Emulating alpha and beta waves in human brain to achieve attention control in memory

Ahmed_magdy102@yahoo.com

Abstract: Observing the current world of computing ,and the single core upgrade stalling behind the number of cores's growth manufacturing direction ,together with the fact that NNs are unaware of the concept of time(we rely on sequences instead) giving rise to many troubles both on the hardware level not being able to take full benefit of all cores and on the memory-related NNs design level. through this study, it is intended to introduce the concept of time to machines and give rise to a unit of measurement for a layer's performance in relation to FLOPS with regards to dynamic working memory alongside suggesting a Design structure to make full use of every core's potential

Introduction:

Neural networks initially were intended to replace classic and fuzzy based algorithms as they required too much labour , were computationally expensive and never coped well with the choatic real world.

RNNs (especially LSTM) were then used to construct memory-based dynamic moving primitives but RNNs are Gradient-based which meant “sequential” expensive computation beside suffering other problems as the vanishing gradient. LSTM was the one that maintained almost constant backward flow in the error signal through methods such as Hessian-free second-order optimization method preserving the gradient by estimating its curvature; or using informed random initialization, then a Kalman filter as added on top of that so Accelerating inference in RNNs became even more difficult due to their inherently sequential nature,

So LSTM was more computationally expensive and sequential; abusing only a percent of the G.P.U.’s cores and the rest are resting at idle.

“Working memory “ is not a storage per se, but a mental workspace utilized during planning, reasoning and solving problems. Most psychologists differentiate WM from “short-term” memory because it can involve the manipulation of information rather than being a passive storage [1].

Along the same lines, Engle et al. [2] argued that WM is all about the capacity for controlled, sustained attention in the face of interference or distraction.

Attention-control is a fundamental component of the WM system and probably the main limiting factor for capacity [3, 4]. Consequently, the inability to effectively parallel process two-attention demanding tasks limits our multitasking performance severely.

Over the past several decades psychologists have developed tests to measure the individual differences in WM capacity and better understand the underlying mechanisms.

These tests have been carefully crafted to focus on the specific aspects of WM such as task-driven attention control, interference and capacity limits [5]. The best known and successfully applied class of tasks for measuring WM capacity is the “complex span” paradigm.

The challenge presented by complex span tasks is recalling the list of items, despite being distracted by the processing task.

Studies show that individuals with high WM capacity are less likely to store irrelevant distractors [6] and they are better at retaining task-relevant information [7].

Developing task-driven strategies for cognitive control are essential for the effective use of WM[8].

Using gates to control admittance of data into deep memory is a task-driven attention control.

Although Reservoir computing has been around for a while now, it seems to be frowned upon still as it seems a bit chaotic and that is what enticed the author to use Echo state networks as they appear to be strongly related to the “order out of chaos” and the “chaos gives birth to a star” ideas.

ESNs are computationally inexpensive, not memory straining and do not suffer the vanishing gradient problem so they were used here to construct the basic deep network, and then concepts were built on it

Methodolgy:

“Esn” and “reservoir” terms are used interchangeably denoting a network comprising a no. of randomly connected units.

ESN’s size must be as small as possible so it could minimize the next reservoir’s “wait time” but call overhead must be in head as well.

ESNs are notorious for not converging easily if not tuned correctly as they retain all past memory so we use a single ESN as an input related oscillating function with as low as 40 units constituting it i.e., when input is big or slow , output frequency is proporionally slow obeying a linear law and we call it ClockESN.

We use two ClockESNs to supply the memory network with two regulating different sine waves, for each sine wave; [0,1] denotes allow and [1,0] denotes block.

one clock generates a relatively fast wave controling the “primal” working memory and the other relatively slow one controls the “primitive” long-term memory.

ClockESN supplies frequency to a number of gates each controlling exogenous input to a stack of ESNs .

