Advances in Artificial Intelligence and Machine Learning; Review 1 (2) 136-154 Received 20-07-21; Accepted 05-08-21; Published 20-08-21

Why Learning and Machine Learning Are Different

János Végh VEGH.JANOS@GMAIL.COM
Kalimános BT, Komlóssy u 26, Debrecen, 4032, Hungary.
ÁdámJózsef Berki BERKI.ADAM@YAHOO.COM
University of Medicine, Pharmacy, Science and Technology of Târgu Mureş, 540142 Târgu Mureş, Romania.

Corresponding Author: János Végh.

Abstract

Machine learning intends to be a biology-mimicking learning method, implemented by means of technical computing. Their technology and methods, however, differ very much; mainly because technological computing is based on the time-unaware classic computing paradigm. Based on the time-aware computing paradigm, the paper discovers the mechanism of biological information storing and learning; furthermore, it explains, why biological and technological information handling and learning are entirely different. The consequences of the huge difference in transmission speed in those computing systems may remain hidden in "toy"-level technological systems but comes to the light in systems having large size and/or mimicking neuronal operations. The biology-mimicking technological operations are in resemblance to the biological operations only when using time-unaware computing paradigm. The difference leads also to the need of introducing "training" mode (with desperately low efficiency) in technological learning, while biological systems have the ability of life-long learning. It is at least misleading to use technological learning methods to complement biological learning studies. The examples show evidence for the effect of transmission time in published experiments.

Keywords: Learning, machine learning,Computing efficiency,Temporal logic, Time-Aware computing, Neuronal computing, Computing paradigm.

1. INTRODUCTION

The ability to learn is a key factor in both the evolution of life and our individual survival. "The broad definition of learning: use present information to adjust a circuit, to improve future performance" needs explanation also [1], what is the "present information", and information at all; furthermore it implies that learning is a temporal process. There are many different definitions what actually 'information' means [2]. In neural science it was observed and theoretically discussed [3] that when using neuronal spikes, "the more precisely spike timing is measured, the more information is gained". However, the conclusion, that timing delivers information, is missing from the information theory. Information handling is an overly complex process, and it includes coding, decoding, transferring, storing, and retrieving processes, among others.

During learning, the organ uses the past information stored in its internal state variables. However, "we should not seek a special organ for 'information storage' — it is stored, as it should be, in every circuit" [1]. The definition also suggests to consider learning (and, because of this: information processing) as a temporal process: the organs have a sensing time, a processing time, a storage access time (despite that "information stored directly at a synapse can be retrieved directly" [1]) and a transfer time (conduction time) to the next computing organ. The general model of computing [4], based on the time-aware computing paradigm [5], correctly describes the general learning process (is underpinned by anatomical evidence, but still needs dedicated experiments, designed with time-aware behavior in mind).

2. THEORY OF TEMPORAL BEHAVIOR

The speed of interaction is finite [6] both in biological computing and technological computing. The finite speed contributes finite transfering timing between computing units. The effect is well known, but neglected. As pointed out [7], von Neumann's computing model mentioned the fact that in principle also transfer time must be considered. In the approximation which targeted implementation using vacuum tubes only, it could be neglected. However, the quickly developing technology invalidated that approximation in a stealthy way. For today, timing relations of technological computing are entirely different from those assumed in the classic computing paradigm, which computing science is based upon.

Von Neumann in his famous 'report' did provide a "procedure" only for the case when the transmission time can be neglected apart processing time. Actually, von Neumanns statement was that

```
if [timing relations of] vacuum_tubes
then Classic_Paradigm;
else Unsound;
```

The more general computing paradigm shall provide the proper procedure instead of the 'unsound' branch, when this omission corresponding to [the time relationships between] electron tubes is not valid.

2.1 The General "Procedure"

In the general computing procedure we need to work – in addition to the spatial coordinates – also with the time coordinate of the computing events; considering that those coordinates are connected by the interaction speed. That is, we introduce a 4-D coordinate system, which we call *time-space* system. This representation is in close resemblance with Minkowski's famous *space-time* system; with the essential difference that a different scale factor is used. In the space-time system all coordinates have dimension of distance, and the time coordinate is given as the corresponding time multiplied by the interaction speed; free and homogeneous propagation is assumed. In our time-space system [8, 6] the spatial distances are divided by the interaction speed and the time represents itself.

That is, all coordinates have dimension of time; this is the appropriate (measurable) quantity given that both technological and biological computing must meet their timing constraints. We assume a sectioned and directed propagation; furthermore the distance is measured along signal path, rather than calculated from the coordinates of its endpoints.

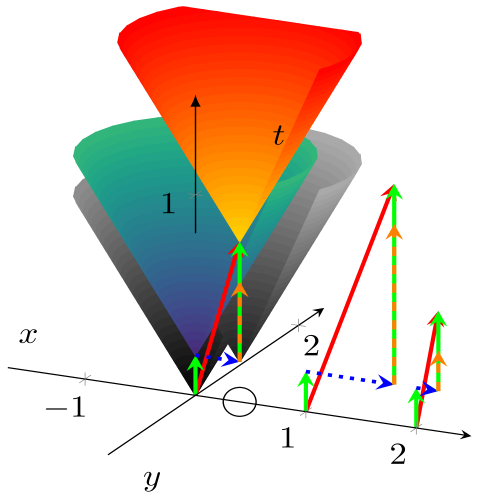

Figure 1: The constrained computing operations in our *time-space* approach. The operating units can be processors, networked computers, gates or neurons; depending on the context. The finite interaction speed and the physical distance of computing elements results in "idle waiting times" (see mixed-color vectors in figure). Idle waiting is one of the major reasons of computing systems' inefficiency.

We describe computing events in this time-space system as constrained vectors, see FIGURE 1. A processing event happens at a given position, but its beginning and end have different time coordinates; that is the processing vector is parallel with axis t. In contrast, a data transfer event changes both spatial and time coordinates simultaneously, that is, the data transfer vectors are not parallel with axis t and are not in the plane perpendicular to it. The vectors are constrained both in the sense that the interaction speed connects their spatial and time coordinates, and that a processing cannot start until its operand arrives, and similarly, delivery of its result cannot start until the processing finished: *the two operations block each other*. This constraint introduces 'idle time' to computing. This idle time in one form leads to inefficiency of processors [9, 7], in another form to inefficiency of parallelized sequential computations [5]. When there are several operands, before the processing may begin, all operands must arrive at their destination.; similarly, all results from multiple processors must be delivered from the output sections of the processing units. These constraints must be emphasized when discussing operation of computing accelerators, especially vector and tensor processors [10, 5].

