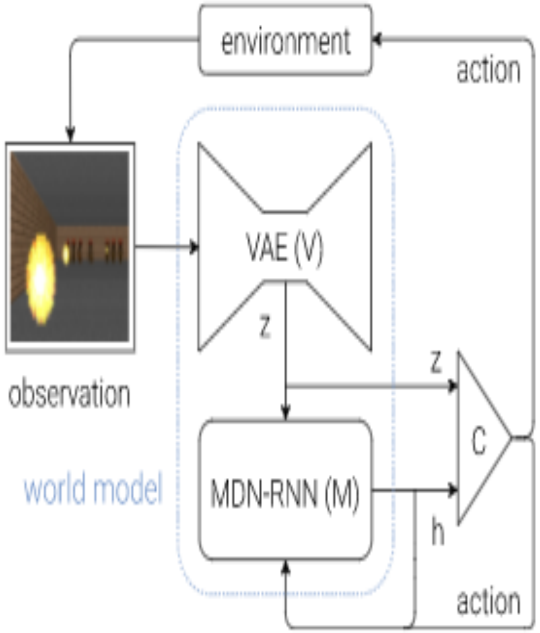
Alex @ IDSIA, led FU Berlin's RoboCup World Champion Team 2004



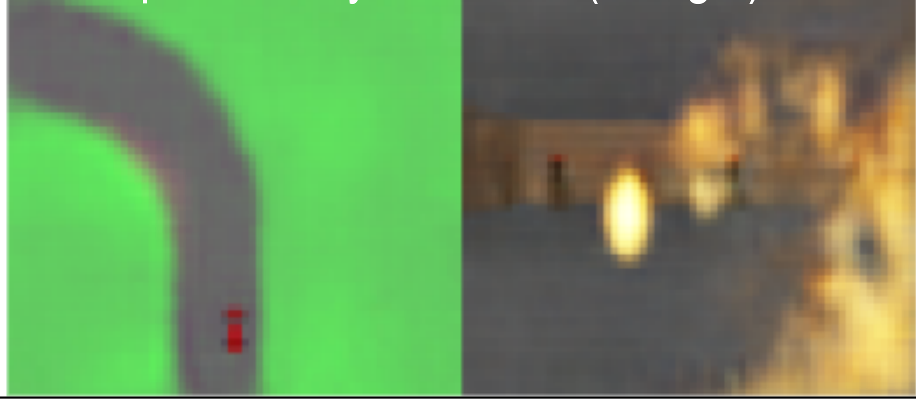
World Models @ NIPS 2018
David Ha, J. Schmidhuber



Train agent inside of its own hallucinated dream generated by its world model, and transfer policy back into actual environment

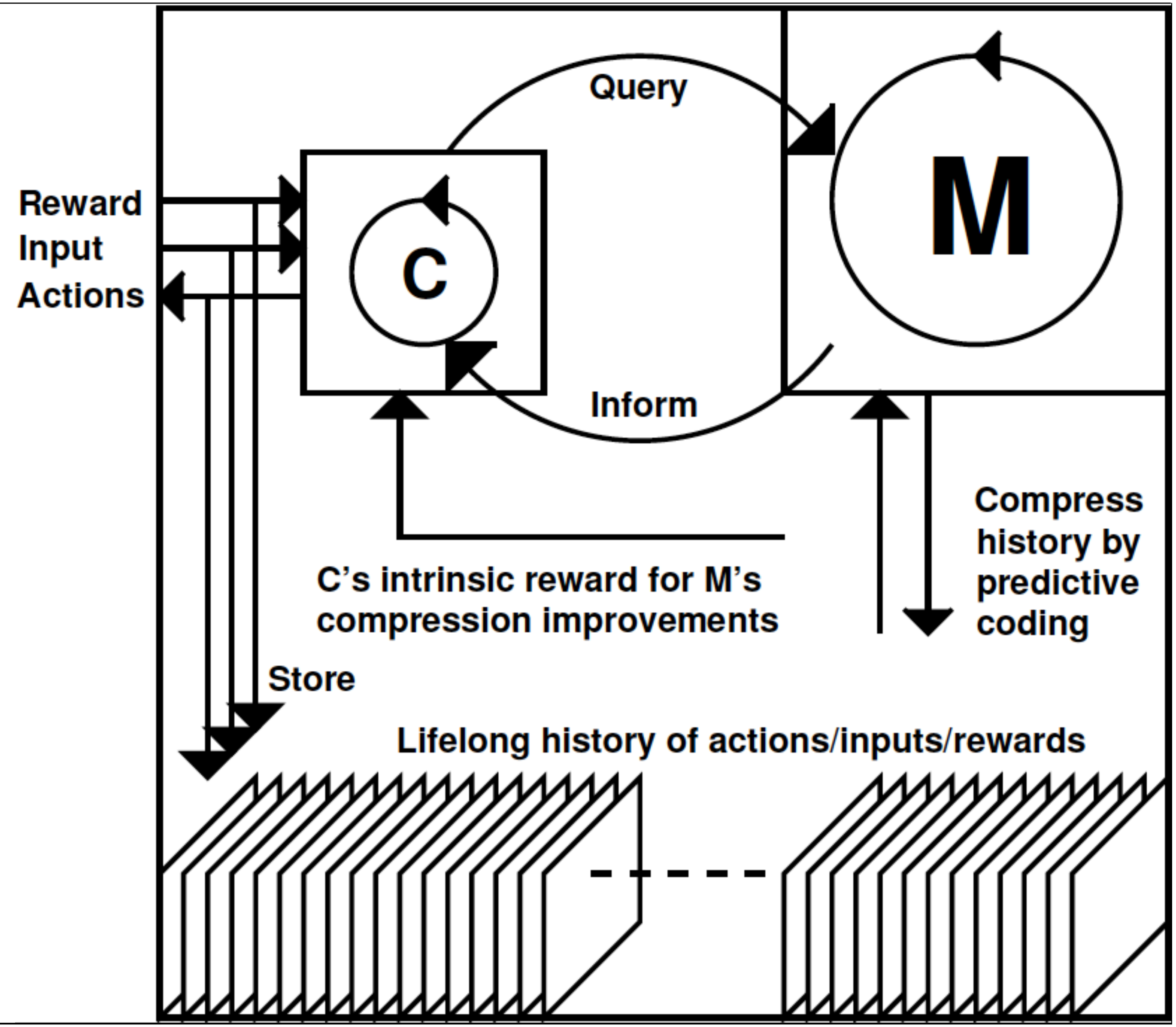


Made possible by David Ha (Google)

