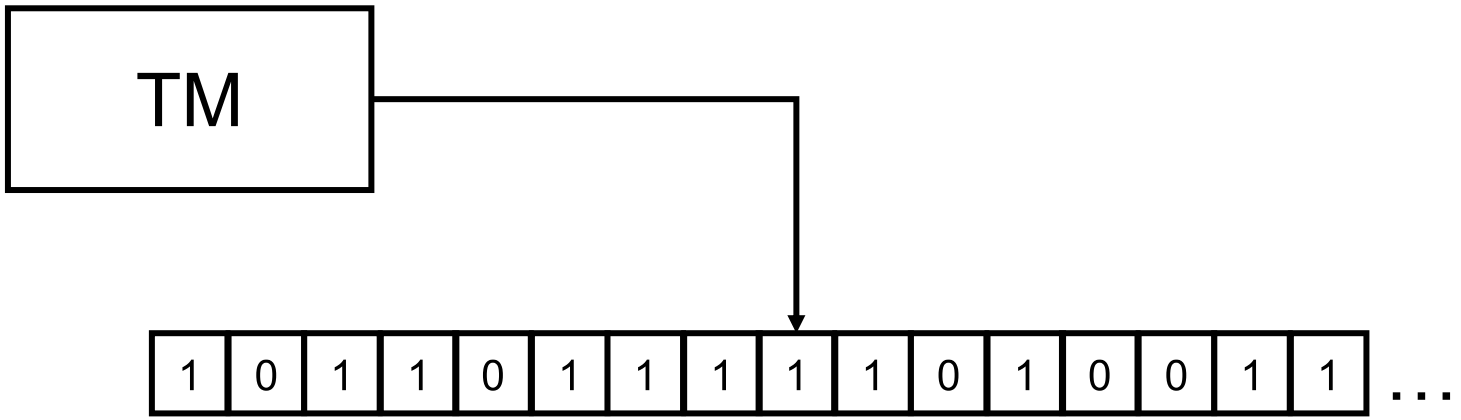
Computational Power and Computational Limits of Neural Networks

Seminar in Deep Neural Networks FS2022

17 May 2022

Martynas Noruišis
Mentor: Peter Belcák

Turing Machines



A. M. Turing, ‘Systems of logic based on ordinals’, *Proceedings of the London Mathematical Society*, Series 2, 1939

Simulating Turing Machines with Neural Networks

A. M. Turing, ‘Systems of logic based on ordinals’, *Proceedings of the London Mathematical Society*, Series 2, 1939

Construction

- A Recurrent Neural Network of 886 processors (nodes)
- Rational weights
- *No slow down (real time simulation)*

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

Sigmoid

Saturated linear function

$$\sigma(x) = \begin{cases} 0 \text{ if } x < 0 \\ x \text{ if } 0 \leq x \leq 1 \\ 1 \text{ if } x > 1 \end{cases}$$

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995
J. Kilian and H. T. Siegelmann, ‘The Dynamic Universality of Sigmoidal Neural Networks’, *Information and Computation*, vol. 128, 1996

$1-$tape TM $=$ $2-$stack Push$-$Down Automaton (PDA) &

$1-$tape TM $=$ $2-$stack PDA $\Longrightarrow$

$p-$tape TM $=$ $2p-$stack PDA

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

Encoding values

$$\omega = \omega_1 \omega_2 \dots \omega_n$$

$$\delta[\omega] = \delta[\omega_1 \omega_2 \dots \omega_n] = \sum_{i=1}^{n} \frac{\omega_i}{2^i}$$

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

10000101

100001010

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

100000000000

011111111111

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

Encoding binary strings

$$\omega = \omega_1\omega_2 \dots \omega_n$$

$$\delta[\omega] = \delta[\omega_1\omega_2 \dots \omega_n] = \sum_{i=1}^{n} \frac{2\omega_i + 1}{4^i}$$

