DISCRETIZING NEURAL TURING MACHINES

James Gleeson & Grant Watson

NTMS (1): QUICK OVERVIEW

<u>NTMs:</u> a fully differentiable computer!

Training input = program input
Training output = desired program output

Learn the *<u>locations of memory</u>* to read/write

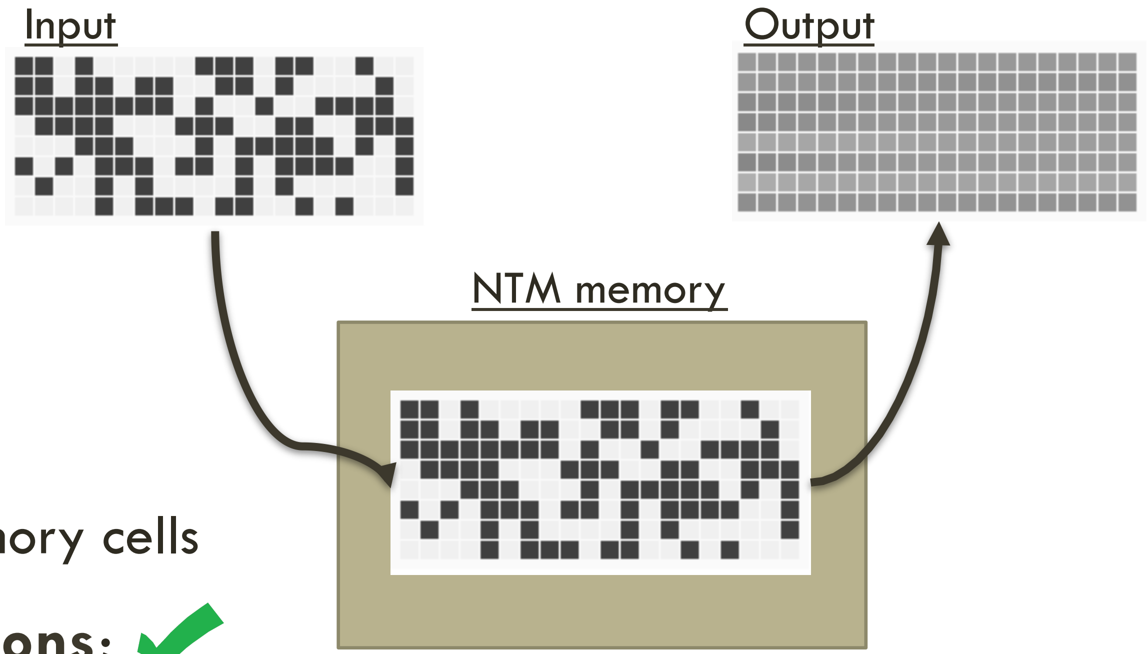
<u>Memory read head</u>

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

Soft attention during reads/writes over the NTM memory cells

- NTM is built from **fully differentiable computations;** learnable via backprop:

As we train over many input/output pairs…



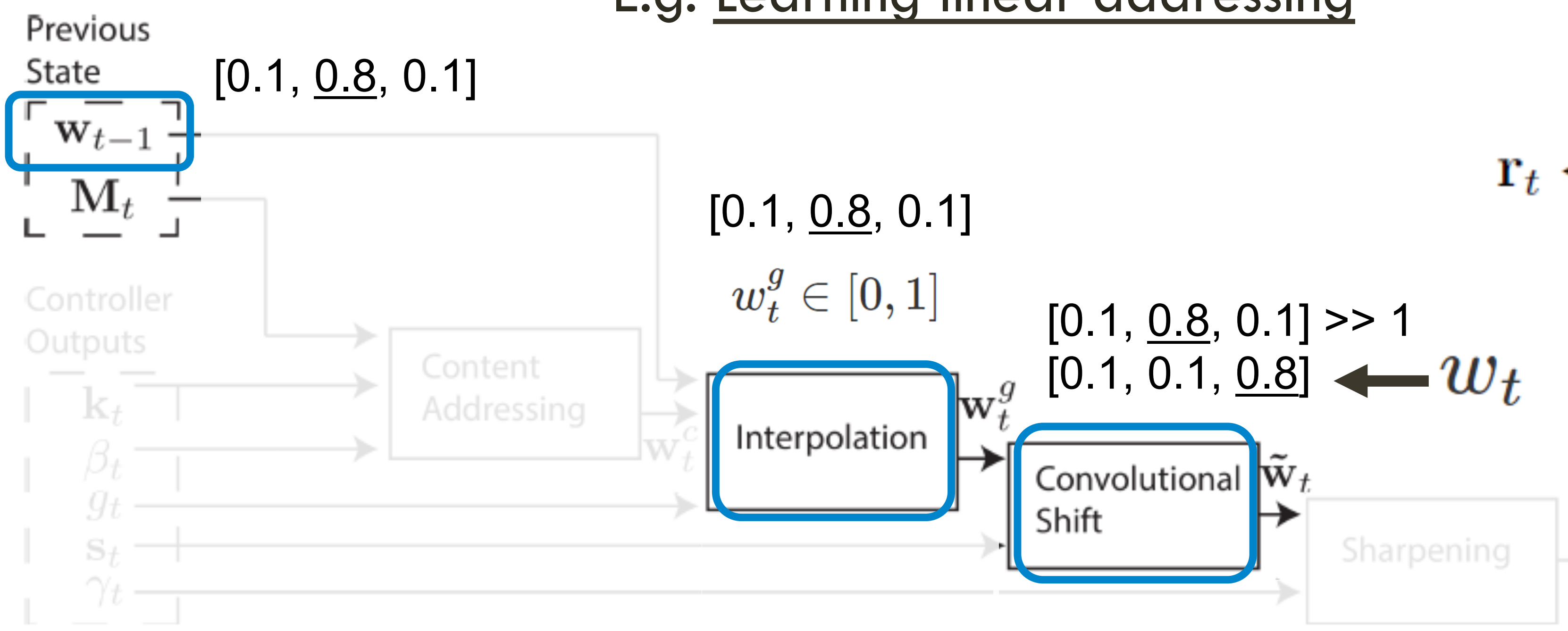
<u>Simple memory copy task:</u>
Learning to ***read/write sequentially***

NTMS (2): LEARNING TO ADDRESS MEMORY

Soft attention during reads/writes over the NTM memory cells

- NTM is built from **fully differentiable computations;** ✓ learnable via backprop:

E.g. <u>Learning linear addressing</u>



<u>Memory read head</u>

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

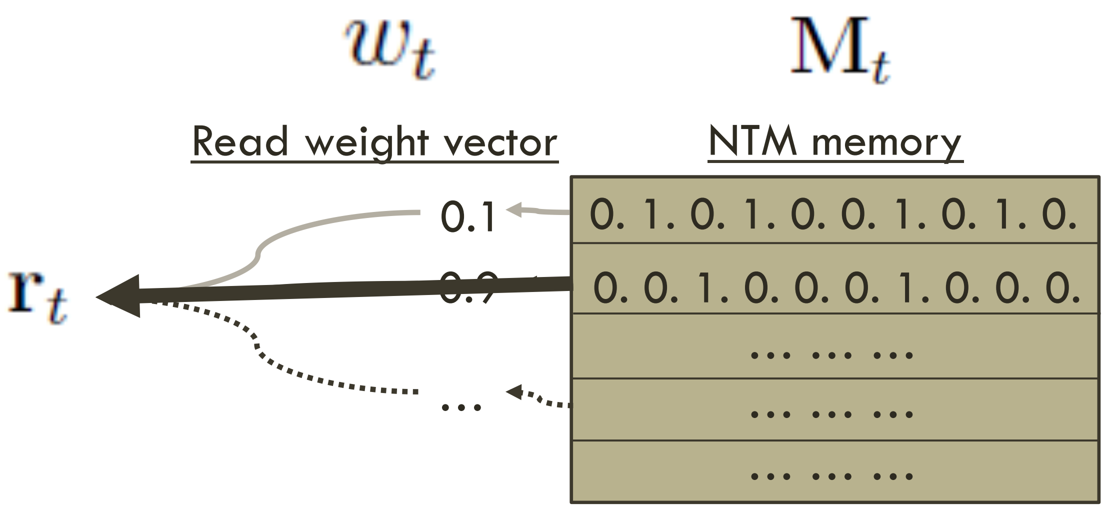
Reads memory at the <u>next</u> location