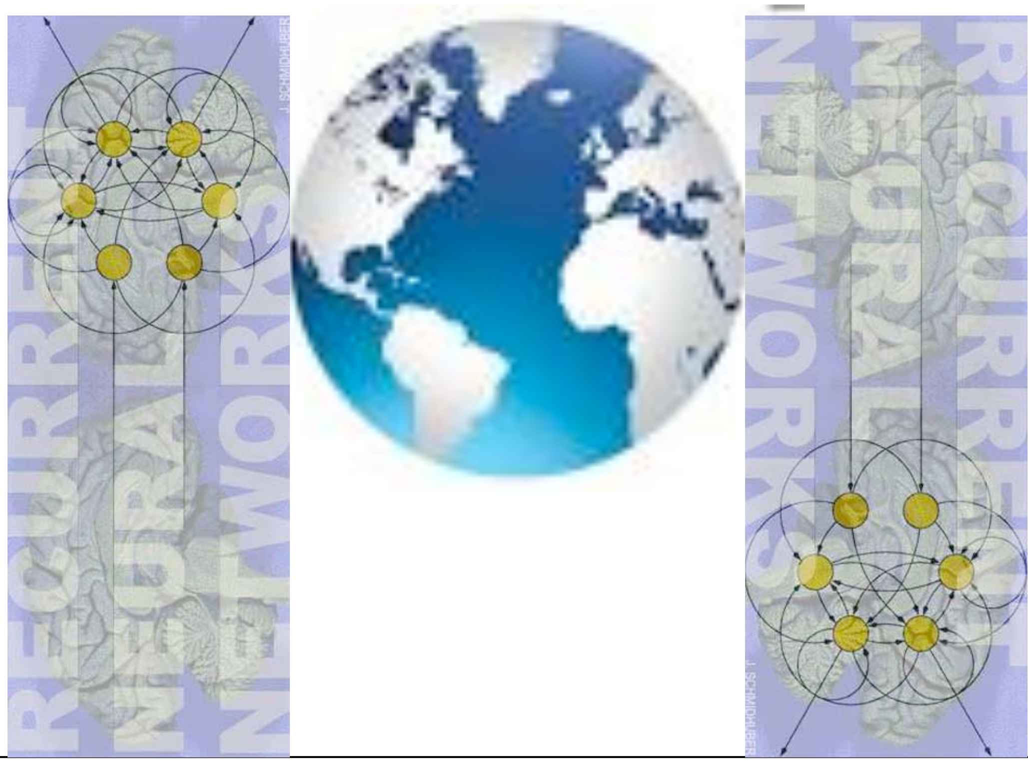


RNNAIssance 2014-2015 On Learning to Think: Algorithmic Information Theory for Novel Combinations of Reinforcement Learning RNN-based Controllers (RNNAIs) and Recurrent Neural World Models

http://arxiv.org/abs/1511.09249



How to motivate the controller to improve the world model?

