Article

On the role of information transfer's speed in technological and biological computations

János Végh [1]* 0000-0002-3247-7810 and Ádám-József Berki [2] 0000-0001-7099-167X

[1] Kalimános BT, Hungary; Vegh.Janos@gmail.com
[2] University of Medicine, Pharmacy, Sciences and Technology of Targu Mures, Romania ; berki.adam@yahoo.com
* Correspondence: Vegh.Janos@gmail.com

Abstract: Information is commonly considered as a mathematical quantity that forms the basis of computing. In mathematics, information can propagate instantly, so its transfer speed is not the subject of information science. In all kinds of implementations of computing, whether technological or biological, some material carrier for the information exists, so the information's propagation speed cannot exceed the speed of the carrier. Because of this limitation, for any implementation, one must consider the transfer time between computing units. We need a different mathematical method to take this limitation into account: classic mathematics can only describe infinitely fast and infinitely small computing system implementations. The difference between the mathematical handling methods leads to different descriptions of the behavior of the systems. The correct handling also explains why biological implementations can have lifelong learning and technological ones cannot. The conclusion about learning evidences matches others' experimental evidence, both in technological and biological computing.

Keywords: computing paradigm; technological computing; biological computing; information transfer speed; information storage; lifelong learning; redundancy; temporal behavior; machine learning; artificial intelligence

1. Introduction

The computing model proposed by von Neumann in his famous "First Draft" [1], is *bio-inspired*, despite the common fallacies: he discussed the computing process implemented in both technological (vacuum tubes) and biological (neurons) computing systems. However, because von Neumann made omissions *validated only for [the timing relations of] vacuum tubes*, his *simplified classic paradigm* cannot be applied to other technologies [2,3]. Because of the omission that the transfer time can be neglected apart from the computing time, the implementations based on the classic paradigm are not *biology-mimicking* ones. Von Neumann, of course, could not foresee the dawn of modern computing technologies, but he warned that *the computing paradigm must be revised when the technology changes* and that it would be ***unsound*** (sic) to apply his *simplified paradigm* (not to be confused with his *model of computation*!) to neural computing. Given that he clearly outlined that his proposal was about *the logical structure* of a computing implementation, it is not a very usable classification criterion whether (the otherwise undefined) von Neumann *architecture* is biomorphic (for a review see [4]) or not.

2. Computing paradigms

No doubt that von Neumann's model is valid for all kinds of computations: the operand(s) (in the form of some physical carrier) must be delivered to the operating unit where the computation takes place. The the computation cannot even start (as pointed out by von Neumann [1]) until the operands are completely delivered to the input section of the computing unit, and similarly, transferring the result cannot even begin until the computation is completed and the result is available in its output section. Because of this, the *information transfer* and *computing* mutually block each other: some "idle" activity is inherently present in the *computing process*.

In the model of computing, one must consider both the time needed to transfer the information and the time to make computation with it, furthermore the presence of the blocking constraint. To calculate the *total time of a computing process* is not simple at all: it depends on both hardware characteristics and workload type, so some neglections must be made. Because of these difficulties, *the classic paradigm neglected the transfer time, so the blocking constraint in the classical paradigm means only logical dependence*. Technological computing is based on the Hardware-Software contract [5]: mathematics provides the strong theoretical basis of computing, but neglects information's transfer time, and technology must adapt itself to the interface defined by von Neumann a three-quarter century ago, and for vacuum tubes only.

We surely know that the simplified model is not valid for our current technology. As the technology develops, it becomes evident that the classic paradigm cannot describe real-world implementations, neither technological (electronic) or biological (neural) ones. Furthermore, in technology, it leads directly [6] to the idea of unlimited computing capacity and workload-independent processing time. In electronics, mainly the issues experienced in connection with building so-called neuromorphic computers led the researchers to the idea that "*More physics and materials needed. Present-day electronics are not enough*" [7]. We can add: *Present-day computing science is not enough: more physics needed.*

However, computing science does not want to admit that computing's physical implementations must also include physics, despite the existence of the ready-made mathematics [8]. In some sense, hundred years after inventing the Minkowski-mathematics, it is still a scandal [9] to consider that *the theory of technological implementation of computing must also include some physics*. The important consequences include (but not limited to) the inefficient processor chips [10], the enormous power consumption [11], the experience of "dark silicon" [12], the stalled supercomputer performance [13,14], the stalled Artificial Intelligence (AI) development [15][6], the failed brain simulation [16]. Interesting, that the that-time "*disciplinary analysis of the reception of Minkowski's Cologne lecture reveals an overwhelmingly positive response on the part of mathematicians, and a decidedly mixed reaction on the part of physicists*" [9] has turned to the exact opposite: the description is generally accepted in physics (and resulted in the birth of a series of modern science disciplines), but completely refused in mathematics-based computing science.

In biology, it was evident from the beginning that the transfer (conduction) time cannot be neglected aside from the computing (synaptic) time. The name "spatiotemporal", and a (separated) time dependence is commonly used [17], in the sense that Precise Firing Sequence (PFS) "*tended to be correlated with the animal's behavior*"; furthermore, that "*the results suggest that relevant information is carried out by the fine temporal structure of cortical activity*" [18]. The "*neural dynamics*" was studied and "*spatiotemporal spreading of population activity was mapped*" [19] by methods used to describe the *static behavior*: interpike intervals histograms, autocorrelation and crosscorrelation. The correct method of description is still missing, given that the major item of the behavior is missed: *the time and position are connected through the information transfer speed* (called conduction velocity).

3. The effect of the information carrier's finite speed

When using the classic computing paradigm, the transfer time is neglected apart from computing time. In other words, "instant interaction" (infinitely large transfer speed, or in other words, infinitely small physical computing system size) is assumed. However, in all physical implementations, the information has some material carrier: inertial mass, electromagnetic or gravitational waves, electrons/ions, neurotransmitters, etc. The kind of the carrier and the transfer mechanism limit the speed of the information transfer, so the physical size of the computing system matters.

In electronic technology, the transfer speed is in the order of 10^8 m/s; in biology (speed of neural transfer), it is $10^1 \dots 10^2$ m/s; that is, the speed of electronic signals is several million times higher than the speed of neural transfer. At dozens of centimeters physical size, the transfer time is also several million times lower than in neural systems (such as our brain). Because of this difference, in biological computing systems, a "spatiotemporal behavior" was assumed from studying the neural operation. In contrast, in (electronic) technological computing systems, the "instant interaction" seemed to be a good approximation initially. However, we have good reasons to introduce a finite interaction speed both in science and computing technology [20].

Initially, both technological and biological *computing* worked on the same *msec* time scale [21]. In contrast, the information *transfer* speed was several million times higher for technological computing than for the biological one. This difference made the classic paradigm a proper approximation for technological computing. However, the evolution of technological computing systems quickly invalidated the classic paradigm's assumption: the processing time moved to the sub-nanosecond region, significantly increasing the transfer-to-computing ratio [14]. Although the *component density* of processors followed Moore's observation for decades, the material carrier's propagation speed, the electromagnetic interaction, remained the same. Similarly, a technological computing system's physical size remained in the region of several dozens of centimeters. Because of these changes, in technological computing systems *the transfer time* remained the same, while their *computing time* became millions of times shorter [22]. The stealthy nature of technological development resulted in that in current technological computing, the transfer time is not only not negligible apart from the computing time, but even it is longer than that. The timing relations of biological computing did not change, so this subtlety of the technological evolution resulted in that the timing relations of current computers are much closer to those of our brain than those assumed in the abstract model. According to the classic paradigm's inventor, it is ***unsound*** to use his simplified paradigm for those timing relations.