ClockESN supplies frequency to a number of gates each controlling input from the previous stack to the next stack

ESN stacks grow bigger in number of reservoirs and no. of units per reservoir as we go deeper and by deeper we mean closer to the most superficial of the decision related networks

big-memo comprises two hemispheres :

one hemisphere of parallel connected stacks of ESNs corresponding to memory cortex

one hemisphere of a sequential stack of ESNs corresponding to behavioral cortex/scratch pad(or muscle memory)

The input to every task is a time-indexed stream of items. At a higher level, we can view the input as a concatenation of various subsequences that represent different functional units of processing. Additionally we use a constant-sized set of special items (called command markers) to both mark the beginning of a subsequence as well as indicate its functional type.

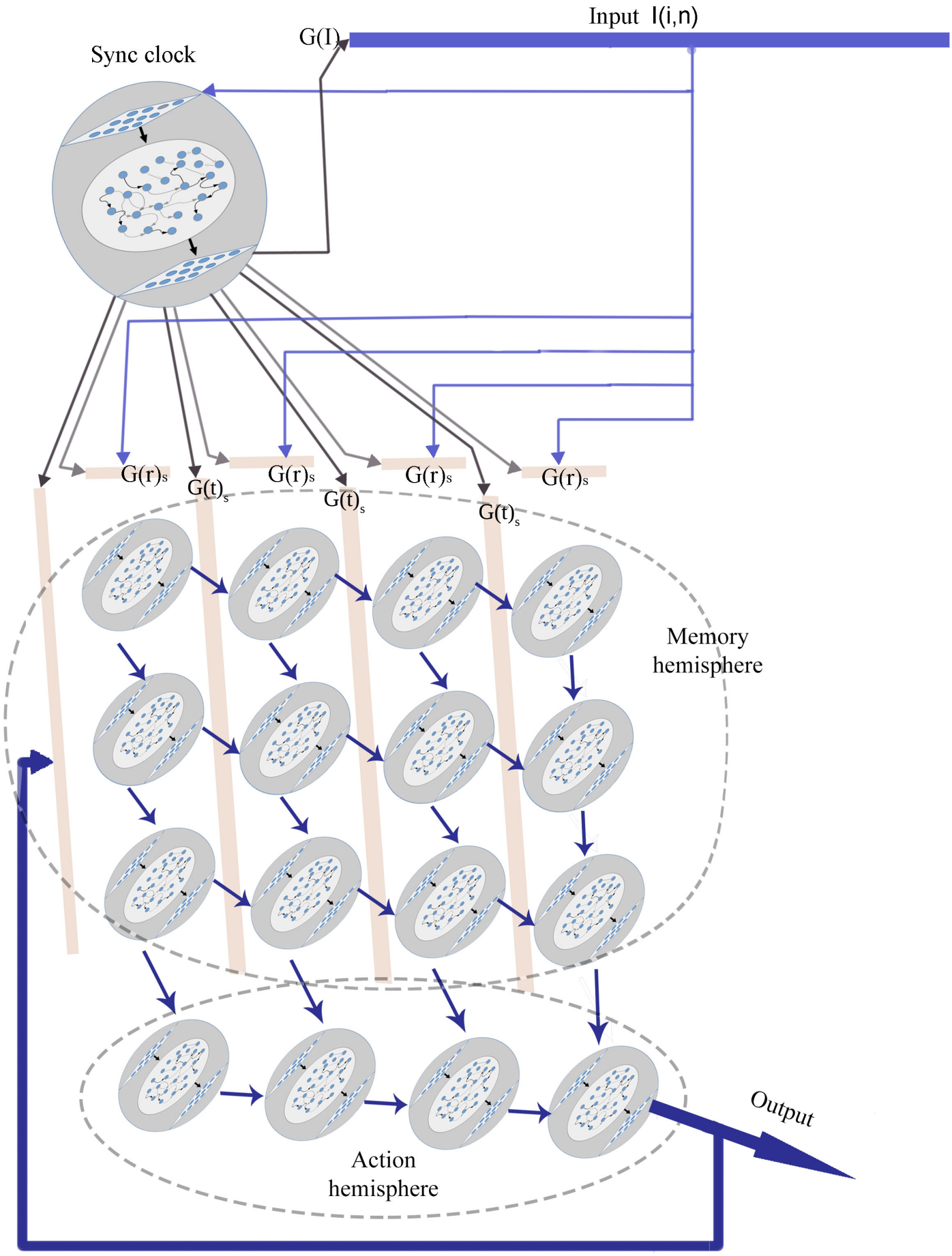
To benchmark each stack’s performance, we supply an input pulse and time the first output coming out of it, then we could scale the stack and reservoirs up or down to achieve full usage of our cores which is super easy for ESNs.