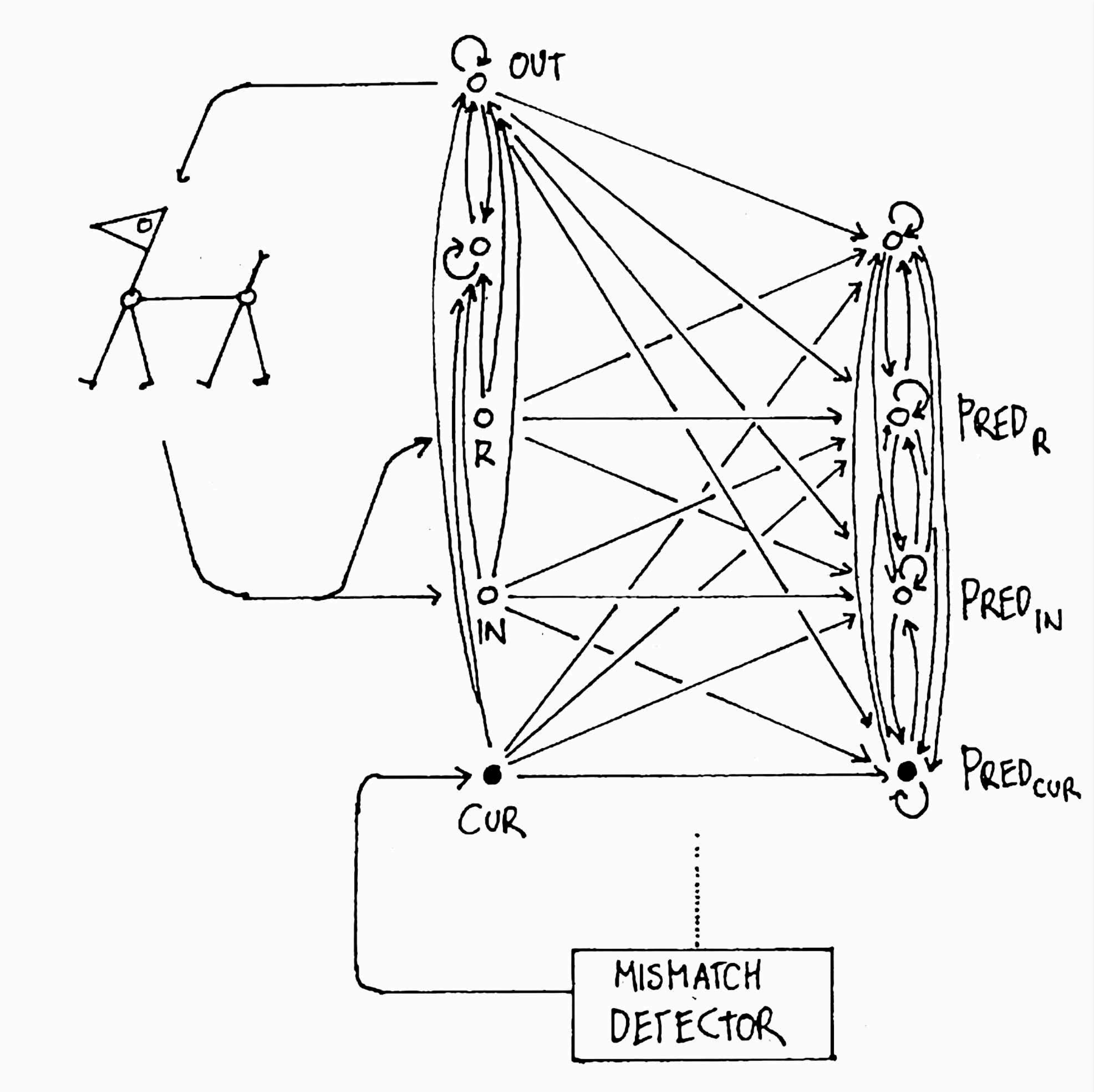


1990s: UNSUPERVISED NEURAL NETS FIGHT EACH OTHER IN A MINIMAX GAME
EACH NET MINIMIZES THE VALUE FUNCTION MAXIMIZED BY THE OTHER
TO LEARN A MODEL OF THE PROBABILITY DISTRIBUTION ON GIVEN DATA

OR TO GENERATE EXPERIMENTS YIELDING INTRINSIC REWARD FOR CURIOSITY

1990: Active Unsupervised Minimax for RL

Adversarial Reinforcement Learning (RL) for agents with Artificial Curiosity (1990): A reward-maximising neural control network C learns to generate action sequences or *experiments* in an environment. It gets intrinsic reward in proportion to the prediction errors of a separate neural network called the world model M. M learns to predict future inputs, given past inputs and actions. Again, in the absence of external reward, C is maximising *exactly* the same value function that M is minimising. This motivates C to invent and generate experiments that lead to "novel" situations where M does not yet know how to predict well [plan1, int1].



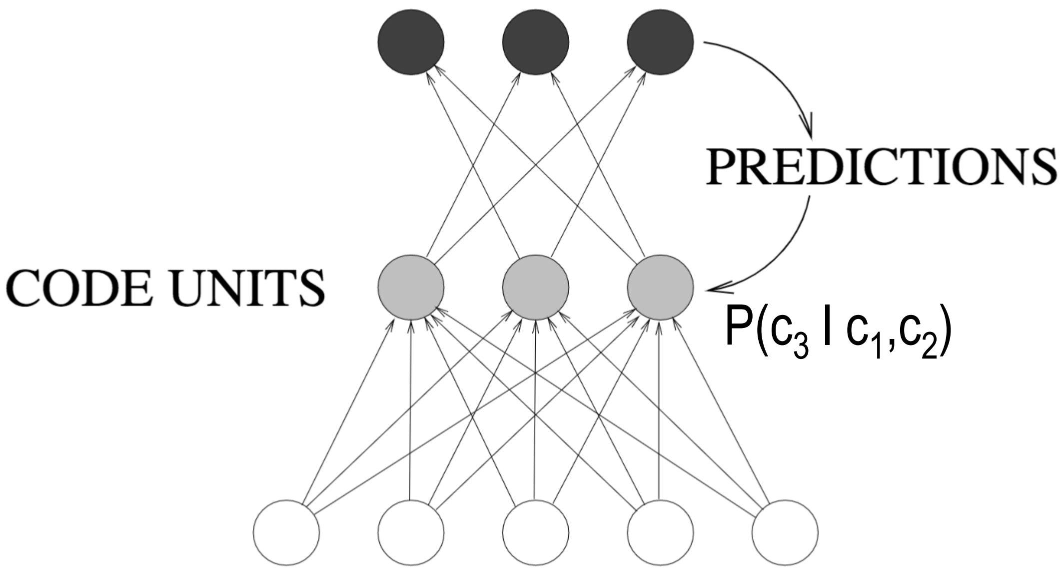
Making the World Differentiable: On Using Self-Supervised Fully Recurrent Neural Networks for Dynamic Reinforcement Learning and Planning in Non-Stationary Environments. J. Schmidhuber, 1990.

[plan1] J. Schmidhuber. Making the world differentiable: On using fully recurrent self-supervised neural networks for dynamic reinforcement learning and planning in non-stationary environments. TR FKI-126-90, TU Munich, November 1990. http://people.idsia.ch/~juergen/FKI-126-90_(revised)bw_ocr.pdf

[int1] J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In Proc. SAB'91, pages 222-227. MIT Press/ Bradford Books, 1991. Based on [plan1].

