Miért ellenzi Neumann János a gépi tanulás gyorsítását?

Why does von Neumann oppose speeding up machine learning?

VÉGH János MTA doktora, egyetemi tanár

Kalimános BT, Debrecen; Vegh.Janos@gmail.com

Abstract

Von Neumann wanted to provide a „fast" and error-free way of solving numerical problems and the operation of the brain inspired his famous paradigm. One of his great ideas was to store the computer's knowledge in the memory of the computer. How became the data-controlled analog brain the model of the instruction-controlled digital computer? Why loading a ready-made program into its memory is the only way to teach the computer? Why is the software-implemented training by machine learning so painfully slow?

Keywords: brain operation; computer operation; data-controlled operation; instruction-controlled operation; accelerating computing

Kivonat

Neumann János numerikus problémák „gyors" megoldására keresett megoldást és nevezetes számítási paradigmáját az agy működése ihlette. Az ő egyik nagy ötlete volt a számítógép tudásának tárolása a memóriában. De hogyan lett az adat-vezérelt analóg működésű biológiai agy az utasítás-vezérelt digitális elektronikus számítógép processzor modellje? Miért csak úgy tanítható a számítógép, hogy beletöltjük a kész tudnivalót? Miért olyan kínosan lassú a gépi tanulás alapú betanítás?

Kulcsszavak: agy működés; számítógép működés; adat-vezérelt működés; utasítás-vezérelt működés; számítás gyorsítás

1 BEVEZETÉS

Manapság számítástudományból megtanuljuk és természetesnek vesszük, hogy digitális számítógépeink programjait azok memóriája tárolja; az ott tárolt adatokat a számítógép központi egysége egy órajel hatására egy utasítás számlálóban tárolt cím alapján megfelelő sorrendben egyesével előveszi, dekódolja (utasításként értelmezi) aztán az így megadott elemi utasítást végrehajtja, majd ugyanezt teszi a következővel, amíg leállító utasítást nem talál. Azt is megemlítjük, hogy Neumann Jánost az agy működése inspirálta nevezetes paradigmájának megalkotására. Az agyról azonban már tudjuk, hogy analóg mennyiséggel (az idővel) számol, nincs központi órajele és utasítás számlálója, külön memória egysége; továbbá adat-vezérelten működik és kiválóan tanul. De akkor hogyan lett az agy jó modell a számítógéphez és miért tanul olyan nehézkesen?

2 A SZÁMÍTÁS MODELLEZÉSE

Amikor valamilyen számítást kell végeznünk, akkor bemenő adatainkhoz műveleteket rendelünk hozzá, akár kézi számolással, akár elektronikusan. Ez a hozzárendelés változatos formákban történhet, a számítást végző személynek szóban adott utasítástól kezdve egy elektronikus egység elindításáig, a számító egységeknek a telefonközpontokban szokásos módon huzalokkal történő összekötésétől egy program mutatónak a tárolt program első utasítására állításáig. A gyakorlati számítások elemi műveletekből állnak és eredményeikkel egymásnak bemenő adatokat szolgáltatnak (ezeket hívjuk láncolt műveleteknek).

Az utasítás is megkeresheti operandusát (utasítás-vezérelt mód), vagy az adat is a szükséges műveletet (adat-vezérelt mód), de mindenképpen szükség van arra, hogy valamilyen módon megadjuk összerendelésüket és külön jelezzük, mikor lehet elkezdeni a műveletet és mikor lehet felhasználni az eredményt; a láncoláshoz pedig meg kell határozni a műveletek végrehajtási sorrendjét, az adat jel-ábrázolását, valamint továbbításának módját, útvonalát és időzítését. A feldolgozás és az adat szállítás egymástól nem választhatók el, azokat együtt kell tárgyalnunk.