Given that a physical carrier delivers the information (in all implementations), the transfer time is crucial in all implementations of computing. As the computing model requires, the operand must reach the operator unit's input section, and for all physical carriers, the transfer speed is finite. Whether the transfer time can be neglected apart from the processing time depends on the implementation technology.

In general, in a somewhat simplified view (see Fig. 1), the information must be transferred between places (x_1, y_1) and (x_2, y_2), that requires *transfer* time. After that, the computation can be done at place (x_2, y_2), that takes *computing* time. The known speed of information transfer enables us to calculate the distance in time units: the temporal and spatial distances are interconnected through the information carrier's speed. The exact form of that dependence was elaborated by Minkowski [8]. In Fig. 1 (as well as in all similar figures), all coordinates are time but labeled as coordinates x and y. The spatial length of transfer operations is represented by horizontal (blue) arrows, the temporal length of computing operation by vertical (green) arrows, while the apparent temporal length of the resulting processing by red arrows. Using this transformation, the total processing time can be calculated using a simple (3D) trigonometry. For math details see [20,23].

The figure shows the picturesque difference between the discussed paradigms: according to the classic (time-unaware) paradigm, the length of the blue vectors (the time needed to transfer information to the other point) is zero, that is, the length of the green vectors (*computing time*) equals to the length of the red vector (total processing time or *apparent computing time*). In other words, the classic paradigm assumes that all computing objects are in the origo, and no time is needed to access them.

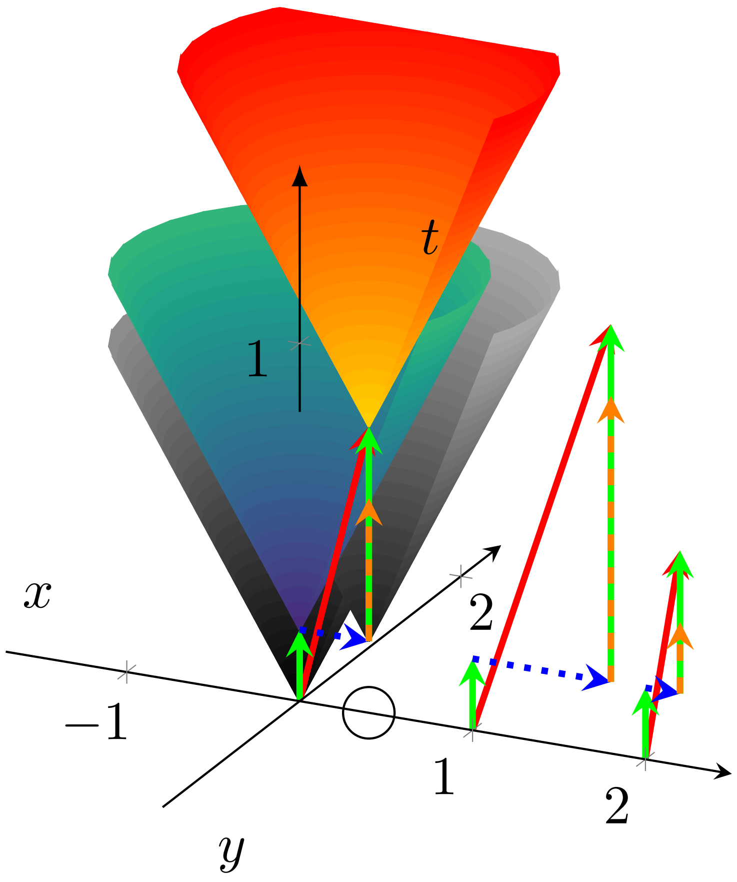

Figure 1. The way of calculation to combine the spatial distance (transmission time, blue arrows) and computing time (green arrows) to result in the apparent processing time (red vector)

4. The effect of synchronization

According to von Neumann's model, there is an implementation-independent need to synchronize the operations data transfer and computation to each other, so the implementation must provide means to keep its operations synchronized. Different means are provided in the different implementations, and they led to drastically different features.

In biological implementations, the source neurons fire (send the information) when they need to, and the target neurons receive it after the charges arrive via their chemical and/or electrical (less common) synapses [24]. In this system, the arrival of the charge to the postsynaptic membrane (i.e., the end of the transfer time) triggers computing (i.e., marks the beginning of computing time). Similarly, firing (when the threshold potential is reached) marks both the end of the computing time and the beginning of the transfer time. All this machinery works in the individual neurons independently.

That is, *the arrival of a spike is a synchronization signal as well*: the zero time of the synaptic conductance function $gsyn(t)$ [25] is set to the arrival of the spike. Notice that while receiving a spike, conductance $gsyn(t)$ of the receiver synapse changes: after reaching its peak conductance, shortly decreases again to zero. In this way, it limits the amount of charge delivered to the membrane (that is, the charge distribution in the received spike and the one reaching the membrane are different. As the spike keeps arriving, the charge continuously reaches the membrane and increases its potential in an analog way. In this way *the length of the computing time* (until membrane's potential reaches its threshold value) *depends on both the amount of the charge and its arrival time*.

After the membrane reaches its threshold potential, the firing period begins, and the computing time ends. However, it takes time (the spike's duration) until the signal reaches the neuron's output section. *This mechanism provides the auto-synchronization of biological computing.* This aspect is consequently neglected in spiking neural networks' technological implementations; for a review, see [26].

Listing 1: The essential lines of source code of the one-bit adder implemented in SystemC

```
SC_MODULE (BIT_ADDER)
{
sc_in<sc_logic> a,b,cin;
sc_out<sc_logic> sum,cout;

//Sensitivity to changes in a, b, & cin.
SC_CTOR (BIT_ADDER)
{
SC_METHOD (process);
sensitive << a << b << cin;
}

//Performed for any change
void process()
{
//intermediate signals
sc_logic aANDb,aXORb,cinANDaXORb;

//Perform intermediate calculations
//when any of the inputs changes
aANDb = a.read() & b.read();
aXORb = a.read() ^ b.read();
cinANDaXORb = cin.read() & aXORb;

//Calculate sum and carry out;
// maybe not final ones
sum = aXORb ^ cin.read();
cout = aANDb | cinANDaXORb;
}
};
```

In contrast, in technological implementations, due to the lack of start signal, a gate (or another digital object) starts to "compute" its result as soon as any of its inputs changes, and it provides the corresponding output after its *fixed length computing time* passes. The gates always have an output signal. Whether this output signal corresponds to the value expected based on the functionality of the gate (AND, OR, XOR, etc.) depends on whether all operands succeeded to arrive at the corresponding input sections of the gate before the computing began and after computing the result arrived to the output section of the unit. The lack of auto-synchronization in technological computations results in internally undefined states (and is the main reason for the inefficient processor chips [10] and the enormous power consumption [11]), as shown in the operating diagram of a one-bit adder in Fig. 2. The corresponding code in SystemC is shown in Listing 1. Notice how the pure logical dependence, a consequence of the time-unaware paradigm (called also "von Neumann style" [27]), is converted to temporal dependence by the technological implementation, and that the adder performs payload (computing) work only in the periods denoted by thick green arrows; the rest is non-payload (transfer) time. Notice also that both computing and transfer times are partly parallel (overlapping).

