On General Problem Solving & How to Learn an Algorithm

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

NNAISENSE
Jürgen Schmidhuber
You_again Shmidhoobuh
The deepest NNs:
RNNs are general computers
Learn program or algorithm = weight matrix

http://www.idsia.ch/~juergen/rnn.html
1997-2009. Since 2015 on your phone! Google, Microsoft, IBM, others, all use LSTM now

http://www.idsia.ch/~juergen/rnn.html
Today’s LSTM RNNs shaped by:

**Ex-PhD students** (TUM & IDSIA)
- Sepp Hochreiter (PhD 1999)
- Felix Gers (PhD 2001, e.g., gated recurrent units or forget gates)
- Alex Graves (PhD 2008)
- Daan Wierstra (PhD 2010)
- Justin Bayer (2009), others

**Postdocs** at IDSIA (2000s)
- Fred Cummins, Santiago Fernandez, Faustino Gomez, …
LSTM learns knot-tying tasklets:
Mayr Gomez Wierstra Nagy Knoll
Schmidhuber, IROS’06
Reinforcement Learning Robot Using LSTM in POMDPs

Bakker, Zhumatiy, Gruener, Schmidhuber, IROS 2003

Robot learns to memorize important bit of information
LSTM: First RNN to win contests: 3 ICDAR 2009 connected handwriting competitions

Connectionist Temporal Classification (CTC):
Graves, Fernandez, Gomez, Schmidhuber ICML 2006

$$O^{ML}(S) = - \sum_{(x,z) \in S} \ln(p(z|x))$$

http://www.idsia.ch/~juergen/handwriting.html
LSTM for speech: 2003 as good as HMMs, 2007: LSTM stack gets best results on keyword spotting in a large corpus (vs HMMs). Today: speech and NLP conferences are full of LSTM ...
A dozen of the many 2014/2015 benchmark records with LSTM RNNs / CTC, often at major IT companies:

1. Large vocabulary speech recognition (Sak et al., Google, Interspeech 2014)
2. English to French translation (Sutskever et al., Google, NIPS 2014)
3. CTC RNNs break Switchboard record (Hannun et al., Baidu, 2014)
5. Prosody contour prediction (Fernandez et al., IBM, Interspeech 2014)
6. Google Voice improved by 49% (Sak et al, 2015, now for >1 billion users)
7. Syntactic parsing for NLP (Vinyals et al., Google, 2014)
8. Photo-real talking heads (Soong and Wang, Microsoft, ICASSP 2015)
9. Social signal classification (Brueckner & Schulter, ICASSP 2014)
10. Image caption generation (Vinyals et al., Google, 2014)
11. Keyword spotting (Chen et al., Google, ICASSP 2015)
12. Video to textual description (Donahue et al., 2014; Li Yao et al., 2015)

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

http://www.idsia.ch/~juergen/firstdeelearner.html
Father of Deep Learning
Ivakhnenko et al, since 1965
Deep multilayer perceptrons with polynomial activation functions
Incremental layer-wise training by regression analysis - learn numbers of layers and units per layer - prune superfluous units
8 layers already back in 1971 still used in the 2000s
Supervised Backpropagation (BP)

<table>
<thead>
<tr>
<th>1990s: Trend from unsupervised to supervised RNNs</th>
<th>2000s: Trend from unsupervised to supervised FNNs</th>
</tr>
</thead>
<tbody>
<tr>
<td>Both trends driven by our team</td>
<td></td>
</tr>
</tbody>
</table>
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
1993: Gradient-based meta-RNNs that can learn to run their own weight change algorithm: J. Schmidhuber. A self-referential weight matrix. ICANN 1993

This was before LSTM. In 2001, however, Hochreiter taught a meta-LSTM to learn a learning algorithm for quadratic functions that was faster than backprop
More Self-Referential Fast Weights in RNNs (1993): Gradient-based RNN learns to control internal spotlight of attention for fast Hebb-anti-Hebb rule

J. Schmidhuber. Reducing the ratio between learning complexity and number of time-varying variables in fully recurrent nets. ICANN 1993
Our Deep GPU-Based Max-Pooling CNNs (IJCAI 2011)

e.g., http://www.idsia.ch/~juergen/deeplearning.html

Some of Our Deep Learning “Firsts”

- First very deep learner (1991-1993) – tasks with >1000 computational stages
- First neural learner of sequential attention (1991)
- First self-referential RNNs that run their own learning algorithm (1993)
- First very deep supervised learner (LSTM, 1995-2009 and beyond)
- First recurrent NN to win international contests (LSTM, 2009)
- First NN to win connected handwriting contests (LSTM, 2009)
- First outperformance of humans in a computer vision contest (MP-CNN, 2011)
- First deep NN to win Chinese handwriting contest (2011)
- European handwriting (MNIST): old error record almost halved (2011)
- First deep NN to win image segmentation contest (2012)
- First deep NN to win object detection contest (2012)
- First deep NN to win medical imaging contest (2012)
- First RNN controller that reinforcement learns from raw video (2013) …
Highway Networks: feedforward LSTM with forget gates (e.g., Gers et al, 2000); train NNs with hundreds of layers