Mind adat-, mind utasítás-vezérlés esetén a láncolt műveleti egységek első elemére adjuk a bemenő adatot, ami működteti a számítási láncot, és a lánc utolsó eleme szolgáltatja az eredményt. Bonyolultabb és

hosszasabb számítás végzéséhez vagy sok elemi számító objektumot kell biztosítani vagy az elemi objektumokat – megfelelő szervezéssel – többször is felhasználni. Az első változat nagy alkatrész pazarlással jár, nagyfokú párhuzamosságot biztosít, de működtetése szinte semmi szervezést nem igényel; a második magasabb rendű szervező egységet igényel, minimális párhuzamosíthatóságot engedélyez, de akár korlátlanul hosszú működést tesz lehetővé.

A feldolgozási művelet elkezdési és befejezési eseményének időpontjai egy ún. idő-ablakot jelölnek ki, amelyek között a művelet végzése történik; az ablak szélessége az *adatfeldolgozás* ideje. Az időablakot a biológia és a technika eltérően jelöli ki. A biológiai feldolgozás esemény vezérelt. Az első beérkező neurális impulzus (spike) nyitja meg az időablakot és a számító egység (a neuron) zárja be azt (önmagát szinkronizálja) amikor elegendő mennyiségű töltés beérkezett a membránra és annak feszültsége így elérte a küszöb potenciált. A számításban az időablakon belül bármikor megérkező összes spike részt vesz. A neuron a két esemény között beérkezett töltés speciális integrálásával méri az időt. A számítás eredménye maga az időablak szélessége. A működés nem matematikai függvényt valósít meg. Bonyolultabb feladatokat csak több neuron együttműködésével lehet megoldani. A technikában az eseményeket a számítógép processzor egy központi órajel használatával imitálja. Az órajel egyik éle határozza meg a művelet kezdetét, egy másik pedig a végét. Az idő-ablak kezdetekor a bemenő adatok már az egység bemenetén elérhetők és az eredmény legkésőbb az időablak végéig megjelenik az egység kimenetén. A tervező felelőssége, hogy a műveletek elkezdésére utasító jel az órajelhez képest megfelelő késleltetéssel érkezzen, mivel mindkét jel terjedési sebessége véges; továbbá hogy a művelet az időablak végéig befejeződjön. Ez azt is jelenti, hogy *valamennyi időzítésnek és valamennyi művelet hosszúságának ismertnek kell lenni tervezéskor*. Minden művelet időablaka egyforma hosszúságú, az esetleg az időablak vége előtt befejeződő műveletek esetén az idő arányos része kihasználatlanul marad és teljesítmény vesztéshez vezet. Az elemi műveletek egy matematikai függvényt valósítanak meg, az elemi egységek önálló műveletekre is használhatók.

Egy másik időablak, ami az eredmény előállításától a lánc következő számítási elemének eléréséig tart, az *adatátvitel* ideje. A biológiában az adatátvitel idejét a neuronok közötti axonok hosszúsága és azokon a jelterjedési sebesség (conduction velocity) meghatározza (és a msec tartományba esik); de minden összekötés közvetlen. *Az átvitt jel indítja a számítást*. A technikában változatos a megvalósítása, a közvetlen és párhuzamos galvanikus összekötéstől az ún. buszon keresztüli szekvenciális arbitrált jelátvitelen át a hálózati átviteli egységek használatáig; ami módokhoz tartozó időablak gyökeresen eltérő szélességű (a psec tartománytól a msec tartományig). *Az átvitt jelnek alkalmazkodnia kell a központi órajel időzítéséhez*. A klasszikus számítási paradigma az elektroncsövek időzítési viszonyait (millisec feldolgozási és mikrosec átviteli idö) alapul véve elhanyagolta az adatátvitel idejét a feldolgozási időhöz képest, így az ma már nemcsak a biológiai, hanem a modern technológiával megvalósított technikai számításokat sem tudja pontosan leírni. Maga a számítási modell azonban időt állónak bizonyult és az általános számítási paradigma (a korlátozott kölcsönhatási sebesség figyelembe vételével, a klasszikus közelítésben használt végtelen nagy sebesség helyett) jól leírja mind a biológiai mind a technikai számítást [1]. Nem meglepő módon, a láncolt működés természetes módon leírható a speciális relativitás elméletéből ismert Minkowski-matematika használatával.

