AIDAN GOMEZ

The Neural Turing Machine

16 MAY 2016

This article serves to briefly outline the design of the Neural Turing Machine (NTM), a backpropogatable architecture that can (among many possibilities) learn to dynamically execute programs.

The original paper.

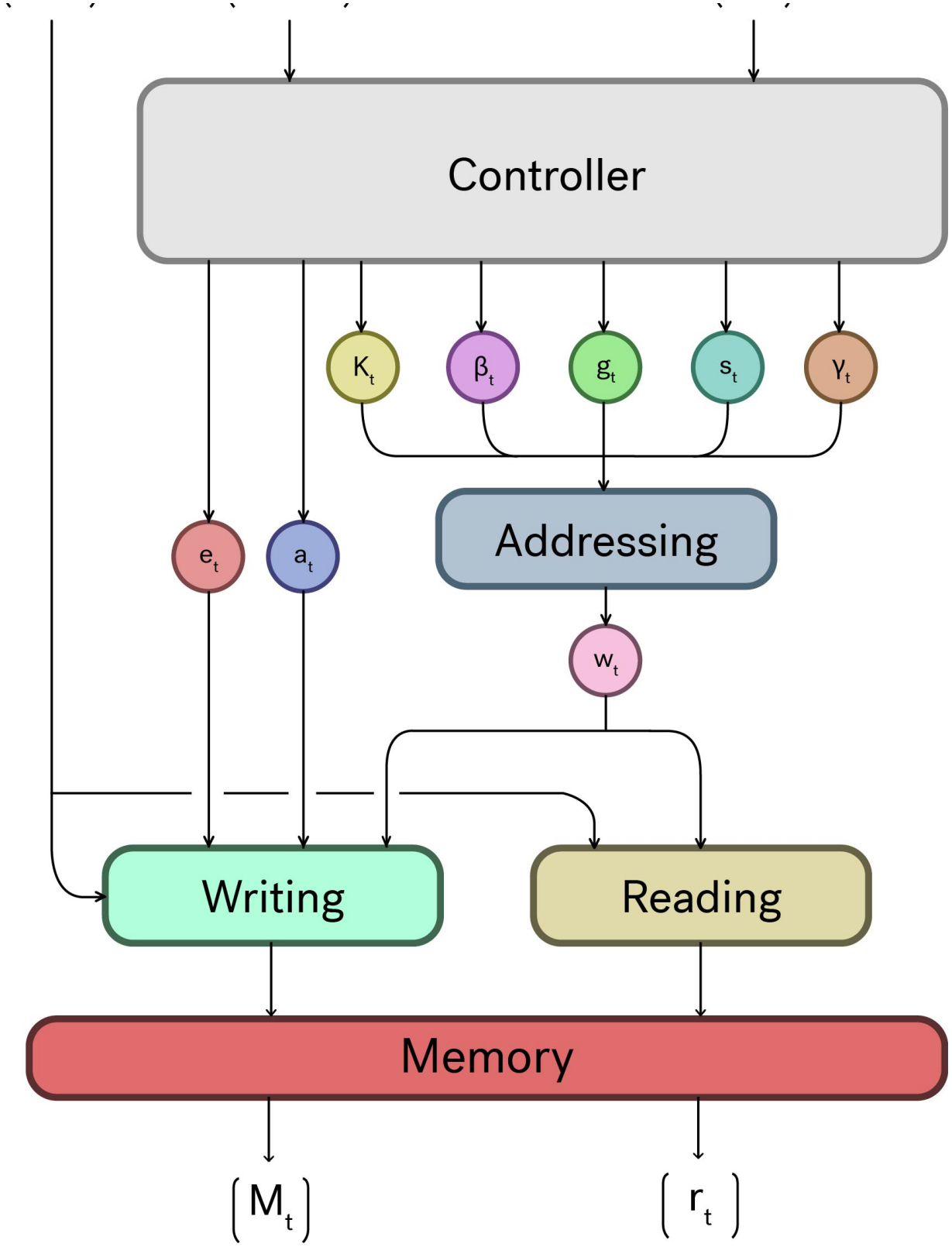
I've added some specifications about the NTM's architecture that the paper excludes for the sake of generality. These will be discussed upon presentation.