NTMS (2): LEARNING TO ADDRESS MEMORY

- Each timestep/clock-cycle, the NTM must **read/write all of memory**!

KEY PROJECT IDEA:

Allow NTM to read/write only a single memory cell at each timestep/clock-cycle



Just like a real computer!

Scales better

➔ More suitable to low-level hardware implementations (e.g. FPGAs, TPUs, etc.)

RELATED  RK

D-NTM / dynamic-NTM: (Gulcehre et al., 2017):

Instead of a weighted combination of memory cells,
sample which memory cell to read/write

$$w_t = [0.1, 0.8, 0.1]$$
$$i \sim Categorical(w_t)$$
$$r_t \leftarrow M_t[i]$$

Limitations:

Sampling makes D-NTM ***not fully differentiable***;

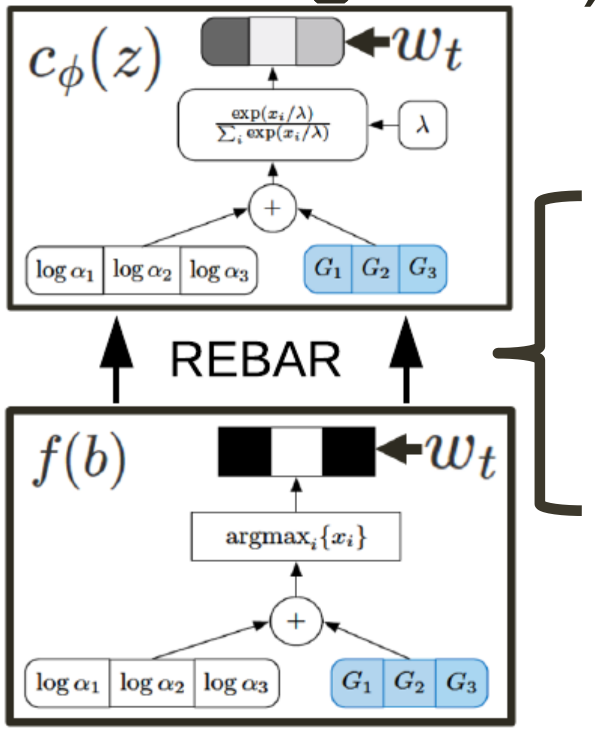
Work-around: they use REINFORCE to train D-NTM.