A két időablak között nem lehet átfedés. Másképpen megfogalmazva, a két tevékenység *kölcsönösen akadályozza egymást*: a számítást nem lehet elkezdeni, amíg az idő-ablakot indító jel meg nem érkezett, továbbá az eredményt nem lehet elszállítani amíg az ablak be nem zárul. A távolabbról érkező vagy hosszadalmasan kiszámítható adatokra várakozni kell, így azok feltartják a számítást és a feladat végrehajtás számítási hatásfoka csökken. Ez a feltétel egyszerű láncolt műveletek esetén jól áttekintető és szabályozható, az alkalmi (és főként több lépcsőben végrehajtott) párhuzamosítások és hurkok (visszacsatolás, back propagation, gyorsító megoldások) azonban nagy odafigyelést igényelnek és számos hiba forrása lehetnek, mivel ezekben az esetekben *az adatfeldolgozás kezdetének és befejeződésének ideje nem ismert a tervezés idején, a központi órajel használata viszont számukra is kötelező*.

3 UTASÍTÁS- ÉS ADAT-VEZÉRELT MŰKÖDÉS

Adat-vezérelt működés esetén minden adatnak saját, kizárólagosan használt útvonala van, ami nagy fokú párhuzamosságot tesz lehetővé. Ennek ára azonban hogy az egyes számító egységek (amik ezúttal *az utasítások*) "egyszer használatosak": ugyanaz a számítási egység mindig ugyanazt a műveletet hajtja végre a hozzá beérkező adatokkal, tehát nem lehet több célra használni. Ezért szükséges az agy számára száz milliárdnyi neuron (azaz lényegében utasítás), mert gyakorlatilag minden elemi műveletnek másik neuront kell használni (még ha van is "újrafelhasználást" biztosító magasabb szintű szervező egység) és ezért tűnik úgy, hogy bármelyik pillanatban csak neuronjaink töredékét használjuk. A modern (főként a sok-magos) processzorok ugyan nagyságrendileg hasonló számú kaput tartalmaznak mint az agy, de a kapuk túlnyomó

többsége csupán az egyetlen utasításfolyam számítási sebességének növelésére szolgál és a ciklikus-szekvenciális utasítás-vezérelt működési mód nem lép fel nagy huzalozási követelménnyel. Ugyanilyen számú kapuból lehetne teljesen adat-vezérelt működésű végrehajtó egységet is építeni, de az célberendezés lenne: minden adatot ugyanolyan módon dolgozna fel és nagyságrendekkel több huzalozást igényelne. A technika számára megoldhatatlan az agy dinamikus működéséhez elengedhetetlen biológiai mechanizmusok utánzása, főként az idő figyelmen kívül hagyása miatt, valamint a szükséges huzalozás kivitelezése is. Emiatt *kompromisszumot kell kötnünk a használható alkatrészek száma és a közöttük levő huzalozás mennyisége, valamint a processzor felépítésének bonyolultsága és működési sebessége között*. Emiatt jelentős különbség van a technikai és biológiai processzor magasabb szintű viselkedésében is.

A technikai számítás *utasítás-vezérelt.* Lényegében egy igen bonyolult ciklikus-szekvenciális processzort használunk, amelyik a bemenetére érkező (a memóriából elővett) *adatot* utasításként értelmezi és annak dekódolása után az utasításnak megfelelően állítja be belső állapotait és adatútjait. Ezután a processzor működését már az utasításban megadott adat vezérli: az elemi számítás végéig a processzor adat-vezérelten működik és az adatok végig siklanak az előzőleg kijelölt adatúton. A processzor egyszerre csak egy utasítást tud végrehajtani, ami jelentősen korlátozza a párhuzamosítást (bár a tényleges megvalósítások rejtetten lehetővé tesznek alkalmi párhuzamosításokat). Az eredmény a kimeneti szekcióba kerül és a processzor ismét utasítás-vezérelt módban dolgozik tovább. Az adat-vezérelt működés során az adatok (pl. negatív-e az eredmény) módosíthatják az utasítás számlálót: a technikai processzor lényegében a kétféle vezérlési mód kombinációját használja. Sőt, a programmegszakítás lényegében egyfajta *esemény-vezérelt* működést valósít meg. *Az átmeneti utasítás-vezérelt mód feltétlenül szükséges, hogy a processzor programozható legyen*.