The Neural Turing Machine was proposed by Graves et al. as a Turing-Complete network capable of learning (rather complex) programs. Inspired by the sequential nature of the brain, and the large, addressable memory of the computer.

The NTM is composed of five modules:

- The controller
- The addressing module
- The read module
- The write module
- The memory

$$\left(M_{t-1}\right) \qquad \left(\text{Input}\right) \qquad \left(r_{t-1}\right)$$



The Controller

The controller acts as the interface between input data and memory. It

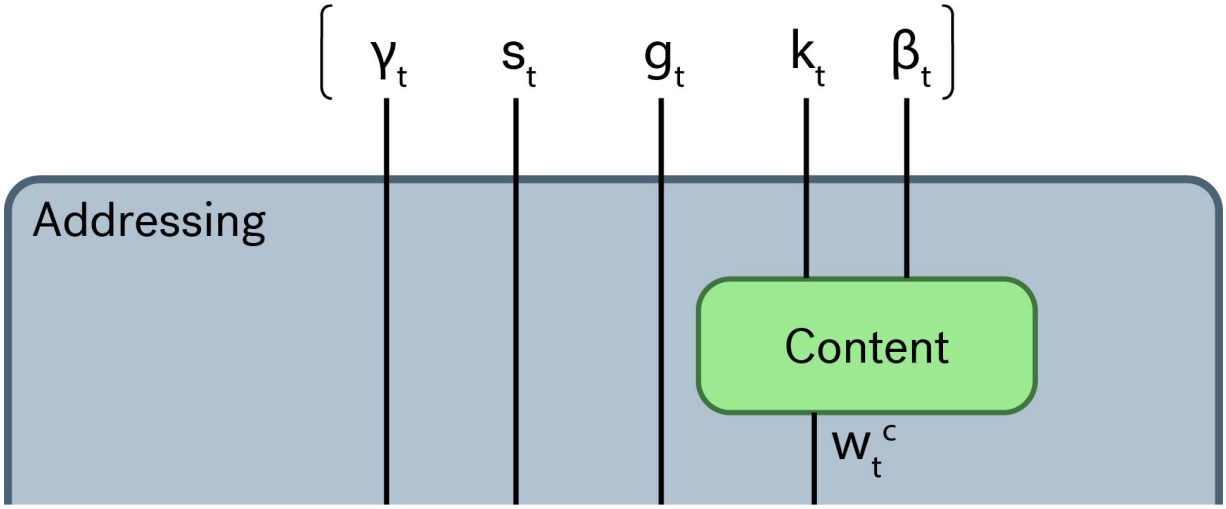
learns to manage its own memory through addressing.