2.2 The General Computing Model

In the computing model, although not explicitly, von Neumann assumed that special events implement the computing constraints, as depicted in FIGURE 2. Those signals (representing the computing events) are handled differently in different implementations. Even, they may be replaced by some 'foreign' signal, or they can be completely omitted.

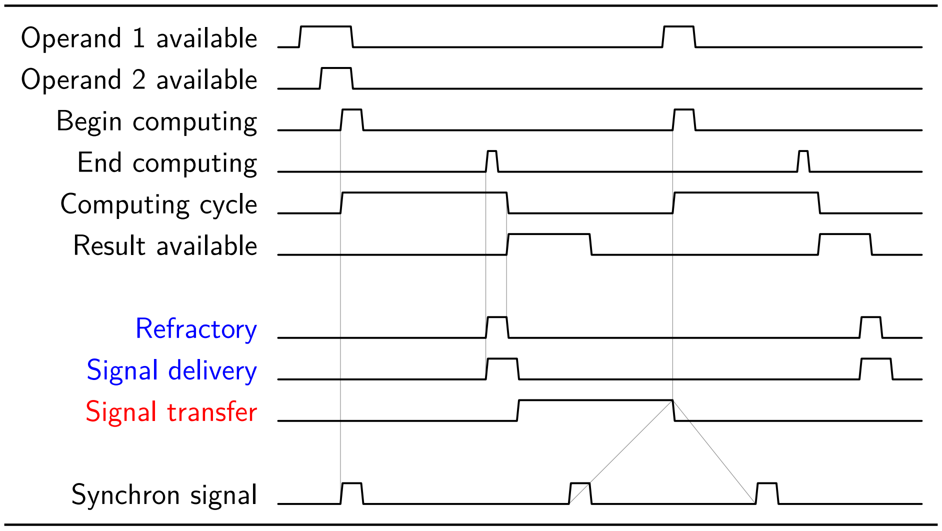

Figure 2: Timing relations of von Neumann's complete timing model, with data transfer time in chained operations; synchronization becomes an issue as the physical size of the computing system grows. Notice that in systems with central clock synchron signals must be bypassed or neglected

Von Neumann warned in his "First Draft" [11], section 5.4, that the operations must be synchronized appropriately (in our wording, the participating processes must be properly aligned to satisfy the blocking constraints).

Although von Neumann suggested to use central synchronization to symplify the initial design, he limited the validity range of his suggestion for the case, when the dispersion of the clock signals is negligible, which is not the case in modern technologies [12, 5]. As the system's physical size grows, the effect of transfer time also grows, or the processing time decreases apart from transfer time. According to von Neumann, the difference in arrival times is critical: "*The emphasis is on the exclusion of a dispersion*" [11]. In addition to its logical contraints, because of its technical implementation, computing faces further limitations.

The operands must arrive at the operating unit before the 'Begin Computing' signal, and the process must issue an 'End Computing' signal. However, the result signal must also be delivered to the "output section" of the unit, so a 'Signal delivery' time is also needed. This latter time must not be confused with the 'Signal transfer' time. In the event of chained operations, this is required, to deliver data from the output section of one computing unit to the input section of another unit (even if, as was the situation with processors, the same physical processor gets used in both operations). *The 'Signal transfer' time, as an approximation, is neglected in the classic computing paradigm.* The approximation, however, is increasingly invalid for modern technological implementations.

3. BIOLOGICAL LEARNING

In biology, the interaction speed (conduction velocity) is million times lower than the electromagnetic wave's propagation speed in wires, so the 'spatiotemporal' behavior of the neuronal operation is commonly accepted (although it is handled by using an inappropriate method, using functions with separable time variable).

3.1 Implementing Synchronization Signals

A spike arriving at a synapse experiences closed gate (zero conductance) [13, 14]. The stalled charge delivery increases the local potential, which opens the voltage-gated channels, enabling charge delivery to the membrane. That is, the 'Operand available' signal is generated by the operand itself; and the arrival of the first operand generates signal 'Begin computing'. The neuron membrane makes the computation: integrates the received charge until it reaches its threshold potential and generates the 'End computing' signal. The time until threshold reached depends both on the synaptic current (conduction speed), and the proportion of charge the $gsyn()$ function (the synaptic strength) of the synapse [13, 14] enables to enter the membrane. When 'End computing' received, inside the neuron membrane a 'Refractory' period begins (prepares for a new computing operation) and at the same time at the hillock a spike is prepared ('Signal delivery'). The 'Signal transfer' period starts when the spike starts towards its destination neuron, outside the source neuron. All signals carry their synchronization signals; biology uses no central clock for synchronization [15]. Instead, it natively synchronizes its signals (aligns them properly). Biology can modulate the interaction speed; delivering the same charge with higher conduction speed means higher synaptic current; an aspect which remains out of sight in technical implementations. The information essentially contains the length of 'Computing cycle' and the end time of 'Signal delivery' (biology uses 'firing rate', the inverse of length of 'Computing cycle', and the phase angle, the offset of the end time of 'Signal delivery' relative to some basic frequency). That is, biology uses *asynchronous auto-synchronization* [16], furthermore it uses a *mixed-mode digital/analog computing method*.

