Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Automatizálási és Alkalmazott Informatikai Tanszék

Gépi tanulás alkalmazásának vizsgálata a kvantuminformatikában

Tudományos Diákköri Konferencia dolgozat

Készítette
Marosits Ádám

Konzulens
Dr. Szegletes Luca

2022. november 1.

Tartalomjegyzék

Irodalomjegyzék 51

Kivonat

A gépi tanulás manapság az egyik legkutatottabb tématerületnek számít, ami nélkülözhetetlen szerepet tölt be a képfeldolgozás, az önvezető autók, internetes keresések, spam szűrés, az analóg és a digitális jelfeldolgozás esetében is. Ezen felül egyre több ipari környezetben is alkalmazzák annak érdekében, hogy minél optimálisabban oldhassanak meg olyan feladatokat, melyekre egzakt megoldást nem, vagy csak nagy számítási kapacitás felhasználásával tudnánk adni.

Egy másik hasonlóan fontos, de az iparban még kevésbé jelen lévő környezet a kvantuminformatika, ahol az állapotok szuperpozícióban lehetnek, ezzel növelve a számítási kapacitást. Jelenleg több cég (pl.: IBM, Google, D-Wave, stb.) foglalkozik kvantumos elvű feldolgozás kutatásaival, a két tématerület összekapcsolása azonban rendkívül újszerűnek számít.

Jelen dokumentumban megvizsgálom különböző problémák megoldásának lehetőségét mind klasszikus-, mind kvantuminformatikával. Ezek a feladatok általában olyan számítási kapacitást igényelnek, melyet a legtöbb esetben nem tudunk biztosítani, ezért relevánssá válik a kvantuminformatika alkalmazásának vizsgálata. A kvantuminformatika és a gépi tanulás elméleti összefoglalása után elsőként röviden megvizsgálom a QAOA (quantum approximate optimization algorithm, kvantumos becslés optimalizációs algoritmus) algoritmust, majd ezután rátérek a neurális hálózatok kvantumos implementációjának vizsgálatára. Dolgozatom célja, hogy bemutassa, miként lehet-e két tématerületet összekapcsolni, a gépi tanulásban ismert metódosuk előnyeit ötvözni a kvantumfeldolgozás gyorsaságával. A munkám során feltárom a kvantuminformatika jelenlegi helyzetét, alkalmazásának lehetőségét.

Abstract

Nowadays machine learning is the most researched topic, that plays an essential role in the case of image processing, self-driving cars, web searching, spam filtering, and analog and digital signal processing. Furthermore, it is used in industrial environments in order to be able to solve problems, that can not be solved or only can be solved using a large computation capacity exactly.

Another important topic, although it is not present in the industrial environment yet, is quantum informatics, where the quantum states can be in superposition, thereby increasing the computational capacity. Several companies (for example IBM, Google, D-Wave, etc.) are engaged in research on quantum processing.

My paper evaluates solutions to different problems with classical and quantum computing. Usually, these problems require a large computation capacity, which is not provided in the computation systems, so the application of quantum computing will be relevant. After presenting the theoretical background of quantum computing and machine learning, the QAOA (quantum approximate optimization algorithm) algorithm is evaluated, then my work introduces the implementation of neural networks with quantum computing.

My paper's objective is to investigate the connection between the two topics using the quantum enhanced features in machine learning. The document expands the status and the possible applications of quantum computing.

1. fejezet

Bevezetés

Manapság egyre nagyobb igény mutatkozik a gépi tanulás használatára, ugyanis jelenleg mind ipari, mind kutatási területen sok olyan problémát ismerünk, melyekre nincs egzakt algoritmus, vagy nem képes olyan pontos megoldást adni, mint a gépi tanulás, melynek lényege, hogy a tanulási folyamatban a tanító adatokból olyan paraméterhalmazt sajátítsanak el, amivel később az adott problémát gyorsan és pontosan meg tudják oldani. Ez lényegében egy emberszerű gondolkodást feltételez, azaz a tapasztalatok alapján hoz döntést a jövőben. Ennek a technológiának jelenleg a legnagyobb relevanciája a képfeldolgozás és jelfeldolgozás esetében van, előbbiben klasszifikálásra, objektumdetektálásra és szemantikus szegmentációra, míg utóbbiban zajszűrésre, mintázat keresésre, esetleg anomália detekcióra lehet rendkívül jól használni. Megjelenik az önvezető autózásban, webkeresések optimalizálásában, illetve a marketing célú alkalmazásban egyaránt, azonban érdemes megemlíteni, hogy a tanítás folyamata rendkívül idő- és erőforrásigényes. További optimalizációs problémák alkalmazása felmerülhet útvonaltervezésnél, hálózati optimalizáció, vagy akár az 5G hálózatok esetében ismert antennatömbös megoldások (MIMO) esetében is. Ezek a problémák NP-nehéz problémák, emiatt nem tudunk rá egyszerűen egzakt megoldást adni, azokat csak közelítő eljárásokkal oldhatjuk meg.

Az optimalizációs problémák modellezhetők energiafüggvénnyel, melyek minimalizálására többféle módszert is ismerünk, klasszikus heurisztikákat, vagy kvantuminformatikából ismert optimalizációs módszereket. A kvantuminformatika és a gépi tanulás szorosan összeköthető folyamatok, mely esetében a klasszikus adatokat kvantumprocesszoron vizsgáljuk, ezzel növelve a számítási kapacitást, melyen mind a számítási sebességet, mind a csökkentett adatméretet értjük, hiszen a kódolt klasszikus állapotoknak képesek vagyunk előállítani a kvantum-szuperpozícióját.