Researchers seem to adopt one of two methods. Some seek to add context parameters to dynamical movement primitives (DMP) to generalize to new objects. Others seek to associate these actions with the different contexts and get an overlap.

~~The author~~ I tried to fuse both approaches by associating time to the shape of the pulses sidebyside with actions while sticking to a relatively deep complex ESN.

To catch something , we move in its general direction based on an action indicating the start of sequence (Ex: baseball player) before getting hold on real info about its current position using prediction which is basically memory of past time-coupled events feeded to it previously .

Based on the above, the Deepest “most primitive” (biggest and slowest)in the memory cortex is connected to the most superficial action network to construct reflexes ,then faster acting yet more complex (going through more stages) networks signal comes later to correct .


Input I(i,n)
G(I)
Sync clock
G(r)s
G(t)s
G(r)s
G(t)s
G(r)s
G(t)s
G(r)s
Memory
hemisphere
Action
hemisphere
Output

Development:

let us start with basic notations:

H_1= memory -hemisphere H_2= action-hemisphere

C_1 = cortex1 C_2 = cortex2

CK_1 = clock1 CK_2 = clock2

R_{ij} is reservoir R in position i and stack j

Input is a tuple of (I, N)

I is input value

N is number of times input is allowed to propagate through reservoirs and change their states before being terminated which basically resembles TTL(time to live) , to achieve a selective memory layer on top of normal memory .

what we need is to decouple reservoir execution time from c.p.u. ‘s frequency so we could benchmark our system independent of the hardware, and to control our response for time-relevant tasks.

Sync clock controls the gate G(I) releasing input to gate G(R) (optional).

Sync clock controls the gate G(R) releasing input to resevoir $(R_{ij})_C$ where i denotes position in stack , j denotes stack No. and c denotes Cortex No.

Sync clock controls the gate G(t) allowing traversal of memory through different cortexes/stacks .

what we need now is a definition of time, independent of a processor’s clock, so we could tie our responses to real time:

assume a sync clock produces a base/carrier frequency $(1/RS)$ such that it produces 10 Giga oscillations in 1 second

so a frequency of $(1/RS)$ is unrelated to c.p.u.'s frequency (a software layer can make sure it stays irrelevant).

one tick (T) equals one "instruction" completed; time between the admittance of input I(n) to a reservoir and first ouput O(n) sprung(assume 0.1 RS which is a lot i.e. freq is $0.1\text{x}10^{10}$ $(1/RS)$).

A consequence of that is, the slowest reservoir will tie the rest of the hemisphere to it, which again is desirable since we need to deflect from complex reservoirs and reside to complex stacks, cortexes and hemispheres so we could exploit the parallel architecture.

A tick is our benchmark of a reservoir's performance.

Next we tune a reservoir sync clock (modulation) to 60T $1/RS$ (60 ticks per RS)and optionally tie it to input so if input is slower/faster then the sync clock will be propotionally slower/faster.

now we try to emulate the alpha and beta waves in the human brain,

so we tune the primitive cortex C_2 to a frequency of 50T RS which corresponds to a frequency of 50 HZ .

,and we tune the primal cortex C_1 to a frequency of 60T RS .

what that means is that we tune the sync clock to produce a tick every $\frac{1}{1/6\,Tx\,10-8}$ RS .