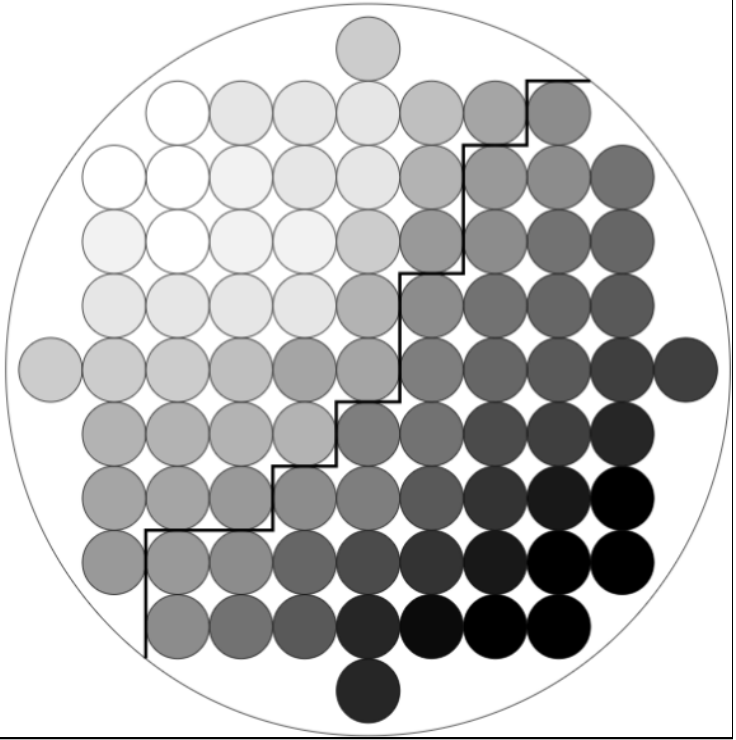
More than 40 follow-up papers on artificial curiosity:
http://people.idsia.ch/~juergen/interest.html
http://people.idsia.ch/~juergen/creativity.html

1991: Predictability Minimization (PM): 2 unsupervised nets fight minimax game to model given data distribution

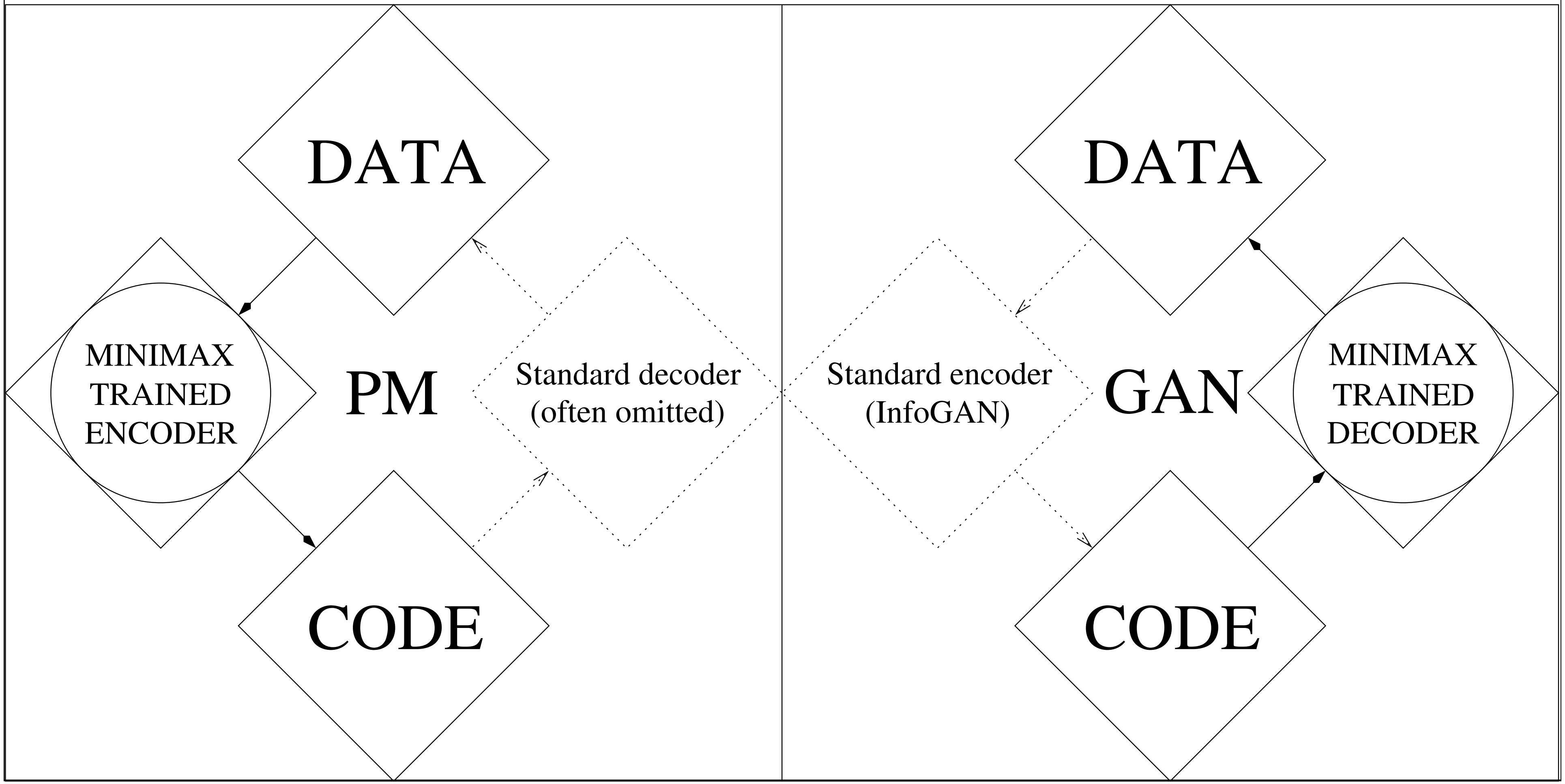


Encoder maximizes objective minimized by predictor. Saddle point = ideal factorial code: P(pattern) = $P(c_1)P(c_2)\ldots P(c_n)$

1996: PM applied to images: learns orientation-sensitive bar detectors, on-center-off-surround detectors, etc