Srivastava, Greff, Schmidhuber, NIPS 2015

Very similar: feedforward LSTM without gates, used by Microsoft to win ImageNet (150 layers): \( y = g(f(x)+x) \)
MICROSOFT WINS IMAGENET THROUGH FEEDFORWARD LSTM WITHOUT GATES

LSTM concepts keep invading CNN territory

JS 2015

http://people.idsia.ch/~juergen/microsoft-wins-imagenet-through-feedforward-LSTM-without-gates.html
Best Segmentation with PyramMiD-LSTM (NIPS 2015)

Stollenga, Byeon, Liwicki, Schmidhuber
BRAINSTORM

http://people.idsia.ch/~juergen/brainstorm.html
Open Source Neural Networks Library by my PhD students K Greff and R Srivastava

JS 2015

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

R-learn & improve learning algorithm itself, and also the meta-learning algorithm, etc…
Success-story algorithm for self-modifying code (since 1994)

R(t): Reward until time t. Stack of past check points $v_1v_2v_3 \ldots$ with self-mods in between. SSA undoes selfmods after $v_i$ that are not followed by long-term reward acceleration up until $t$ (now):

$$\frac{R(t)}{t} < \frac{[R(t) - R(v_1)]}{(t-v_1)} < \frac{[R(t) - R(v_2)]}{(t-v_2)} < \ldots$$
### Internal State

<table>
<thead>
<tr>
<th>Addresses</th>
<th>CONTENTS</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5321</td>
</tr>
<tr>
<td>1</td>
<td>-44</td>
</tr>
<tr>
<td>2</td>
<td>810</td>
</tr>
<tr>
<td>3</td>
<td>-2</td>
</tr>
<tr>
<td>4</td>
<td>-3322</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>6</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>9</td>
<td>-189</td>
</tr>
<tr>
<td>10</td>
<td>2</td>
</tr>
<tr>
<td>11</td>
<td>237</td>
</tr>
<tr>
<td>12</td>
<td>6</td>
</tr>
</tbody>
</table>

### Instruction Pointer

Parameters

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Probability</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = ADD(a1, a2, a3)</td>
<td>0.001 0.0014 0.9 0.24 0.001 0.0014 0.9 0.9</td>
</tr>
<tr>
<td>1 = MUL(a1, a2, a3)</td>
<td>0.001 0.0014 0.04 0.01 0.001 0.0014 0.04 0.04</td>
</tr>
<tr>
<td>2 = SUB(a1, a2, a3)</td>
<td>0.99 0.0014 0.01 0.01 0.99 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>3 = JMLEQ(a1, a2, a3)</td>
<td>0.001 0.99 0.01 0.01 0.001 0.99 0.01 0.01</td>
</tr>
<tr>
<td>4 = MOVEAGENT(a1, a2)</td>
<td>0.001 0.0014 0.01 0.7 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>5 = InvokersS(0)</td>
<td>0.004 0.0014 0.01 0.01 0.004 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>6 = INCP(0, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>7 = DECP(0, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
</tbody>
</table>

### Perceptions

#### External Environment

#### Variable Probability Distributions
<table>
<thead>
<tr>
<th>Addresses</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Contents</td>
<td>5321</td>
<td>-44</td>
<td>810</td>
<td>-2</td>
<td>-3322</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>-189</td>
<td>2</td>
<td>237</td>
<td>6</td>
</tr>
</tbody>
</table>

**Instruction Pointer**

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = ADD(a1, a2, a3)</td>
<td>0.001 0.0014 0.9 0.24 0.001 0.0014 0.9 0.9</td>
</tr>
<tr>
<td>1 = MUL(a1, a2, a3)</td>
<td>0.001 0.0014 0.04 0.01 0.001 0.0014 0.04 0.04</td>
</tr>
<tr>
<td>2 = SUB(a1, a2, a3)</td>
<td>0.99 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>3 = JMLEQ(a1, a2, a3)</td>
<td>0.001 0.99 0.01 0.01 0.001 0.99 0.01 0.01</td>
</tr>
<tr>
<td>4 = MOVEAGENT(a1, a2)</td>
<td>0.001 0.0014 0.01 0.7 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>5 = InvokeSSA()</td>
<td>0.004 0.0014 0.01 0.01 0.004 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>6 = INCREB(a1, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>7 = DECREB(a1, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
</tbody>
</table>