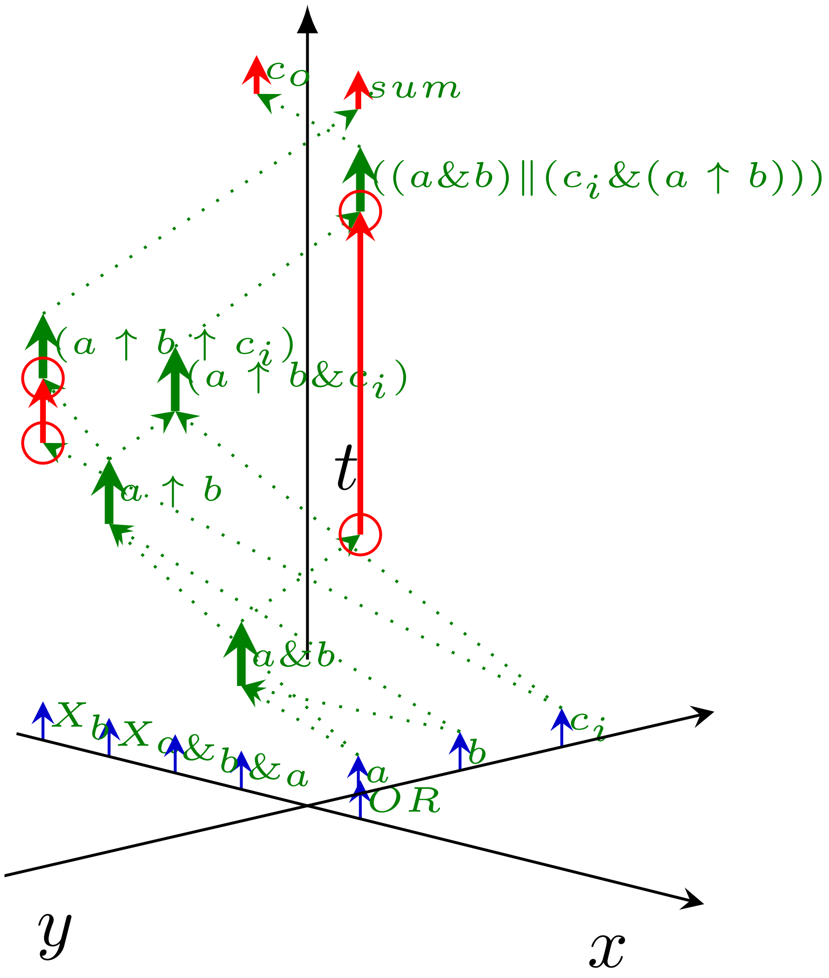

Figure 2. The operation of a technological one-bit adder, with "pointless" synchronization (red circles), see Listing 1. The input signals a, b and c_i are aligned along axis y (the input section), the computation takes part in gates aligned alog axis x, and the output signals c_o and sum aligned again along axis y (the output section).

At the red circles at the bottom of the red arrows, one operand (the result of a previous computation) arrives at its target gate (green arrowhead in the red circle), and (after the computing time) it may change the state of the target gate (depending on its previous state). However, the state, *may be or may not be* the final result of the operation: the second operand is still missing (although the input section has a well-defined signal level). After some time (the red arrowhead in the upper red circles), the second operand also arrives, and it may change the state of the target gate (depending on its previous state). In the time fraction corresponding to the red vector, the gate's output value is undefined, and so is the result of the one-bit addition. Also, notice that upon arrival of the second operand and performing the computation the gate represents, the signal still must arrive at the adder's output section. Without synchronization, especially in a several-bit adder, the information available in the output section of the adder may change several times during the operation [20]. Also, notice that unlike in biological implementation, the gates take power even if their input signals are missing.

Von Neumann emphasized the role of synchronization. His classic analysis resulted in one more important (and, for the intended vacuum-tube implementation, correct) conclusion that the initial design is significantly simplified by using a central synchronization clock signal. This is why a synchronization signal is used to validate the output signal: for the external world, signal c_o is only achievable after the central clock signal arrives. Given that the output signal c_o is, at the same time, the input signal c_i of the next one-bit adder, the clock signal must reach the first and the last one-bit adder at different times. This need anyhow results in some "skew" in the clock signal. Given that the total idle time increases with the number of bits comprised (from the beginning of the idle time in the first bit to the end of idle time in the last bit), in larger designs, the need for introducing several clock domains [11] appears.

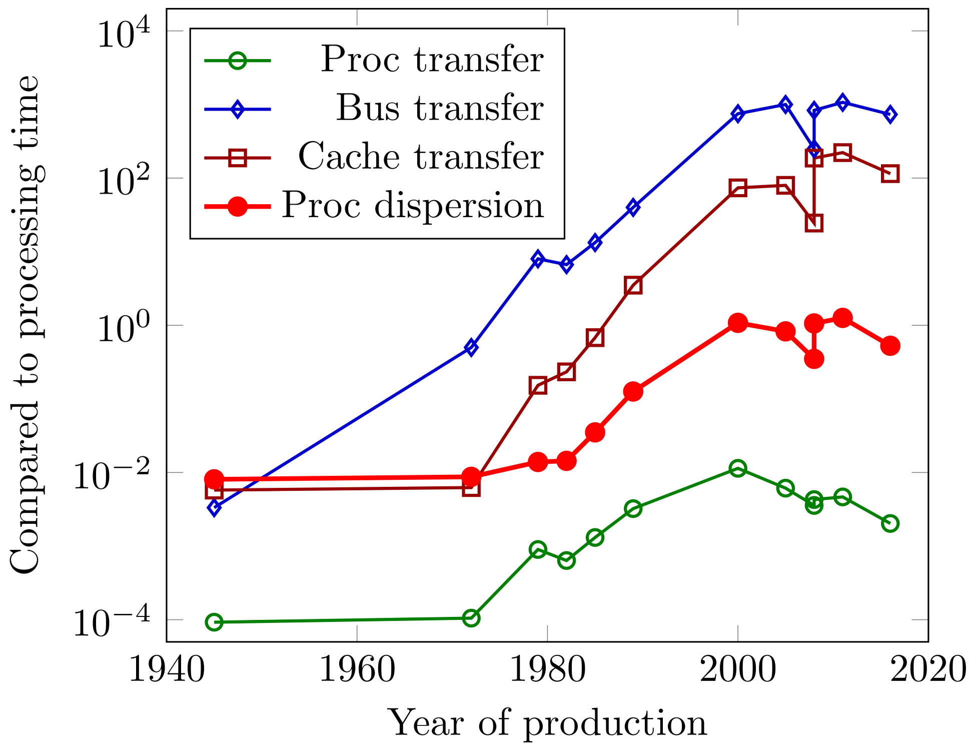

Figure 3. The history of dispersion of processors in technological computing

Von Neumann's detailed analysis has shown that using a central synchronous signal is advantageous only when the clock signals' dispersion is negligible. However, the evolution of technological computing quickly increased the dispersion (see Fig. 3 and the detailed discussion in [20,28,29]). In current technological implementations is already noticed that the advantages of using central synchronization are lost, so recently, the idea of asynhronous operation was proposed [30]. *In our current technologies, the dispersion is not negligible* even within processors, and especially not in computing systems, from the several centimeter long buses in our PCs [11] through the wafer-scale systems [31] to the hundred meter long cables in supercomputers [13].