3.2 The Learning Mechanism

Given that spikes arriving to neurons' synapses synchronize themself, a neuron has basically two ways to adjust its time of spiking. Given that in neuronal communication essentially charge collection takes place, the neuron either uses more charge from its synaptic input or uses higher current. In the first case the neuron increases the temporal width of its synaptic function $gsyn()$ [13, 14] through increasing local concentration of neurotransmitters; this mechanism is commonly mimicked in machine learning through modifying the "synaptic weight" W_i of the i-th synapse. In the second case it increases the synaptic current (increases the delivery speed), so the same amount of charge arrives at an earlier time; a mechanism which has no equivalent in machine learning. The dynamic balance of the two mechanisms enables not only for learning, but also for forgetting, replacing, redundancy, rehabilitation, etc. *Noticeable that in both cases only the receiver neuron adjusts itself, the sender neurons need not to adapt themselves.*

The effect in both cases is the same: the threshold at an earlier time is reached by neuron. Biology notices their effect as an increased firing rate (the inverse of the repetition time). The first method is quick (in the millisecond range) but energy wasting: keeping the concentration gradient needs a lot of ion pumping.[1] The second method is slow (in the range of days to weeks): needs anatomical changes, but the operation needs lower energy consumption. That is, biology finds the best timing quickly (short-term learning), but if the external condition persists, it finalizes storing that information: myelinates the corresponding axon (long-term learning; for anatomical evidence see cited references in [17])[2] , and the higher delivery speed enables to decrease the local concentration gradient.These two mechanisms are facets of the same coin: biological neuron adjusts its processing and transfer times. That is, biology stores information locally, in the form of adjusting time (a dynamic form). Any study attempting to grasp biology's information storage and processing in some static form, fails.

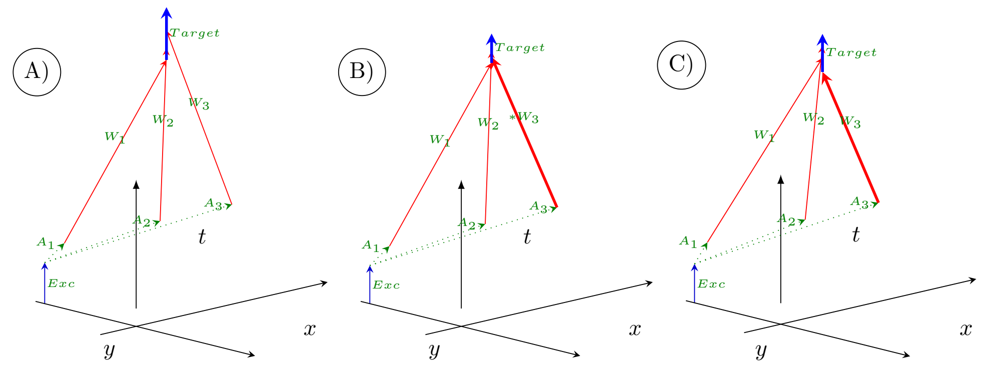

Figure 3: How neurons learn. A) The initial state B) Short time learning, changing synaptic weight by +50% C) Long time learning, changing conduction velocity by +10%.

The cooperation of these two mechanisms is in accordance with Hebb's original observation [18]. He actually observed the effect of long-term learning, in the form that "*the persistence or repetition of a reverberatory activity (or "trace") tends to induce lasting cellular changes*". Actually, Hebbian learning is a two-step process: at the beginning of learning, the synaptic strength is increased. One can presuppose that exceeding membrane's threshold repeatedly using the same pattern, "freezes" neurotransmitter concentration, enabling to notice the changed gradient from the outside world. As Hebb already described: "*When one cell repeatedly assists in firing another, the axon of the first cell develops synaptic knobs (or enlarges them if they already exist) in contact with the soma of the second cell*." In the second step, synaptic strength may decrease to near of its original value (enabling to economize energy), and the learned information is stored anatomically, as the changed thickness of the myelination of the corresponding axon. Notice that here the anatomic consequences are noticed; "the persistence or repetition of a reverberatory activity" actually means a special noise averaging. The primary change, in the first step, is hard to measure.

The equivalence of the 2 mechanisms in getting to know is illustrated in FIGURE3, displaying how the version explains in what form the data is saved and adjusted in biological neuron. Furthermore, it

[1] For mimicking this mechanism, on one side one must consider the finite speed of the neurotransmitters, that is, that the speed of change in the value of W_i is limited. On the other side, transmitters are physically moved from the region correspondig to one synapse to another, that is the sum of all weights must be the same (renormalization by biology).

[2] The evidence that neural networks must "invest" in information processing, is known, see for example chapter "Pricing Neural Information" in [3]. However, connection between different anatomical characteristics and learning, furthermore the "economic aspects" of operation are not discussed.

explains and joins those 2 learning modes. In FIGURE 3 (FIGURE 1 shows an additional coordinate system within it) We believe that the activation of a neuron at the coordinates will be beneficial $(-1,0,0)$ happens and a neuron that fires to participants of an assembly (aligned alongside coordinate axis y at x values -0.3, 0.1 and 0.4). That figure assumes that the activation (after being forwarded to the point of branching) branches toward 3 participants of assembly A. At location, the spikes are transmitted to their common target neuron. $(-1.5, 0.7)$ by A_i members. W_1,W_2,W_3 (assumed of being equal for A) are the analogous synaptic weights of the targeted neuron, and 3acquired spikes may also cause the destination neuron to fire (the aggregate of the three spikes' ability contributions is simply over the threshold).

Given that the location of the assembly participants and their firing timings barely vary, so vary their arrival timings (see the red arrowheads) of the spikes from the assembly participants at goal's position.The frequently cited simplified form of Hebb's rule "cells that fire together wire together" is probably more correct in the form "neurons wire together *if* they fire together" [19]: neural spikes to assembly members shall follow very similar path to be able to fire together.

The $Target$ is initially in state of rest. Once the initial spike arrives, it will increase the membrane's potential, and so do rather later the 2nd spike and third spike, too. With the charge introduced by the 3rd spike, The target reaches its verge, and (after some "sign transport time": charging the membrane) it fires. The computation's length is denoted according to the length of the blue arrow: Its bottom is at the arrival time of the initial spike, its top head is at the start of refractory period of the neuron. Given that once accomplishing the threshold, a while is wanted to membrane's charge to its working potential, the measurement of the arrow consists of a further contribution. For more details see [3]. Notice that some learning details are pre-programmed and that the learned new knowledge is equivalent to the preprogrammed (maybe genetic) one, furher more that learning is a native and life-long ability, as the elegant experiment [20] demonstrated.