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

Stack operations

$$\text{stack } q = \sum_{i=1}^{n} \frac{2\omega_i + 1}{4^i}$$

peek

$$\text{top}(q) = \sigma(4q - 2)$$

push 0

$$\frac{q}{4} + \frac{1}{4}$$

push 1

$$\frac{q}{4} + \frac{3}{4}$$

pop

$$4q - (2\,\text{top}(q) + 1)$$

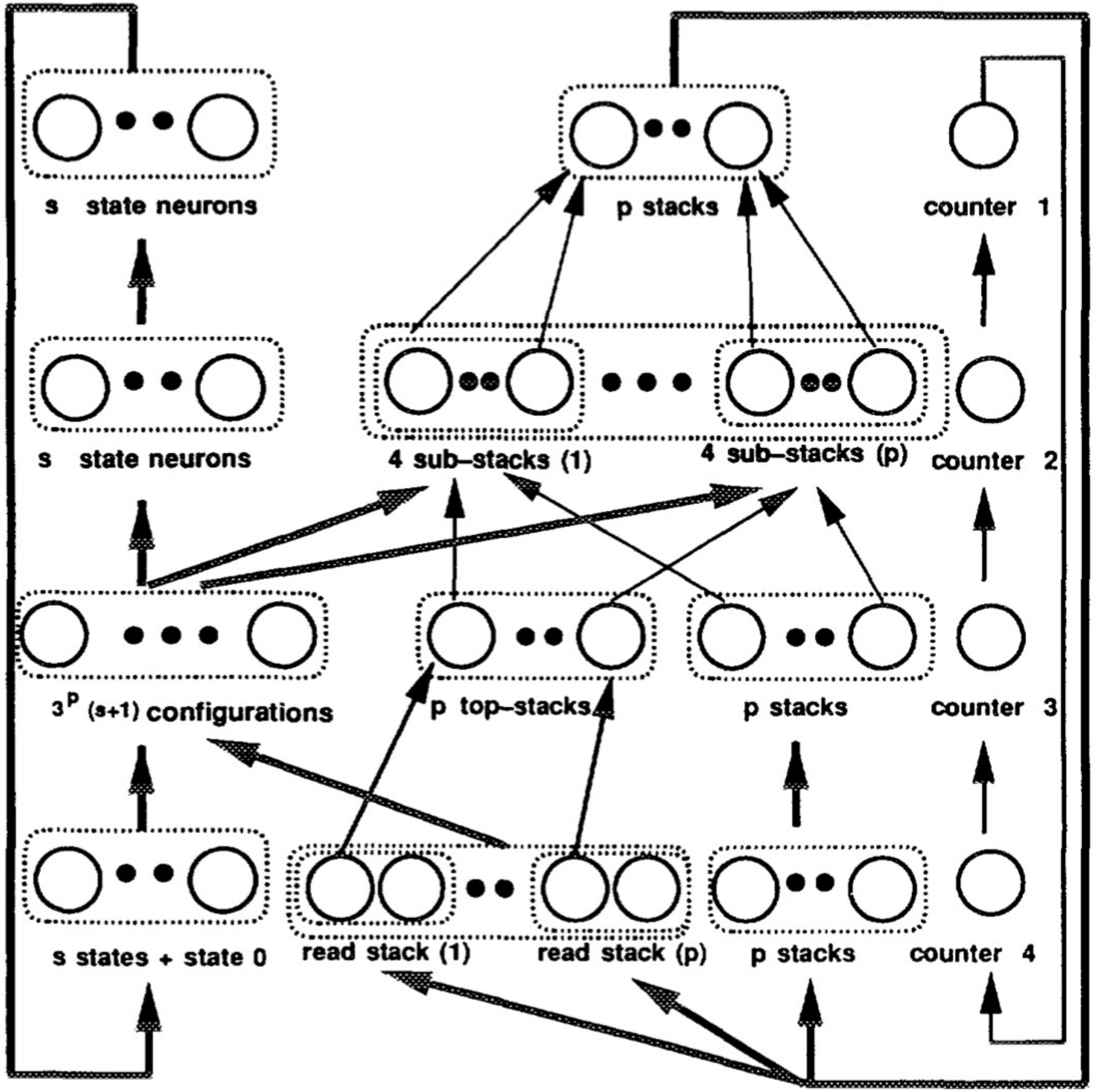
non-empty stack

$$\sigma(4q)$$

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

Layout of the construction

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995



F_1

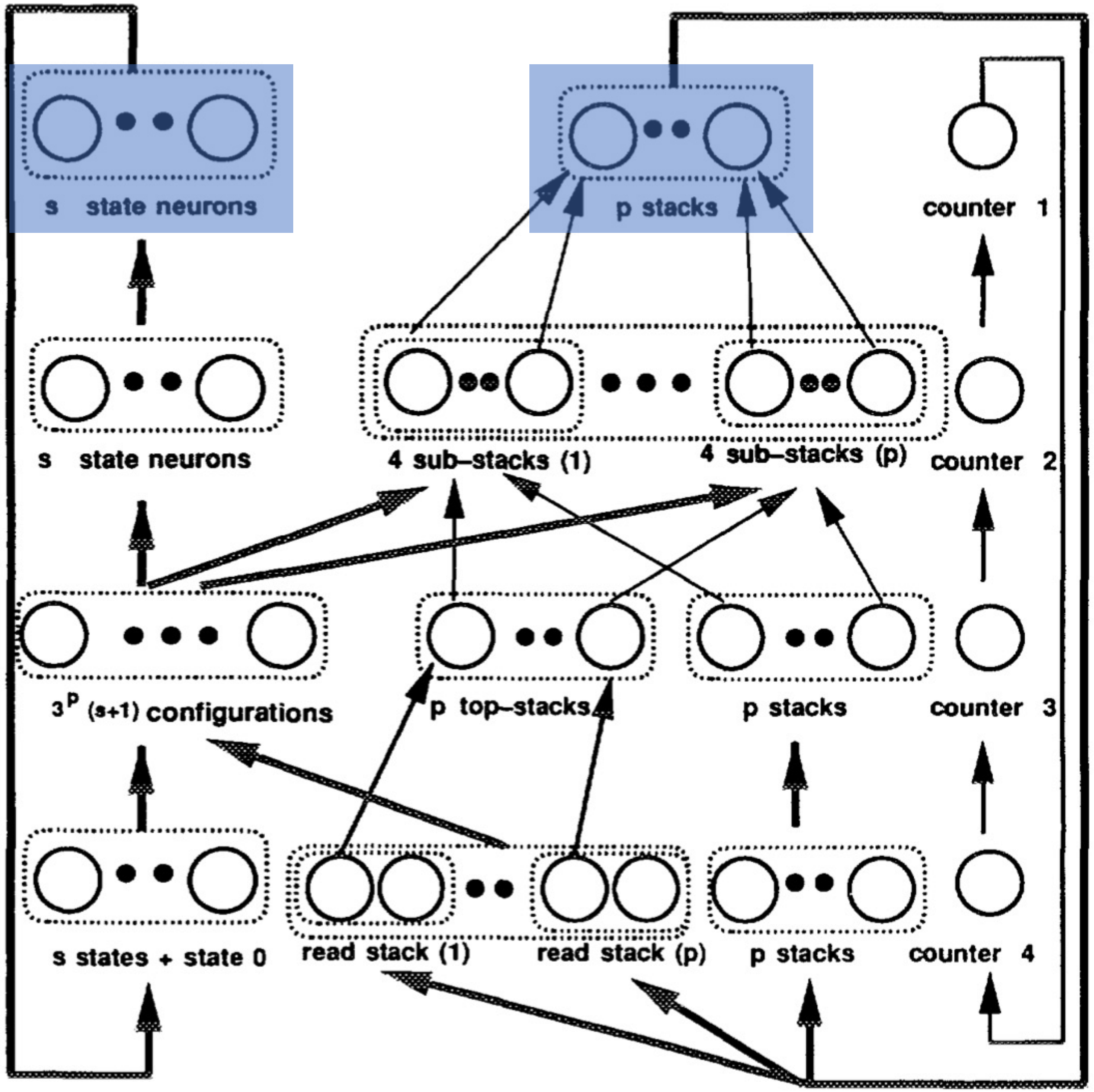
F_2

F_3

F_4

H. T. Siegelmann and E. D. Sontag, 'On the computational power of neural nets', *JCSS*, 1995

$$(x_1, \dots, x_p, q_1, \dots, q_p) \in \mathbb{Q}^{s+p}$$



F_1

F_2

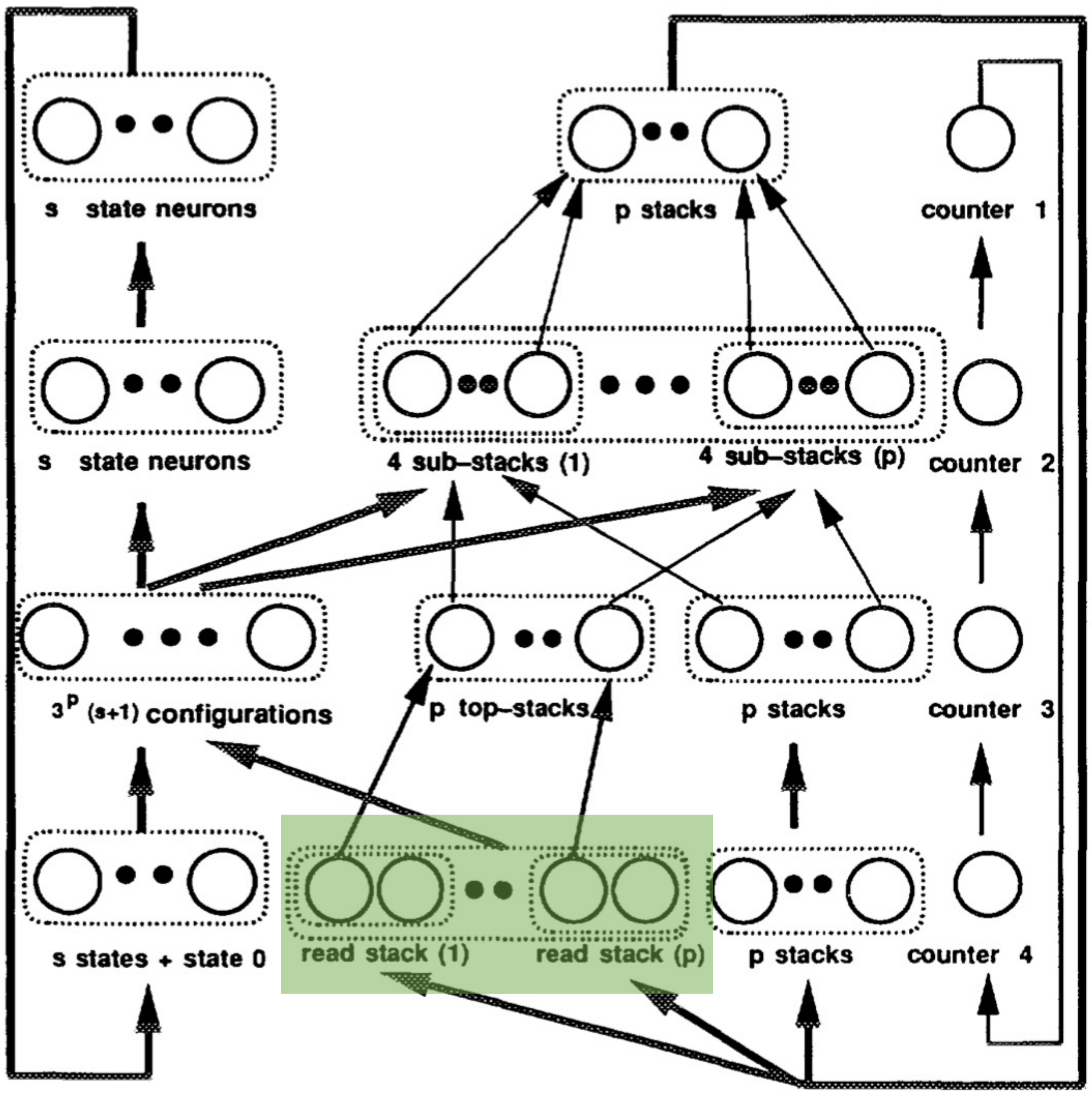
F_3

F_4