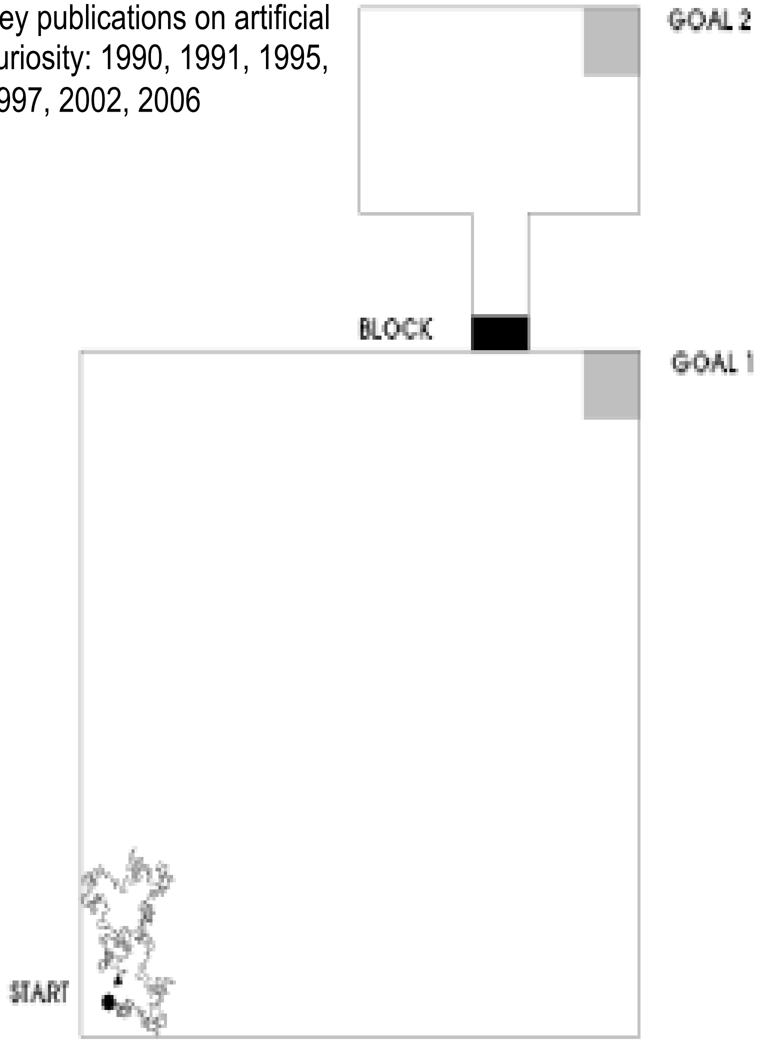
PM v GAN: latent space v original data space

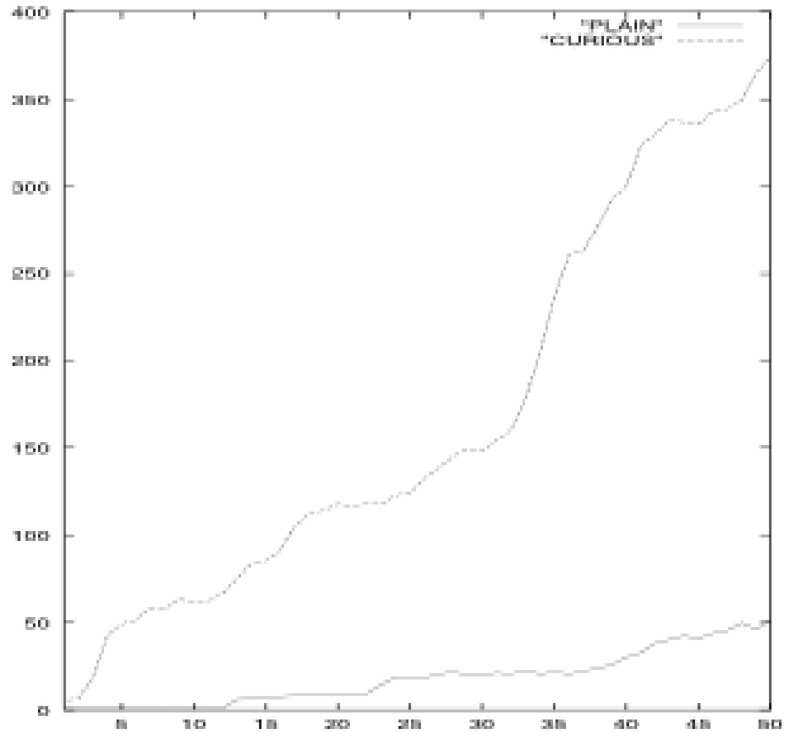


1997-2002: More Sophisticated Unsupervised Minimax for RL: What's interesting? Exploring the predictable

Two dueling, reward-maximizing modules (both general computers) called *left brain* and *right brain* collectively design an *experiment:* a (probabilistic) program that defines how to execute an action sequence in the environment, and how to compute the final experimental outcome through an instruction sequence implementing a computable function (e.g., a binary yes/no classification) of the observation sequence triggered by the experiment. Both brains can predict experimental outcomes before they are known. If their predictions or hypotheses differ, after having generated and executed the experiment, the surprised loser pays an *intrinsic* reward to the winner in a *zero sum game.* Each brain is maximising the value function minimised by the other. This may also accelerate the intake of external reward [int5-7].

Key publications on artificial curiosity: 1990, 1991, 1995, 1997, 2002, 2006





1997-2002: artificial curiosity through active unsupervised minimax accelerates real reward

