Neural Reasoning

Daniele Di Sarli

Intelligent Systems for Pattern Recognition, University of Pisa

Computation Time

Similar problems may require very different computation time to be solved.

- sorting **10** numbers
- sorting **10k** numbers

However, most ML algorithms will try to solve both problems in a **similar amount of time**.

Observation

With an input sequence of N numbers and a **fixed** cost of H for each step, a RNN takes $O(NH) = O(N)$ time to **sort** the whole sequence. Optimal alg. is $\Omega(N \log N)$.

Since RNNs are able to **simulate a Turing Machine**[4], there must be room for improvement.

Adaptive Computation Time

ACT: general idea

Idea: what if we could make the network **ponder** for a variable (*learned*) amount of time at each input step, before emitting the output? [2]

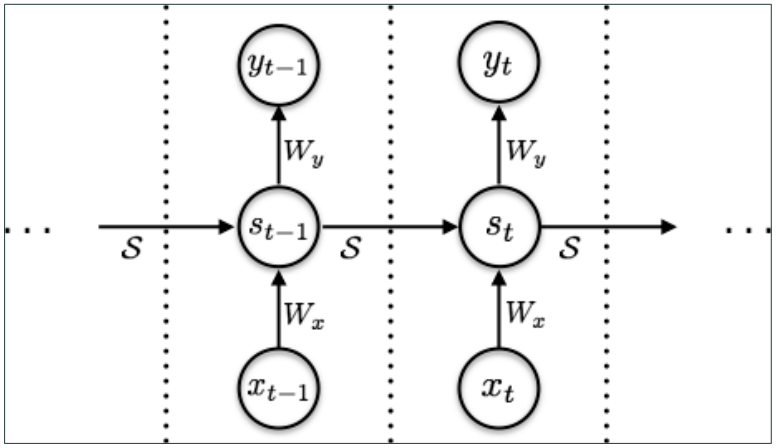

Figure 1: Unrolled RNN

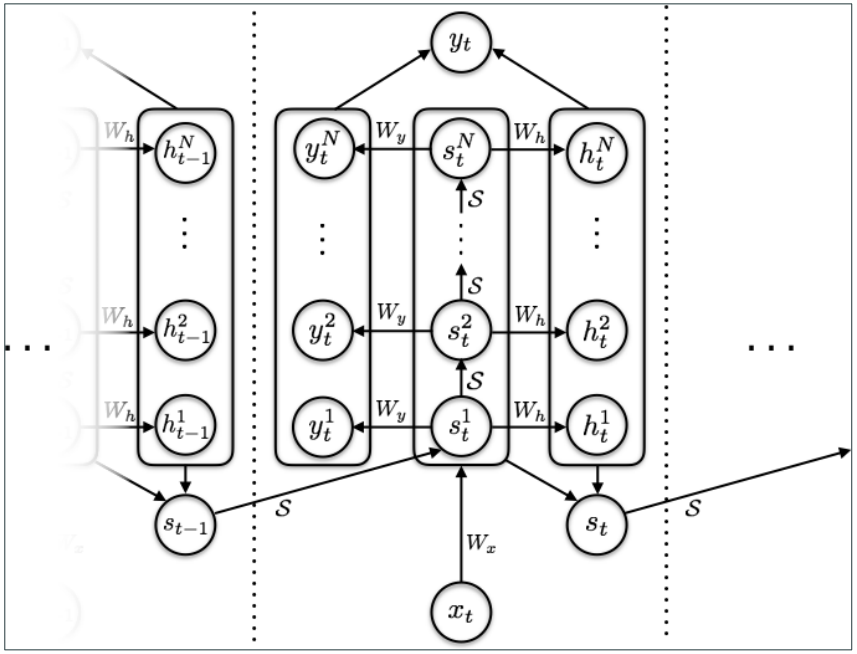

Figure 2: Unrolled RNN with ACT

ACT: intermediate states

Standard RNN

$$s_t = \mathcal{S}(s_{t-1}, W_x x_t)$$
$$y_t = W_y s_t + b_y$$

RNN with ACT

$$s_t^n = \begin{cases} \mathcal{S}(s_{t-1}, x_t^1) & \text{if } n = 1 \\ \mathcal{S}(s_t^{n-1}, x_t^n) & \text{otherwise} \end{cases}$$
$$y_t^n = W_y s_t^n + b_y$$

$$1 \leq n \leq N(t)$$
$$x_t^n = x_t + \delta_{n,1}$$

How to decide $N(t)$?
How to compute s_t and y_t?

ACT: halting units

A **halting unit** h_t^n is used to decide, given the current *intermediate state*, the **halting probability** p_t^n of that *intermediate step*.