A dekódolt utasításnak hivatkozni kell tudni a feldolgozandó adatra. A művelet végzéséhez szükséges adatokat közvetlenül (vezetékkel) az utasítás számára elérhetővé tenni egyrészt megoldhatatlanul sűrű vezetékezést jelentene, másrészt nem biztosítana lehetőséget hogy az utasítás különböző adatokat használjon (pl. egy vektor különböző elemeit) a végrehajtás során. Ezért adat hivatkozásként egy tároló-beli címet adunk meg, ami tárolót egy speciális vezeték rendszeren (a buszon) át lehet elérni. *A tárolót egyidejűleg csak egy számítási egység használhatja, ami a párhuzamos működés további jelentős korlátozását jelenti és a buszon át leszállított adat érkezési ideje a tervezéskor nem ismert*. Az újra-konfigurálható rendszerek éppen attól hatékonyak, hogy előre (és nem utasítás dekódolás után) beállított adatutakat használnak és lehetővé teszik az adatutak párhuzamos használatát. Lehetne azonban egy közbülső (configware) réteget is használni a hardver és szoftver rétegek között.

A biológiai számítás *adat- és esemény-vezérelt.* A számítás elemi egysége a neuron, aminek az absztrakt megfelelője egy feszültség-vezérelt kondenzátor [2]. Neumann korában a neuront a McCulloch-Pitt modell alapján, lényegében digitális számító elemként írták le, ezért nem okozott gondot technikai megfelelőjének leírása digitális egységként. Ma már tudjuk, hogy a neuron viselkedése matematikailag igen nehezen írható le és bonyolult elektronikával modellezhető. Nagy számú (több ezer) bemenete van, és egy-egy számítási műveletben csupán tucatnyi bemenet súlyozott és kapuzott jelét használja. A jel továbbítás és feldolgozás során erősen figyelembe kell venni az alacsony vezetési sebességet, azaz a klasszikus számítási paradigma egyáltalán nem használható. Mind az argumentumok, mind az eredmény idő jellegű; a jelfeldolgozás analóg módon történik és események vezérlik. A töltés a kondenzátorból folyamatosan "szivárog", ezért a bemeneteire beérkező töltések érkezési ideje nagyon sokat számít. Az eredményhez az idő-ablak nyitva tartási idején belül beérkező töltések mind hozzájárulnak, növelik vagy csökkentik a membrán potenciált. A neuronnak (különböző biofizikai mechanizmusokkal megvalósított) memória-szerűen működő állapotai vannak; a számítás újraindítható és eredménye az aktuális neurális környezettől, valamint a neuron saját érzékenységétől is függ. Az összes tényező ismeretében ugyan leírható a működés, de a nagy számú befolyásoló tényező és főként az időfüggés sokrétűsége miatt a működés sokkal kevésbé determinisztikus, mint a technikai számításé. *Az időablakban a számítás során figyelembe vett bemenetek, azok súlya és beérkezési ideje műveletről műveletre változhatnak, ami egyrészt nehezen meghatározhatóvá teszi a művelet eredményét, másrészt – a tanulási mechanizmussal együtt – lehetőséget nyújt a magasabb szintű idegi működés számára a redundancia, rehabilitáció, intuíció, asszociáció, stb. fogalmak megvalósítására*.