A dolgozatomban szeretném a fentebb megemlített elvet kihasználni. Elsőként bevezetem a kvantuminformatika és a gépi tanulás alapjait, majd bemutatom a két tématerület összefonódását. Ezután egy-egy példán keresztül bemutatom az optimalizációban ismert kvantumbecslés optimalizációs algoritmust (Quantum Approximate Optimization Algorithm, - QAOA), és a szupport vektor gépeknél (Support Vector Machine - SVM) megismert kernel metódust. Végezetül pedig ismeretetem a konvolúciós neurális hálózaton végzett vizsgálatok eredményeit. A dolgozat célja egy olyan összehasonlítás bemutatása, mely szemlélteti a módszer gyakorlati alkalmazhatóságának jelenlegi helyzetét, lehetőségeit.

2. fejezet

Kvantuminformatika

Az olyan rendszerek leírását, melyek működésére a klasszikus fizika megkötései nem adnak megfelelő magyarázatot, valószínűségi alapon adjuk meg, a kvantummechanika törvényei segítségével. A mai számítógépek, habár egyre növekvő számítási kapacitással rendelkeznek, sok esetben nehezen, vagy egyáltalán nem képesek megoldani különböző problémákat. Jelenleg is rengeteg optimalizációs problémát ismerünk (pl.: útvonaltervezés, hálózati optimalizáció, stb.), melyek NP-nehéz problémák, ugyanis ezen komplex rendszerek állapottere exponenciálisan nő. Ezen feladatok megoldására alkalmazhatunk klasszikus heurisztikákat (pl.: szimulált lehűtés), vagy a teljes eseményteret egyidőben bejáró kvantumos elvű megoldásokat. Ezek a már jól ismert klasszikus bitek helyett kvantumbitekkel operálnak, amit reprezentálhatunk egy elektron spinjével vagy egy foton polarizációjával. Ezen kvantumbitek a rendszer állapotainak szuperpozíciójában vannak, így a teljes állapottér egyidőben előáll számunkra.

2.1. Kvantum kapus elven alapuló kvantumszámítógép

Az elsőként kialakuló (és máig a legismertebb) modell, a kvantum kapus modell, melynek elméleti alapjait David Deutsch dolgozta ki[2]. Ebben a modellben a klasszikus logikai kapukhoz hasonló elven működő kvantumkapuk (unitér transzformációk) követik egymást. A kvantumbit, kvantumregiszter, kvantumkapu fogalmak ismertetésére a továbbiakban kerül sor. Ilyen kvantumszámítógépes modellt alkalmaz az IBM, a Google, és a Microsoft kvantumszámítógépe is.

2.1.1. A kvantuminformatika posztulátumai

A kvantuminformatika jelenleg négy alapvető feltevésen nyugszik, melyeket a következőkben szeretnék bemutatni.

2.1.1.1. I. Kvantum bit

Zárt fizikai rendszer állapota egy Hilbert-térbeli egységhosszú vektorral írható le. A Hilbert-tér egy olyan komplex vektortér, amiben értelmezve van a Hermite-féle alak (belső szorzat) [3].

Egy kvantumbit a $|0\rangle$ és a $|1\rangle$ kvantum-állapotok szuperpozíciójával adható meg. Ez azt jelenti, hogy a Hilbert-térben minden kvantumbit egy egységvektor, ami felírható a $|0\rangle$ és a $|1\rangle$ állapotok lineáris kombinációjaként, azaz a következő módon:

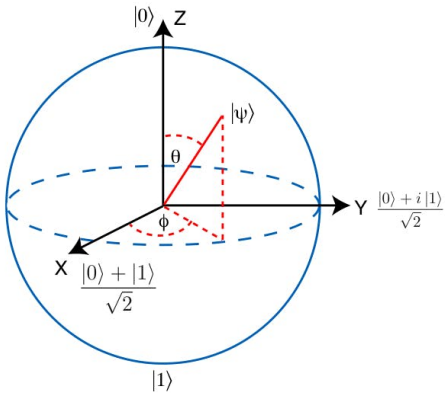
$$|\phi\rangle = a\,|0\rangle + b\,|1\rangle\,, \tag{2.1}$$

ahol a és b a két állapot valószínűségi amplitúdói. Ebben az esetben $a^2 + b^2 = 1$, hiszen az állapotok egység hosszú vektorral írhatók le.

Érdemes bevezetni a Bloch-gömb fogalmát: minden normalizált tiszta kvantumállapot leírható az alábbi módon:

$$|\Phi\rangle = cos(\frac{\Theta}{2}) + e^{i\phi}sin(\frac{\Theta}{2}). \tag{2.2}$$

Ily módon szemléltethetünk egy kvantum-állapotot a következő módon:



2.1. ábra. Bloch gömb [4]

2.1.1.2. II. Kvantum rendszer fejlődése

Egy zárt kvantumrendszer időbeli fejlődése leírható az időfüggő Schrödinger egyenlettel:

$$i\hbar\frac{\delta\,|\phi\rangle}{\delta t} = H\,|\phi\rangle\,, \tag{2.3}$$

ahol H a Hamilton-függvény (melyet a rendszer energiafüggvényének nevezünk, erre később részletesebben is kitérek), $\hbar = \frac{h}{2\pi}$, ahol h a Planck-állandó. Általában a kvantuminformatika területén számunkra elegendő ezen egyenlet diszkretizált változatát ismerni, melyet a következőképpen irhatunk fel: egy zárt rendszer időbeli változása a t_0 és t_1 időpontok között unitér transzformációval írható le:

$$|\phi_{t_1}\rangle = U\,|\phi_{t_0}\rangle\,. \tag{2.4}$$

Ezen unitér transzformáció levezethető a Schrödinger egyenletből:

$$|\phi_{t_1}\rangle = \exp\left(\frac{iH(t_1 - t_0)}{\hbar}\right)|\phi_{t_0}\rangle\,, \tag{2.5}$$

$$U(t_0, t_1) = \exp\left(\frac{iH(t_1 - t_0)}{\hbar}\right)|\phi_{t_0}\rangle\,. \tag{2.6}$$