**Internal State**

**External Environment**

**Perceptions**

**Variable Probability Distributions**
<table>
<thead>
<tr>
<th>CONTENTS</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Addresses</td>
<td>5321</td>
<td>-44</td>
<td>810</td>
<td>-2</td>
<td>-3322</td>
<td>5</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>-189</td>
<td>2</td>
<td>237</td>
<td>6</td>
</tr>
</tbody>
</table>

| Instruction Pointer Parameters |

<table>
<thead>
<tr>
<th>Instruction</th>
<th>Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = ADD(a1, a2, a3)</td>
<td>0.001 0.0014 0.9 0.24 0.001 0.0014 0.9 0.9</td>
</tr>
<tr>
<td>1 = MUL(a1, a2, a3)</td>
<td>0.001 0.0014 0.04 0.01 0.001 0.0014 0.04 0.04</td>
</tr>
<tr>
<td>2 = SUB(a1, a2, a3)</td>
<td>0.99 0.0014 0.01 0.01 0.99 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>3 = JMLEQ(a1, a2, a3)</td>
<td>0.001 0.99 0.01 0.01 0.001 0.99 0.01 0.01</td>
</tr>
<tr>
<td>4 = MOVEAGENT(a1, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>5 = InvokeSSA()</td>
<td>0.004 0.0014 0.01 0.01 0.004 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>6 = INCFROB(a1, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
<tr>
<td>7 = DECROB(a1, a2)</td>
<td>0.001 0.0014 0.01 0.01 0.001 0.0014 0.01 0.01</td>
</tr>
</tbody>
</table>

Variable Probability Distributions

External Environment

Perceptions
<table>
<thead>
<tr>
<th>ADDRESS</th>
<th>CONTENTS</th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 = ADD(a1, a2, a3)</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.9</td>
<td>0.24</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.9</td>
<td>0.9</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1 = MUL(a1, a2, a3)</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.04</td>
<td>0.01</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.04</td>
<td>0.04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2 = SUB(a1, a2, a3)</td>
<td>0.99</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td>0.99</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3 = JMLEQ(a1, a2, a3)</td>
<td>0.001</td>
<td>0.99</td>
<td>0.01</td>
<td>0.01</td>
<td>0.001</td>
<td>0.99</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4 = MOVEAGENT(a1, a2)</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.7</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5 = InvokeSSA()</td>
<td>0.004</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.07</td>
<td>0.004</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6 = INCROB(a1, a2)</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7 = DECROB(a1, a2)</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td>0.001</td>
<td>0.0014</td>
<td>0.01</td>
<td>0.01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
1997: Lifelong meta-learning 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.
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 M. by asymptotically fastest method for all well-defined problems

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

$$n^3 + 10^{10000} = n^3 + O(1)$$

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

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, \(n=1\): \(\text{movdisk}\)
- 1 day, \(n=1,2\): \(\text{c4 c3 cpn c4 by2 c3 by2 exec}\)
- 3 days, \(n=1,2,3\): \(\text{c3 dec boostq defnp c4 calltp c3 c5 calltp endnp}\)
- 4 days: \(n=4, n=5, \ldots, n=30\): 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:
  
  \[
  \text{Hanoi(Source,Aux,Dest,n)}:\ IF\ n=0\ exit;\ ELSE\ BEGIN
  \text{Hanoi(Source, Dest, Aux, n-1)};\ move\ top\ disk\ from\ Aux\ to\ Dest;
  \text{Hanoi(Aux, Source, Dest, n-1)};\ END
  \]

- Prefix boosts instructions of previously 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
Back to RNNs, because physics dictates that future hardware will be 3D-RNN-like: many processors connected by many short and few long wires
Finds Complex Neural Controllers with a Million Weights – RAW VIDEO INPUT!
Faustino Gomez, Jan Koutnik, Giuseppe Cuccu, J. Schmidhuber, GECCO 2013
Octopus-arm control: 82 in, 32 out, 3'680 weights, only 20 DCT coefficients, compression 1:184

Octopus-arm with low-level vision, 32x32 in, 32 out, 33'824 weights, 160 DCT, compression 1:211

TORCS driving video game, low-level vision, 64x64 in, 3 out, 1'115'139 weights, 200 DCT, compression 1:5575

http://www.idsia.ch/~juergen/compressednetworksearch.html
Google bought DeepMind for 600 M to do Machine Learning (ML) & AI. First DeepMinders with PhDs in ML & AI: my lab’s ex-PhD students Legg (co-founder) & Wierstra (#4). Background of the other co-founders: neurobiology & video games (Hassabis) & business (Suleyman).

DeepMind hired 2 more ex-PhD students of mine: Graves (CTC @ ICML 2006, NTM, …) & Schaul (on our 2010 Atari-Go paper).
IJCNN 1990, NIPS 1991: Reinforcement Learning with Recurrent Controller & Recurrent World Model

A bit like AIXI, but with feasible local search
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
2015: all of this is coming back!
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
On Learning to Think: Algorithmic Information Theory for Novel Combinations of Reinforcement Learning RNN-based Controllers (RNNAs) and Recurrent Neural World Models

http://arxiv.org/abs/1511.09249
Maximize Future Fun(Data X,O(t)) ~
∂CompResources(X,O(t))/∂t

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

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

https://www.youtube.com/watch?v=OTqdXbTEZpE
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.
Next: build small animal-like AI that learns to think and plan hierarchically like a crow or a capuchin monkey.

Evolution needed billions of years for this, then only a few more millions for humans.
now talking to investors
neural networks-based
artificial intelligence

THE DAWN OF AI