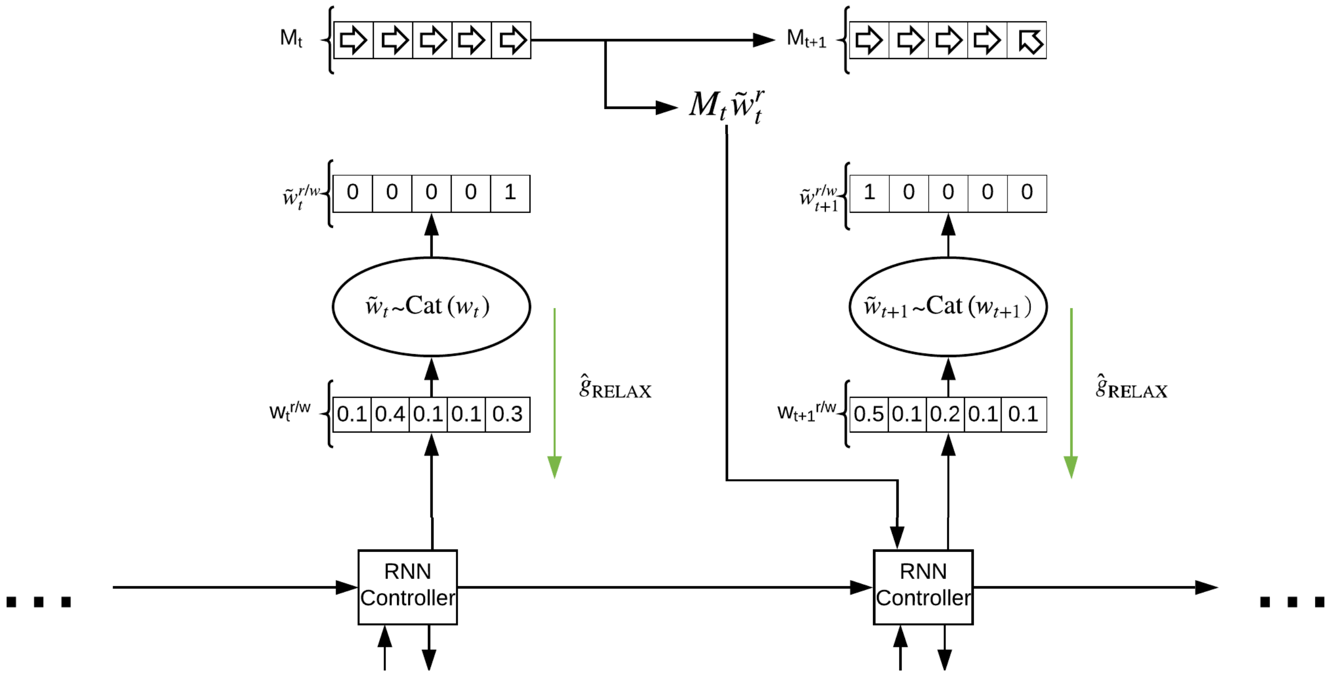
\+ many tweaks to combat this high variance gradient estimator

Other work: TARDIS (Gulcehre et al., 2017), RL-NTM (Zaremba & Sutskever, 2015)

DISCRETIZING NEURAL TURING MACHINES

Figure 1)