Látható, hogy a fenti unitér transzformáció normatartó, hiszen a hossza egységnyi [3]. A leggyakoribb transzformációk a Pauli-X, Pauli-Y, Pauli-Z és a Hadamard kapu (itt a transzformációt kvantumkapu elnevezéssel illetjük, később így hivatkozunk rá), melyek rendre a következők:

$$X = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \tag{2.7}$$

$$Y = \begin{bmatrix} 0 & -i \\ i & 0 \end{bmatrix}, \tag{2.8}$$

$$Z = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}, \tag{2.9}$$

$$H = \frac{1}{\sqrt{2}} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}. \tag{2.10}$$

2.1.1.3. III. Mérés

A kvantum mérés leírható M_m mérési operátorok halmazával, mely operátorok esetében m az m állapothoz tartozó mérés operátora. Amennyiben a rendszer $|\phi\rangle$ állapotban van, akkor annak a valószínűsége, hogy a mérés során m-et mérünk[3]:

$$p(m) = \langle\phi| M_m^\dagger M_m |\phi\rangle, \tag{2.11}$$

és a rendszer mérés utáni állapota:

$$\frac{M_m |\phi\rangle}{\sqrt{\langle\phi| M_m^\dagger M_m |\phi\rangle}}. \tag{2.12}$$

2.1.1.4. IV. Többites kvantum rendszer

Egy összetett rendszer állapota megadható a részrendszerek állapotainak tenzorszorzatából:

$$|\Phi\rangle = |\phi_1\rangle \otimes |\phi_2\rangle \otimes ... |\phi_n\rangle. \tag{2.13}$$

Ebben az esetben $|\Phi\rangle$ állapotot kvantumregiszternek nevezzük. Egy n kvantumbites rendszer (kvantumregiszter) leírásához klasszikusan 2^n bitre lenne szükségünk [3].

2.1.2. Összefonódás

Kvantum összefonódásról akkor beszélünk, ha egy több állapotból álló fizikai rendszert nem tudunk szétbontani tenzorszorzatokra. Egzakt matematikai formában leírva ez azt jelenti, ha egy tiszta kvantumállapot $|\phi_{AB}\rangle$ nem írható le a következő formában: $|\phi_A\rangle \otimes |\phi_B\rangle$. Ilyen állapotok például a Bell-állapotok [3]:

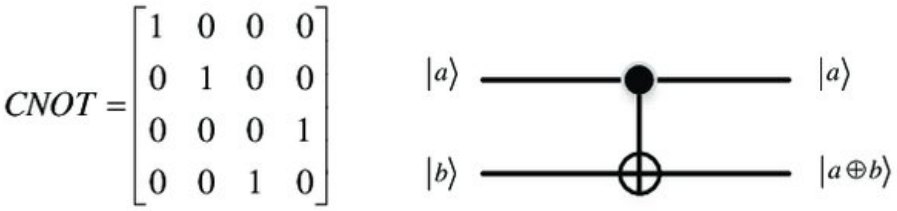
$$|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle + |11\rangle) \tag{2.14}$$

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|01\rangle + |10\rangle) \tag{2.15}$$

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|00\rangle - |11\rangle) \tag{2.16}$$

$$|\phi\rangle = \frac{1}{\sqrt{2}}(|01\rangle - |10\rangle) \tag{2.17}$$

A Bell-állapotok előállításához használhatunk CNOT kaput, mely lényegében egy vezérelt NOT kapu. A transzformáció mátrixa, illetve kvantumkapus megjelenítése az alábbi ábrán látható:



2.2. ábra. CNOT kapu [5]

2.1.3. IBM kvantumszámítógép

Az IBM egy 1911-ben alapított amerikai cég, mely az elsők között hozott létre kvantumprocesszorokat. A legutóbb kiadott processzor a 127 kvantumbittel rendelkező IBM Eagle processzor, mely a 65 kvantumbites Hummingbird, illetve a 27 kvantumbites Falcon processzort követte. Az áttörést azzal érték el, hogy több fizikai szintet kötöttek össze, illetve az Eagle processzor már az újonnan fejlesztett erősen-hexagonális struktúrában rendezi a kvantumbiteket [6]

Az IBM kvantumszámítógépéhez online elérést biztosít. Az IBM Quantum System One az első, bárki számára elérhető integrált rendszer. Jelenleg a 27 kvantumbites Falcon processzor használható általa, azonban fejleszthető a komplexebb verziókra is (az Eagle processzor elérhetősége például 2023-ra várható). 2023-ban várhatóan ezt követi az IBM Quantum System Two 433-1,121 kvantumbittel rendelkező processzorokkal operáló rendszere [7]. Érdemes megemlíteni ezen felül a Qiskit-et, mely egy nyílt forráskodú IBM által fejlesztett Python modul, amely hozzáférést biztosít a cég kvantumszámítógépéhez, illetve könnyebb implementálhatóságot nyújt az elérhető függvényeknek köszönhetően.

2.3. ábra. IBM Eagle kvantumszámítógép [8]

2.2. A lehűtésen alapuló kvantumszámítógép

A másik kvantumprocesszor modell a lehűtésen alapuló kvantumszámításon alapszik. Ebben a fejezetben összefoglalom az ehhez tartozó fogalmakat, más alkalmazási lehetőségeket, illetve bemutatom a D-Wave cég kvantumprocesszorát.