4 IMITÁLT NEURÁLIS SZÁMÍTÁSOK

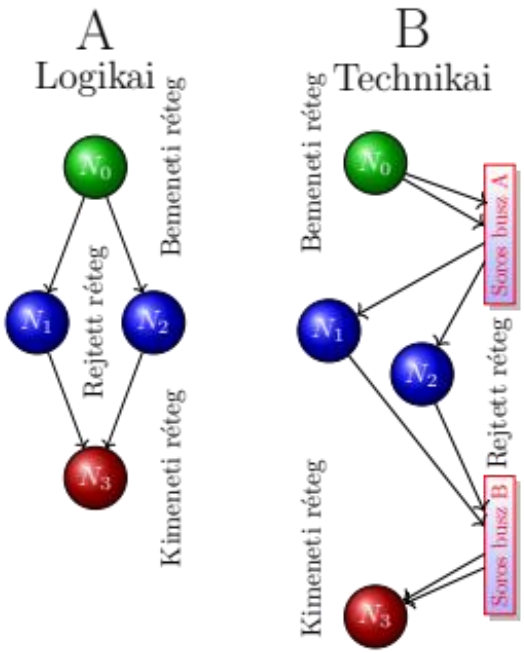
A biológiai számítás technikai imitálásához neurális hálózatot használunk, ahol a „neuronok" egymással össze vannak kötve és egymás számára adatokat továbbítanak, a [3] 1.c ábrájának megfelelő módon. Az alsó neuron réteg neuronjaira adjuk a bemenő adatokat, amelyekből valamilyen, a neuronok szinaptikus érzékenységét imitáló, egyedi súlyokkal számítjuk ki a következő réteg bemenő adatait, amit a réteg neuronjai

hasonlóan továbbítanak a következő réteg számára. Természetesen, a következő réteg csak akkor tudja a számításokat elvégezni, ha az előző réteg már teljesen befejezte a számítást. *A két réteg idő-ablakai nem fednek át, és közöttük van még a szállítási idő-ablak is*. A csak CPU-t használó (és egyéb szekvenciális) megvalósítás esetén ez a feltétel biztosított. Szokásos azonban a réteg számításait (gyorsítás céljából) Graphic Processing Unit (GPU) használatával végezni.

A GPU részben párhuzamosítja a számítást, továbbá adatszállítást is végez, ezért ilyenkor az időablakok értelmezése kétségessé válik. A GPU bemeneteire oda kell szállítani a bemenő adatokat és a számítás elvégzése után el kell szállítani az eredményt, azaz az adatok érkezése/elszállítása/feldolgozása fokozatos és részben egyidejű. A műveleti idő akár egyetlen órajelre is rövidülhet, ha valamennyi adat beérkezését megvárjuk, de akkor a szállítási idő nagyon hosszú, mivel az egyes adatok oda- és elszállítása is egy-egy órajelet igényel (a memória rekeszeket egyesével kell írni/olvasni, bármilyen rejtett módon történik is). Lehet azonban a beérkező adattal azonnal műveletet végezni: a GPU bonyolult stalling/interleaving mechanizmusa biztosítja, hogy idővel valamennyi adat feldolgozódik. Amennyiben az adatok szállítása buszon keresztül történik (ez a jellemző), *a busz arbitráció jelentős bizonytalanságot hoz be az időablak tervezésébe,* mivel *megváltoztatja az argumentumok szállítási idejét, sőt akár sorrendjét is.* Az idő-ablak szélessége tervezéskor nem ismert, de azt rögzíteni kell. Az idő-ablak szélessége nem lehet túl nagy, mert akkor romlik a műveleti sebessége. Ha viszont kicsi tartalékkal tervezik és futtatáskor az arbitráció jelentős mértékű késést okoz (a buszt a GPU adatforgalmától idegen forgalom is terheli), az adatok egy része nem érkezik meg az idő-ablakon belül. Ennek veszélye bemenő és kimenő adatok esetén is fennáll. A GPU memória-rekeszekkel dolgozik és attól függetlenül elvégzi a műveletet, hogy azok az általunk vélt argumentumot vagy eredményt tartalmazzák-e. Ha az arbitráció miatt a busz nem a tervezett időben szállít (az adat az idő ablakon túl ér oda), akkor a GPU „idegen" adattal dolgozik. A jelenség észrevehetetlen a video alkalmazások esetén, ahol egy GPU egy irányba továbbít adatot egy buszon keresztül, továbbá az esetleges hiba is csak milliomod rész pixelt érint 10 msec nagyságú időre. A mély rétegű tanulás esetén viszont számos GPU adatai keverednek egyetlen buszon és az arbitráció miatt egymást késleltetik; ráadásul az eredmények számítási céllal használódnak és a súlyokban valamennyire tartósan megmaradnak. A GPU-ban egyszerre folyik a memória töltése és ürítése, valamint a művelet elvégzése. A szállítás és feldolgozás időablaka (a forgalomtól függően) átfedhet, a számítási paradigma elvárt követelményét nem teljesíti.