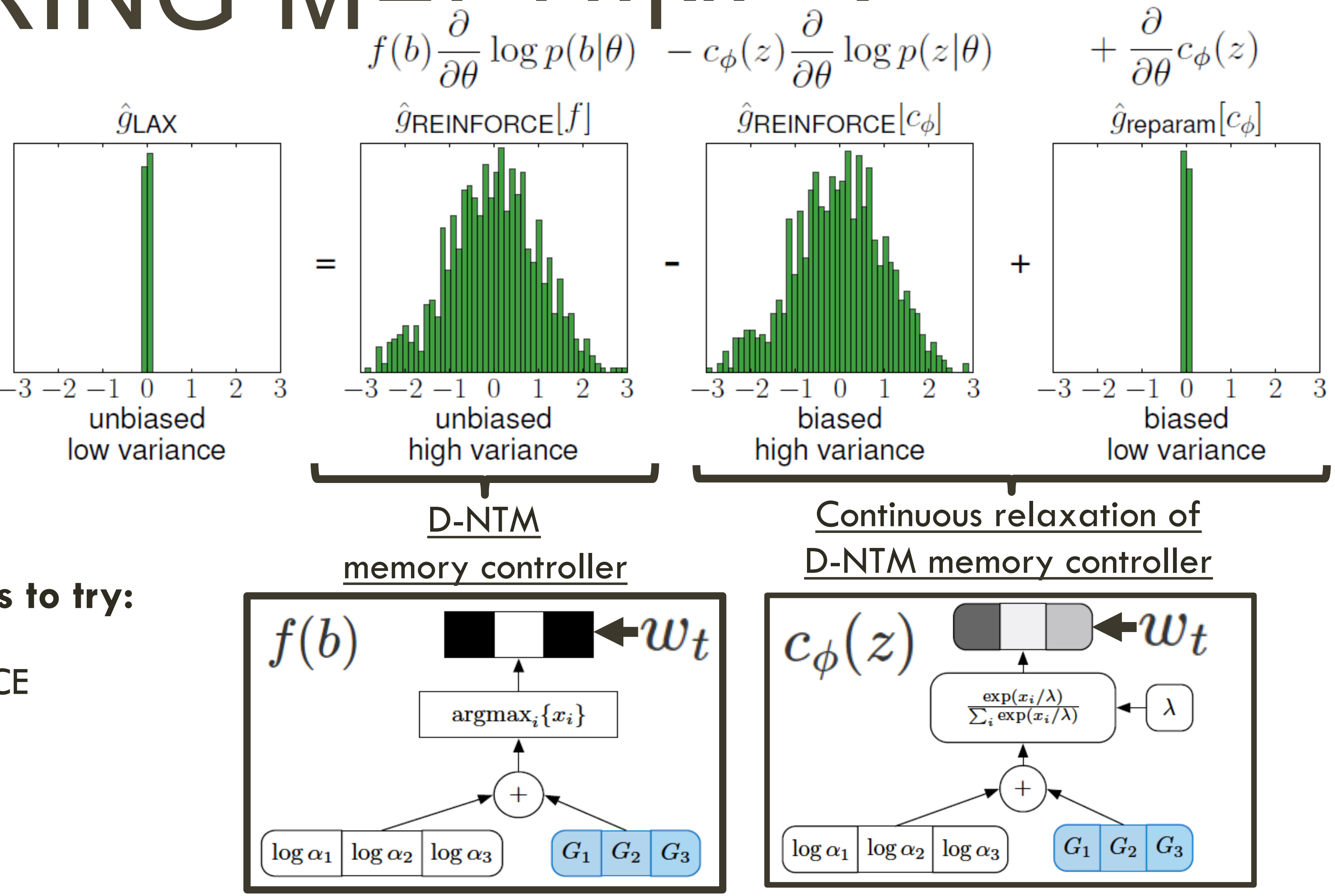


$$\hat{g}_{\text{RELAX}} = [f(b) - c_\phi(\tilde{z})] \frac{\partial}{\partial\theta} \log p(b|\theta) + \frac{\partial}{\partial\theta} c_\phi(z) - \frac{\partial}{\partial\theta} c_\phi(\tilde{z})$$

$$b = H(z), z \sim p(z|\theta), \tilde{z} \sim p(z|b, \theta)$$

DISCRETIZING NEURAL TURING MACHINES

Figure 1)



Gradient estimators to try:

- REINFORCE
- REBAR
- RELAX

PROGRESS  RT!