Implementation:

Basic guidelines for constructing a stable ESN are:

-The spectral radius of the matrix must be less than for a stable matrix but a recent strudy supported that this constraint does not hold all the time a local lyapunov exponent serves better.

-If data < 1+ No. of RD units + No. of input units ,then expect overfitting.

-Think sparse not big , connections rather than No. of units.

-Input data normalization and scaling is always a good thing to avoid saturation in the activation nonlinearities(memory loss).

-Leaking rate controls reservoir dynamics hence increasing duration of short term memory.

-First training steps ‘s(like a 100) resulting errors are harmful if collected , ignore them and collect afterwards.

-A time step weighting array can be used to assig different importance ot different time steps when training .

-Use output feedback only if necessay.

-Teacher forcing is more efficient and helps to a faster converge .

-Regularization or noise addition is a lways a good idea.

-ESNs seem to benefit the most from initial chaotic activations while being teacher forced.

-Feedback weights were chosen from [-0.5 , 0.5]

-Noise term insertion was sampled from a uniform distribution from over [-0.001 , 0.001] to [-0.0000001 , 0.00000001]

-Due to the relative cheapness with regards to computing power requirement , many Methods of error compensation can be incorporated and some are added to the framework.

-Noise term range of values can be changed overtime and is preferably high then lowered as time passes.

-if large noise was added to output feedback , larger noise was added to input to achieve stability but generally more noise leaded to better stability.

-Some Dimensionalit y reduction algorithms are also incorporated in the framework such as (PCA) and (KPCA) to be used instead of ridge regression or LMS.

For the hemispheres:

Each gate can assume one of two states :

[0,1] allow.

[1,0] block.

gate G(I) withholds external input I(n)and can be controlled by RS clock or can be normally open.

G(I) sends input to G(s) and RS clock (optional)

$G(s)_i$ is a gate on each stack of memory reservoirs for i= 1, ... n

Each gate is a parent, if it gets an allow signal then it activates all underlying stack reservoirs

each reservoir recieves input I(n) from other nodes which is a tuple of (I, N) ;

I denoting input and N denoting number of steps data is allowed to move before death (N = 1) , if (N=0) then it is subject to memory fading effects only.

En is the enable bit : ‘0’ is enable , ‘1’ is disable. Which is checked by a parent function to activate all underlying stack reservoirs processing input

[1] Nelson Cowan. The many faces of working memory and short-term storage. Psycho-nomic Bulletin and Review, 24(4):1158–1170, 2017.

[2] Randall W Engle, Stephen W Tuholski, James E Laughlin, and Andrew RA Conway.

Working memory, short-term memory, and general fluid intelligence: a latent-variable

approach. Journal of experimental psychology: General, 128(3):309, 1999.

[3] Andrew RA Conway and Randall W Engle. Working memory and retrieval: A

resource-dependent inhibition model. Journal of Experimental Psychology: General,

123(4):354, 1994

[4] Randall W Engle and Michael J Kane. Executive attention, working memory capacity,

and a two-factor theory of cognitive control. Psychology of learning and motivation,

44:145–200, 2004.

[5] Klaus Oberauer and Hsuan-yu Lin. An interference model of visual working memory.

Psychological Review, 124(1):1–39, 2017.

[6] Edward K Vogel, Andrew W McCollough, and Maro G Machizawa. Neural mea-

sures reveal individual differences in controlling access to working memory. Nature,

438(7067):500, 2005.

[7] Ashleigh M Maxcey-Richard and Andrew Hollingworth. The strategic retention of

task-relevant objects in visual working memory. Journal of Experimental Psychology:

Learning, Memory, and Cognition, 39(3):760, 2013.

[8] Learning to Remember, Forget, and Ignore using Attention

Control in Memory

IBM Research AI, Almaden Research Center, San Jose, USA † 28 sep2018