Halting unit
$$h_t^n = \sigma(W_h s_t^n + b_h) \qquad \in (0, 1)$$

Halting prob.
$$p_t^n = \begin{cases} R(t) & \text{if } n = N(t) \\ h_t^n & \text{otherwise} \end{cases} \qquad \in (0, 1)$$

\# steps
$$N(t) = \min\{n' : \sum_{n=1}^{n'} h_t^n \geq 1 - \epsilon\} \qquad \in \mathbb{N}^+$$

Remainder
$$R(t) = 1 - \sum_{n=1}^{N(t)-1} h_t^n \qquad \in (0, 1)$$

ACT: halting units, an insight

Halting prob. $$p_t^n = \begin{cases} R(t) & \text{if } n = N(t) \\ h_t^n & \text{otherwise} \end{cases} \in (0, 1)$$

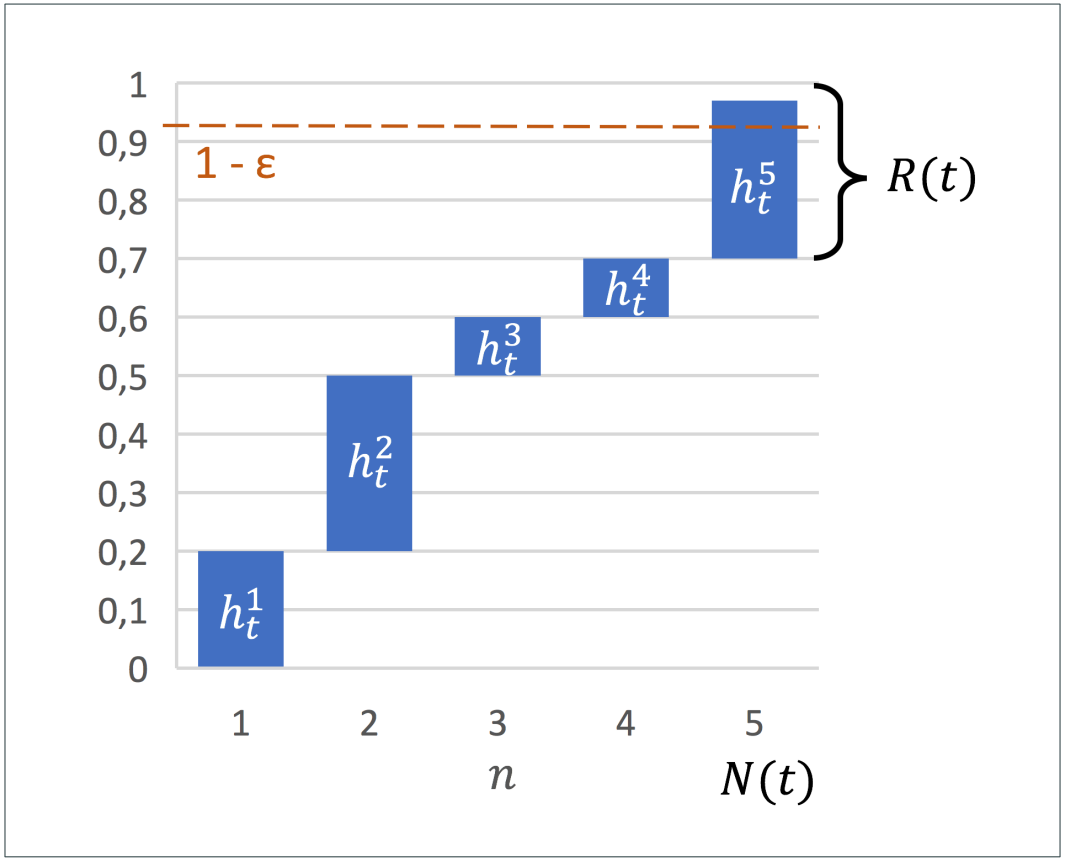

Figure 3: Halting probabilities

ACT: final state and output

We still need to compute s_t and y_t.

A natural choice would be:

- **sample** $\hat{n}$ from p_t^n
- set $s_t = s_t^{\hat{n}}$, $y_t = y_t^{\hat{n}}$

Sampling is non-differentiable!

What we do instead:

$$s_t = \sum_{n=1}^{N(t)} p_t^n s_t^n \qquad y_t = \sum_{n=1}^{N(t)} p_t^n y_t^n$$