5. Time, information storage and learning in biological implementation

In biology, it is obvious from the beginnings that the measurable quantities change with time and space: their behavior is called "spatiotemporal". It is similarly evident that the brain, unlike technological computing systems, does not feature a unique, perfectly synchronous clock to regulate communication and computing [32]. The common experience [25] shows that the outputs of biological neurons depend not only on their inputs (they compute with their inputs) but also on their internal state (they store information [33,34][1]). Furthermore, biological systems can adapt themselves to both short-term and long-term changes in the external world: they can learn. We use "*the broad definition of learning: use present information to adjust a circuit, to improve future performance*" [33]. However, the definition needs to explain also, what is the "present information" which the circuit adjusts during learning. Fortunately, the time-aware paradigm offers a natural explanation for those phenomena. Fig. 4 shows how the model explains in what form the information is stored and adjusted in biological neurons. Furthermore, it explains and connects those two learning modes.

In Fig. 4 (in the coordinate system shown also in Fig. 1), we assume that the excitation of a neuron at coordinates (-1,0,0) occurs and a neuron fires to an assembly (aligned along coordinate axis y at x values -0.3, 0.1 and 0.4). The figure assumes that the excitation (after passing to the branching point) branches towards three members of the assembly A. The assembly members A_i send spikes to their common *Target* neuron at position (-1.5,0.7). The corresponding synaptic weights of the target neuron are W_1, W_2, W_3 (assumed to be of equal for A)), and three received spikes may cause the target neuron to fire (the sum of the potential contribution of the three spikes is just above the threshold). Given that the position of the assembly members and their firing times slightly differ, so differ their arrival times (see the red arrowheads) of the spikes from the assembly members at the target's position.

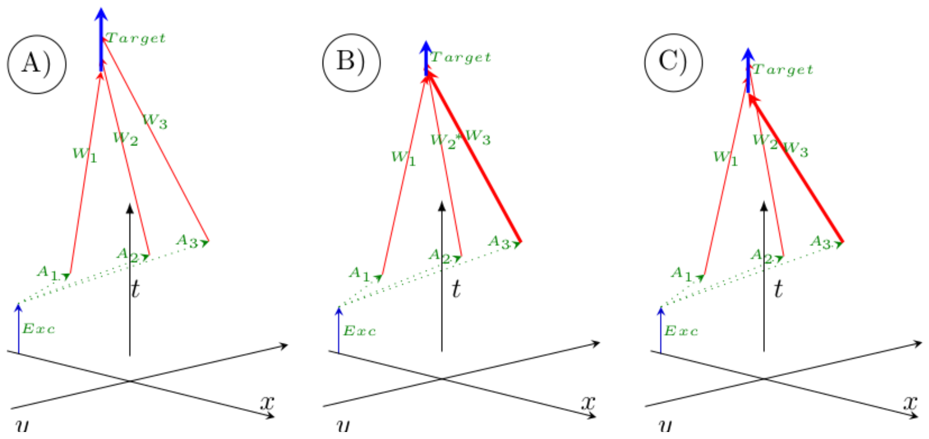

Figure 4. How neurons learn. A) The initial state B) Short time learning, changing synaptic weight by +50% B) Long time learning, changing conduction velocity by +10%

The *Target* is initially in rest. When the first spike arrives, it increases the membrane's potential, and so do somewhat later the second and third spikes, too. With the charge delivered by the third spike, *Target* reaches its threshold, and (after some "processing time": charging the membrane) it fires. The blue arrow length denotes the length of the computation: its bottom is at the first spike's arrival time, its head is at the beginning of the neuron's refractory period. Given that after reaching the threshold, some time is needed to charge the membrane to its operating potential, the length of the arrow includes an extra contribution.

The total operation cycle of the neuron is given as

$$T_{Computing} = T_{Triggering} + T_{Charging} + T_{Refractory}$$

Here we assume that the computation will not fail. $T_{Triggering}$ is the time the *Target* needs to collect the potential contributions from its synapses to reach the threshold, $T_{Charging}$ is the time needed to charge the membrane to its maximum potential, $T_{Refractory}$

[1] This feature can easily be misidentified as memristance [35], in the time-unaware model

is needed to reset the neuron to its initial state. Usually, some idle time T_{Idle} (no neural input) also follows between the "computing operations", which can make $T_{Computing}$ longer. That is, the operating frequency (its firing rate, the inverse of $T_{Computing}$) of the cyclic operation is

$$F = \frac{1}{T_{Triggering} + T_{Charging} + T_{Refractory} + T_{Idle}}$$

Given that $T_{Charging}$ and $T_{Refractory}$ are defined anatomically, the only way to adjust neurons' firing rate is to change the $T_{Triggering}$ and T_{Idle} times (the latter being relevant only in periodic operation: how long 'pause' between computations is enabled): how quickly the neuron's membrane reaches its activation threshold potential (in this simplified discussion we do not consider that between the arrivals of spikes the membrane looses some charge). That is, the (anatomically defined) maximum operating frequency is

$$F_{Max} = \frac{1}{T_{Charging} + T_{Refractory}}$$

The time $T_{Triggering}$ can be adjusted by the neuron either collecting more charge to its membrane from its synapses or collecting the same charge in a shorter time, or combining them. Biology provides mechanisms for both ways of adjusting the triggering time. The two ways differ in their implementation speeds and operating costs, enabling the biological systems to find a proper balance between the two.

One mechanism is to collect more charge from the input spike that hits its synapse: to increase the synaptic weight W_i corresponding to assembly member A_i. It can be carried out by changing the shape of function $gsyn_i(t)$: a wider function (providing more transmitters) has such an effect. The ions in the spike anyhow reach the target at its presynaptic point, and increasing W_i actually means delivering more charge to the (post-synaptic) membrane, causing a larger increase in the membrane's potential. The "unused" (not transmitted to its membrane) ions must be pumped out, which is a rather energy-consuming action. However, tuning $gsyn()$ is a fast process: increasing the transmitters' concentration can be implemented in a few *msec* periods. As observed [36], "*elementary channel properties can be rapidly modified by synaptic activity*".

Another mechanism is to decrease the time needed to transfer a spike from A_i to *Target*: biology can change the corresponding axon's thickness. Given that the transfer speed (conduction velocity) on thicker axons gets higher, the spike arrives in a shorter time and in this way contributes to the membrane's potential at an earlier time (i.e., it reaches its threshold potential earlier, too). This mechanism is less expensive in energy consumption but needs significantly longer times to implement it. The aspect that changes in conduction velocity "*could have profound effects on neuronal network function in terms of spike-time arrival, oscillation frequency, oscillator coupling, and propagation of brain waves*" [37], and that "*Node of Ranvier length [can act] as a potential regulator of myelinated axon conduction speed*" [38,39] have been noticed, but *the role of time in storing information and learning has not yet been discussed*.

Given that both mechanisms result in shorter $T_{Triggering}$ times, they cause the same effect: the computing time (or in other words: the firing rate) changes. When the firing rate changes, one cannot tell for sure which mechanism caused it. This equivalence is why nature can combine the two mechanisms: the neuron can increase its firing rate quickly (as a trial), and (on success, that is, if that learned condition is durable) it may decide to reimplement ("remodel" [40]) its knowledge in a less expensive way. It makes the corresponding axon thicker. After that, the needed weight W_i, which was increased for the short-term learning (experimental) period, can be decreased to conserve the learned long-term (stable) knowledge to reduce the energy consumption. The effect (the learned knowledge) remains the same.