Information may have different meanings in different contexts, and neuronal information theory [3] claims that when using neuronal spikes, as observed: "*the more precisely spike timing is measured, the more information is gained*". In the case shown in FIGURE3, the assembly members fire after they receive the spike from neuron *Exc*, so their spikings are not independent. Similarly, *Target's firin g is not independent at all from firing of the other mentioned neurons; just the opposite.* As a consequence, the fundamental assumption of a Poisson model that spikes are generated independently, is not fulfilled. Furthermore, the presence of a spike is digital datum (one bit)[3], the rest is analog. Firing carries the information learned.

The beginning of the spike may shift and the firing rate may change[4]; an analog information content. Our findings are in good accordance with the claim [3], that the information is mainly encoded in the "temporal precision" of the spikes and is estimated to need up to three bits/spike. We can also add that this "temporal precision information" is contributed by the learning process. Given that estimated information content of a spike is between one and three bits [1], the analog information may even dominate. However, the claim that *timing delivers information*, is missing from the mathematical theory; given that it assumes instant interaction. *Using information theory for neural information transfer needs revision* (for a definition of neural information see [3]).

[3] The shape may carry further digital information. According to [23], if the slope of a received spike exceeds a current threshold, it leads to immediate spike generation; enabling direct sychronization. Worth to notice that this *digital* information piece is hidden in the *analog* information (spike shape).

[4] At the level of biological neuronal networks, firing rate is a chief way of information transfer.

4. MACHINE LEARNING

The most attractive feature of biological learning is that the ability of learning seems to be a native feature of the network of circuits, which can cope with the enormous amount of data from their environment, without needing programming efforts. Computing technology attempts to organize various number of computing units (ranging from a few-neuron "toy" systems to supercomputers) into networks and attempts to mimic neuronal operations, including learning. To operate those networks, HW/SW solutions and mathematical methods have been developed, both (more or less) mimicking biology.

By analogy, and to distinguish it from biological learning, the methods used in those networks are called 'machine learning' (sometimes mistakenly identified as "artificial intelligence" [21]). Those networks are built from components designed for processor-based computing, and those "biology-mimicking" systems are missing essential features of biological computing. Under these conditions, in some very specialized fields (such as playing a game or identifying a pattern), those systems can show up remarkable successes. However, in general, given that the performance scaling is strongly non-linear [9], "*core progress in AI has stalled in some fields*" [22].

Although it is hard both to list the crucial differences between technological components attempting to mimic a biological component and to separate completely the HW and SW issues, some of the most important ones are mentioned below. In addition, some neuronal-operation specific issues are also discussed. The final reason is in all cases the time-unaware computing paradigm.

4.1 Implementing Synchronization Signals

Special care must be taken when handling signals 'Begin computing', 'End computing' and 'Signal delivery' signals[5]. An operating unit always has input signal levels for its operands and provides output signal levels for its results, but the values of those signals correspond surely to their expected values only if the timing signals are properly aligned. All operands ought to be brought to the enter phase of the computing unit before 'Begin Computing' arrives. Similarly, 'Signal delivery' can begin only after 'End computing' arrives, and it can terminate only after all operands delivered to the corresponding output sections. Especially in the case of more number of operands and results, one may need a shared bus for delivering them, which can prolong the delivery time; to an unpredictable value.

For real-life computations chained computing operations must be considered and their phases must be appropriately synchronized. Synchronization can be accomplished by different means. The availability signal for an operand can be generated either by a central control unit (synchronous operation) or on a per-operand basis (asynchronous operation). Most of the today's technical computing systems apply a central clock. As an admission that the commonly utilised synchronised operating mode is extremely inefficient in non-dispersion less systems, the concept of asynchronous operation is around [24-27]. The central synchronization means that signals 'Operand available', 'Begin computing', 'Result available', etc. are derived from central clock signal as an internal offset time to it. Signal 'Refractory' is implemented as a short internal reset of circuits, immediately before

[5] Modern accelerating solutions in implementations, such as pipelining, 'out of order' execution, branch predicting, etc. make details of description much more complex, but the logical need of considering these signals persists.

the falling edge of the synchron signal. Similarly, 'Signal delivery' must finish before the falling edge of the synchron signal arrives. Given that the length of the 'Signal transfer' is undefined, in technical designs transfer time is not included; that is, immediate interaction (infinitely large transfer speed) is assumed.

At inter-component level the "skew" of clock signal leads to introducing clock domains and clock distribution, and causes dozens of percentage loss in energy efficiency [12, 28]. Neglecting the intra-component skew introduces non-intended flops. Both effects increase the dispersion of clock signals [7], and is majorly responsible for the experienced inefficiency [9] of computing. Considering transfer time leads to, among others, discovering the performance limitation of supercomputers [29], Artificial Neural Network ANNs [10] and brain simulation [30] and explains why "Core progress in AI has stalled in some fields" [22].

We can add: the proper synchronization needs special care when using accelerated and parallelized computations. Similarly, one must think about the temporal sequence when considering results from feedback and recurrent relations. For examples see section 5. On one side, it is true that "*The analog memristor*[6] *array is effectively the neural network laid out in the form of a crossbar, which can perform the entire operation* ***in one clock cycle***" [27]. On the other side, as in brackets fairly added: *"(not counting the clock cycles that may be required to fetch and store the input and output data)"*; which action may take clock cycles in the order of the number of operands; furthermore, it may comprise very time-consuming memory access or I/O operations. All the memristor array's operands must arrive at its input section (They must also be computed or otherwise manufactured before they may be used), furthermore the operation terminates only when all results are sent to their destination from its output section. To form a fair analogy, the total computing time (including data transfer time) shall be compared to conventional operations' total computing time. The total time of operation must include data delivery to and from the memristor array. Even when continuous-time data representation is utilised, the problem still exists [32], where the need for additional clock cycles is not mentioned.

4.2 Hardware Issues

In technological computing, we have technology blocks, among others, memory chips. In biology, "we should not seek a special organ for 'information storage' — it is stored, as it should be, in every circuit"; the idea of "in-memory computing". The information "is stored directly at a synapse and can be retrieved directly" [1]. In technological computing, registers are essentially another memory with short address and short access time; simulating data stored directly at synapse would need a huge number of registers. Using conventional "far" memory instead of the direct register memory introduces two-three orders of magnitude worse performance. Summing synaptic inputs means scanning all registers, which needs several thousands of machine instructions; introduces another three-four orders of magnitude worse performance.