ACT results: Sorting Task

Figure 4: Sorting task training example

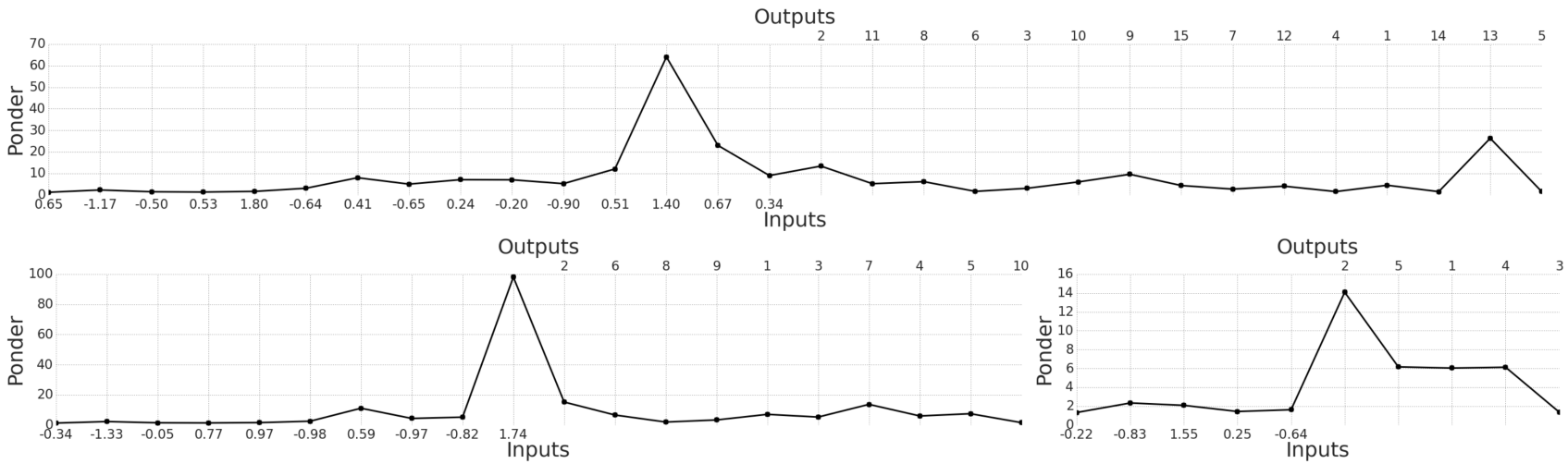

Figure 5: Ponder time during sorting

ACT results: Wikipedia Character Prediction

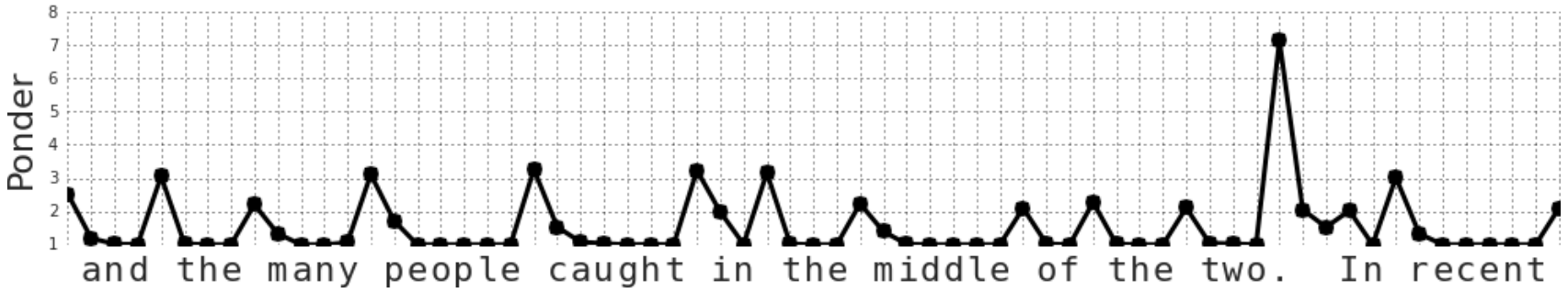

Figure 6: Ponder time

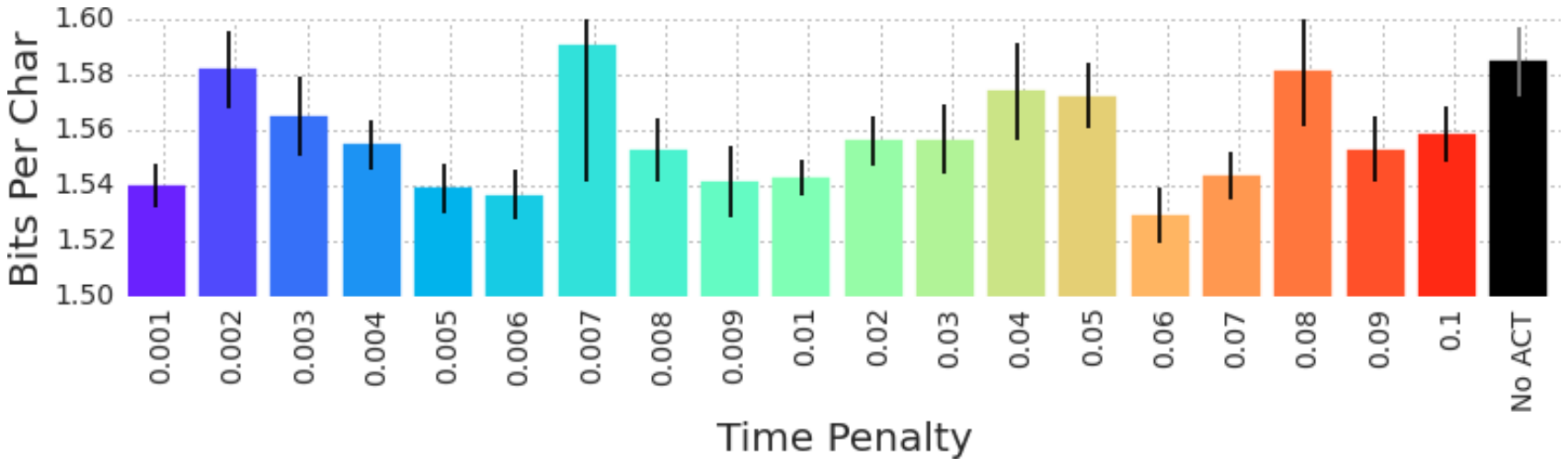

Figure 7: Error rates w.r.t time penalty

ACT results: Addition Task

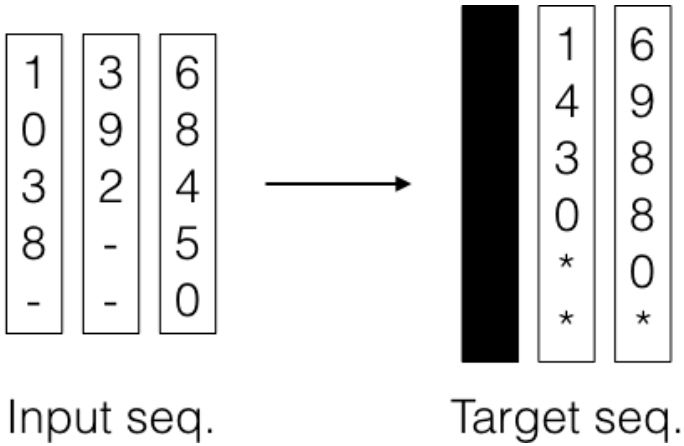

Figure 8: Training example. Each input digit is represented by a size 10 one hot encoding.