2.2.1. Ising-modell

A modellt Wilhelm Lenz német fizikus alkotta meg 1920-ban, nevét Ernst Ising szintén német fizikusról kapta, aki elsőként megoldást talált az egydimenziós problémára 1925-ben [9]. Az Ising-modell alkalmas fizikai rendszerek matematikai leírására. A modell szerint egy rendszer különböző állapotainak tulajdonsága (például egy mágneses dipól esetén a spin-ek orientációja) diszkrét változókkal írhatók le, melynek lehetséges értékei a $\sigma = -1$ és a $\sigma = 1$. Egy ilyen rendszer tehát megadható a következő matematikai leírással:

$$\Omega_N = \{-1, +1\}^{xN} = \{(\underline{\sigma} = (\sigma_1, \sigma_2, ..., \sigma_N) : \sigma_k = \pm 1\} |\Omega_N| = 2^N. \tag{2.18}$$

2.2.2. Hamilton-függvény

Egy rendszer energiafüggvénye, azaz Hamilton-függvénye megadható az Ising-modell segítségével. A rendszer összenergiájához kétféle hozzájárulás lehet, azaz kétféle interakció növelheti, illetve csökkentheti a Hamilton-függvényünk értékét: minden spin interakcióban van egy külső térrel ($h_i \cdot \sigma_i$) és fellép egy kölcsönhatás a szomszédos spinek között is ($-J_{i,j} \cdot \sigma_i \cdot \sigma_j$). J nagysága reprezentálja, hogy milyen erős a kölcsönhatás a spinek között, az előjele pedig azt, hogy az azonos, vagy az ellentétes orientáció van jobban előnyben részesítve az energiaminimumra törekvés során. A fizikai rendszer teljes energiája ezen energiák összege. Eszerint a Hamilton-függvény Ising-modell esetén megadható az alábbi egyenlettel:

$$H(\sigma) = -\sum_{<i,j>}^{N} J_{i,j} \cdot \sigma_i \cdot \sigma_j - \sum_{i=1}^{N} h_i \cdot \sigma_i. \tag{2.19}$$

A fenti absztrakt leírás helyett az Ising-modell illusztrálható egy mágneses dipólussal. Példánkban tekintsünk egy olyan esetet, melyben a spinek egy kétdimenziós rácsban helyezkednek el. Minden spin kölcsönhatásban van egy külső mágneses térrel (ennek nagysága h_i, ami reprezentálja a mágneses tér erősségét, iránya pedig azt mutatja meg, hogy az i. spin milyen irányítottságot preferál). A külső mágneses tér teljes hatását, annak előjeles összegzésével kaphatjuk. A második tag, ami az energiafüggvényben szerepel a szomszédos spinek egymásra hatása ($J_{i,j}$). Amennyiben $J_{i,j} > 0$, a kölcsönhatást ferromágnesesnek nevezik, amennyiben $J_{i,j} < 0$ antiferromágnesesnek, ha pedig $J_{i,j} = 0$ a spinek nincsenek kölcsönhatásban. Ferromágneses esetben a spinek egyirányúsága kisebb hozzájárulással rendelkezik, mintha nem ugyanabba az irányban állnak. Ellenkező esetben a rendszerben lévő spinek főként az ellenirányítottságot preferálják. Nyilvánvalóan a teljes hozzájárulást ezek előjeles összegzésével kapjuk.

Az informatikában sokszor használják az egzaktul megoldható Sherrington-Kirkpatrick modellt [10], mely lényegében az Ising-modell távolabbi ferromágneses és antiferromágneses hatásokkal. A modell alapján megadható a Hamilton függvény a következőképpen:

$$H(\sigma) = -\sum_{i<j}^{N} J_{i,j} \cdot \sigma_i \cdot \sigma_j. \tag{2.20}$$

A Sherrington-Kirkpatrick modellnek nagy jelentősége van például neurális hálós alkalmazásokban vagy NP-nehéz optimalizációs problémák megoldásában. Egy lehetséges konfiguráció létrejöttének valószínűsége az Ising-model szerint [11]:

$$P(\sigma) = \frac{e^{-H(\sigma)\beta}}{Z_\beta}. \tag{2.21}$$

ahol $\beta = \frac{1}{k_B \cdot T}$, és k_B a Boltzmann-állandó, T a hőmérséklet, Z_β pedig a normalizációs függvény vagy más néven állapotösszeg, ami az alábbi módon adható meg:

$$Z_\beta = \sum_\sigma e^{-\beta \cdot H(\sigma)}. \tag{2.22}$$

Végtelen hőmérsékleten ($\beta = 0$), minden konfiguráció ugyanakkora valószínűséggel állhat elő. Nagy, de nem végtelen hőmérséklet esetében kis mértékű korreláció van a szomszédos bitek között. Ebben az esetben a mágnesezettség a spinek átlagértékének összege, ami a következő egyenlettel írható le:

$$M = \frac{1}{N} \sum_{i=1}^{N} \sigma_i. \tag{2.23}$$

Ekkor minden M mágnesezettségű konfigurációhoz tartozik egy $-M$ mágnesezettségű konfiguráció ugyanazzal a valószínűséggel (mivel ugyanakkora energiával is rendelkeznek), ezért a rendszer ugyanannyi időt tölt az M mágnesezettségű konfigurációban, mint a -M mágnesezettségűben. Ebben az esetben az átlagos mágnesezettség a teljes időre nézve nulla. A korreláció csak a szomszédos bitek között lép fel, mivel $J_{i,j}$ nagyon kicsi, ha $i.$ és $j.$ spinek szomszédosak. Amennyiben $i.$ és $j.$ spinek nem szomszédosak $J_{i,j}$ elhanyagolható lesz. Kis hőmérsékleten ($\beta >> 1$) a konfiguráció közel van a legkisebb energiájú rendszerállapothoz, ahol minden spin ugyanolyan irányítottságú. Rudolf Peierls bizonyította [12],hogy egy rendszer, amelyben minden spin a kijelölt pozitív irányba áll, soha nem fluktuál át abba a konfigurációba, ahol minden spin negatív irányítottságú.