[6] The temporal behavior of components and their materials can easily be misidentified in the time-unaware model. Even *memristance* has been introduced [31] 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. It happened five decades ago; for today there are some serious doubts as to whether a genuine memristor can actually exist in physical reality [34]. In our analysis demostrated, some temporal behavior surely exists; the question is how much it is related to material or biological features, if our time-aware computing method is followed.

Biological axons, from computational point of view, represent a private connection between computing units, but billions of such (very low speed) “buses” work in parallel, without contention. The usual technological implementation is to use a single very high speed bus, which must be “owned” for every single communication. When using a serial bus, Arbitration consumes the great majority of probable processing time (see FIGURE 4 for a temporal diagram and [8] for a case study). Given that the required arbitration (which grows with the number of neurons!) rather than the bus speed limits the transfer time, utilising a single high-speed bus in large-scale systems is at best dubious: “The idea of using the popular shared bus to implement the communication medium is no longer acceptable, mainly due to its high contention.” [33]. Bus arbitration, addressing, and latency all increase time to the transfer process (and reduce system efficiency). This form of communicational burst can easily result in a “communicational collapse [35],” but it can also result in accidental “neuronal avalanches” [36]. According to [8], the shared medium is the largest contributor to computing time consumption at a large number of communicating units.

FIGURE 4 represents the case of multiple processing units connected to the bus in terms of time-aware computing. It explains why linking current computer components with high-speed buses results in significant performance degradation. The processors are positioned at (−0.3,0) and (0.6,0) on the temporal diagram, whereas the bus is at location (0,0.5). The two processors wish to convey their results to their destinations when they’ve completed their computations (green arrows at processor locations). We assume they want to communicate at the same moment in the diagram. They must first be able to board the shared transport (red arrows). Because the core at (−.3,0) is closer to the bus in terms of proximity, it receives priority. The grant signal is then sent to the asking core, which starts the bus operation, and the data from the core is sent to the bus. When the data reaches to the bus, it is sent because of the bus’s rapid speed. The other core’s bus request can then be granted, and the second core’s computed result can ultimately be bused. In both situations, notice the variation in temporal distances between the ‘bus requested’ and ‘bus granted’ messages, FIGURE 4.

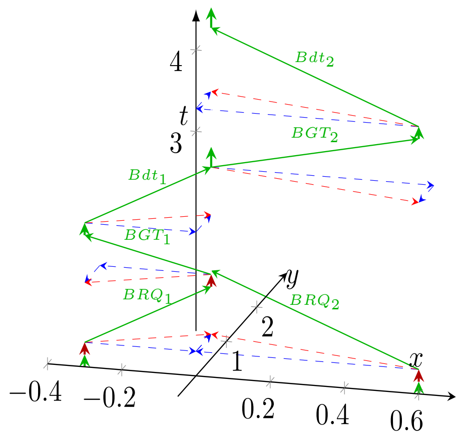

Figure 4: The temporal operating diagram of a technological high-speed single bus: the bus delivers data only in the fractions denoted by vertical green arrows.

In the figure, we pretended that our two cores wanted to communicate using a single shared bus at the same time. The effective processing time is several times longer than the actual processing time since they have shared access to the bus. Furthermore, if a single high-speed bus is employed, the number of cores linked to the bus increases the overall time of operation linearly. Because the (temporal) distance between cores is often in the picoseconds region, but the bus (and the arbiter) are at a distance much beyond nanoseconds, the real temporal behaviour of the bussed data (and the idle time resulting from it) is significantly worse than the figure depicts.

When using a shared bus, raising the processing or communication speeds has no linear effect on the total execution time. Furthermore, performance is not limited by bus speed. A relatively modest change in transfer time can result in a quite substantial change in processing time value. This change causes an inexplicable slowdown in the computing system: The slowest component plus mutual blocking determines the system's efficiency.

Given that conventional processors are implemented in SPA, as Amdahl coined the wording [37], communication between them is implemented through I/O instructions. This approach is why "*artificial intelligence, it's the most disruptive workload from an I/O pattern perspective*[7]". Better performance can be hoped only in a drastically different approach [38]. Technological computing optimizes for single-thread performance without networking, biology uses simple processors with very good networking ability.

Biology uses three-state logic, which enables a very low energy consumption. Neurons are "off" until some input is received by their synapses, then for some time they get "open", after some time "inactivated", and again after some time "off" again. As discussed in [17], the three-state operation mode is desirable from several points of view; among others it defines the direction of time. The present design point of view, replacing "inactivated" state with a "down" edge, enables reaching higher operating frequency [4], but large designs shed light on the limitations, caused by attempting to replace the energetically needed three-state operation [39, 40] with a simplified two-state operating mode.

4.3 Software Issues

As discussed in connection with biological neurons, signal timing has crucial role in network's operation. SW methods, not handling explicitly the simulated time, cannot align computing constraints properly. The usual SW organization is prepared to imitate single-thread processes. There are sofware libraries (such as SystemC [41]) which provide special engine enabling event scheduling on top of OS's event scheduling. However, handling events is only possible through using services of the OS, which is very costly in terms of non-payload instructions [42, 43]. Handling events at a reasonable non-payload performance, one needs to have entirely different architecture [38].

It was already admitted that "building this new hardware [neuromorphic computing] necessitates reinventing electronics"; furthermore that "more physics and materials needed" [26]. 'Rebooting computing' is needed, from the ground up.

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

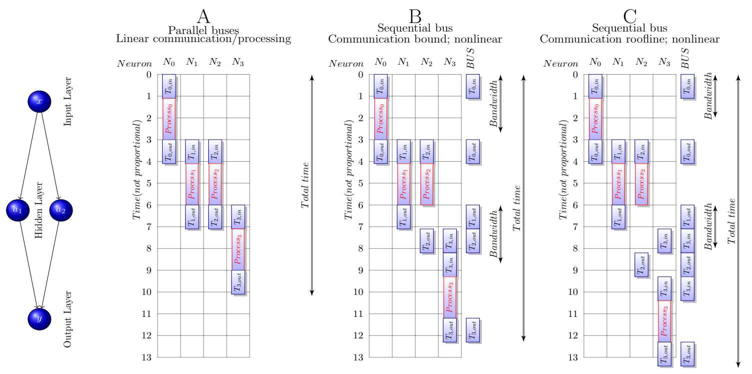

Figure 5: Implementing neuronal communication in different technical approaches. For a more detailed legend, see text. A (the biological implementation): the parallel bus; B and C(the technical implementation): the shared serial bus, before and after reaching the communication "roofline" [41].