H. T. Siegelmann and E. D. Sontag, 'On the computational power of neural nets', *JCSS*, 1995

$(\zeta[q_i], \tau[q_i]) \in \{0,1\} \times \{0,1\}$

$(\text{top}(q_i), \text{non\_empty}(q_i))$



F_1

F_2

F_3

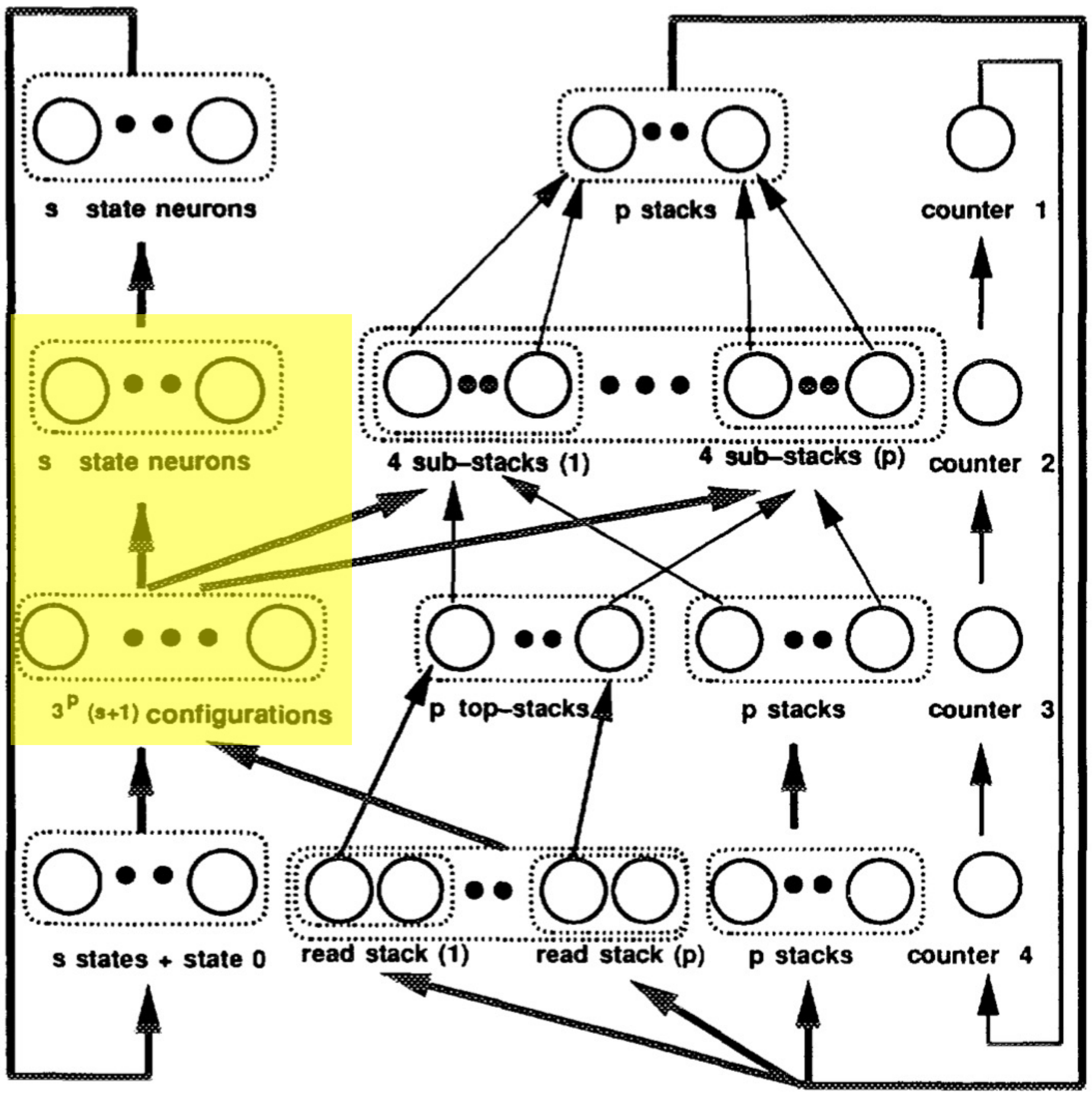
F_4

H. T. Siegelmann and E. D. Sontag, 'On the computational power of neural nets', *JCSS*, 1995

$$x_i^{+} := \sum_{j=0}^{s} \beta_{ij}(\zeta[q_1], \dots, \zeta[q_1], \tau[q_i], \dots, \tau[q_p]) x_j$$

$\beta_{ij}(\zeta[q_1], \dots, \zeta[q_1], \tau[q_i], \dots, \tau[q_p]) = 1$
$\Leftrightarrow$ there is a transition from j to i with the configuration

$$\beta(d_1, d_2, \dots, d_t)x = \sum_{r=1}^{2^t} c_r\, \sigma(v_r \cdot \mu)$$



F_1

F_2

F_3

F_4

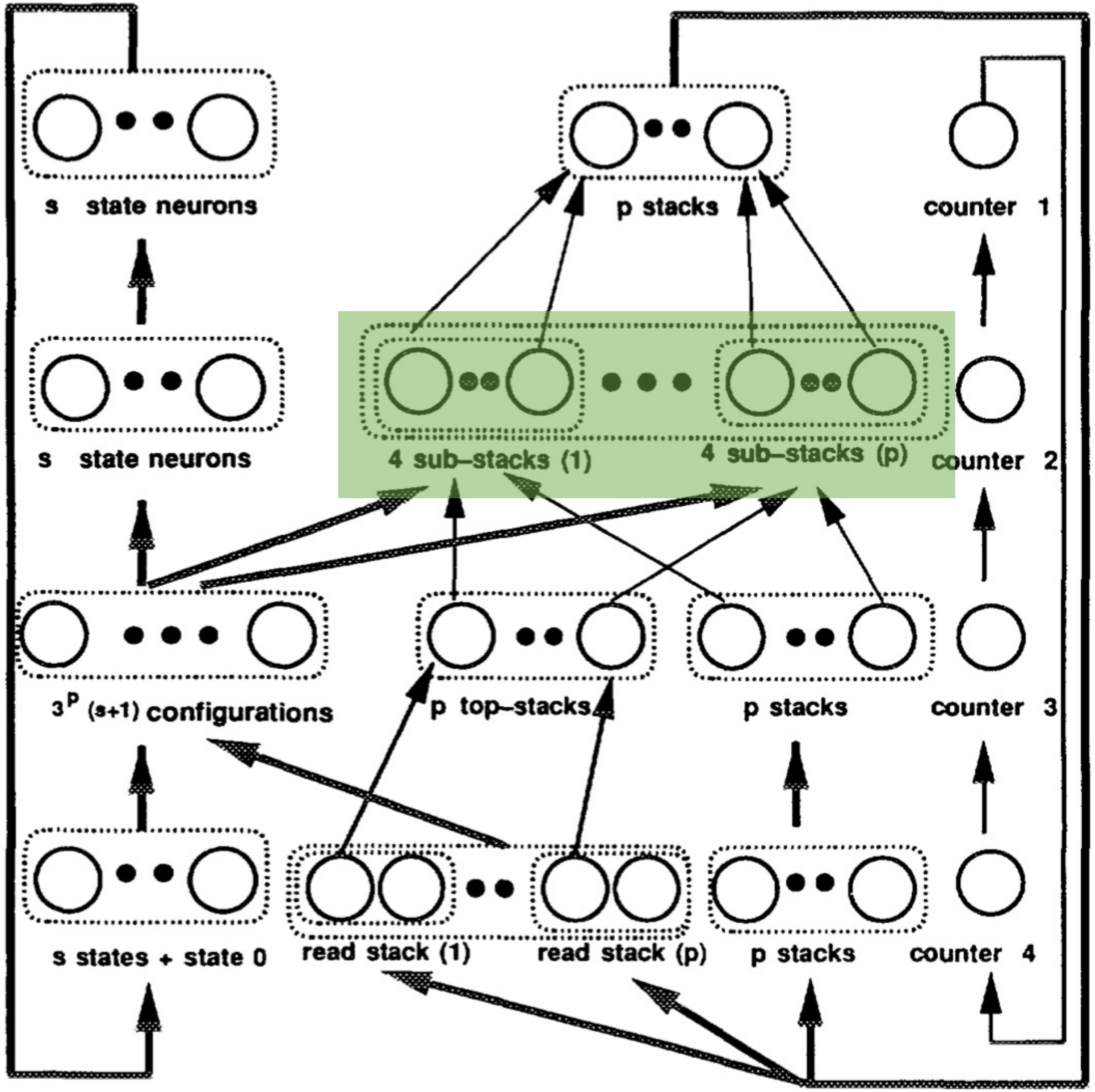
H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