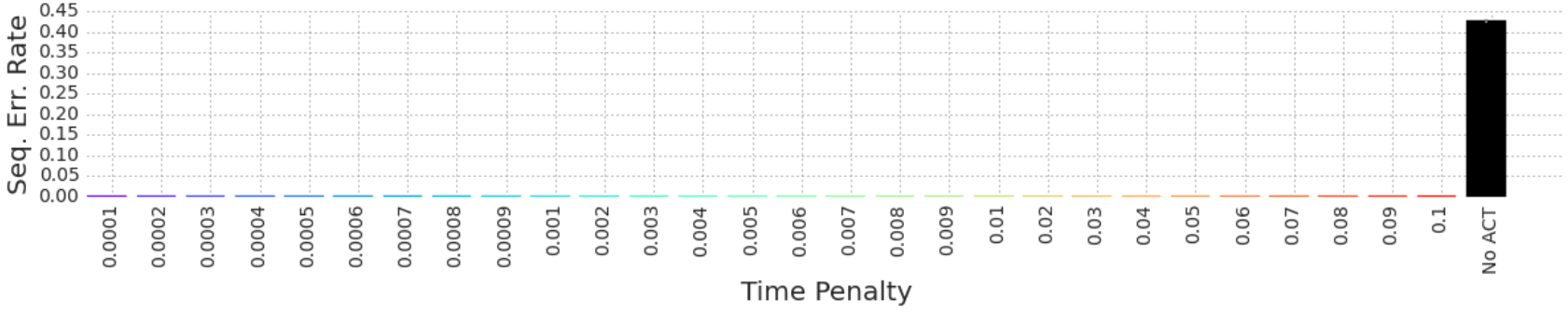

Figure 9: Error rates w.r.t time penalty

ACT for Textual Entailment Recognition

Neumann et al. (2016) showed the usage of **ACT** with a more advanced model exploiting **attention**. [3]

Textual Entailment examples:

"Jack is arguing with John."
⇓ *entails*
"John and Jack are in disagreement."

"Jack is arguing with John."
⇊ *contradicts*
"John believes that Jack is right."

"Jack is arguing with John."
? *neutral*
"Everyone is at the beach."