4.4 Neuronal Issues

Given that neurons must exchange information about the result of their elementary operations after every single operation, an extra handicap is introduced by ANN type workload. A simple neuromorphic use case is shown in the inset of FIGURE 5.3. one output neuron and One input neuron are connected by a hidden layer. The hidden layer comprises only two neurons.

FIGURE 5A shows a system of "parallel buses", the biological implementation: (axons). In a parallel implementation like this, with increasing either payload processing speed or non-payload communication speed, the total time (processing + transmitting), that is system's performance, increases linearly.

FIGURE 5B shows the technological implementation of a high-speed shared bus for exchnging information.; see also FIGURE 4. The action that loads the bus at the stated time is to the right of the grid. The communication bandwidth is represented by a double arrow, which has a length proportionate to the amount of packages the bus can convey in a given time unit. We assumed that, input neuron may convey all of its information in a single message to the hidden layer, and that processing in the hidden layer begins and ends at the same time. Because neurons compete for access to the bus, only one of them is capable of delivering a message right away, others will have to wait until the bus arrives. The message can only be received by the output neuron once the first neuron has finished transmitting it. The second message from the bus must first be obtained by the output neuron, and it can only begin processing after both input arguments have arrived. During non-payload processing and payload processing, this limitation generates successive bus delays in the hidden layer and output neuron. Adding more neurons to the layer causes the delay to increase.

There are two possible outcomes at this stage. Either the 2ndneuron must wait until its second input arrives (in biology, a spike also carries its synchronisation signal and triggers integration), or it can update its output in real time as inputs arrive, if its processing speed permits. (in "technical neurons," this synchronisation facility isn't used because continuous levels are used instead of pulses).In the latter scenario, however, until the second input arrives, the neuron produces an output signal that is different from the one predicted by the mathematical dependence (and is processed). The output signal, which may be briefly incorrect [8], is known in the field of electronics, and those "glitches" are eliminated by applying a "worst-case" delay for the output signal. Incorporating a serial bus into that calculation would significantly increase the required "worst-case" delay.

Because the bus must be reached twice throughout the T_B time, (not just the operand given to the bus, but also the time spent waiting for arbitration: the right to use the shared bus). The physical delivery T_d through bus must be included, making the transfer time $T_t = 2.T_B+T_d+X$. "Foreign contribution" is represented by the **X**. Any other traffic burdens the bus if it is not devoted to "neurons in this layer exclusively." Processing may be slowed by messages from other layers as well as general system communications (and can contribute to faking the biological function that is being imitated).

Even if the hidden layer has only one neuron, it must apply the process of bus sharing on a case-by-case basis. A direct link to a nearby neuron takes longer than physical delivery to the bus (The arbiter and the bus are both in the cm range, implying many n sec transfer periods; direct transfers between connected gates might be in the p sec range.) If there are more neurons on the bus (such as a buried layer) working in parallel, they must all wait for the bus. The high-speed bus is only mildly loaded when only a few neurons are present. The number of neurons in the buried layer rises linearly with the load on the bus (or, possibly, all neurons in the system), but the bus's temporal behaviour is different.

When mimicking neuronal operation, all inputs from the buried layer must be waited for by the second neuron. The neuron behind the concealed layer has a transmission time of $T_t = L.2.T_B + T_d + \mathbf{X}$ if there are L neurons in the hidden layer.

The reason for this is explained by the temporal behaviour. "*shallow networks with many neurons per layer scale worse than deep networks with less neurons*" [45]: *If the layer organisation requires several arbitrations to reach the bus, both the physical bus delivery time T_d and the processing time T_p become negligible*: transfer time is determined by the amount of neurons in the buried layer. Although the common bus continues to limit communication in deeper networks, the system's messages may be transmitted at different times in each of its layers (despite having independent buses across the layers). There is no method to organise the message traffic because there is just one bus.

The role of the system's burden becomes clear here: both neurons in the hidden layer wish to communicate using the single common bus at the same time. The perceived processing time is several times greater than the physical processing time because they obstruct each other, and it increases in direct proportion to the amount of neurons in the hidden layer (if a single high-speed bus is employed, and possibly also with the total number of neurons in the system).

As the scale of the system grows, the fraction of time required to forward data on the high-speed bus gradually reduces. Bus speed becomes marginal in excessive systems, especially when they attempt to imitate neuromorphic workload.The times depicted in the graph are not in any way proportional. The (temporal) distances between cores are in the hundreds of picoseconds range, whereas the bus (and the arbiter) are well above nanoseconds. The real temporal behaviour (and the resulting idle time) is far worse than the graph depicts.

In addition to wasting valuable time for contenting for the exclusive use the single serial bus, utilization of the bit width (and package size) is also inefficient. The information is mainly encoded in the "temporal precision" of the spikes and is estimated to need up to three bits/spike [3]. The time-unaware time stamping of artificial spikes does not encode temporal precision, given that the message comprises *sending* the time coordinate of the event, instead of the relative time to a local synchron signal (a base frequency); furthermore the conduction time is not included in the message: the time coordinate of *receiving* the time-stamped event is missing.

5. TRAINING ANNS

One of the most startling characteristics of ANNs is their multi-week training duration, even for (in comparison to brain functionality) easy tasks. The mathematical methods, on the other hand, do not include time-dependence; nevertheless, the technical implementation of ANNs does. New neuronal outputs (which provide as new neural inputs as well) are only loosely coupled in terms of delivery times, as discussed in [8]: the assumption that producing an output means simultaneously producing an input for some other neuron only works in timeless "classic computing," as discussed in [8].

Show the temporal diagram of a 1-bit adder in [8], to see what effect considering temporal behaviour might have. When we use adders, we have a set amount of time to read out the result. We don't want "glitches," so we specify a maximum period until all bits relax, and we'll only get the final result (at the cost of some performance); the adder is synchronised.