Our findings seem to be supported by anatomical evidences that "*individual anatomical parameters of myelinated axons can be tuned to optimize pathways involved in temporal* *processing*" and that "*the internode length decreases and the node diameter increases progressively towards the presynaptic terminal, and . . . these gradations are crucial for precisely timed depolarization*" [38]. However, the change of axons' thickness is measurable only after weeks or months; presumably, so is the decrease in the transfer time to the synapse. It was observed [40] that "*neuronal activity can rapidly tune axonal diameter*" and "*activity-regulated myelin formation and remodeling that significantly change axonal conduction properties are most likely to occur over time-scales of days to weeks*".

This mechanism reveals that short-term and long-term learning perform the same action: they reduce processing time using two different biological implementations. Increasing neurotransmitter concentration leads to shorter computing time, increasing axon thickness leads to shorter transmission time. In both cases, as our model predicts, the processing time decreases (the firing rate increases). It means that short time and long time learning are just two sides of the same coin. Our analysis seems to underpin that it was correctly explained [33] "*we should not seek a special organ for 'information storage' — it is stored, as it should be, in every circuit.*" Mainly because "*information stored directly at a synapse can be retrieved directly*". Besides, our analysis adds the discovery that remained hidden: *the information is stored through handling the time* or, in other words, *adjusting temporal processing*. Biology takes advantage of the low speed of information propagation[2]; one more evidence for the neural design principle [33]: "*Send only information that is needed, and send it as slowly as possible*". Basically, this information handling is why biological systems have life-long learning property without implementing switches "Learn On/Off" and "Short/Long term".

Biological systems have a complex network of neurons, with partly pre-programmed neurons, with well-defined initial synaptic weights (as reflected by their firing rates). As the elegant investigation [41] proved, as an implied side result[3], that the result of learning manifests in that the affected neurons change their firing frequency, using the mechanism described above. Our time-aware paradigm offers the chance to investigate and understand the phenomena quantitatively. Furthermore, a way to find out directly whether it is a long or short time learning (whether the change in $T_{Triggering}$ correlates with the change in the corresponding conduction speed or the synaptic strength): the firing rate is the inverse of the governing rule, the temporal relations.

Notice that in Fig. 4A) the spike from A_3 arrives last, immediately before *Target* fires. According to Hebb's observation, its synaptic weight W_3 is increased in Fig. 4B) by 50%; putting A_2 in the winner's position for the next learning cycle. This effect provides the dynamical behavior needed for life-long learning.

Temporal behavior is also the key to understanding redundancy. As one can conclude from the experimental results in [42], using *simultaneous* firing, 6-7 spikes would be sufficient for charging the membrane above its threshold value. Given that the membrane discharges between the arrival times of the charge contributions from the individual spikes and their arrivals are poorly concerted, up to 20 spikes are needed in practice to produce firing. This phenomenon and the learning mechanism given above explain that if one of the dominant assembly members dies out, some formerly 'obsolete' assembly signal's synaptic strength can increase. After some training, the operation stabilizes using the new assembly member.

6. Time, information storage and learning in technological implementation

In technology, from the point of view of mimicking biology, the most harmful consequence of neglecting transfer time (that is, assuming "instant interaction") is that the paradigm can be used to describe neither how information is stored nor changing the transfer speed. Technology must use drastically different implementation methods that manifest in drastically different behavior of technological and biological systems.

[2] An interesting historical parallel that computer EDVAC used delay lines with *msec* processing time for information storage
[3] The major goal of that investigation was to prove that learning to some measure is "pre-wired"

The lack of possibility of storing information through changing timing adequately led to the need to use a separated "memory" unit, where special signals (unlike "stored and retrieved directly" [33] in biology) are used to store and retrieve the information. The input and output sections of the model are implemented as numbered storage cells. Given that computing systems are assembled from pre-fabricated functional blocks [43], those sections must be wired to the processing unit. The synaptic weights W_i are stored in those memory cells and accessed through a shared medium: the bus. The shared medium must be made "private" for the time of transferring data. Most of the time is spent with contenting for the right of owing the bus, see Fig. 5, and in detail [20]. This non-payload time is especially disproportional for the neural-mimicking communication, where the neural messages comprise just 1. . . 3 bits [33] of information.

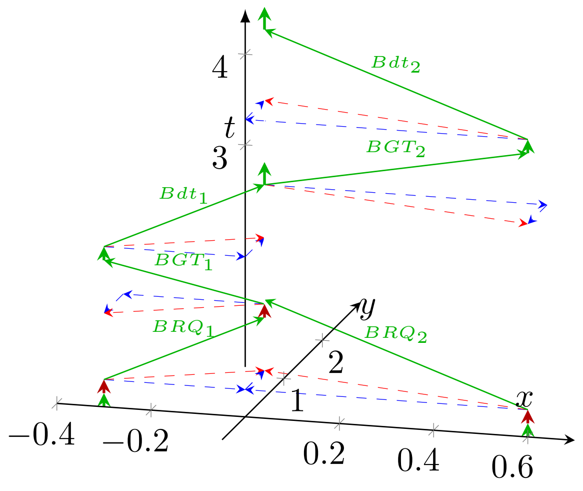

Figure 5. The temporal operating diagram of a technological high-speed single bus: the bus delivers data only in the fractions denoted by vertical green arrows

Storing information in synapses has two major advantages: it is directly wired to the computing unit, and all synaptic weights can be reached quickly and simultaneously [33]. In technological implementation, it would need storing all synaptic weights in processor registers, which not only can be accessed "instantly" (without addressing contention), but the processor can perform operations with all of them simultaneously. The sequential operation of processors alone reduces their operating speed by orders of magnitude for many synapses. The distance between storage and computing units makes the transfer time orders of magnitude higher [13] and, in this way, surely not negligible apart from computing time. The temporal behavior of communication also reduces the achievable performance by orders of magnitude [20]: only one communication action can use the shared medium at a time, and a considerable offset time must be used for contending for a private use of the shared medium. As analyzed in detail in [6], these temporal effects reduce computing efficiency for vast computing systems to a tragically low level, see Fig. 6. In addition to these effects, the Single Processor Approach (SPA) [44] requires using Input/Output (I/O) instructions (and non-payload time to organize communication), raising ratio of the non-payload portion [6] drastically. This is why, especially for neural simulation, bitterly admitted that: "*artificial intelligence, . . . it's the most disruptive workload from an I/O pattern perspective.*"[4] This is why a growing portion of supercomputers reduce their number of processors when running heavier workloads. Benchmarks High Performance Conjugate Gradients (HPCG) decreases the achievable efficiency by order(s) of magnitude (and the top efficiency can be achieved using an

[4] https://www.nextplatform.com/2019/10/30/cray-revamps-clusterstor-for-the-exascale-era/ :

order of magnitude less processors), and the AI workload makes it hopeless to run an application with reasonable efficiency.

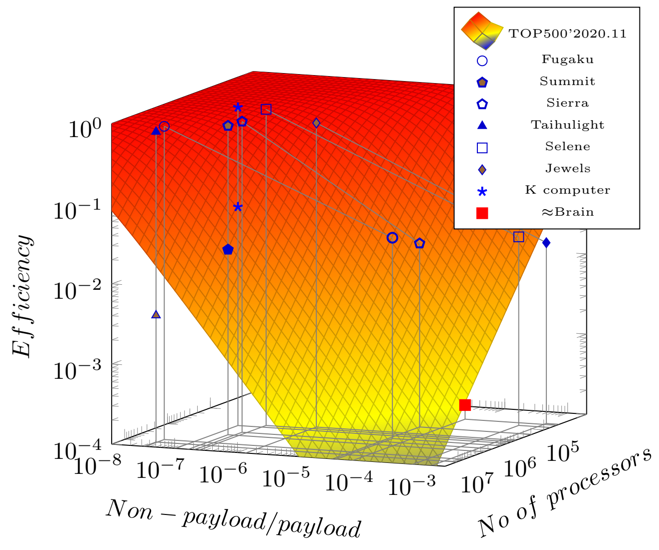

Figure 6. The temporal behavior of the technological components results in that the *payload* efficiency of vast computing systems sharply decreases as the number of processors increases; the architecture defines the parallelization efficiency. Notice the reasoned guess for the efficacy of simulating the brain, resulting from the vast numbers of computing units and the disruptive workload.