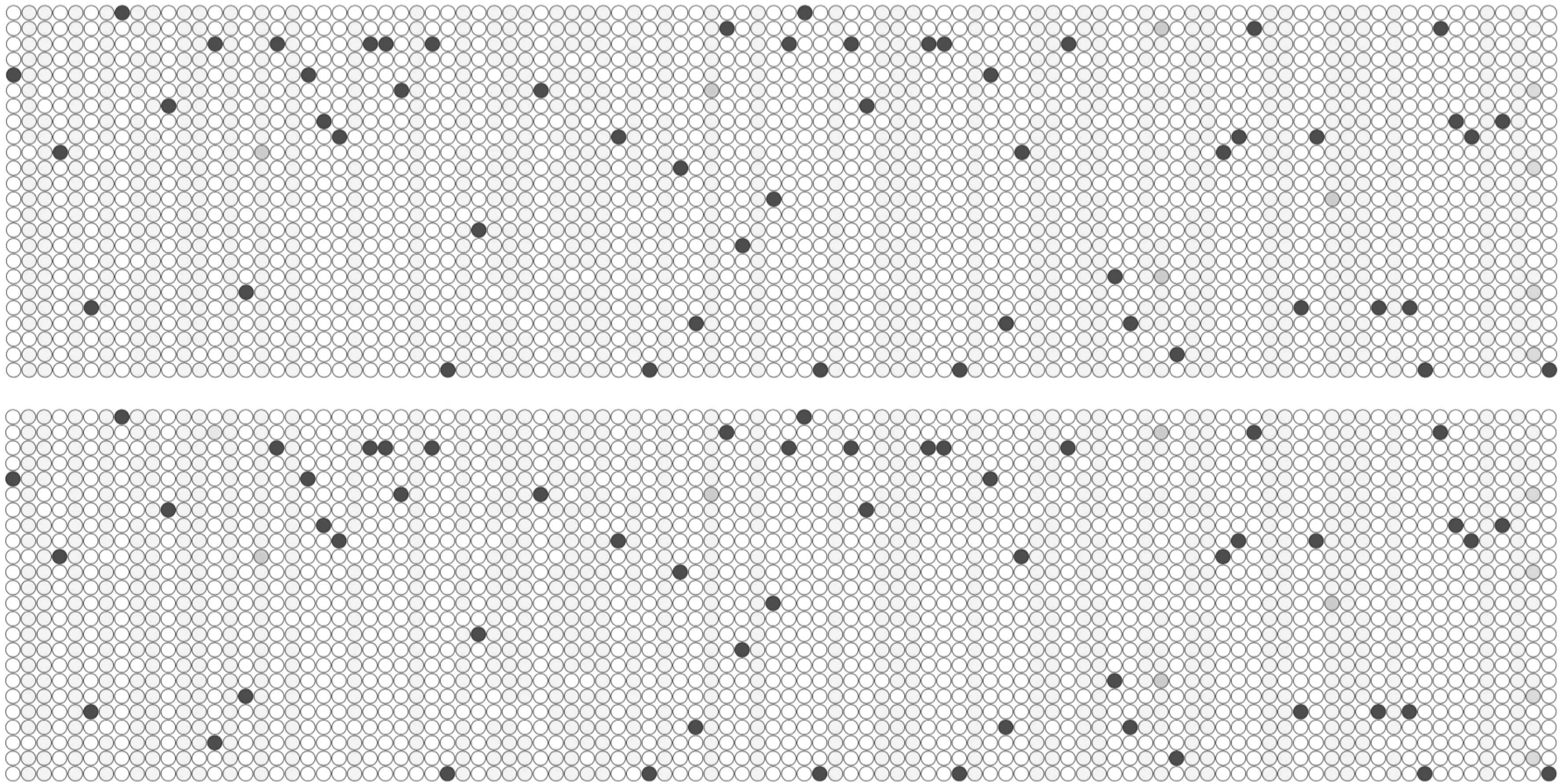
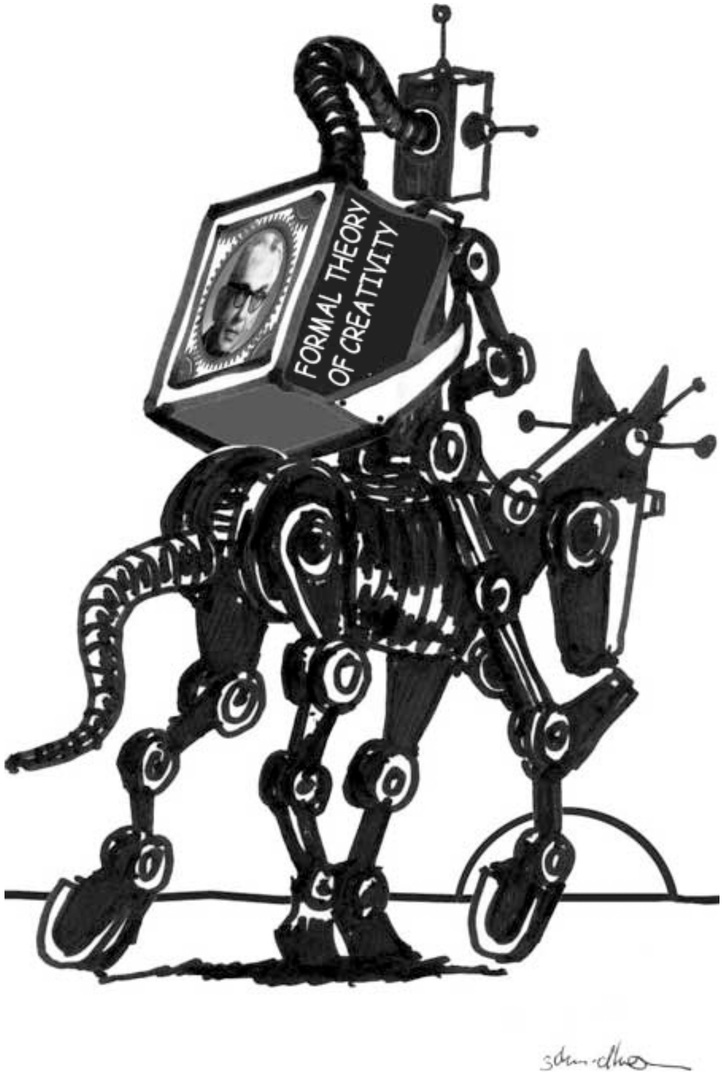


Figure 8: *Experiment 2a:* LEFT*'s (top) and* RIGHT*'s first 100 (of 576) probability distributions after simulation 1. Grey scales indicate probability magnitudes (white = close to 0, black = close to 1). The probability mass of many (but not all) columns is concentrated in a single value. Both brains are almost identical due to SSAandCopy PLAs. Their stacks are quite different though.*