The case of ANNs, on the other hand, is unique. The bits are plugged directly in the adder, and the outputs of bit n are the inputs of bit n + 1 (there is no feedback). The signals in ANNs are provided via a bus, and the type and sequence of interconnections are determined by a variety of criteria (starting with the type of tasks and ending with the actual inputs). The case becomes more complicated when ANNs are being trained.

The sole constant in timing is that a neural input will always arrive after a partner has generated it. The timing of delivered events, on the other hand, is uncertain: it is determined by technological delivery factors rather than the logic that generates them. There are two awful options when it comes to time stamping. Option one is for neurons to have a (biological) sending time-ordered input queue and to start processing once all partner neurons have sent their messages. This necessitates the use of a synchronisation signal and results in significant performance loss (in parallel with the one-bit adder). New neuronal outputs (which function as new neuronal inputs) are only weakly coupled in terms of delivery times. This technical approach allows us to provide feedback to a neuron that fired later (based on its time stamp) and define a new neuronal variable state; this is a "future state" while processing a message received physically later but with a time stamp pointing to a biologically earlier time.

A third, perhaps better, alternative is to keep a biologically-timed queue and send out feedback and output in time windows (much less time than the commonly used "grid time"), or to independently evaluate received events and transmit feedback and output promptly. In both circumstances, it's worth considering whether their impact is significant (exceeds a certain tolerance threshold as compared to the previous condition) and whether they can reduce the need for communication in this way.

We start showing an input when training ANNs, and the system starts to work. It validates its synaptic weights before displaying that input. Its initial weights might be randomised, or they could be based on past input data. The system sends correct signals, a receiver, on the other hand, processes a signal only after it has been physically transmitted[8], meaning that it may happen that it (and its dependents) start to adjust their weights to a state that is not yet defined. In **[8]**, the indefinite time of the first AND gate is quite a short, while the OR has a long indefinite time.

When playing chess against another player, a quicker computer can be utilised to analyse upcoming moves more effectively. Even before its opponent makes the next move, it can calculate all possible future moves. However, before publishing its next move, it must wait for its opponent's next move, Otherwise, it might publish a wrong move. Sending the feedback to its opponent as soon as the next move is computed –i.e., without synchronization– results in a quickly computed but maybe wrong move. The faster the computer is, the lower its performance as a chess player is without synchronisation. When it comes to synchronisation, the faster the computer is, the more idle activity it has.

At the operation's beginning, Some of the network's component neurons may have undefined states and weights. In their operation (essentially an iteration), without synchronization, In most circumstances, the actors employ incorrect input signals, and they will almost certainly adjust their weights to false signals at first, and with significant time delay at a later stage.

If we're lucky (and keep in mind that in the case of more complicated systems, we're dealing with unstable states), the system will converge, albeit painfully slowly. Alternatively, nothing at all. Neglecting the temporal nature of networks results in excruciatingly sluggish and shaky convergence.

Even in ANNs, synchronisation is required. When using accelerators, feedback, and recurrent networks, we must exercise caution. The time matters. Computing neural outcomes faster in order to deliver faster feedback isn't going to help. It takes time to deliver feedback information, and in order to do so, we must use the same common media, which has its own set of drawbacks. The magnitudes of "computing time" and "communication time" in biology are similar.

Communication time in computing can be substantially longer than calculation time for a variety of reasons. In other words, the received feedback returns state variables that were valid a long time ago. In biology, spiking is also a "look at me" signal: the feedback shall be addressed to that neuron, which caused the change[9]. Neurons receive data regarding "the influence of all other neurons, including myself," without taking spiking time into account (organised neuron polling in a software cycle). The beginning of a signal's validity is defined by receiving a spike; the "expiration period" is defined by "leaking." Spiking networks rely on their temporal dynamics.

Because of extensive waiting, some result/feedback events must be dropped in excessive systems in order to provide a false sense of improved performance.The logical reliance that feedback is computed based on the results of the neuron that receives the feedback, the computing system's

[8] Whether or not the message envelope contains a time stamp

[9] See the Hebbian learning: the neuron uses its inputs and output, exclusively.

physical implementation becomes time-dependent [8]. The neuron will receive the feedback later (even if at the same biological time, according to the time stamp they contain), therefore those messages will be at the end of the line. As a result, they're quite likely to be "dropped if the receiving process is congested over numerous delivery cycles" [46]. The feedback in the learning process in excessive systems involves outputs based on undefined inputs and feedbacks, and the computed and (perhaps correct) feedback may be overlooked.

An excellent "experimental proof" of the claims above is provided in [47]. With the words of that paper: "*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*". "*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.*" The intention to compute feedback faster, based maybe on undefined inputs, reaches the previous layer's neurons faster, but the speed has its price: "*As λ_s increases, the running time decreases, but so does performance.*" The addition of ANNs' spatio-temporal behaviour boosted the efficacy of video analysis greatly [48], Even in its most basic form, the use of segregated (i.e., not connected in the way proposed above and in [8]) time and space contributions to describe them.

Investigations in the time domain confirmed directly the role of time (mismatching): "*The CNN models are more sensitive to low-frequency channels than high-frequency channels*" [49]: When opposed to rapid changes, the feedback can follow the slow changes with less effort.

6. CONCLUSION

Machine learning arised from the intention of mimicking biological learning on technical implementations, having several million times higher transfer and processing speed. However, the speed difference sheds light to a common fallacy of their operation: their design does not consider that chained operations must consider also transfer time; furthermore that the computing operation and the transfer operation are blocking each other. Methods of learning and machine learning have mostly orthonal methods, because of their "implementation": biological operation natively considers transfer time, technical implementation assumes immediate interaction (neglects transfer time).

References

[1] Sterling P, Laughlin S. Principles of Neural Design, 1st ed. The MIT Press;2017.

[2] Floridi L. Information - A Very Short Introduction. Oxford University Press;2010.

[3] Stone JV. Principles of Neural Information Theory. Sebtel Press;2018.

[4] Vegh J. In The 2021 International Conference on Computational Science and Computational Intelligence; Foundations of Computer Science FCS'21. IEEE. 2021;FCS4404, in print.

[5] Vegh J. Revising the Classic Computing Paradigm and Its Technological Implementations. Trans Computers, 2021;14:1-13.

[6] Vegh J. Why do we need to Introduce Temporal Behavior in both Modern Science and Modern Computing, With an Outlook to Researching Modern Effects/Materials and Technologies. Global Journal of Computer Science and Technology: Hardware & Computation. 2020;20:13.