Besides, the analog computation performed by the membrane is imitated by integrating the received charge in short time periods (grid time), which makes the "quantal nature of computing time" visible [45]. This is why neither special-purpose hardware (HW) brain simulators, nor software (SW) simulators running on general-purpose supercomputers can simulate *the operation (not to be confused with filling up the equivalent memory capacity with data [46])* only a tiny fraction of our brain [47]. These technical nuances cause that "*any studies on processes like plasticity, learning, and development exhibited over hours and days of biological time are outside our reach"* [47].

The serial operation adds one more issue. The SW repeats adding change contributions in a cycle, without being aware at what time that contribution arrived and what is the relation of the partial sum of potential contributions to that of the membrane potential. In this way, the time of firing can happen only after finishing the cycle, which on one side smears the time of firing over the operation of the cycle. On the other side it makes it hard to interpret Hebb's observation, given that all contributions (inputs at the synapses) arrive at the "same time". Because of this difference, for changing synaptic weights, special mathematical procedures (such as gradient descent) had to be introduced. They introduce their specific issues (such as vanishing gradient or the need to calculate the gradient of noisy signals) and distribute the effect of synaptic learning to "foreign" synapses. Handling the synapses grouped in vector or tensor means that the effect of learning, in a single synapse, experienced in the many-parameter space is shared between all participating synapses, whether they contributed to learning or not. This latter effect greatly contributes to the experienced unreasonably long training times and over-fitting [48]; accompanied by the lack of explicit time parameters and mis-fitting.

One of the significant features in technological computing systems is that neither their computing time nor the transmission speed (of the electromagnetic waves) can be changed. Correspondingly, the learning mechanisms (even in the so-called neuromorphic systems) must be drastically different. The lack of time in technological computing does not enable implementing learning mechanisms similar to those known in biology, and machine learning requires introducing "training" or "test" mode switches, for a review see [4]. (Changing the weights of synaptic inputs is arbitrary and has no effect on the computing time.) The other significant point is that technological computing lives in a world of "instant transmission", i.e., that only logical sequencing exists, but no temporal sequencing. When analyzing algorithms, the "time of computing" is included instead of the correct "processing time", including transfer time. This difference results in unrealistic estimations in the scaling of algorithms.

The issue mentioned in connection with Fig. 2, the idle time, leads to further troubles in the case of more complex systems. The gates marked by red circles in Fig. 2 are directly wired. However, when transferring large amounts of data needed for the operation, for example, elements of a vector or a matrix, on a finite width bus, the data must be serialized at the sender and deserialized at the receiver. A typical example is that deep learning figures show *logical wiring*. Still, the *physical wiring* involves also some bus, so the real temporal behavior of neural messages is as shown in Fig. 5. The effect was pointed out both experimentally [49] and theoretically [6], and is a major reason why AI development stalled [15]. As the model requires, the computation (with a vector or a matrix) cannot even start until the last element delivered to its corresponding input section. This requirement automatically increases the transfer time as many fold as many elements are considered, and correspondingly, it increases the idle time and decreases efficiency.

However, one can *not* consider the needed synchronization (it is easy to do so, given that no auto-synchronization occurs) and start the calculation as soon as the overall computing process begins. Analogously to the case of a one-bit adder, some results will always be available in the tensor's output section. After all inputs arrived at the input section of the tensor and the corresponding computation performed, the output section's result will match the result expected based on the mathematics. Before that time, we can read out the output section, but it has not much sense: we can be sure we have the correct result only when the computation in the above sense is completed. *Before that time, the result may or may not be correct.* The faster the tensor unit computes, the worse.

A popular and frequently occurring idea is to use memristors for storing synaptic information [50,51][5] It sounds good that "*The analog memristor array is effectively the neural network laid out in the form of a crossbar, which can perform the entire operation* ***in one clock cycle***" [53]. In brackets, however, fairly added, that *(not counting the clock cycles that may be required to fetch and store the input and output data)*. Yes, all operands of the memristor array must be transferred to its input (and previously, they must be produced), and the results must be transferred to their destination. The total time of the memristor-related operations shall be compared to conventional operations' total time to make a fair comparison.

The case is made even more complicated when introducing different buffers and time stamping. The time stamping preserves the correct biological time. Still, it may arrive at a physical time when the computing process already performed the state's calculation at that biological time, in the absence of the data still waiting somewhere in the queue. A signal coming from a physically distant position of an overloaded system has a good chance of being delayed in one buffer. It arrives at the destination with considerable delay to its expected arrival time. One possible handling is that "spikes are processed as they come in and *are dropped if the receiving process is busy over several delivery cycles*" [47]. This handling is believed to enhance the apparent efficiency of the system. Given that the physically distant artificial neurons will have a longer delivery time, they

[5] Five decades ago, even *memristance* has been introduced as a fundamental electrical component, meaning that the memristor's electrical resistance is not constant but depends on the history of current that had previously flowed through the device [35]. There are, however, some serious doubts as to whether a genuine memristor can actually exist in physical reality [52]. In the light of our analysis, some temporal behavior definitely exists; the question is how much it is related to material or biological features, if our time-aware computing method is followed.

will go to the end of the queue of the input events, so they have good chances to be dropped (in biology, many times, distant neurons are controlling local neural assemblies; so in their technical implementation the remote control will be missing).

If they are not dropped because of their late technological delivery, they will arrive at a simulated time that has already passed. Given that the simulated time at the time of their technological processing is already gone, the two bad options are to drop them after investigating their time stamp or, without investigating it, process their message content at the wrong time. A special chance to mishandle those events, that some delayed event will survive in their buffer the end signal of the experiment, and after terminating the experiment (providing no more inputs), they will arrive as valid events (with valid time stamp, from the past), providing the sensation that "Lack of Sleep Could Be a Problem for AIs"[6].

In vast artificial neural systems "*Yet the task of training such networks remains a challenging optimization problem. Several related problems arise: very long training time (several weeks on modern computers, for some problems), the potential for over-fitting (whereby the learned function is too specific to the training data and generalizes poorly to unseen data), and more technically, the vanishing gradient problem*" [48]. In the light of our analysis, we can make an important addition: as the *time* (in its many manifestations) is not present in the many-parameter fitting in explicit form, its effect is attributed to some synaptic weight(s), resulting in mis-fitting.

Some works already guessed that the technological speed of propagating information may be an issue for artificial networks. A trivial way to provoke faster propagation is to include fewer computing nodes in the network. "*The immediate effect of activating fewer units is that propagating information through the network will be faster, both at training as well as at test time.* [48]" However, a natural consequence is that (see their Fig. 5): "*As λ_s increases, the running time decreases, but so does performance.*"

One can expect that considering the temporal behavior, in any form, can enhance learning. Introducing the spatio-temporal behavior of ANNs, even in its simple form, using separated (i.e., not connected in the way proposed in [23]) time and space contributions to describe them significantly improved the efficacy of video analysis [54].