Textual Entailment Recognition, example i

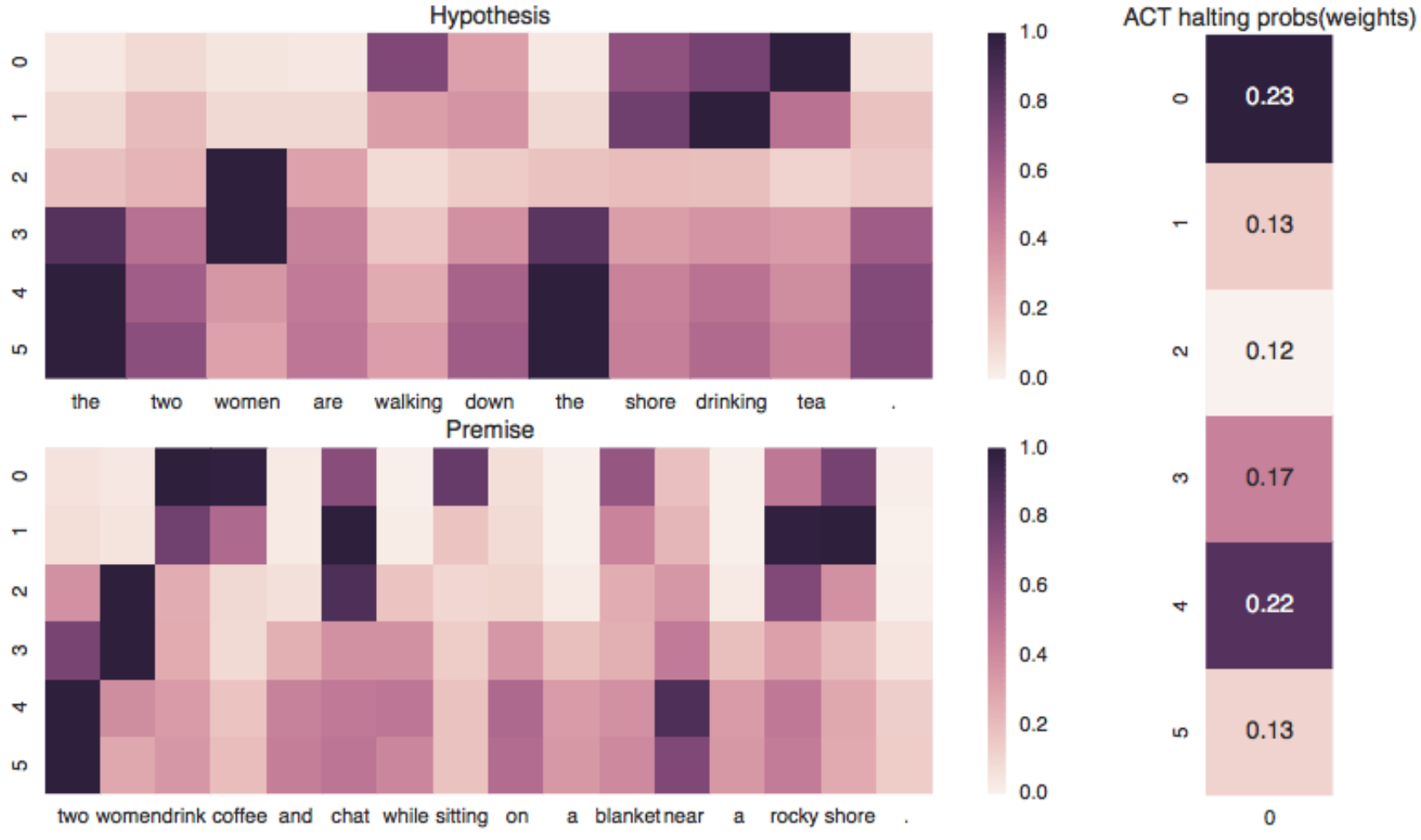

Figure 10: Attention weights at each inference step

Textual Entailment Recognition, example ii

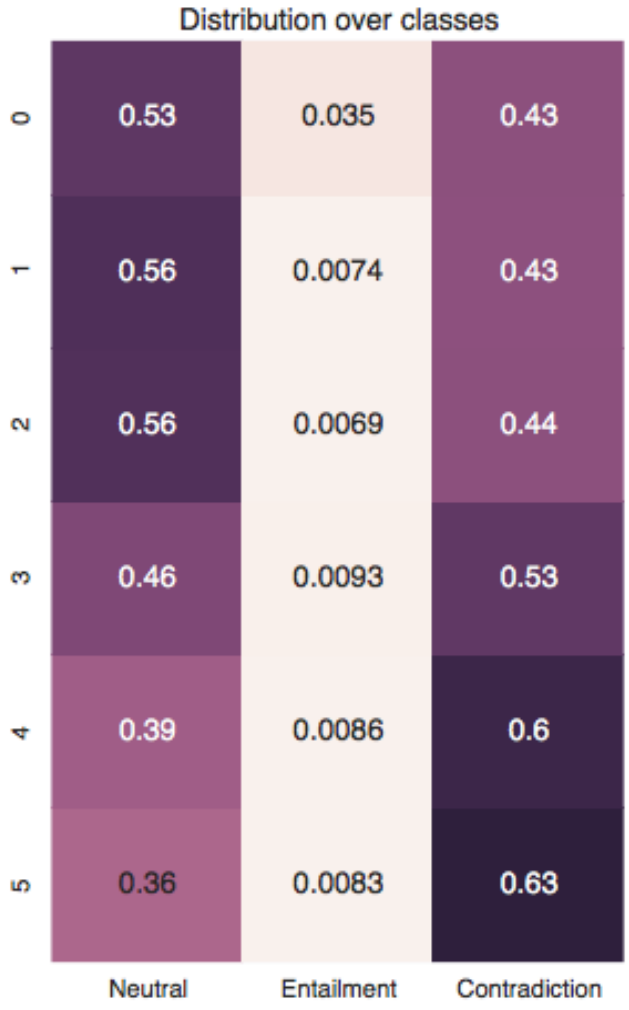

Figure 11: Network intermediate outputs at each inference step

Reasoning for Scene Understanding

Attend, Infer, Repeat

A different approach for **reasoning** within **variational autoencoders**, by Eslami et al. (2016) [1]