[7] J Vegh. In Proceedings of the 2020 International Conference on Computational Science and Computational Intelligence, CSCI'20, 2020. Las Vegas, Nevada, USA. IEEE Computer Society, 2020:CSCI2019.

[8] https://arxiv.org/abs/2006.01128

[9] Hameed R, Qadeer W, Wachs M, Azizi O, Solomatnikov A, et. al. Understanding Sources of Inefficiency in General-Purpose Chips. ACM SIGARCH Computer Architecture News.2010;38:37-47.

[10] http://arxiv.org/abs/2005.08942

[11] J. von Neumann.First Draft of a Report on The EDVAC. IEEE Annals of the History of Computing.1993;15, 27-75.

[12] http://www.imperial.ac.uk/ wl/teachlo-cal/cuscomp/notes/chapter2.pdf

[13] Johnston D, Sin Wu SM. Foundations of Cellular Neurophysiology. Massachusetts Institute of Technology.1995:710.

[14] Koch C. Biophysics of Computation, Oxford University Press, 1999.

[15] Antle MC, Silver R.Circadian Insights into Motivated Behavior. Trends Neurosci. 2015;28,145-151.

[16] Vegh J, Berki A J. In The 2021 International Conference on Computational Science and Computational Intelligence; Foundations of Computer Science FCS'21.IEEE. 2021; FCS4378, in print.

[17] https://www.preprints.org/manuscript/202103.0414/v1

[18] D. Hebb. The Organization of Behavior, Wiley: New York. 1949.1999;50:437.

[19] Lowel S, Singer W. Selection of Intrinsic Horizontal Connections in the Visual Cortex by Correlated Neuronal Activity. Science. 1992;255,209.

[20] https://www.biorxiv.org/content/10.1101/803577v1

[21] Jordan MI. Artificial Intelligence—The Revolution Hasn't Happened Yet. Harvard Data Science Review. 2019.

[22] Hutson M.Core progress in AI has stalled in some fields. Science. 2020;368:927. 146

[23] Losonczy A, Magee J.Integrative Properties of Radial Oblique Dendrites in Hippocampal CA1 Pyramidal Neurons. Neuron. 2006;50, 291-307.

[24] Furber S, Temple S.Neural Systems Engineering. J R Soc Interface. 2007;4:193.

[25] https://arxiv.org/abs/1705.06963

[26] Markovic D, Mizrahi A, Querlioz D, Grollier J. Physics for Neuromorphic Computing. Nature Reviews Physics. 2020;2;499-510.

[27] Kendall JD, Kumar S. The Building Blocks of a Brain-Inspired Computer. Appl Phys Rev. 2020;7:011305.

[28] Waser R. Advanced Electronics Materials and Novel Devices. Nanoelectronics and Information Technology, 3rdEdition.Wiley. 2012.

[29] https://arxiv.org/abs/2001.01266

[30] J. Vegh. How Amdahl's Law Limits the Performance of Large Artificial Neural Networks. Brain Informatics. 2019;6:4.

[31] Strukov DB, Snider GS, Stewart DR, Williams RS. The Missing Memristor Found. Nature. 2008;453:80-83.

[32] Wang C, Liang SJ, Wang CY, Yang ZZ, Ge Y, et al.Scalable Massively Parallel Computing Using Continuous-Time Data Representation in Nanoscale Crossbar Array. Nature Nanotechnology.2021.

[33] Mourelle LDM, Nedjah N, Pessanha FG. Reconfigurable and Adaptive Computing: Theory and Applications (CRC press, 2016), Chap. 5: Interprocess Communication via Crossbar for Shared Memory Systems-on-chip, 1st Edition, 2016.

[34] Abraham I. The Case for Rejecting the Memristor as a Fundamental Circuit Element. Scientific Reports. 2018;8:10972.

[35] Moradi S, Manohar R. The Impact of on-Chip Communication on Memory Technologies for Neuromorphic Systems. Journal of Physics D: Applied Physics. 2018;52;014003.

[36] Beggs JM, Plenz D. Neuronal Avalanches in Neocortical Circuits. Journal of Neuroscience. 2003;23, 11167.

[37] Amdahl GM. Validity of the Single Processor Approach to Achieving Large Scale Computing Capabilities. In AFIPS Conference Proceedings. 1967; 30:483-485.

[38] https://arxiv.org/abs/2006.00532

[39] Goychuk I, Hanggi P, Vega JL, Miret-Artes S.Non-Markovian stochastic resonance: Three-state model of ion channel gating. Phys. Rev. 2005; E 71, 061906.

[40] Linder B, Garcia-Ojalvo J, Neiman A, Schimansky-Geier L. Effects of Noise in Excitable Systems. Physics reports 2004;392,321.

[41] http://www.accellera.org/downloads/standards/systemc

[42] Tsafrir D.The Context-Switch Overhead Inflicted by Hardware Interrupts (and the enigma of do-nothing loops). In Proceedings of the 2007 Workshop on Experimental Computer Science,ACM, New York, NY, USA, 2007, ExpCS '07. 2007:4.

[43] David FM, Carlyle JC, Campbell RH. In Proceedings of the 2007 Workshop on Experimental Computer Science. (ACM, New York, NY, USA, 2007), ExpCS '07.

[44] Williams S, Waterman A, Patterson D.Roofline: An Insightful Visual Performance Model for Multi-Core Architectures. Commun. ACM. 2009;52:65.

[45] Keuper J, Pfreundt FJ.Distributed Training of Deep Neural Networks: Theoretical and Practical Limits of Parallel Scalability. In 2nd Workshop on Machine Learning in HPC Environments, MLHPC. IEEE. 2016:1469-1476. 147

[46] van Albada SJ, Rowley AG, Senk J, Hopkins M, Schmidt M, et al. 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.

[47] https://arxiv.org/pdf/1511.06297

[48] Xie S, Sun C, Huang J, Z Tu, Murphy K. In Computer Vision – ECCV 2018, ed. by V. Ferrari, M. Hebert, C. Sminchisescu, Y. Weiss .Springer International Publishing, Cham. 2018: 318–335.

[49] https://arxiv.org/abs/2002.12416