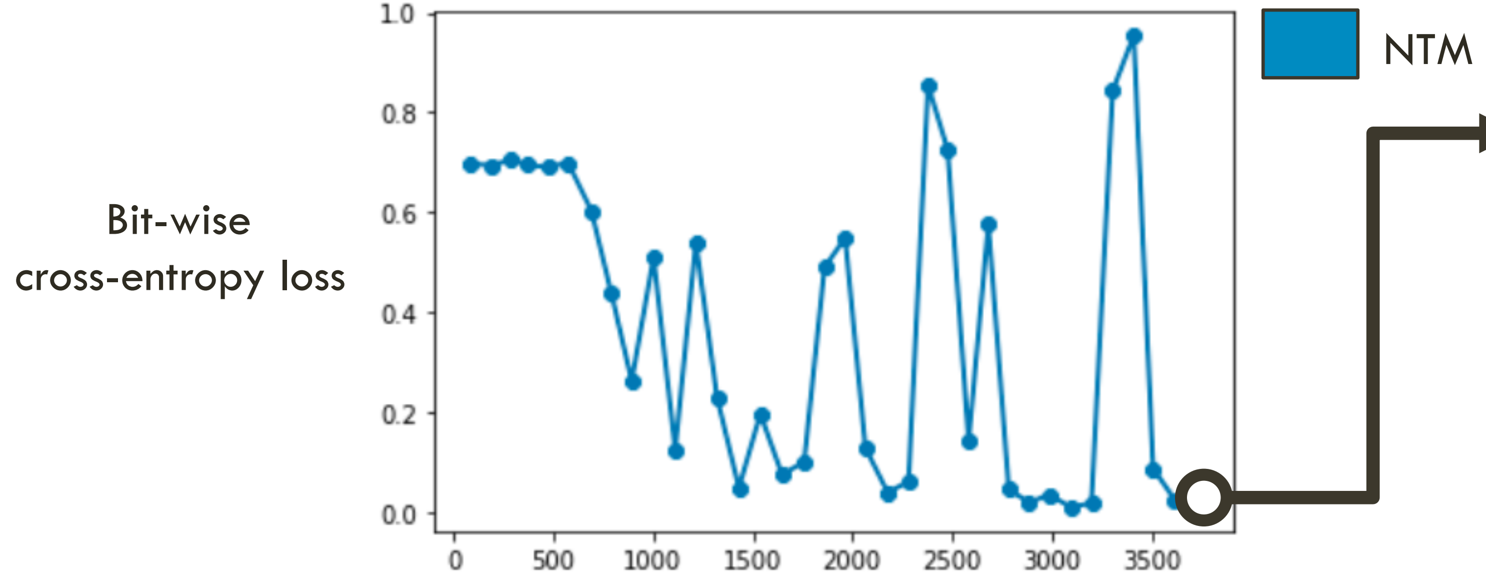
Successfully reproduced prior work:

(1) NTM: copy task performance

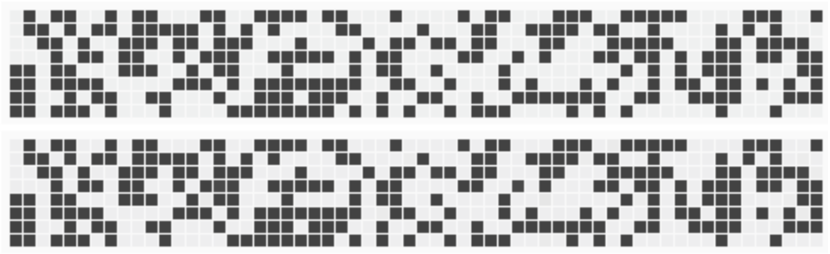
(2) Gradient estimators: applying REINFORCE/RELAX/REBAR to a toy Bernoulli(θ) problem

Training convergence:

Memory copy task



Target output



NTM Output

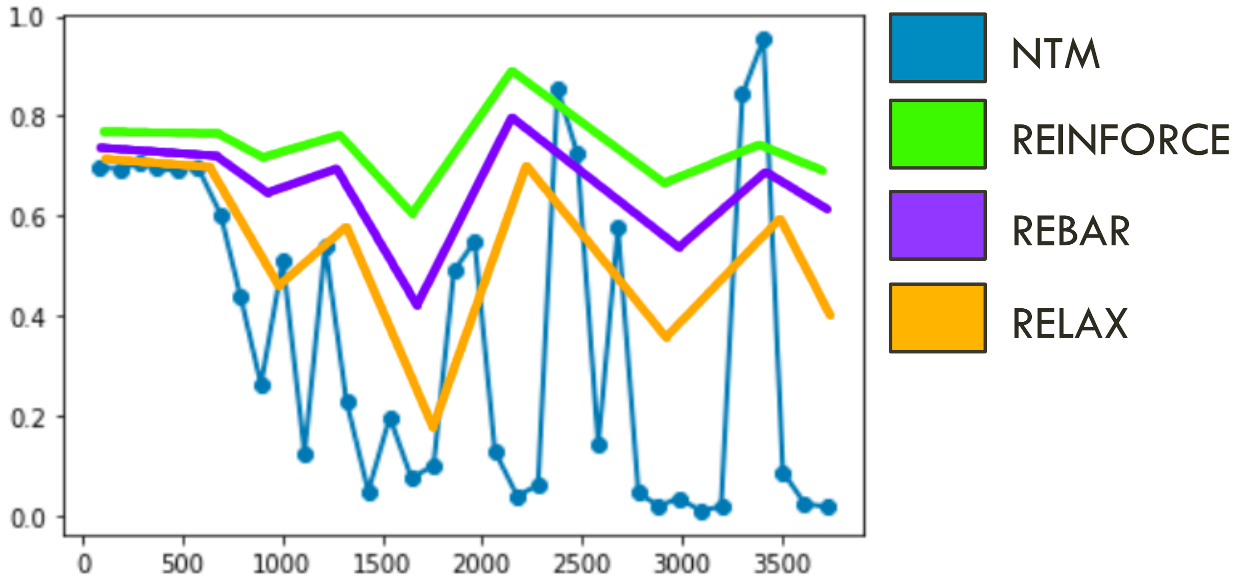
PROGRESS  RT!