$$q_i^+ := \left(\sum_{j=0}^{s} y_{ij}^1(a_1, \dots, a_p, b_1, \dots, b_p)x_j\right) \times q_i$$

$$+ \left(\sum_{j=0}^{s} y_{ij}^2(a_1, \dots, a_p, b_1, \dots, b_p)x_j\right) \times \left(\frac{1}{4}q_i + \frac{1}{4}\right)$$

$$+ \left(\sum_{j=0}^{s} y_{ij}^3(a_1, \dots, a_p, b_1, \dots, b_p)x_j\right) \times \left(\frac{1}{4}q_i + \frac{1}{4}\right)$$

$$+ \left(\sum_{j=0}^{s} y_{ij}^4(a_1, \dots, a_p, b_1, \dots, b_p)x_j\right) \times (4q_i - 2\zeta[q_i] - 1)$$



F_1

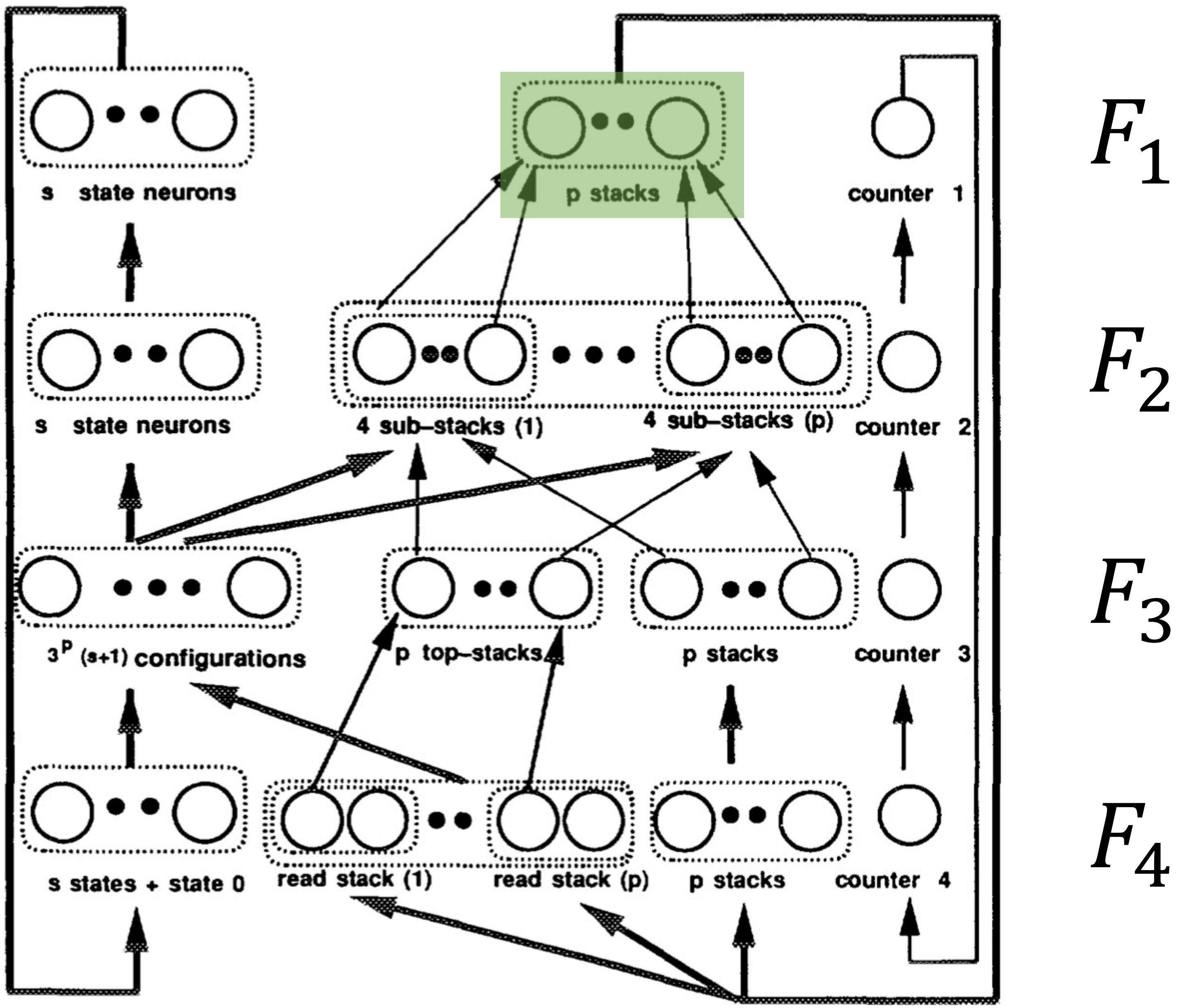
F_2

F_3

F_4

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

$$q_i^+ := \left(\sum_{j=0}^{s} y_{ij}^1(a_1,\dots,a_p,b_1,\dots,b_p)x_j\right) \times q_i$$
$$+ \left(\sum_{j=0}^{s} y_{ij}^2(a_1,\dots,a_p,b_1,\dots,b_p)x_j\right)$$
$$\times \left(\frac{1}{4}q_i + \frac{1}{4}\right)$$
$$+ \left(\sum_{j=0}^{s} y_{ij}^3(a_1,\dots,a_p,b_1,\dots,b_p)x_j\right)$$
$$\times \left(\frac{1}{4}q_i + \frac{1}{4}\right)$$
$$+ \left(\sum_{j=0}^{s} y_{ij}^4(a_1,\dots,a_p,b_1,\dots,b_p)x_j\right)$$
$$\times (4q_i - 2\zeta[q_i] - 1)$$



 H. T. Siegelmann and E. D. Sontag, 'On the computational power of neural nets', *JCSS*, 1995

Construction

- An RNN of 886 processors (nodes)
- Rational weights
- No slow down (real time simulation)

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995

Are Neural Networks more powerful than TMs?