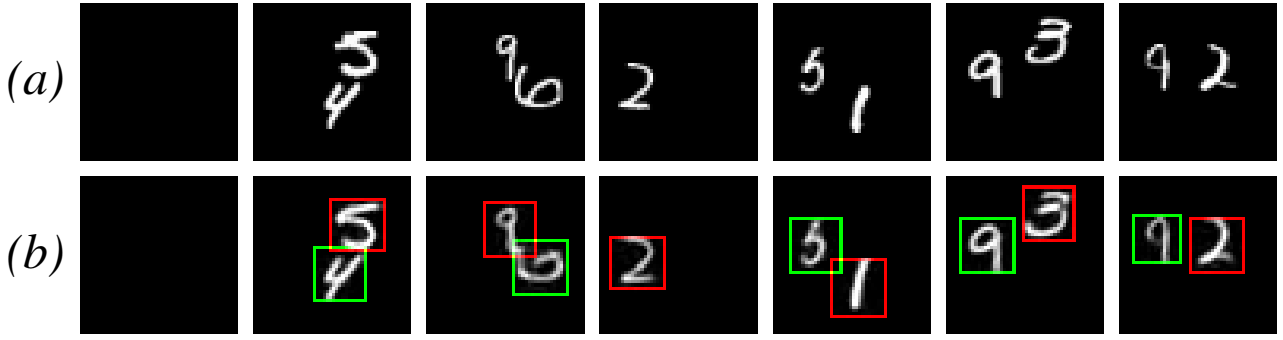

Figure 12: a) input; b) reconstructed output (annotated)

If $\mathbf{x}$ is the image and $\mathbf{z}$ its latent description, our model is

$$p_\theta^x(\mathbf{x}|\mathbf{z})p_\theta^z(\mathbf{z})$$

and we're interested in $p(\mathbf{z}|\mathbf{x}) = p_\theta^x(\mathbf{x}|\mathbf{z})p_\theta^z(\mathbf{z})/p(\mathbf{x})$

AIR: model i

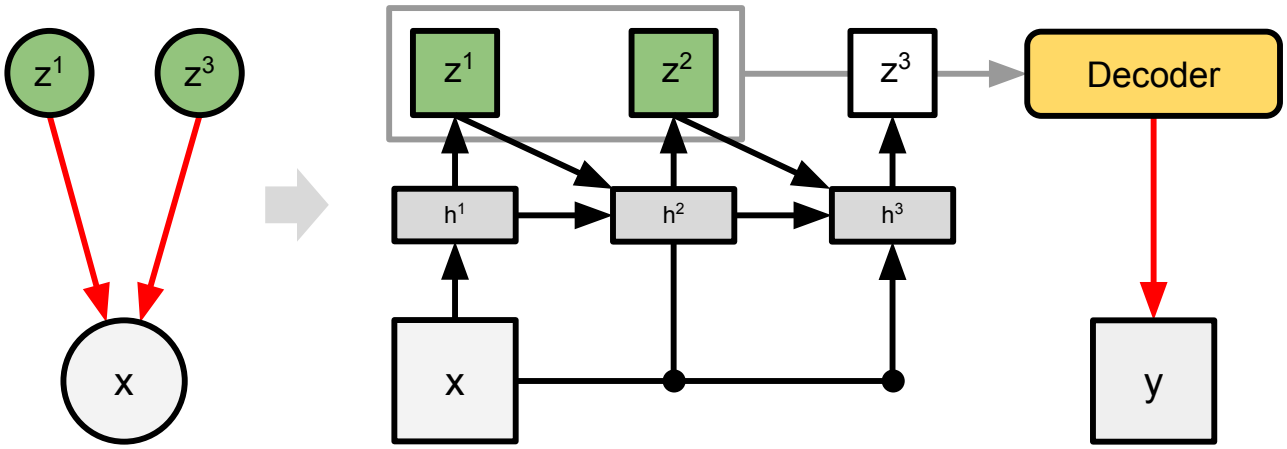

Figure 13: Black arrows: **inference network**. Red arrows: **generative model**.

We want to 1) **interpret** the model, 2) use **one step per digit**.

So we assume $\mathbf{z} = (\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^n)$ (one per digit)

We also assume $\mathbf{z}^i = (\mathbf{z}^i_{\text{what}}, \mathbf{z}^i_{\text{where}})$

And define $\mathbf{z}_{\text{pres}}$ as the variable length latent vector $(\overbrace{1\ 1\ \cdots\ 1}^{n}\ 0)$