[pm1] J. Schmidhuber. Learning factorial codes by predictability minimization. Neural Computation, 4(6):863-879, 1992. Based on TR CU-CS-565-91, Univ. Colorado at Boulder, 1991.
[pm2] J. Schmidhuber, M. Eldracher, B. Foltin. Semilinear predictability minimzation produces well-known feature detectors. Neural Computation, 8(4):773-786, 1996.
[int5] J. Schmidhuber. What's interesting? TR IDSIA-35-97, IDSIA, July 1997. (Co-evolution of unsupervised RL adversaries in a zero sum game for exploration. See also [int3].)
[int6] J . Schmidhuber. Artificial Curiosity Based on Discovering Novel Algorithmic Predictability Through Coevolution. In P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, Z. Zalzala, eds., Congress on Evolutionary Computation, p. 1612-1618, IEEE Press, Piscataway, NJ, 1999. Based on [int1].
[int7] J. Schmidhuber. Exploring the Predictable. In Ghosh, S. Tsutsui, eds., Advances in Evolutionary Computing, p. 579-612, Springer, 2002. Based on [int1].

More on Predictability Minimization (PM): http://people.idsia.ch/~juergen/ica.html
More on artificial curiosity: http://people.idsia.ch/~juergen/interest.html
http://people.idsia.ch/~juergen/creativity.html



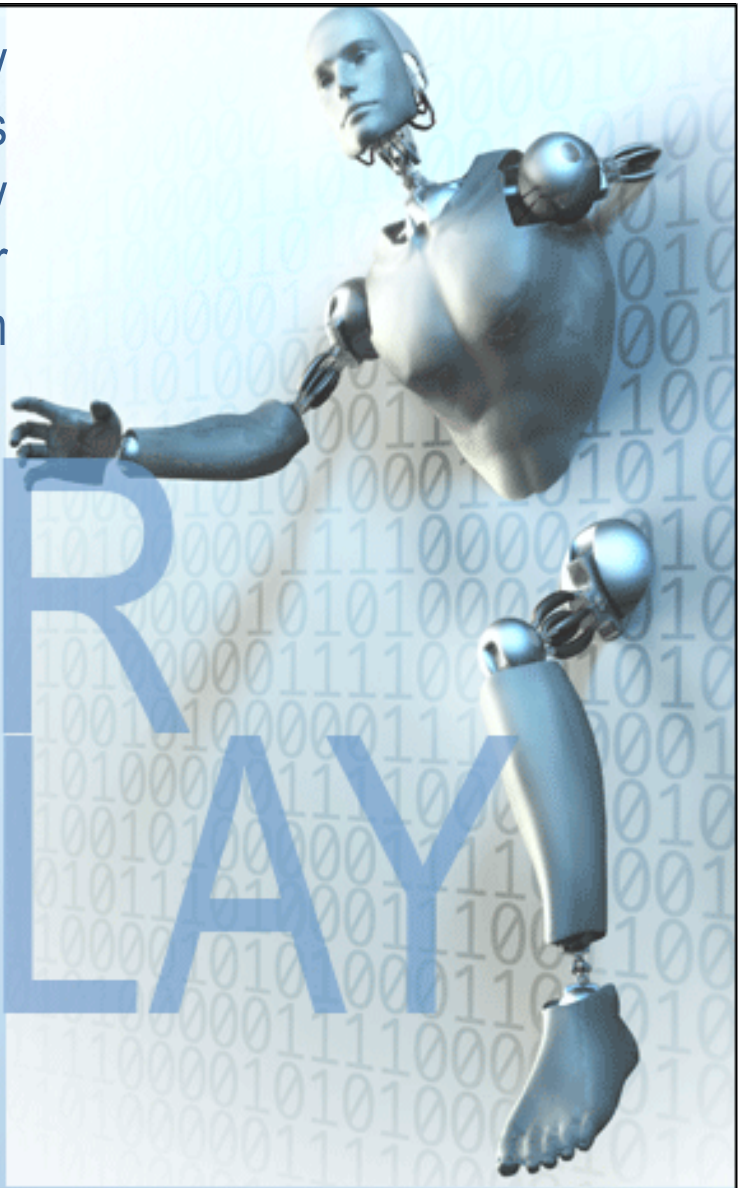
Maximize Future Fun(Data X,O(t))~ ∂CompResources(X,O(t))/∂t

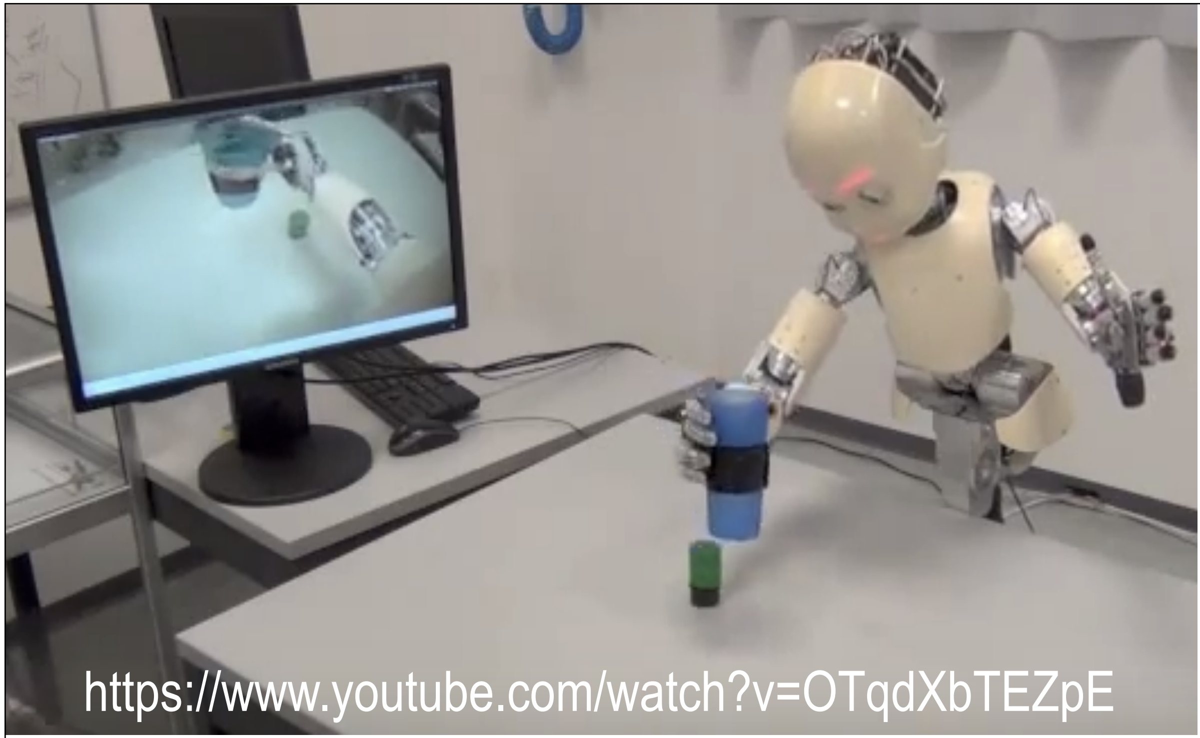
My formal theory of fun & novelty & surprise & attention & creativity & curiosity & art & science & humor