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995
H. T. Siegelmann and E. D. Sontag, ‘Analog computation via neural networks’, *Theoretical Computer Science*, vol. 131, 1994
J. L. Balcazar, R. Gavalda, and H. T. Siegelmann, ‘Computational power of neural networks: a characterization in terms of Kolmogorov complexity’, *IEEE Trans. Inform. Theory*, vol. 43, Jul. 1997

Super-Turing computation

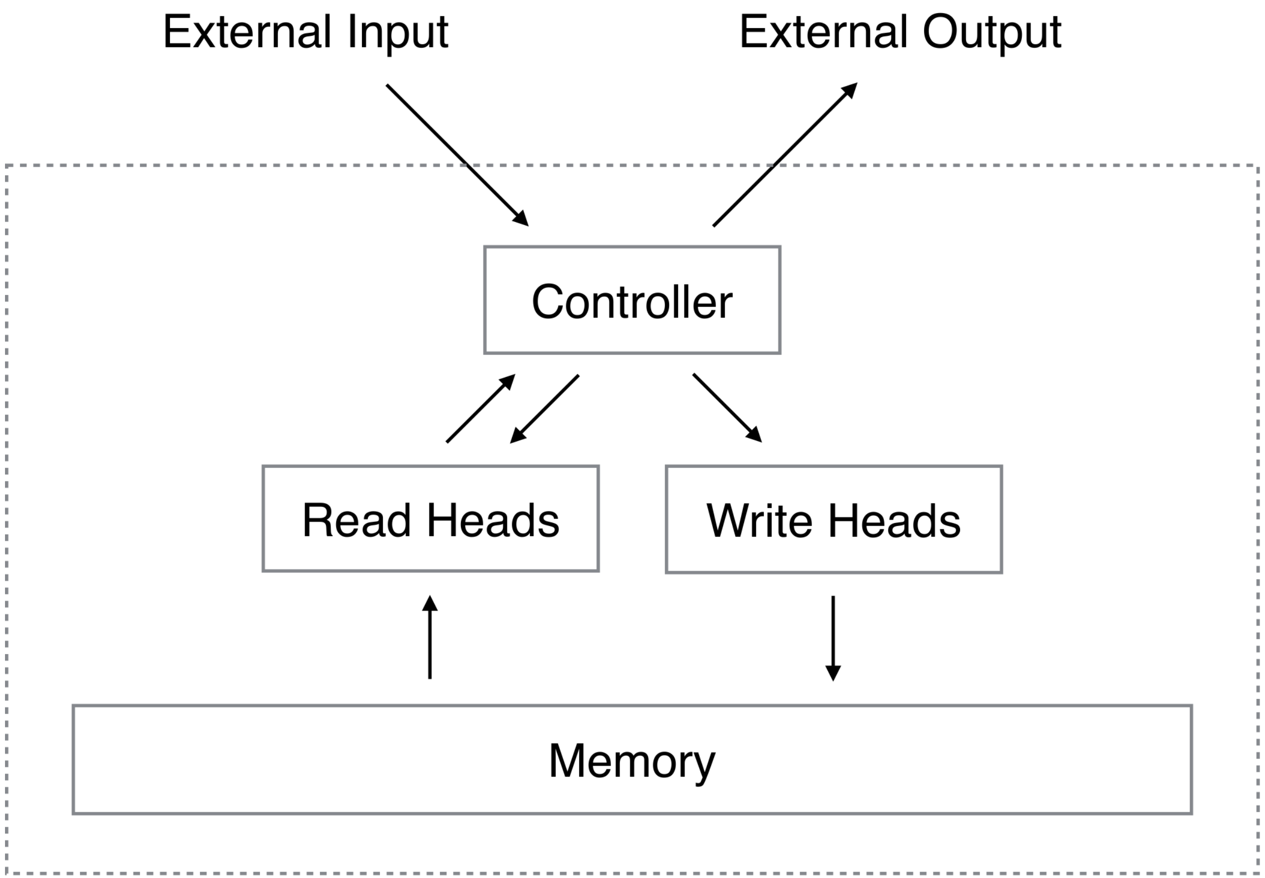
- NNs with rational weights = as powerful as Turing Machines
- NNs with real weights = more powerful than Turing Machines (Super-Turing!)
  - Can decide any language in exponential time
  - In polynomial time accept languages in P/poly

H. T. Siegelmann and E. D. Sontag, ‘On the computational power of neural nets’, *JCSS*, 1995
H. T. Siegelmann and E. D. Sontag, ‘Analog computation via neural networks’, *Theoretical Computer Science*, vol. 131, 1994
J. L. Balcazar, R. Gavalda, and H. T. Siegelmann, ‘Computational power of neural networks: a characterization in terms of Kolmogorov complexity’, *IEEE Trans. Inform. Theory*, vol. 43, Jul. 1997

Mimicking Turing Machines

Neural Turing Machines

Neural Turing Machines



A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

Reading

read vector

$$\mathbf{r}_t \leftarrow \sum_i w_t(i)\mathbf{M}_t(i)$$

Writing

erase

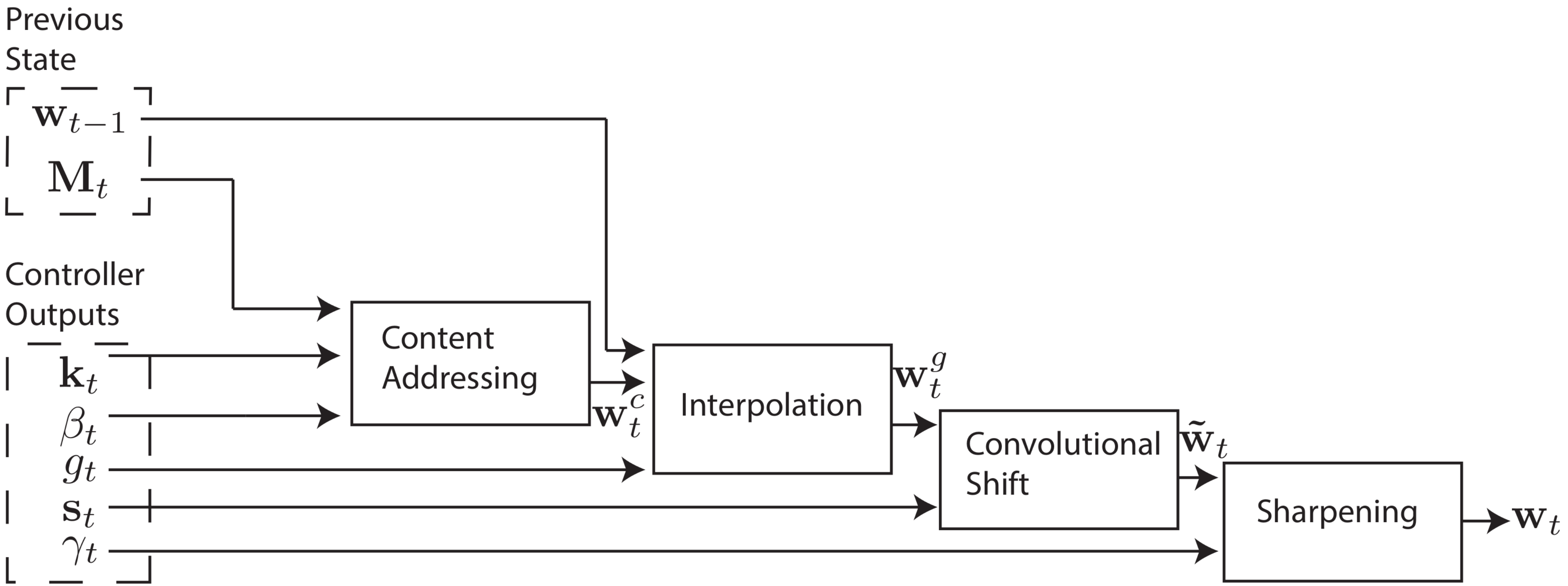
$$\widetilde{\mathbf{M}}_t(i) \leftarrow \mathbf{M}_{t-1}(i)[\mathbf{1} - w_t(i)\boldsymbol{e}_t]$$

add

$$\mathbf{M}_t(i) \leftarrow \widetilde{\mathbf{M}}_t(i) + w_t(i)\boldsymbol{a}_t$$