7. The effect of periodic operation

In biology, the observer starts his/her observation "in medias res": in a living organism, where a stationary, stable "parameter space" exists. However, learning may definitely change the pre-wired settings [41], both in the synaptic strengths and in the number of participating network nodes. Studying learning (and especially pre-wired states) needs extreme caution and is challenging not only from the technical point of view.

In technology, the synaptic weights, connections, and other factors influencing neural operation must be set up by the experimenter, requiring pre-wiring the system. The lack of information about the initial parameter space settings, and the need to produce a valid initial space setting for the beginning of the training, needs to make a bargain. Usually, the system is allowed to learn (i.e., set its weights according to the rules) from an initial random or uniform or some otherwise pre-set state. Given that no synchronization signal is provided, the system assumes all signals to be valid, including the different feedback, recurrent relations, and other signals (for a review see [4,26]).

This also means that *the computed feedback may reach the neurons in the previous layer before the previous layer could compute their needed inputs*. Because of the lack of synchrony signal, the computation considers the feedback signal all the time as a full-value signal, even if it is based on uninitialized signals. The result is that even the weights that were initially correct may be destroyed and the starting point moved in the parameter space to a wrong position, even if originally the point was at the right position [6]. This effect

[6] https://www.lanl.gov/discover/science-columns/top-columns-and-blogs/2020/sciam-artificial-sleep.php

is topped by the difference that biology naturally implements a limitation for the speed of change for its biological weights. The "need for speed" in technological computing does not implement such moderation.

The role of time in learning comes to light demonstratively when analyzing video recordings. This corresponds to the case that the training is based on a series of slightly different samples, where different objects are varying at different speeds from frame to frame. One time constraint is how frequently the sample object changes, the other one how quickly its analysis is performed. One can expect that a slow change gives more time for the system to learn; the rapid changes cannot be learned: seen too few times. Our expectation on the role of time (mismatching) is confirmed directly via making investigations in the time domain. "*The CNN models are more sensitive to low-frequency channels than high-frequency channels*" [55]: the feedback can follow the slow changes with less difficulty compared to the faster changes.

8. Conclusions

The time needed to transfer information is not considered in technological computing and considered incompletely in biological computing. The computing model, proposed by von Neumann, correctly describes both kinds of computing, but the improper neglections cause failures. The different implementations lead to drastically different features of computing, which makes true biology-mimicking simulations at least hard, when exceeding toy-level complexity. The technology could be made able to fabricate more appropriate biology-mimicking computing systems, but first, it should be considered that in the two implementations, the information transfer's speed differs by several million times, and this difference leads to conceptual errors manifesting in failed vast (by intention:) biology-mimicking systems. The time, crucial for biological neural operation, cannot appropriately handled in current technological neural imitations.

Acknowledgement

Project no. 136496 has been implemented with the support provided from the National Research, Development and Innovation Fund of Hungary, financed under the K funding scheme.

References

1. von Neumann, J. First Draft of a Report on the EDVAC. https://web.archive.org/web/20130314123032/http://qss.stanford.edu/~godfrey/vonNeumann/vnedvac.pdf, 1945.
2. Williams, M.R. The Origins, Uses, and Fate of the EDVAC. *IEEE Ann. Hist. Comput.* **1993**, *15*, 22–38. doi:10.1109/85.194089.
3. Godfrey, M.D.; Hendry, D.F. The Computer as von Neumann Planned It. *IEEE Annals of the History of Computing* **1993**, *15*, 11–21.
4. Schuman, C.; et.al. A Survey of Neuromorphic Computing and Neural Networks in Hardware **2017**.
5. Asanovic, K.; Bodik, R.; Demmel, J.; Keaveny, T.; Keutzer, K.; Kubiatowicz, J.; Morgan, N.; Patterson, D.; Sen, K.; Wawrzynek, J.; Wessel, D.; Yelick, K. A View of the Parallel Computing Landscape. *Comm. ACM* **2009**, *52*, 56–67.
6. Végh, J. Which scaling rule applies to Artificial Neural Networks. Computational Intelligence (CSCE) The 22nd Int'l Conf on Artificial Intelligence (ICAI'20). IEEE, 2020, pp. Accepted ICA2246, in print, [2005.08942].
7. Markovic, D.; Mizrahi, A.; Querlioz, D.; Grollier, J. Physics for neuromorphic computing. *Nature Reviews Physics* **2020**, *2*, 499–510. doi:https://www.nature.com/articles/s42254-020-0208-2.pdf.
8. Hermann Minkowski. Die Grundgleichungen für die electromagnetischen Vorgänge in bewegten Körpern. *Nachrichten von der Königlichen Gesellschaft der Wissenschaften zu Göttingen (in German)* **1908**, p. 53–111.
9. Walter, S. Hermann Minkowski and the scandal of spacetime. *ESI News* **2008**, *1*, 6–8.
10. Hameed, R.; Qadeer, W.; Wachs, M.; Azizi, O.; Solomatnikov, A.; Lee, B.C.; Richardson, S.; Kozyrakis, C.; Horowitz, M. Understanding Sources of Inefficiency in General-purpose Chips. Proceedings of the 37th Annual International Symposium on Computer Architecture; ACM: New York, NY, USA, 2010; ISCA '10, pp. 37–47. doi:10.1145/1815961.1815968.
11. Waser, R., Ed. *Advanced Electronics Materials and Novel Devices*; Nanoelectronics and Information Technology, Wiley, 2012.
12. Esmaeilzadeh, H.; Blem, E.; St. Amant, R.; Sankaralingam, K.; et al.. Dark Silicon and the End of Multicore Scaling. *IEEE Micro* **2012**, *32*, 122–134.
13. Végh, J. Finally, how many efficiencies the supercomputers have? *The Journal of Supercomputing* **2020**, *76*, 9430–9455.
14. Simon, H. Why we need Exascale and why we won't get there by 2020. Exascale Radioastronomy Meeting, 2014, AASCTS2.
15. Hutson, M. Core progress in AI has stalled in some fields. *Science* **2020**, *368*, 6494/927. doi:10.1126/science.368.6494.927.