2.2.3. Alkalmazások

Az eredeti motiváció az Ising-modellre a ferromágnesesség jelensége volt. A mágnesezettség annak köszönhető, hogy az elektronok spinje az anyagban ugyanolyan irányítottságú. Az Ising-modell azt vizsgálja, hogy az elektronok nagyrésze hogyan kerülhet olyan irányítottságba, hogy a kijelölt halmazban a spinek mind pozitív, vagy negatív irányítottságúak legyenek úgy, hogy csak a szomszédos spinek tudnak korrelálni egymással. Emellett hasonló alkalmazása még a spin-üvegek problémája, ahol a spinek egy adott T_f hőmérséklet alatt "befagynak", azaz egy olyan rendezetlen, metastabil állapotban ragadnak, amely állapot nem a lehető legkisebb energiájú állapot [13]. Nevét a spin üvegek mágnesezettsége és a kémiai üveg térbeli rendezettlensége közti analógiáról kapta.

Az Ising-modell alkalmas gázatomok mozgásának modellezésére is. A modell generál egy téridő rácsot, és azt vizsgálja, hogy az egyes pozíciók tartalmazzák-e az atomot („1”-es bitet rendelve hozzá), vagy nem („0”-s bitet rendelve hozzá). A rendszer energiája (ami az interakcióban lévő atomokból fakad) leírható a következőképpen: az atomok közti korrelációt a $-4 \cdot J_{i,j} B_i B_j$ egyenlet adja meg, ez kontrollálható hozzáadott kémiai potenciállal (μ). A rácsgáz energiája:

$$H(B) = -\frac{1}{2} \cdot \sum_{i,j} 4 \cdot J_{i,j} B_i B_j + \sum_i \mu B_i. \tag{2.24}$$

A biteket helyettesítve a spin terminológiával: ($B_i = \frac{\sigma_i + 1}{2}$):

$$H(\sigma) = -\frac{1}{2} \cdot \sum_{i,j} J_{i,j} \sigma_i \sigma_j + \sum_i (4 J_{i,j} - \mu) \sigma_i. \tag{2.25}$$

Látható, hogy ez az egyenlet már formailag is megegyezik a Hamilton-függvénnyel. A biológiai rendszerekben is a rácsgáz modell egy módosított verzióját használják, hogy leírják a kötések viselkedések, vagy a DNS kondenzációt [14]. A neuronok aktivitása szintén

modellezhető az Ising-modell segítségével. Ebben az esetben a spin állapotok a neuronok aktív és inaktív állapotához vannak társítva. Minden neuronhoz társítható egy gyulladási ráta (h_i), illetve mivel ezek nem teljesen függetlenek egymástól, minden neuronpárhoz társítható egy korrelációs ráta is ($J_{i,j}$). Ebben az esetben az energiafüggvény megadható a következő módon:

$$H(\sigma) = -\frac{1}{2} \cdot \sum_{i,j} J_{i,j}\sigma_i\sigma_j + \sum_i h_i\sigma_i. \tag{2.26}$$

2.2.4. Kvantum-lehűtés

A kvantum-lehűtés (Quantum Annealing - QA) egy metaheurisztikus folyamat függvények globális minimumának megtalálására kvantum-fluktuáció segítségével. Kvantum lehűtés esetén a rendszer előállítja az összes lehetséges állapot szuperpozícióját ugyanakkora valószínűséggel, majd fejlődik az időfüggő Schrödinger egyenlet szerint. Ekkor a lehetséges állapotok amplitúdója megváltozik. Ebben az esetben ha a változás elegendően lassú a rendszer az alapállapotba kerül (ezt hívják adiabatikus kvantum-lehűtésnek). Ha a változás gyors a rendszer elhagyhatja az alapállapotot. Végezetül tehát, ha a rendszer az energiaminimumba kerül megoldást kapunk a megadott Ising-modellre, ami az optimalizációs problémánk megoldását is jelenti.

A kvantum-alagutazás egy kvantummechanikai folyamat, ahol a szubatomi részecskék képesek átlépni az adott potenciálgátakat anélkül, hogy elegendő energiával rendelkezzenek. Ezt nyilvánvalóan a klasszikus mechanika törvényei nem engedik meg, azonban számos folyamatban játszanak nagyon fontos szerepet (például: radioaktiv sugárzás). A jelenség az anyag hullámtermészetéből fakad. A kvantum lehűtés ezen jelenséget használják ki a lokális minimumokból történő kilépésre, mesterséges probabilisztikus eljárások helyett [15].

2.2.5. Kvadratikus kényszermentes bináris optimalizáció

A QUBO (Quadratic Unconstrained Binary Optimization) problémák optimalizálása NP-nehéz probléma. Ezen problémák QUBO-modellje olyan másodfokú függvénnyel rendelkezik, melynél a változók értéke a $\{0; 1\}$ halmazból kerül ki. A QUBO modellben a változók kvantumbitekkel vannak reprezentálva, az optimalizációs probléma leírható a következő egyenlettel [16, 17, 18]:

$$y = x^T \cdot Q \cdot x, \tag{2.27}$$

ahol x a bináris döntési változók által alkotott vektor, Q a QUBO mátrix (ami egy felsőháromszög mátrix), mely tartalmazza az adott bináris változók kölcsönhatását. A megoldandó optimalizációs problémában a változók értékének lehetséges halmaza miatt négyzetes változók helyettesíthetők saját elsőfokú tagjukkal [16].