A fenti működés (távolról nézve) hasonlít a valódi neurális működésre, lényegében adat-vezérelt. A valódi neuronok viszont önállóan tanulnak, a bemenetükre érkező adatok idő-viszonyai alapján szelektíven állítják saját szinaptikus érzékenységüket. A mesterséges hálózatok tagjait azonban tanítani kell, például a back-propagation algoritmus felhasználásával, lásd [3] 1.c ábráját. Ami azt jelenti, hogy a működést felügyelő program (utasítás-vezérelt módon) számítást végez és az eredményeket az előzőhöz képest fordított irányban küldi át a hálózaton (a módszernek nincs biológiai relevanciája). Ami természetesen csak akkor tehető meg, ha az „előre" irány számítása már befejeződött (tehát a két időablak nem fed át és közöttük van a szállítás időablaka is), továbá természetesen az előzőhöz hasonló korlátozás érvényes a rétegek közötti számolásra is. Itt újabb lehetőség kínálkozik a felelőtlen gyorsításra: ha a két rétegben független GPUk állnak rendelkezésre a számításhoz, az előre és hátra számításokat egyszerre végezhetik. Azonban a súlyok megváltoztatása a „hátra" irányú számítás által az „előre" számítás egy részére is hatni fog: felülírjuk a súlyok egy részét, lényegében kiszámíthatatlan módon. Az eljárás következtében igen gyors lesz a számítás, de valójában nagyrészt nem a matematikailag elvárt eredményt kapjuk, mivel a kétféle irányú számítás idő-ablakai átfednek. Ez a gyorsítás többszörös menyiségű művelet elvégzését jelenti (azaz jelentős többlet energia fogyasztást is), valójában azonban jelentősen lassítja a matematikai konvergenciát.

Rekurrens működtetés esetén tovább rontjuk a helyzetet. Mint [3] ki is mondja, *a mély rétegű tanulás rekurzív módja még csekély számú réteg esetén is igen nagy (elméletileg végtelen) számú, azonos súlyokat használó, lineáris rétegnek felel meg, lásd [3] 5. ábrája.* A korrekt matematikai működéshez faktoriálisan növekvő számú számítást kellene elvégezni: egyetlen lépés lényegében egy végtelenszer ismétlődő számítási sort indít el (és minden ismétlés egy újabb végtelenszer ismétlődő számítási sort indít el), azaz szigorúan vett értelemben egy elindított számítás soha nem fejeződne be. Az egy lépéshez rendelhető időablaknak végtelen szélesnek kellene lenni, azaz az észszerű működési sebesség érdekében csonkolni kell a számítást.