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

Addressing



A. Graves, G. Wayne, and I. Danihelka, 'Neural Turing Machines', *arXiv:1410.5401 [cs]*, Dec. 2014

Focusing by Content

$$w_t^c(i) \leftarrow \frac{\exp(\beta_t K[\boldsymbol{k}_t, \mathbf{M}_t(i)])}{\sum_j \exp(\beta_t K[\boldsymbol{k}_t, \mathbf{M}_t(j)])}$$

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

Focusing by Location

interpolation

$$\mathbf{w}_t^g \leftarrow g_t \mathbf{w}_t^c + (1 - g_t)\mathbf{w}_{t-1}$$

convolutional shift

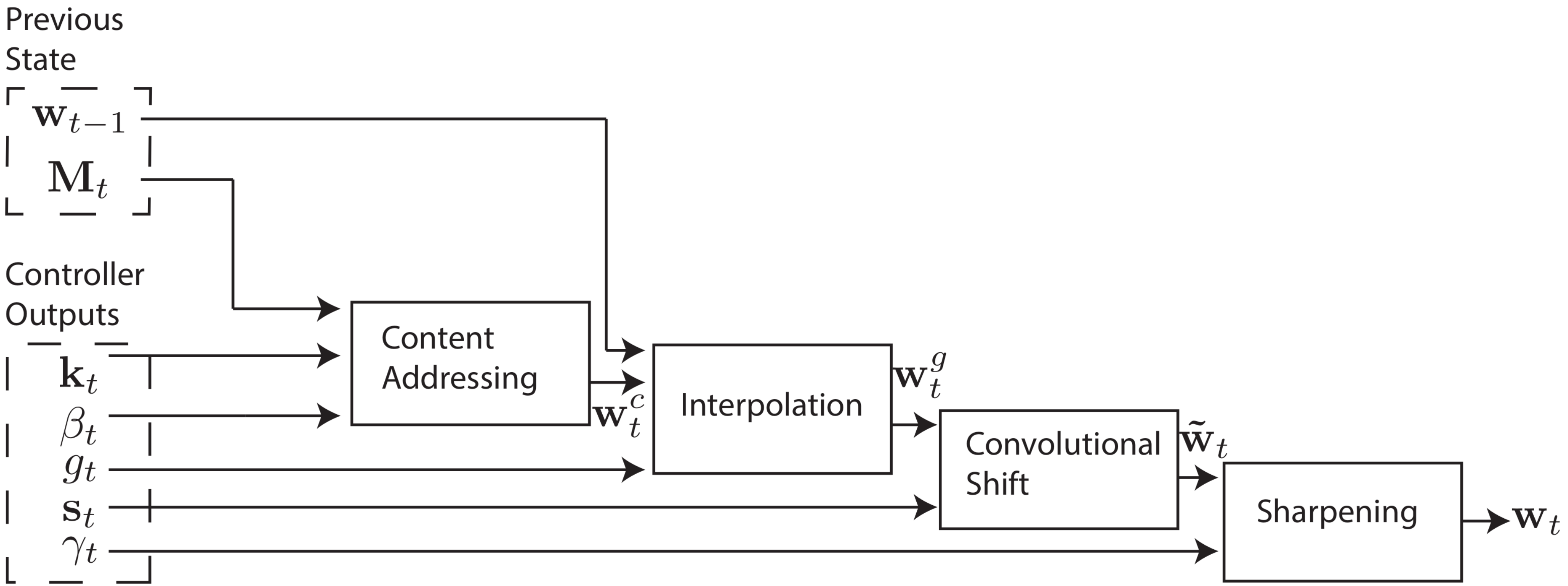
$$\tilde{w}_t(i) \leftarrow \sum_{j=0}^{N-1} w_t^g(j) s_t(i-j)$$

sharpening

$$w_t(i) \leftarrow \frac{\tilde{w}_t(i)^{\gamma t}}{\sum_j \tilde{w}_t(j)^{\gamma t}}$$

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

Addressing



A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

LSTM

- Small internal memory
- Less interpretable

Feed-forward

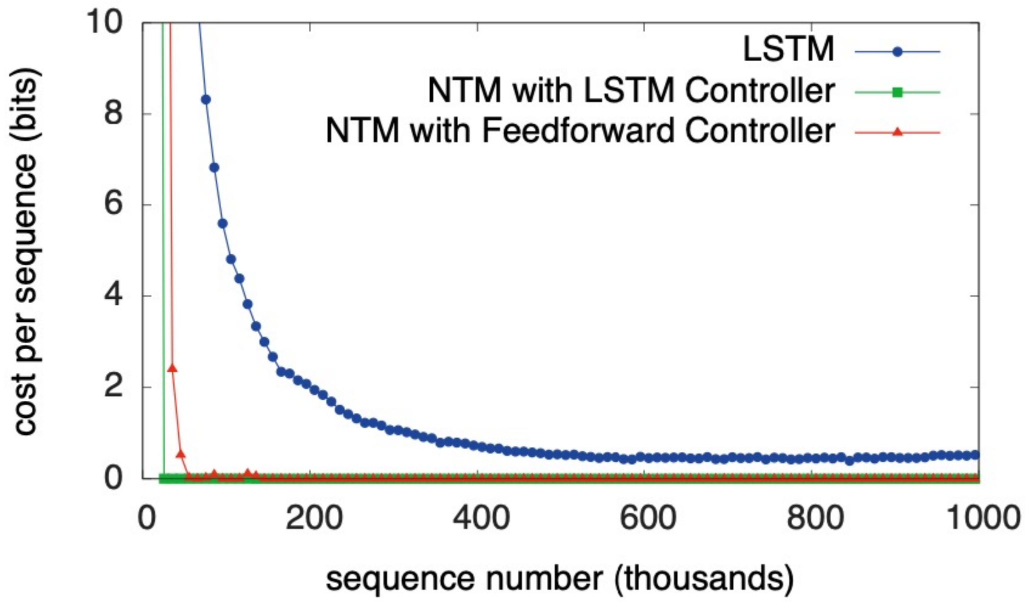
- No internal memory
- More interpretable