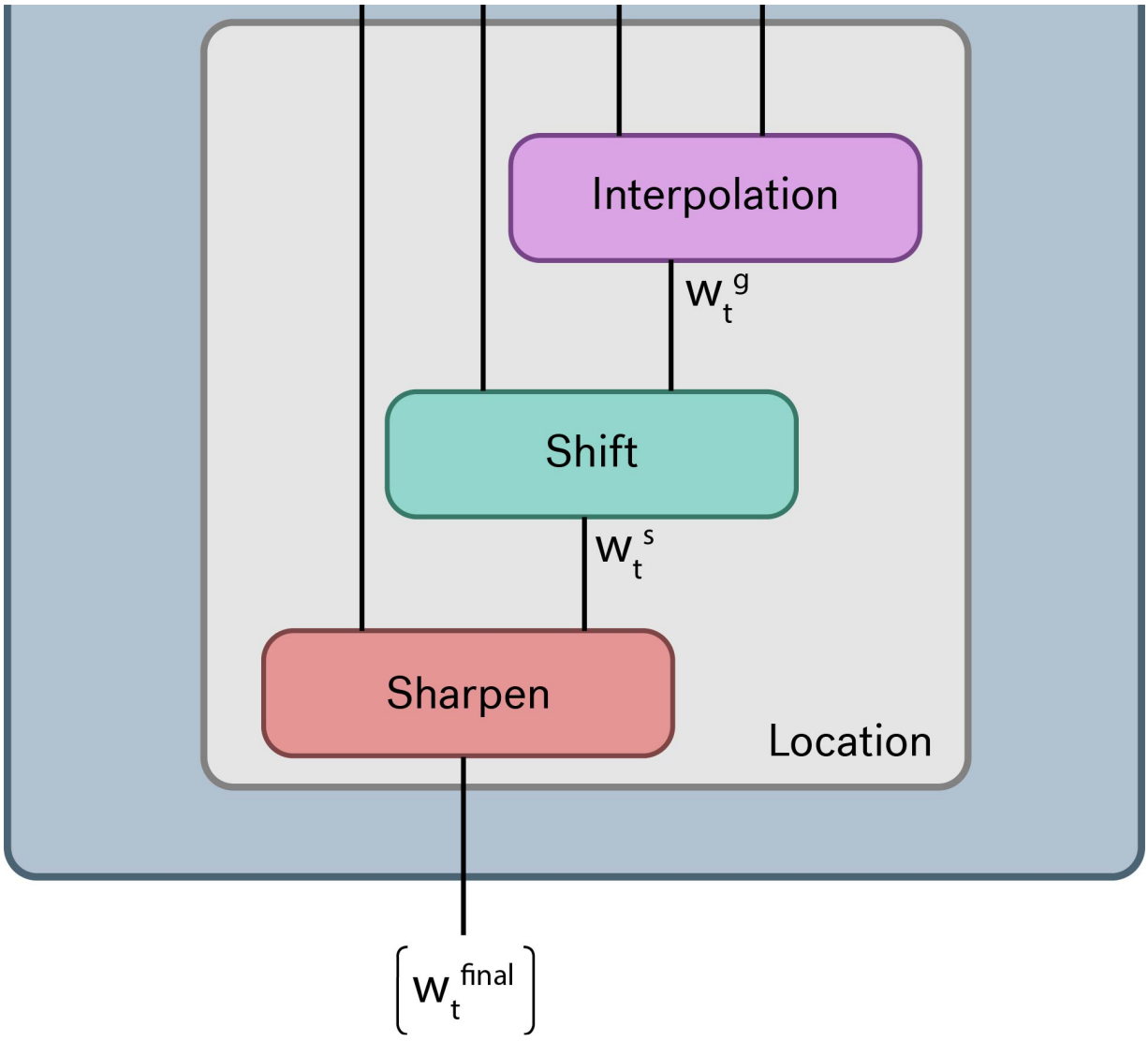
The paper maintains that the controller can be of any format, it simply needs to read in data and produce the outputs required by the sub-modules. They choose an LSTM for their implementation.

The parameters depended upon by the sub-modules are:

- $\mathbf{k_t}$ the key vector; Compared against the memory when addressing by content similarity.
- β_t the key strength; A weighting effecting the precision of the content addressing.
- $g_t \in (0, 1)$ the blending factor; A weight to blend between content addressing and previous time-step addressing.
- $\mathbf{s_t}$ the shift weighting; A normal distribution across the allowed shift amounts.
- γ the sharpening exponent; Serves to sharpen the final address
- $\mathbf{e_t} \in (0, 1)^N$ erase vector; Similar to an LSTM, decides what memory from the previous time-step to erase.
- $\mathbf{a_t}$ add vector; The data to be added to memory.

The Addressing Module





This module generates a window over the memory for the read and write heads.

Accessing by content similarity

This module effectively laps over the memory comparing each block to the key vector $\mathbf{k_t}$ and creates a normally distributed weighting over the memory based of the similarity.