AIR: model i

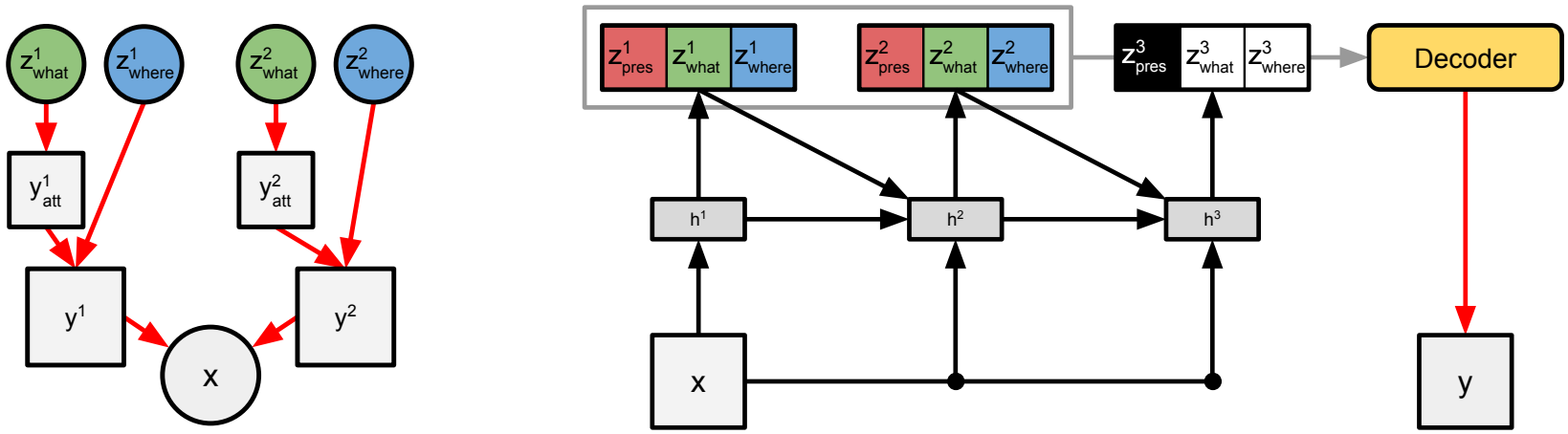

Figure 13: Black arrows: **inference network**. Red arrows: **generative model**.

We want to 1) **interpret** the model, 2) use **one step per digit**.

So we assume $\mathbf{z} = (\mathbf{z}^1, \mathbf{z}^2, \ldots, \mathbf{z}^n)$ (one per digit)

We also assume $\mathbf{z}^i = (\mathbf{z}^i_{\text{what}}, \mathbf{z}^i_{\text{where}})$

And define $\mathbf{z}_{\text{pres}}$ as the variable length latent vector $(\overbrace{1\ 1\ \cdots\ 1}^{n}\ 0)$

AIR: model ii

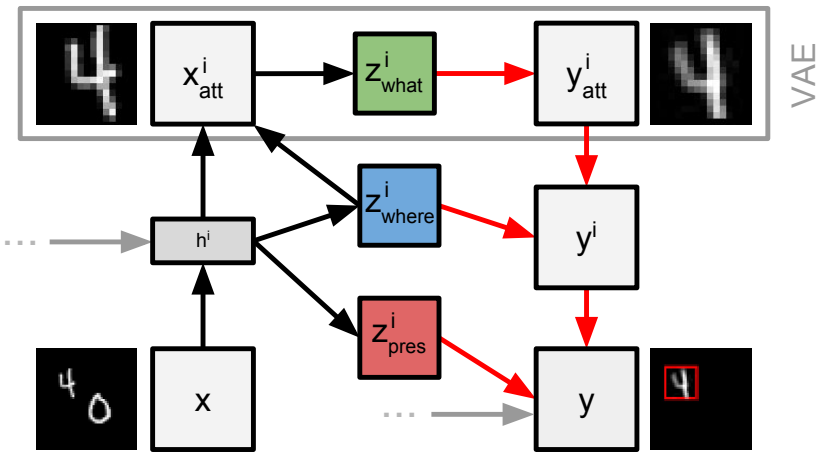

Figure 14: Interaction between **inference** and generator networks at every time-step.

AIR: learning

$$p(\mathbf{z}|\mathbf{x}) = p_\theta^x(\mathbf{x}|\mathbf{z})p_\theta^z(\mathbf{z}) \;\; / \;\; \underbrace{p(\mathbf{x})}_{\text{intractable}}$$

Approximate $p(\mathbf{z}|\mathbf{x})$ by learning a distribution $q_\phi(\mathbf{z}, n\,|\mathbf{x})$ that minimizes $\mathrm{KL}[q_\phi(\mathbf{z}, n\,|\mathbf{x}) \,||\, p_\theta^z(\mathbf{z}, n\,|\mathbf{x}))]$

Actual shape of the posterior:

$$q_\phi(\mathbf{z}, \mathbf{z}_{\text{pres}}\,|\mathbf{x}) = q_\phi(z_{\text{pres}}^{i+1} = 0|\mathbf{x}) \prod_{i=1}^{n} q_\phi(\mathbf{z}^i, z_{\text{pres}}^i = 1|\mathbf{x}, \mathbf{z}^{1:i-1})$$