A Hamilton-függvény(melyet megismertünk az Ising-modellnél), egy kvadratikus függvény, mely tartalmazza a kvantumbitek lineáris és kvadratikus részét. A különbség a két modell között pusztán annyi, hogy az Ising-modell esetében a paraméterek a $\{-1; 1\}$ halmazból választódnak [16, 17], emiatt a Hamilton függvény könnyedén átkonvertálható QUBO formába a következő módon:

$$x_i = \frac{\sigma_i + 1}{2}. \tag{2.28}$$

A QUBO-modell nem tartalmaz megkötéseket azon felül, hogy a változók binárisak. Mindazonáltal számos probléma nem modellezhető különböző megkötések nélkül, erre a célra alkalmaznak a QUBO modellek büntetéseket. A büntetések értéke nulla, amennyiben a megoldás megfelelő, ellenkező esetben viszont nullától különböző értékkel járulnak hozzá

az energiafüggvényhez. A büntetésekkel tűzdelt modell optimalizálása lesz tehát az optimalizálási feladatok során felhasznált modell.

2.2.6. D-Wave QPU

Végül szeretném bemutatni a másik nagy csoporthoz tartalmazó, talán manapság legjobban hasznosítható kvantumprocesszort. A kanadai D-Wave céget 1999-ben alapították, mint az első kvantuminformatikai nagy- vállalatot. A cég által fejlesztett kvantumprocesszor (Quantum Processing Unit - QPU) képes megoldani a QUBO vagy Ising-modellel reprezentált NP-nehéz problémákat kvantumlehűtést alkalmazva. Jelenleg két jelentős architektúrával rendelkezik.

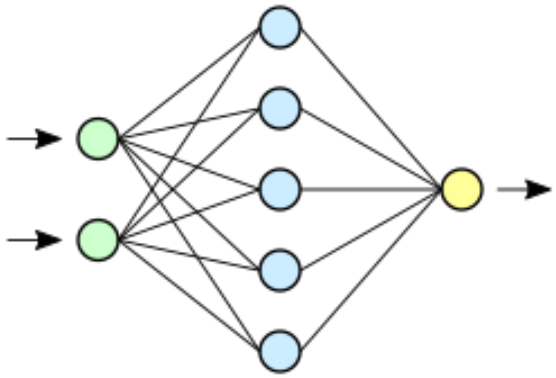
2.4. ábra. D-Wave Advantage QPU [19]

3. fejezet

Gépi tanulás

A mesterséges intelligencia lényegében azt jelenti, hogy egy gép emberszerű intelligenciával rendelkezik. A gépek egy adott bemenetre egy adott kimenetet szolgáltatnak, és a tanulási folyamat alatt a kimenet megfelelőségének maximalizálása érdekében módosítják belső működésüket. Ilyen gépi tanuló algoritmusok például a neurális hálózatok.

Neurális hálózatnak nevezzük azt a hardver vagy szoftver megvalósítású párhuzamos, elosztott működésre képes információfeldolgozó eszközt, amely azonos, vagy hasonló típusú – általában nagyszámú – lokális feldolgozást végző műveleti elem, neuron többnyire rendezett topológiájú, nagymértékben összekapcsolt rendszeréből áll, rendelkezik tanulási algoritmussal, mely általában minta alapján való tanulást jelent, és amely az információfeldolgozás módját határozza meg, illetve rendelkezik a megtanult információ felhasználását lehetővé tevő információ előhívási, vagy röviden előhívási algoritmussal [20].



3.1. ábra. Egy neurális hálózat egyszerű szemléltetése

3.1. Neuron

Ahogyan azt már fentebb leírtam a neurális hálózatokat neuronok építik fel. Egy neuronnak több bemenete és egy kimenete van, azok között nemlineáris kapcsolat van. A nemlinearitást nevezzük aktivációs függvénynek, melyek közül többfélét is megkülönböztetünk (szigmoid, relu, stb.). Egy neuron rendelkezhet memóriával is (rekurrens neurális hálózatok).

3.2. Tanulás

A mesterséges intelligencia esetében általában tanuló módszereket alkalmazunk, melyek egy általános, paraméterezhető modellt nyújtanak úgy, hogy a tanulás során tanító adathalmazt használnak fel annak érdekében, hogy a modellt úgy állítsák be, hogy az adott problémák pontos megoldására alkalmas legyen. Ezen paraméterek általában statisztikai módszerekkel, numerikus optimalizálással, heurisztikákkal határozhatók meg, így nem feltétlenül mindig a helyes eredményt kapjuk.

Az egzakt matematikai leírás a következő: adott $\overline{x}$ bemeneti adatvektorunk, és f modellünk. Ezáltal a neurális hálózatunk kimenete a következőképpen alakul:

$$\hat{y} = f(\overline{x}, \Theta) \tag{3.1}$$

Három féle tanulási módszert különböztetünk meg:

- Felügyelt/ellenőrzött tanulás.
- Felügyelet nélküli, nem ellenőrzött tanulás.
- Analitikus tanulás.

Jelen dokumentum témája a felügyelt tanulás lesz, így a későbbi részekben erre külön nem hivatkozom. Érdemes megemlíteni néhány metrikát, melyet később használok. Pontosság alatt azt értem, hogy az adott bemenethez tartozó predikciók hány százalékban egyeznek meg a hozzájuk tartozó címkékkel. Hiba alatt a címke értékének és a predikció értékének különbségét értem. Epoch alatt a teljes adatbázis feldolgozását értem egy tanítási és egy validációs folyamaton. Konfúziós mátrixnak nevezzük azt a mátrixot, melyet a predikció (nálam sorok) és aktuális címke (nálam oszlopok) összeállításából tudunk képezni.