1. ábra
A neurális átvitel logikai és biológiai, valamint technikai módja

Már maga a matematikai módszer is magyarázza, miért olyan lassú a betanítás: a virtuálisan rögzített paramétereket még a helyes tanulási módszer is csak kínosan lassan tudja változtatni. Ironikus módon, azért tapasztalható tanulás az azonos súlyok ellenére, mert a busz működése által okozott késleltetések keverik a rögzített súlyokkal rendelkező virtuális és az adjusztált súlyokkal rendelkező valódi rétegekben levő neuronok üzeneteit. Három lépést előre, két lépést hátra. Ez azt jelenti, hogy tanulás szempontjából a rekurrens működtetés a lehető legrosszabb megoldás: a "tanítás" eredménye annál lassabban jelenik meg, minél nagyobb a rendszer, mivel a virtuális rétegek nagyon is valós kommunikációs igénnyel lépnek fel. A működés csupán azért nem szinguláris, mert a véges végrehajtási sebesség (plusz adatszállítási idő, tipikusan a buszon át) miatt nem végtelenül kis idő alatt lép fel az új számítás igénye. Nincs nem-átfedő adatszállítási idő, azaz bármelyik pillanatban esetleges, hogy mit szállítunk el eredményként. Sajnos, a matematika újra felfedezi az Achilleusz és a teknősbéka paradoxonját, ahol végtelen sok művelettel (végtelen sor) próbálják előállítani az egyszerű szorzás/osztás művelettel is kiszámítható eredményt. Amit aztán (a számítógép elméleti működésének meg nem értése miatt) az elektronikát gyártó cégek pontosan ilyen formában valósítanak meg, a végsőkig fokozva a számítás gazdaságtalanságát [4]. Lényegében ez okozza, hogy „*a gépi tanulás abba az irányba tart, hogy felélje az összes előállítható energiát; olyan modellt használ, ami költséges, alacsony hatékonyságú, és fenntarthatatlan*" [5]. Ez a gyakorlatban azt jelenti, hogy pl. a Google PaLM rendszerének betanítása két hónapon át összesen 3.4 gigawatt-óra energiát vett igénybe és több tíz tonna szén-dioxid-kibocsátásával járt [6]. A felhasznált energiának csupán egyik fele szükséges a tényleges használatra, a másik fele a számítógép működési elvéből következő "tétlenség" és az infrastruktúra fenntartására szükséges [7].

5 AZ ÁTVITEL TECHNIKAI MEGVALÓSÍTÁSA

A neurális hálózatok elvi rajza a biológiai összekötéssel megegyező, amint a 3. ábra bal oldala mutatja. A technikailag megvalósított adatszállítás viszont a buszon keresztül történik, annak összes hátrányával együtt. Azaz, *egy adatot először a busznak kell továbbítani, és a másik node-nak a busztól kell megkapnia*. Az elvi működés (számítástudomány) megegyezik az ábra bal oldalán mutatottal, a valódi működés (számítógéptudomány) jelentősen eltér, lásd az ábra jobb oldalát, komoly tulajdonságbeli eltéréseket vonva maga után. A buszt csak kizárólagos joggal lehet használni, ezért minden adatot küldeni akaró neuronnak "versenyeznie" kell a busz használatáért (arbitráció), minden egyes küldés esetén. Az arbitráció az egyes átviteli időkhöz egy ál-véletlen átviteli időt ad, ami a rendszer méretétől függően akár dominánssá is válhat. A

biológia (és a matematika) esetében párhuzamosan továbbított adatok a technikai megvalósításban a kizárólagos használat miatt szerializálódnak. A buszon egyidejűleg csak egy adat továbbítódhat. Ha a buszra sok neuron csatlakozik és egyidejűleg akarnak adatot továbbítani, akkor a busz sávszélessége jelentősen korlátozza az adatok átvitelének idejét. *A busz használata határozatlanná teszi a számítás idő-ablakát*; minél nagyobb a rendszer, annál nagyobb mértékben.

Szekvenciális busz használata eleve *megszünteti az adatok egyidejű érkezését*, de az igazán kedvezőtlen hatása hogy *az üzenet továbbítás sorrendje nem meghatározott*. A bonyolultabb (sok neuront és több mélyebb réteget tartalmazó) hálózatok esetén előfordulhat, hogy egy réteg valamely csomópontja később kapja meg a magasabban fekvő node-tól az adatot, mint a többi. Ha az egyes rétegek (amint szokásos) ugyanazt a nagy sebességű de egyetlen buszt használják, a különböző rétegek adatforgalmai akadályozhatják egymást, még ha az egyes rétegek számításait külön-külön GPU végzi is. Ennek kiküszöbölésére vagy meg kell várni valamennyi neurális input beérkezését (aszinkron számítás) vagy enélkül folytani a számítást, bízva abban, hogy az abban a memóriában levő tartalom nem tér el jelentősen a helyes értéktől, vagy pedig az újabb inputok beérkezésekor a mélyebb réteg neuronjai (és az általa produkált adatok elküldése után a következők) újból elvégzik a tőlük elvárt számítást (megismétlik az egész idő-ablakot), amitől exponenciálisan megnő a kapuk (szükségtelen) billenéseinek száma, ezáltal rendkívüli mértékben megnő a neurális hálózat teljesítmény felvétele és tovább csökken a hasznosított számítógépes műveletek aránya (a hatásfok). “A népszerű közös buszt használni kommunikációs közegként ma már nem fogadható el ötletként” [8].