Figure 2a) Memory copy task

<u>Training convergence:</u>

Memory copy task

Bit-wise
cross-entropy loss



Training time (seconds)

Sanity check.

Simple task that the NTM should be able to learn quickly.

PROGRESS  RT!

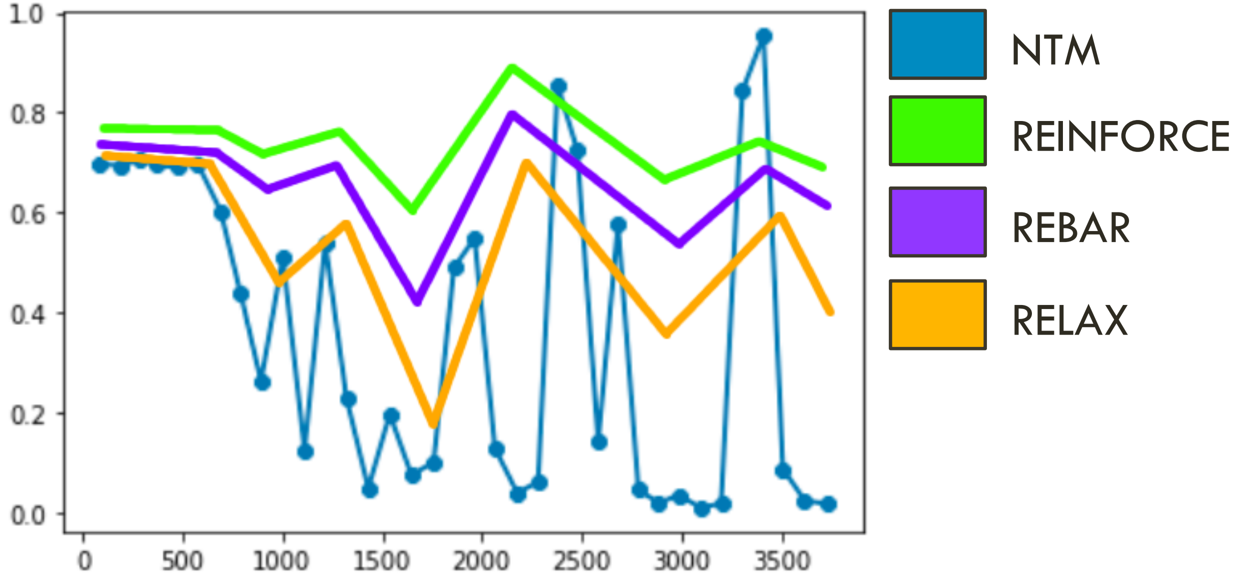
e.g.

Task 15: Basic Deduction
Sheep are afraid of wolves.
Cats are afraid of dogs.
Mice are afraid of cats.
Gertrude is a sheep.
What is Gertrude afraid of? A:wolves

Figure 2b) bAbI question/answer task

Training convergence:
bAbI question/answer task

Accuracy (%)



Training time (seconds)

Can NTM be trained on difficult tasks with better accuracy than REINFORCE?