For learning, **jointly optimize** ϕ (inference net.) and θ (model) by **maximizing** the **ELBO**:

$$\log p_\theta(\mathbf{x}) \geq \mathcal{L}(\theta, \phi) = \mathbb{E}_{q_\phi}\left[\log \frac{p_\theta(\mathbf{x}, \mathbf{z}, n)}{q_\phi(\mathbf{z}, n\,|\mathbf{x})}\right]$$

AIR: results

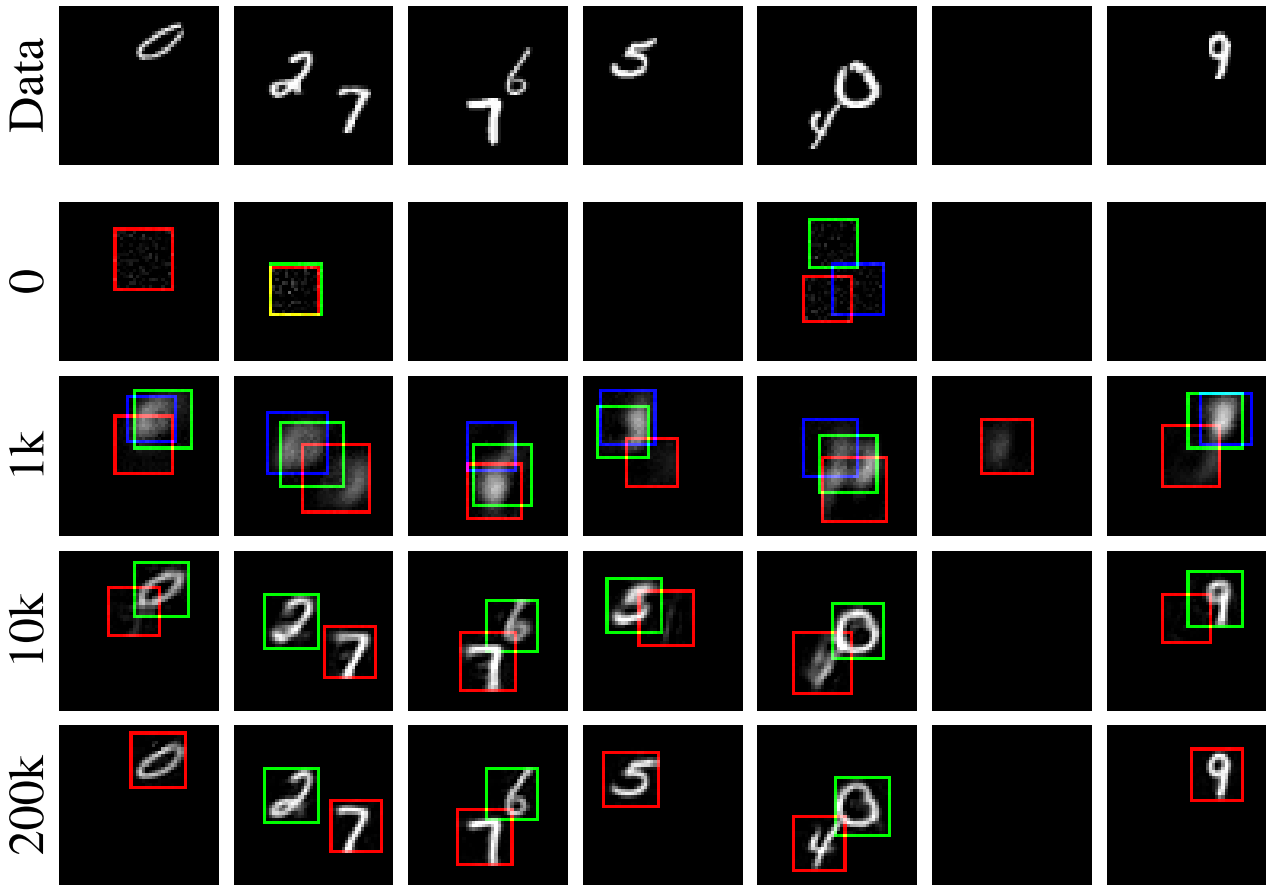

Figure 15: Reconstruction at different training stages.

Conclusion

Summary

Adaptive computation time is the norm in **classical algorithms** (and biology!).

ACT:

- elegant way to **augment any RNN**
- ponder over the **complexity** of the task

AIR:

- **generative** model
- specific to **images**
- "ponder" over the **structure** of the task

Bibliography

[1] S. A. Eslami, N. Heess, T. Weber, Y. Tassa, D. Szepesvari, G. E. Hinton, et al.
Attend, infer, repeat: Fast scene understanding with generative models.
In *Advances in Neural Information Processing Systems*, pages 3225–3233, 2016.

[2] A. Graves.
Adaptive computation time for recurrent neural networks.
arXiv preprint arXiv:1603.08983, 2016.

[3] M. Neumann, P. Stenetorp, and S. Riedel.
Learning to reason with adaptive computation.
arXiv preprint arXiv:1610.07647, 2016.

[4] H. T. Siegelmann and E. D. Sontag.
On the computational power of neural nets.
In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 440–449. ACM, 1992.