6 ÖSSZEFOGLALÁS

A számítógép elméleti működésének modelljét az agy inspirálta. A számítógép technikai megvalósítása azonban számos kompromisszum kötését tette szükségessé, amelyek pontos betartása lassúvá teszi annak működését. Az agy neurobiológiai mechanizmusainak és konnektivitásának reprodukálása technológiailag nem lehetséges. A biológiai agy egyik karakterisztikus funkcióját, a tanulást, a technikai agy hardvere egyáltalán nem tudja megvalósítani: egy tevékenység megtanulása csak a kész utasítássorozatnak a számítógép memóriába töltésével lehetséges. A biológiai agy tanulásának imitálása teljesen a számítógép szoftverre marad. A tanulás matematikája biológiailag nem releváns módszereket használ. Továbbá az agy és a számítógép eltérő működési elvei egyúttal eltérő tulajdonságokat, például tanulási képességet is jelentenek. A szoftveres tanulás kétségbe ejtően lassú és alacsony hatékonyságú.

A számítógép teljes története a működés mind gyorsabbá tételéről szól. Neumann János alapvetésében bemutatta, hogy a számítási műveletek során az argumentumok szállítása és a művelet elvégzése összefüggő és egymást akadályozó tevékenységek. A matematikailag korrekt működés elérésére gondoskodni kell arról, hogy a két művelet időablakai ne fedjék át egymást. Ennek helyes megvalósítása azonban lassúvá teszi a szekvenciális processzoron futó program működését. A gyorsítási ötletek a szekvenciális számítás helyenkénti legalább részletes párhuzamosításáról, vagy éppen a számítási sorrend megváltoztatásáról szólnak. A szekvenciális lineáris működést feltételező paradigma alapján párhuzamosításokat és hurkokat építeni (akár matematikai akár elektronikus megvalósítással), elméleti megalapozás és a technikai működés ismerete nélkül, felelőtlenség. A számítógép időbeli működésének figyelmen kívül hagyása már eddig is tragikusan nagy energia fogyasztást eredményezett (a tervezett szuperszámítógépek esetén már arról folynak megbeszélések, hogy hány dedikált erőművet kell a számítógép működtetésére szánni), *a tanulás számítógépes megvalósítása a működés megértése nélkül fenntarthatatlanná teszi számítógép használatot*. Emellett a számítási idő-ablak csonkítása miatt az eredmény helyessége is sérül.

IRODALMI HIVATKOZÁSOK

[1] Végh J.: Revising the Classic Computing Paradigm and Its Technological Implementations, Informatics, 4/8(2021)171. doi = 10.3390/informatics8040071. https://www.mdpi.com/2227-9709/8/4/71

[2] Végh J, Berki Á. J. On the Role of Speed in Technological and Biological Information Transfer for Computations, Acta Biotheoretica 70/4(2022) 26,doi = 10.1007/s10441-022-09450-6

[3] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. Nature, 521:436-444, 2015.

[4] A. Mehonic and A.J. Kenyon. Brain-inspired computing needs a master plan. Nature, 604:255-260, 2022.

[5] AI-power-consumption-exploding, 2022. https://semiengineering.com/ai-power-consumption-exploding

[6] In AI, is bigger always better? Nature 615(2023) 202 https://www.nature.com/articles/d41586-023-00641-w

[7] Luccioni A. S., Viguier S. Anne-Laure Ligozat A-L, Estimating the Carbon Footprint of BLOOM, a 176B Parameter Language Model, 2022, https://arxiv.org/pdf/2211.02001.pdf

[8] Macedo Mourelle L., Nedjah N, and Pessanha F. G. Reconfigurable and Adaptive Computing: Theory and Applications, chapter 5: Interprocess Communication via Crossbar for Shared Memory Systems-on-chip. CRC press, 2016.