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

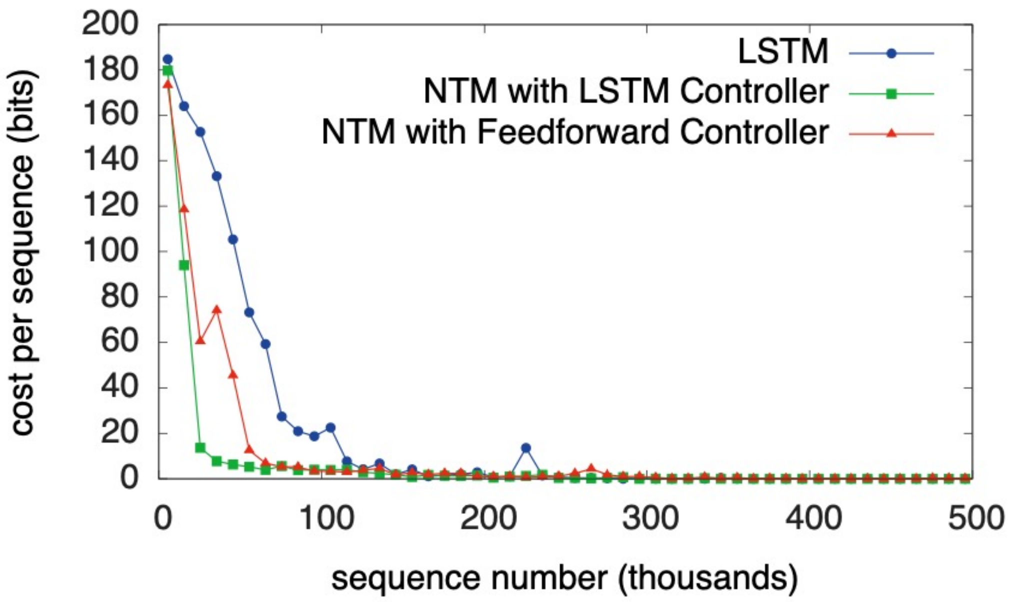
Experiments

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

Copy and Repeat Copy



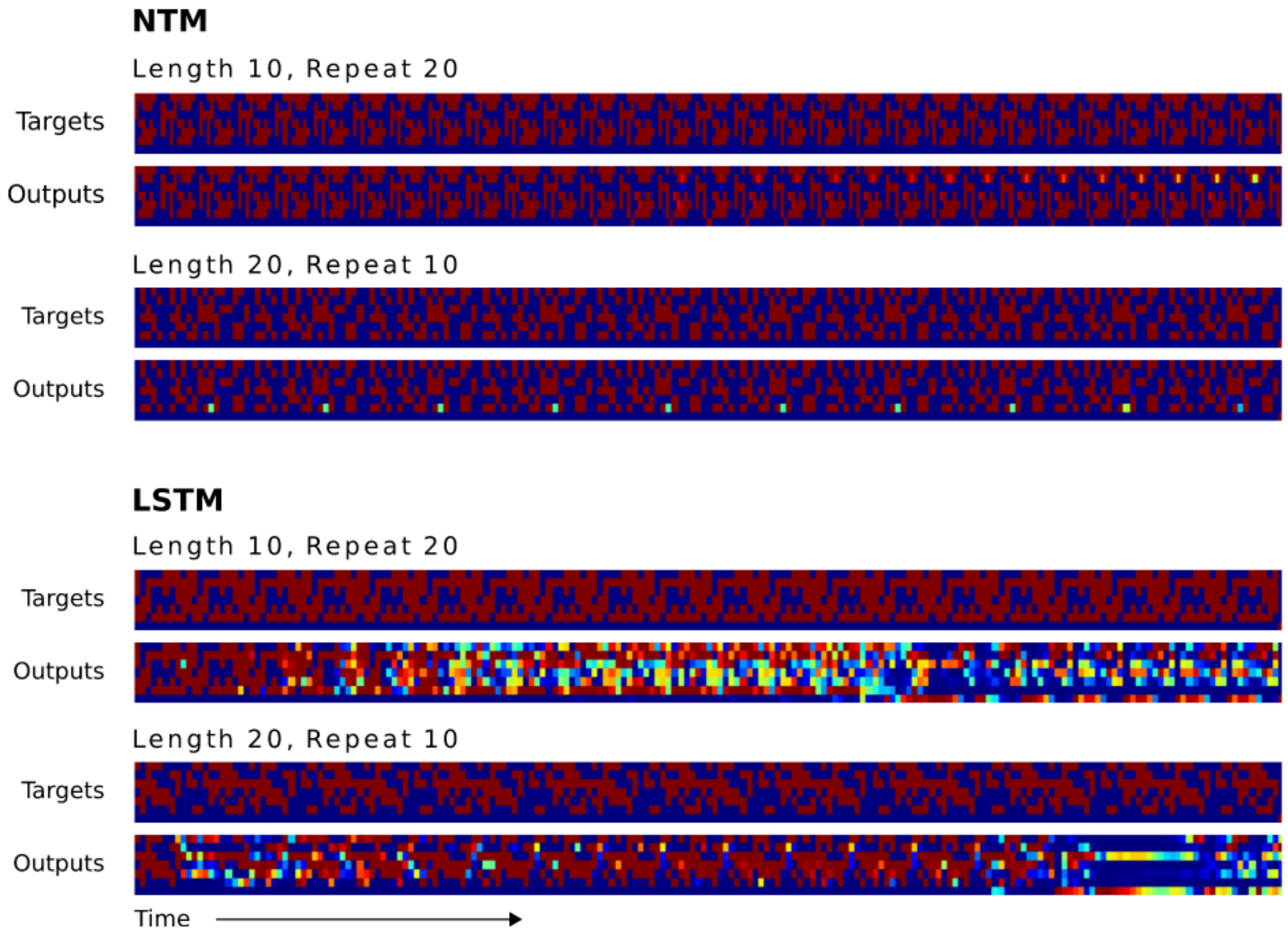
copy



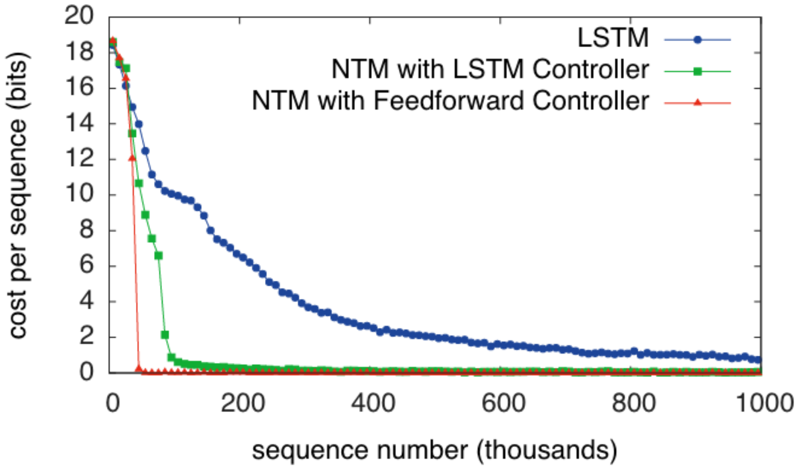
repeat copy

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

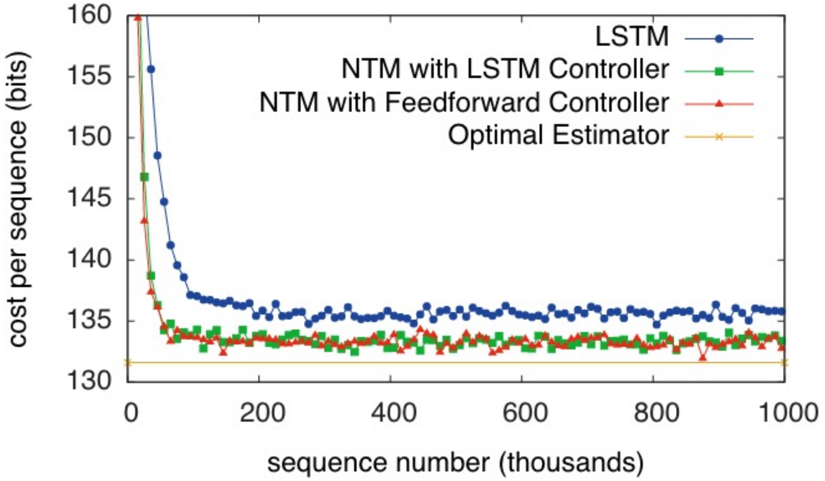
Repeat Copy Generalisation



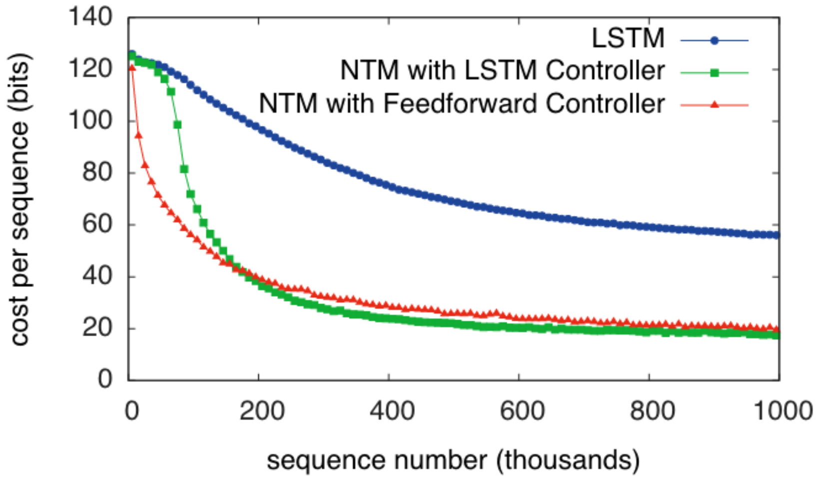
A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014