$w_t^c(i) = softmax(\beta_t \delta(K_t, M_t(i)))$ where $\delta(a, b) = \frac{a \cdot b}{|a| \cdot |b|}$ is an example difference function.

Accessing by location

This module has 3 steps:

1. The controller decides how much of the previous time-step's weighting $\mathbf{w}_{t-1}^{final}$ should be preserved (using g_t).

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

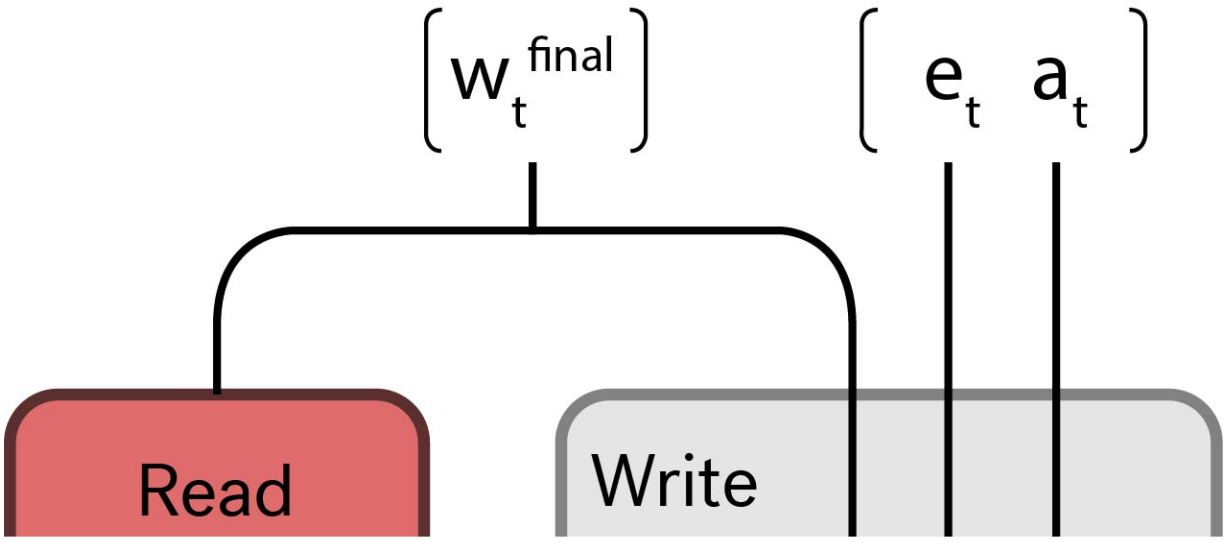
2. It then performs a shift of the weighting (using $\mathbf{s}_t$).