3.2.1. Felügyelt tanulás

Ebben az esetben ismerjük a neurális hálózat összetartozó be- és kimeneti értékeit, ezek alapján tanítjuk a hálót. Megadhatjuk, hogy a hálózat által predikált eredmény, illetve a valódi eredmény között milyen kapcsolat áll fennt, ezek alapján definiálhatunk egy költségfüggvényt:

$$C(\hat{y}, y) = C(f(\overline{x}, \Theta), y), \tag{3.2}$$

ahol C az adott költségfüggvény (például: MSE (mean squared error) esetén $C(\hat{y}, y) = (\hat{y} - y)^2$). A költségfüggvény szerint módosíthatjuk modellünket, ezzel javítva annak pontosságát. Általában az eljárást iteratívan hajtják végre. Olyan eset is fennállhat, amikor nem áll rendelkezésre a helyes válasz, csak az, hogy a bemenetre adott kimenet helyes vagy helytelen-e, ezt megerősítéses tanulásnak nevezzük [21].

3.2.2. Felügyelet nélküli tanulás

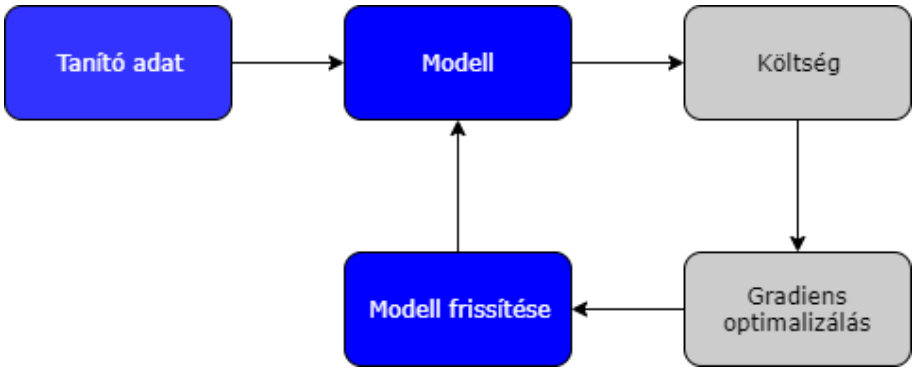
Ebben az esetben nem állnak rendelkezésünkre adott bemenetekhez tartozó kimenetek, a neurális hálózatnak a bemeneti adatok közti kapcsolatot kell megtalálnia. Általában ez a tanulás is iteratív [20]. A két eddig említett tanulás között van a félig felügyelt tanítás, ahol az adatok egy része címkézett, míg a másik része nem az.

3.2.3. Analitikus tanulás

Ebben az esetben a háló viselkedése szintén a bemenetek alapján alakul ki, azonban itt nem az általuk adott információ által, hanem analitikus módszerekkel.

3.3. Tanítás optimalizálása

Az elérhető adathalmazunkat több részre osztjuk, az adat nagyobbik részét (nagyjából 80%-át) tanításra használjuk fel, míg a kisebbik részét tesztelésre és validációra. A validációs halmaz elemei arra szolgálnak, hogy a tanítás során megvizsgáljuk a modellt, majd a túltanulás elkerülése érdekében módosítsuk a hiperparamétereket. A teszthalmazt kizárólag a modellünk végső értékelésére használjuk, az a tanítás folyamatában nem vesz részt. A tanulás folyamatát mutatja be a 3.2. ábra.



3.2. ábra. Tanulási folyamat felügyelt tanulás esetén

Ebben a részben szeretném bemutatni azt, ha már ismerjük a költségfüggvényt, hogyan tudunk beavatkozni a modell megváltoztatásába.

3.3.1. Elsőrendű gradiens alapú optimalizálás

Tekintsünk egy lineáris osztályozási modellt, melyre:

$$y = \overline{\Theta} \cdot x, \tag{3.3}$$

ahol $\overline{\Theta}$ a súlyvektor. A teljes költségfüggvény nem ismert, azonban a költségfüggvény, a modell által definiált súlyvektor általi deriváltjai meghatározhatók. Ezen deriváltak irányába történő elmozdulás a paramétertéren a hibafüggvény maximális növekedéséhez járulna hozzá, ha ezzel ellentétes irányba mozdulunk, akkor pedig a legnagyobb mértékű csökkenést kapjuk. Ebbe az irányba kell lépnünk, majd iteratívan megismételve ezt egy lokális minimumba juthatunk. Általában nem a teljes tanító adatbázis hibáját értékeljük ki, csak egy-egy batch nagyságnyi mintáét (sztochasztikus gradiens módszer) [22].A gradiens alapú optimalizálással az i. iteráció alapján kapható súlyvektor legyen $\overline{\Theta}_{i+1}$, amelyre a következő egyenlet teljesül:

$$\overline{\Theta}_{i+1} = \overline{\Theta}_i - \alpha \cdot \frac{\Delta C(f(\overline{x_{train}}, \overline{\Theta}_i), y)}{\Delta \overline{\Theta}_i}, \tag{3.4}$$

ahol α a tanulási ráta. Amennyiben α minél nagyobb, annál jobban tudunk az optimum felé haladni, azonban ha túl nagyra választjuk előfordulhat, hogy nem találunk bele pontosan a lokális minimumba. Ezért fontos a tanulási ráta ütemező alkalmazása. A célunk a globális optimum megtalálása, ennek érdekében érdemes a gradiens "momentumát" kihasználni: emiatt a gradiens érték számítása a korábbi két értékből tevődik össze.