associative recall



dynamic n-gram



priority sort

A. Graves, G. Wayne, and I. Danihelka, ‘Neural Turing Machines’, *arXiv:1410.5401 [cs]*, Dec. 2014

Memory Networks

Components

- $\boldsymbol{I}$ (input feature map)
  - convert x to an internal representation $I(x)$
- $\boldsymbol{G}$ (generalisation)
  - $m_i = G(m_i, I(x), m)\ \forall i$
- $\boldsymbol{O}$ (output feature map)
  - $o = O(I(x), m)$
- $\boldsymbol{R}$ (response)
  - $r = R(o)$

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Basic model

memories stored at the next available slot

$$\mathbf{m}_N = x, N = N + 1$$

k relevant memories chosen by O

$$o_1 = O_1(x, \boldsymbol{m}) = \mathrm{argmax}_{i=1,\dots,N}\ s_O(x, \boldsymbol{m}_i)$$

$$o_2 = O_2(x, \boldsymbol{m}) = \mathrm{argmax}_{i=1,\dots,N}\ s_O([x, \boldsymbol{m}_{O_1}], \boldsymbol{m}_i)$$

$$\dots$$

$$o_k = O_k(x, \boldsymbol{m}) = \mathrm{argmax}_{i=1,\dots,N}\ s_O([x, \boldsymbol{m}_{O_1}, \dots, \boldsymbol{m}_{O_k}], \boldsymbol{m}_i)$$

J. Weston, S. Chopra, and A. Bordes, ‘Memory Networks’, *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Basic model

choose best word to output

$$r = \operatorname{argmax}_{w \in W} s_R([x, \boldsymbol{m}_{O_1}, \ldots, \boldsymbol{m}_{O_k}], w)$$

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Training

minimise (for $k = 2$)

$$\sum_{\bar{f} \neq \mathbf{m}_{o_1}} \max(0, \gamma - s_O(x, \mathbf{m}_{O_1}) + s_O(x, \bar{f}))$$
$$+ \sum_{\bar{f}' \neq \mathbf{m}_{o_2}} \max(0, \gamma - s_O([x, \mathbf{m}_{O_1}], \mathbf{m}_{O_2}) + s_O([x, \mathbf{m}_{O_1}], \bar{f}'))$$
$$+ \sum_{\bar{r} \neq r} \max(0, \gamma - s_R([x, \mathbf{m}_{O_1}, \mathbf{m}_{O_2}], r) + s_R([x, \mathbf{m}_{O_1}, \mathbf{m}_{O_2}], \bar{r}))$$

J. Weston, S. Chopra, and A. Bordes, ‘Memory Networks’, *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Extensions

J. Weston, S. Chopra, and A. Bordes, ‘Memory Networks’, *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Efficient memory via hashing

J. Weston, S. Chopra, and A. Bordes, ‘Memory Networks’, *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Efficient memory via hashing

The capital of France is Paris

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Efficient memory via hashing

1. Train embedding matrix U_O
2. Use K-Means to cluster word vectors $(U_O)_i$

J. Weston, S. Chopra, and A. Bordes, ‘Memory Networks’, *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Modelling write time

$$s_{O_t}(x, y, y') = \Phi_x(x)^{\mathrm{T}} U_{O_t}(\Phi_y(y) - \Phi_y(y') + \Phi_t(x, y, y'))$$

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Experiments

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Large-scale QA

- Open question answering (QA) on a 14M statement dataset (Fader et al. (2013))

Method	F1
(Fader et al., 2013)	0.54
(Bordes et al., 2014b)	0.73
MemNN (embedding only)	0.72
MemNN (with BoW features)	0.82

Method	Embedding F1	Embedding + BoW F1	Candidates (speedup)
MemNN (no hashing)	0.72	0.82	14M (0x)
MemNN (word hash)	0.63	0.68	13k (1000x)
MemNN (cluster hash)	0.71	0.80	177k (80x)

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Simulated world QA

Joe went to the kitchen. Fred went to the kitchen. Joe picked up the milk.
Joe travelled to the office. Joe left the milk. Joe went to the bathroom.
Where is the milk now? A: office
Where is Joe? A: bathroom
Where was Joe before the office? A: kitchen

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

Simulated world QA

	Difficulty 1			Difficulty 5	
Method	actor w/o before	actor	actor+object	actor	actor+object
RNN	100%	60.9%	27.9%	23.8%	17.8%
LSTM	100%	64.8%	49.1%	35.2%	29.0%
MemNN $k = 1$	97.8%	31.0%	24.0%	21.9%	18.5%
MemNN $k = 1$ (+time)	99.9%	60.2%	42.5%	60.8%	44.4%
MemNN $k = 2$ (+time)	100%	100%	100%	100%	99.9%

J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015

References

1. A. M. Turing, 'Systems of logic based on ordinals', *Proceedings of the London Mathematical Society*, Series 2, 45, pp. 161-228, 1939
2. H. T. Siegelmann and E. D. Sontag, 'On the computational power of neural nets', *Journal of Computer and System Sciences*, 50(1), pp. 132-150, 1995, doi: 10.1006/jcss.1995.1013
3. H. T. Siegelmann and E. D. Sontag, 'Analog computation via neural networks', *Theoretical Computer Science*, vol. 131, no. 2, pp. 331–360, Sep. 1994, doi: 10.1016/0304-3975(94)90178-3
4. J. L. Balcazar, R. Gavalda, and H. T. Siegelmann, 'Computational power of neural networks: a characterization in terms of Kolmogorov complexity', *IEEE Trans. Inform. Theory*, vol. 43, no. 4, pp. 1175–1183, Jul. 1997, doi: 10.1109/18.605580
5. J. Kilian and H. T. Siegelmann, 'The Dynamic Universality of Sigmoidal Neural Networks', *Information and Computation*, vol. 128, no. 1, pp. 48–56, Jul. 1996, doi: 10.1006/inco.1996.0062
6. S. Chung and H. T. Siegelmann, 'Turing Completeness of Bounded-Precision Recurrent Neural Networks', in *Advances in Neural Information Processing Systems*, 2021, vol. 34, pp. 28431–28441. Accessed: May 08, 2022. [Online]. Available: https://proceedings.neurips.cc/paper/2021/hash/ef452c63f81d0105dd4486f775adec81-Abstract.html
7. A. Graves, G. Wayne, and I. Danihelka, 'Neural Turing Machines', *arXiv:1410.5401 [cs]*, Dec. 2014, Accessed: Mar. 05, 2022. [Online]. Available: http://arxiv.org/abs/1410.5401
8. J. Weston, S. Chopra, and A. Bordes, 'Memory Networks', *arXiv:1410.3916 [cs, stat]*, Nov. 2015, Accessed: Mar. 05, 2022. [Online]. Available: http://arxiv.org/abs/1410.3916
9. A. Fader, L. Zettlemoyer, and O. Etzioni, 'Paraphrase-driven learning for open question answering', in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers),* pp. 1608-1618, Aug. 2013.