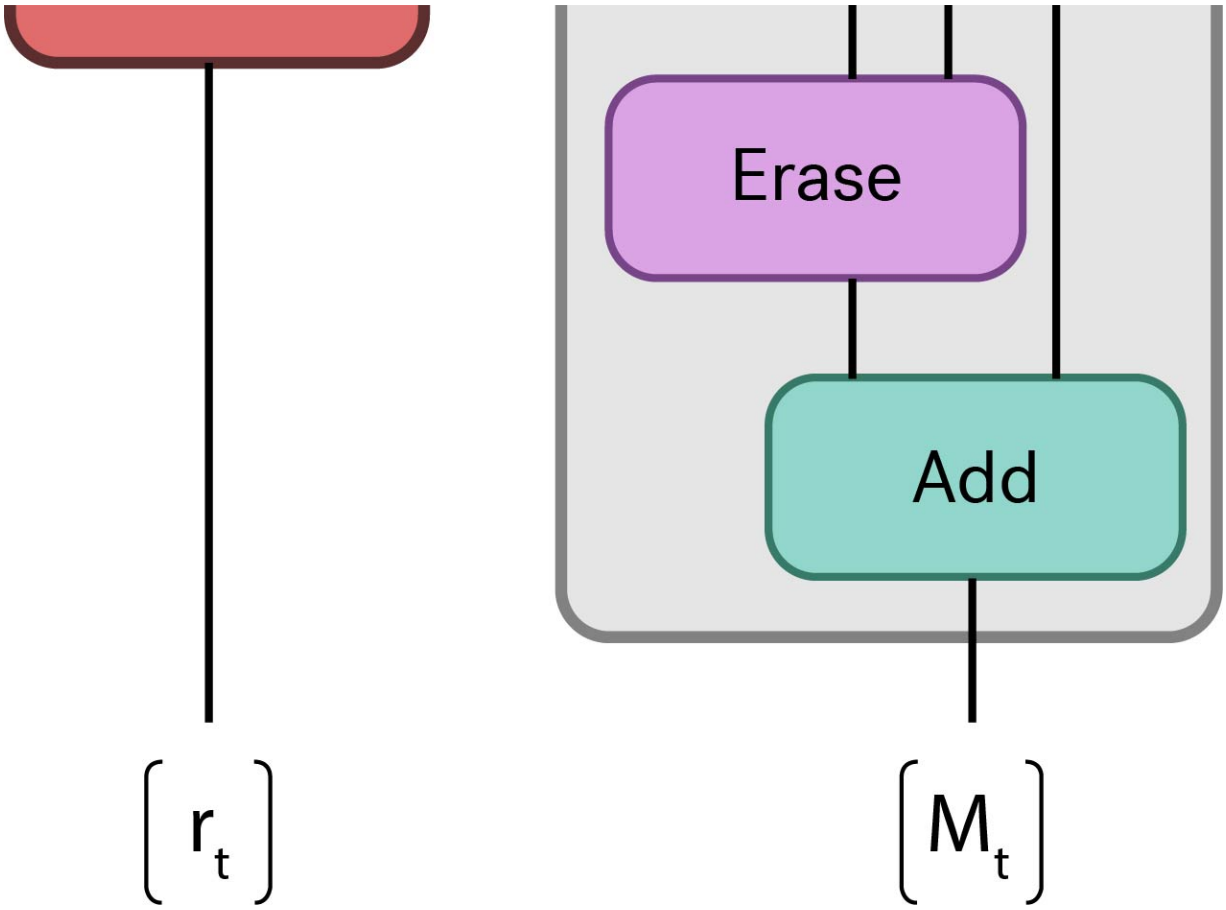
$$w_t^s(i) = \sum_j w_t^g(j) \cdot s_t(i - j)$$

3. Finally it sharpens (using γ_t) and normalizes the weighting.

$$w_t^{final}(i) = \frac{w_t^s(i)^{\gamma_t}}{\sum_j w_t^s(j)^{\gamma_t}}$$

Read & Write





Reading

Pretty straight-forward.

$$\mathbf{r}_t = \sum_i w_t^{final}(i) \cdot \mathbf{M}_t(i)$$

Writing

Writing is performed in two steps, similar to how an LSTM updates its state.

1. The erase vector $\mathbf{e_t}$ removes memory that is no longer relevant.

$$\mathbf{M}_t(i) = \mathbf{M}_{t-1}(i)[1 - w_t^{final}(i) \cdot \mathbf{e}_t]$$

2. The new data is placed in memory (using $\mathbf{a_t}$).

-

$$\mathbf{M}_t(i) = \mathbf{M}_t(i) + w_t^{final}(i) \cdot \mathbf{a}_t$$

Significance

This architecture has already had an impact on a multitude of research projects (notably, the Neural GPU) and I have great faith it will continue to do so. There's been much discussion over the past couple of years about computers programming themselves, and I believe this has been the greatest stride towards that end-goal.

The Neural Turing Machine is a reactionary computer, changing behaviour based on its "environment". It certainly will play a major role in setting precedence for the way neural networks will be applied for the purpose of AI.

I look forward to reading more of the work done by the Google DeepMind and OpenAI teams.

If there are any errors in my description please do not hesistate to reach out to me at hello@aidangomez.ca.