16. Abbott, A. Documentary follows implosion of billion-euro brain project. *Nature* **2020**, *588*, 215–216. doi:10.1038/d41586-020-03462-3.
17. Maass, W.; Natschläger, T.; Markram, H. Real-Time Computing Without Stable States: A New Framework for Neural Computation Based on Perturbations. *Neural Computation* **2002**, *14*, 2531–2560, [https://doi.org/10.1162/089976602760407955]. doi:10.1162/089976602760407955.
18. Prut, Y.; Vaadia, E.; Bergman, H.; Haalman, I.; Slovin, H.; Abeles, M. Spatiotemporal Structure of Cortical Activity: Properties and Behavioral Relevance. *J. Nourophysiol* **1998**, *79*, 2857–2874.
19. Plenz, D.; Aertsen, A. Neural dynamics in cortex-striatum ci-cultures – II. Spatiotemporal Characteristics of Neural Activity. *Neuroscience* **1996**, *70/4*, 893–924.
20. Végh, J. Why do we need to Introduce Temporal Behavior in both Modern Science and Modern Computing. *Global Journal of Computer Science and Technology: Hardware & Computation* **2020**, *20/1*, 13–29.
21. J. P. Eckert, J.; Mauchly, J.W. Automatic High-Speed Computing: A Progress Report on the EDVAC. Technical Report Report of Work under Contract No. W-670-ORD-4926, Supplement No 4, Moore School Library, University of Pennsylvania, Philadephia, 1945.
22. Luk, W. Imperial College London, textbook. http://www.imperial.ac.uk/~wl/teachlocal/cuscomp/notes/chapter2.pdf, 2019.
23. Végh, J. Introducing Temporal Behavior to Computing Science. 2020 CSCE, Fundamentals of Computing Science. IEEE, 2020, pp. Accepted FCS2930, in print, [2006.01128].
24. Pereda, A.E. Electrical synapses and their functional interactions with chemical synapses. *Nature Reviews | Neuroscience* **2014**, *15*, 250–263.
25. Koch, C. *Biophysics of Computation*; Oxford University Press, 1999.
26. Schliebs, S.; Kasabov, N.K. Evolving spiking neural networks: A Survey. *Evolving Systems 4(2)* **2013**, *2*. doi:10.1007/s12530-013-9074-9.
27. Backus, J. Can Programming Languages Be liberated from the von Neumann Style? A Functional Style and its Algebra of Programs. *Communications of the ACM* **1978**, *21*, 613—-641.
28. Végh, J.; Berki, A.J. On the spatiotemporal behavior in biology-mimicking computing systems. *Researchgate* **2020**.
29. Végh, J. Revising the classic computing paradigm and its technological implementations. *Computing* **2020**, p. in review.
30. Kumar, S.; et al. Acceleration of an Asynchronous Message Driven Programming Paradigm on IBM Blue Gene/Q. 2013 IEEE 27th International Symposium on Parallel and Distributed Processing; IEEE: Boston, 2013.
31. Grubl, A.; Billaudelle, S.; Cramer, B.; Karasenko, V.; Schemmel, J. Verification and Design Methods for the BrainScaleS Neuromorphic Hardware System. *Journal of Signal Processing Systems* **2020**, *92*, 1277–1292. doi:https://www.nature.com/articles/s42254-020-0208-2.pdf.
32. Antle, M. C. and Silver, R.. Orchestrating time: arrangements of the brain circadian clock. *Trends Neurosci.* **2015**, *28*, 145–151.
33. Sterling, P.; Laughlin, S. *Principles of Neural Design*, 1 ed.; The MIT Press, 2017.
34. Buzsáki, G. *The Brain from Inside Out*, 1 ed.; Oxford University Press, 2019.
35. Strukov, D.B.; Snider, G.S.; Stewart, D.R.; Williams, R.S. The missing memristor found. *Natures* **2008**, *453/7191*, 80–83.
36. Benke, T.A.; Lüthi, A.; Isaac, J.T.R.; Collingridge, G.L. Modulation of AMPA receptor unitary conductance by synaptic activity. *Nature* **1998**, *393*, 1629–1636.
37. Pajevic S, Basser PJ, F.R. Role of myelin plasticity in oscillations and synchrony of neuronal activity. *Neuroscience* **2014**, *13*, 135–147. doi:10.1016/j.neuroscience.2013.11.007.
38. Ford, M.; Alexandrova, O.; Cossell, L. Node of Ranvier length as a potential regulator of myelinated axon conduction speed. https://www.nature.com/articles/ncomms9073, 2015.
39. Arancibia-Cárcamo, I.L.; Ford, M.C.; Cossell, L.; Ishida, K.; Tohyama, K.; Attwell, D. Node of Ranvier length as a potential regulator of myelinated axon conduction speed. https://pubmed.ncbi.nlm.nih.gov/28130923/, 2017. doi:doi:10.7554/eLife.23329.
40. Almeida, R.G.; Lyons, D.A. On Myelinated Axon Plasticity and Neuronal Circuit Formation and Function. *J. Neuroscience* **2017**, *37*, 10023–10034.
41. McKenzie, S.; Huszár, R.; English, D.F.; Kim, K.; Yoon, E.; Buzsáki, G. Preexisting hippocampal network dynamics constrain optogenetically induced place fields. 10.1101/803577, 2019. doi:10.1101/803577.
42. Losonczy, A.; Magee, J. Integrative properties of radial oblique dendrites in hippocampal CA1 pyramidal neurons. *Neuron* **2006**, *50*, 291–307.
43. Patterson, D.; Hennessy, J., Eds. *Computer Organization and design. RISC-V Edition*; Morgan Kaufmann, 2017.
44. Amdahl, G.M. Validity of the Single Processor Approach to Achieving Large-Scale Computing Capabilities. AFIPS Conference Proceedings, 1967, Vol. 30, pp. 483–485. doi:10.1145/1465482.1465560.
45. Végh, J. How Amdahl's Law limits performance of large artificial neural networks. *Brain Informatics* **2019**, *6*, 1–11.
46. Kunkel, S.; Schmidt, M.; Eppler, J.M.; Plesser, H.E.; Masumoto, G.; Igarashi, J.; Ishii, S.; Fukai, T.; Morrison, A.; Diesmann, M.; Helias, M. Spiking network simulation code for petascale computers. *Frontiers in Neuroinformatics* **2014**, *8*, 78. doi:10.3389/fninf.2014.00078.
47. van Albada, S.J.; Rowley, A.G.; Senk, J.; Hopkins, M.; Schmidt, M.; Stokes, A.B.; Lester, D.R.; Diesmann, M.; Furber, S.B. Performance Comparison of the Digital Neuromorphic Hardware SpiNNaker and the Neural Network Simulation Software NEST for a Full-Scale Cortical Microcircuit Model. *Frontiers in Neuroscience* **2018**, *12*, 291.

48. Bengio, E.; Bacon, P.L.; Pineau, J.; Precu, D. Conditional Computation in Neural Networks for faster models. ICLR'16:, 2016, [arXiv:cs.LG/1511.06297].
49. Keuper, J.; Preundt, F.J. Distributed Training of Deep Neural Networks: Theoretical and Practical Limits of Parallel Scalability. 2nd Workshop on Machine Learning in HPC Environments (MLHPC). IEEE, 2016, pp. 1469–1476. doi:10.1109/MLHPC.2016.006.
50. Chicca, E.; Indiveri, G. A recipe for creating ideal hybrid memristive-CMOS neuromorphic processing systems. *Applied Physics Letters* **2020**, *116*, 120501, [https://doi.org/10.1063/1.5142089]. doi:10.1063/1.5142089.
51. Building brain-inspired computing. *Nature Communications* **2019**, *10*, 4838.
52. Abraham, I. The case for rejecting the memristor as a fundamental circuit element. *Scientific Reports* **2018**, *8*, 10972. doi:10.1038/s41598-018-29394-7.
53. Kendall, J.D.; Kumar, S. The building blocks of a brain-inspired computer. *Appl. Phys. Rev.* **2020**, *7*, 011305. doi:10.1063/1.5129306.
54. Xie, S.; Sun, C.; Huang, J.; Tu, Z.; Murphy, K. Rethinking Spatiotemporal Feature Learning: Speed-Accuracy Trade-offs in Video Classification. Computer Vision – ECCV 2018; Ferrari, V.; Hebert, M.; Sminchisescu, C.; Weiss, Y., Eds.; Springer International Publishing: Cham, 2018; pp. 318–335.
55. Xu, K.; Qin, M.; Sun, F.; Wang, Y.; Chen, Y.K.; Ren, F. Learning in the Frequency Domain. https://arxiv.org/abs/2002.12416, 2020, [arXiv:cs.CV/2002.12416].