E.g., Connection Science 18(2):173-187, 2006
IEEE Transactions AMD 2(3):230-247, 2010
http://www.idsia.ch/~juergen/creativity.html

PowerPlay not only solves but also continually invents problems at the borderline between what's known and unknown - training an increasingly general problem solver by continually searching for the simplest still unsolvable problem

POWER PLAY

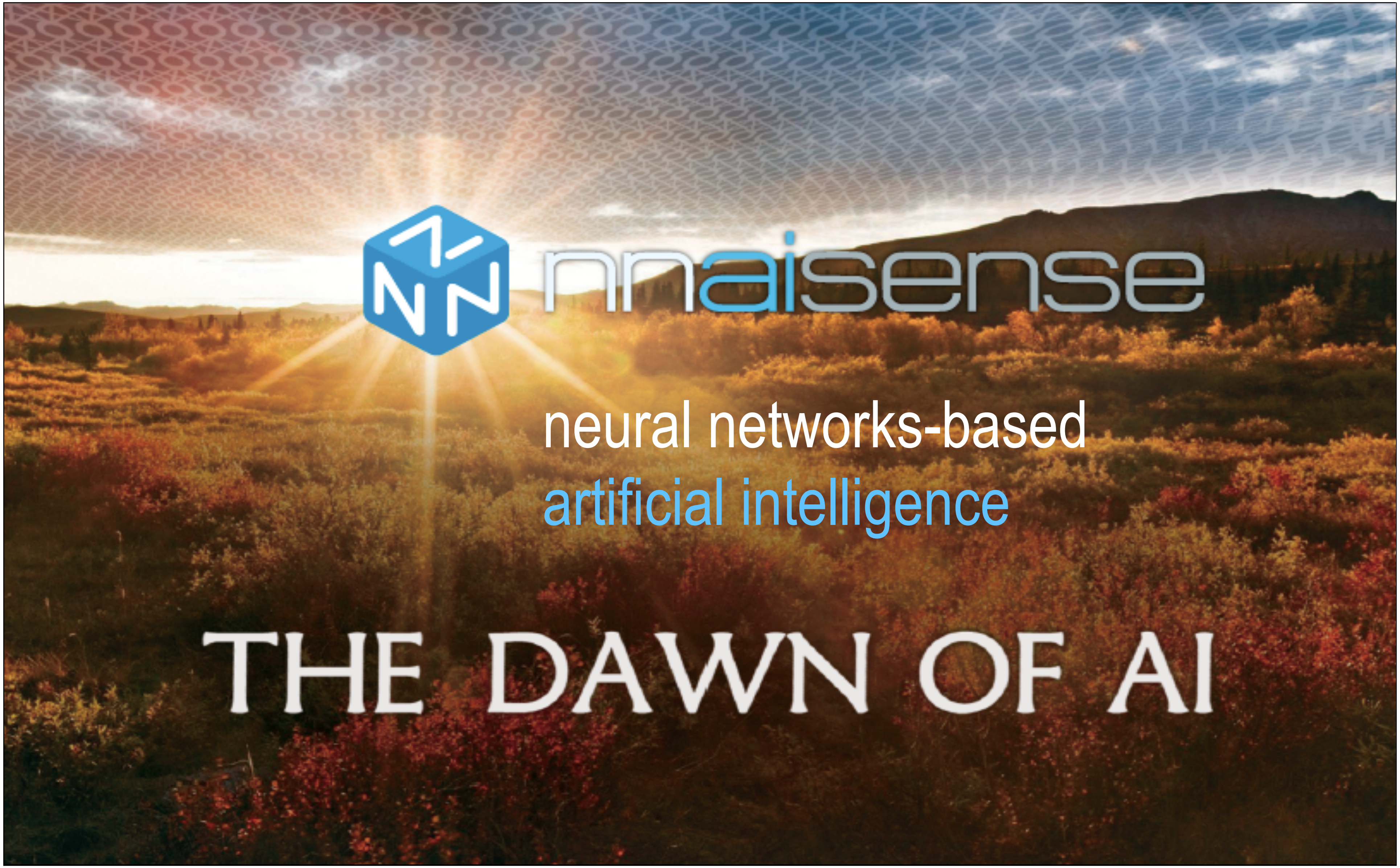




Continual curiosity-driven skill acquisition from high-dimensional video inputs for humanoid robots. Kompella, Stollenga, Luciw, Schmidhuber. Artificial Intelligence, 2015


YouTube
AAAI 2013 BEST STUDENT VIDEO AWARD
w. M Stollenga, K Frank, J Leitner, L Pape, A Foerster, J Koutnik

DRAWBACKS OF CURIOSITY


nnaisense
neural networks-based
artificial intelligence
THE DAWN OF AI

http://people.idsia.ch/~juergen/erc2017.html

www.nnaisense.com