3.3.2. Másodrendű gradiens alapú optimalizálás

Az előző fejezetben bemutatott eljárás az optimum közelében könnyen oszcillálhat, illetve a konvergálás az optimumhoz relatíve lassú, így érdemesebb a másodrendű gradiens alapú

optimalizálást használni. Ez a következőképp írható le:

$$\overline{\Theta}_{i+1} = \overline{\Theta}_i - \cdot \frac{f(\overline{x_{train}}, \overline{\Theta}_i)'}{f(\overline{x_{train}}, \overline{\Theta}_i)''}. \tag{3.5}$$

Ezt a módszert Newton iterációnak nevezik. Hátránya, hogy a Hesse-mátrix kiszámítása jóval nagyobb számítási kapacitást igényel.

3.3.3. Felmerülő problémák a tanítás során

A tanítás során felmerülhet számos probléma a modell bonyolultságának és az adathalmaznak köszönhetően. Amennyiben túl egyszerű a modellünk, nehezen tanulja meg a mintákat, emiatt pontatlan lesz az eredmény. Ezt a jelenséget underfittingnek nevezzük. Ellenkező esetben az overfitting jelentkezik, ami azt jelenti, hogy a modell konkrétan rátanul a tanítóadatbázisra ahelyett, hogy csak a szabályszerűségeket ismerné fel, emiatt más adatbázison nem fog megfelelően működni. Ezt overfittingnek nevezzük. Előbbit a modell bonyolultabbá tételével, utóbbit, dropout rétegekkel, kimeneti nemlinearitásokkal lehetséges csökkenteni.

Nehézséget okoz még továbbá a hiperparaméterek optimalizálása (pl.: tanulási ráta, momentum, stb.). Gondoljunk bele, ha egy több hiperparaméterből álló rendszerünk van, akkor azok egy sokdimenziós teret feszítenek ki, ahol meg kell találnunk a globális optimumot. Erre használható rácsos, vagy random elrendezés is, de léteznek ennél jóval szofisztikáltabb módszerek is, mint például a Bayes optimalizáció [23].

3.4. Előre haladó neurális háló

Amennyiben összecsatolunk neuronokat egy előre haladó neurális hálózatot kapunk. A neurális háló összekapcsolt neuronjai kezdetben random paraméterkészlettel rendelkeznek, valamint egy-egy réteg bemeneteiből megállapíthatjuk a súlyvektor ismeretében a kimenetét, ezt előreterjesztésnek nevezzük. Ekkor azonban a láncszabály miatt a deriváltakat is ismerjük. A kimenetet hátraterjesztve (backpropagation) meghatározhatjuk az összes súly gradiensét, így végrehajtva az optimalizációt.

Egy előrehaladó lineáris neurális háló esetén egyetlen perceptron a következőképpen adható meg:

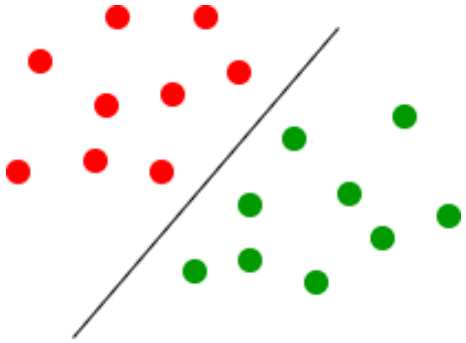
$$\underline{\sigma} \cdot (\overline{\Theta}^T \overline{x}), \tag{3.6}$$

ahol $\underline{\sigma}$ az aktivációs függvény, másnéven a kimeneti nemlinearitás. A többszintes neurális háló (N rejtett réteggel rendelkező) pedig a következő módon írható le:

$$\underline{\sigma}_n \cdot (\overline{\Theta}_N^T (...\underline{\sigma}_1 \cdot (\overline{\Theta}_1^T \overline{x})). \tag{3.7}$$

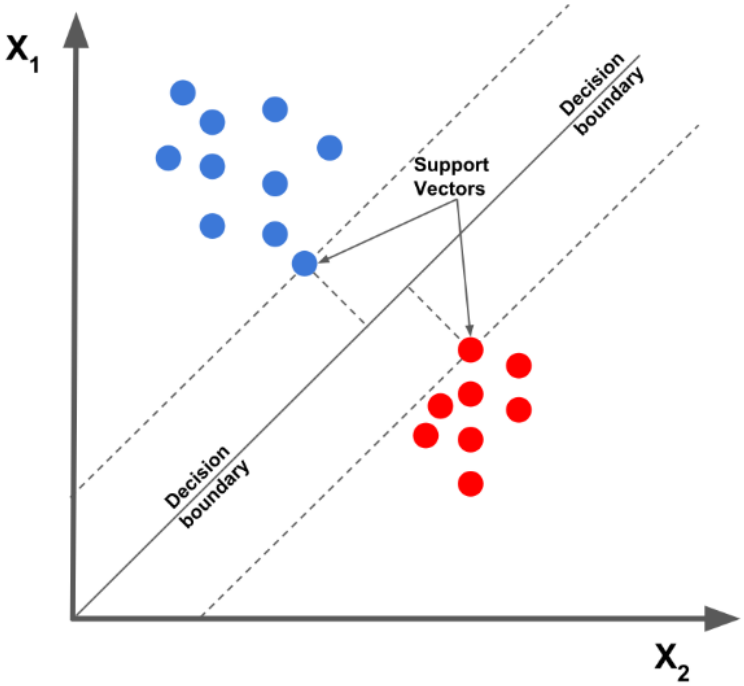
3.5. Szupport vektor gép

A szupport vektor gépek [24] egy felügyelt tanulási modellel rendelkező klasszifikációra vagy regresszióra használt neurális háló modell. Lineáris és bináris optimalizációt képesek végrehajtani. Tekintsünk újra egy lineáris osztályozási modellt. Ebben az esetben a modell képes egy adathalmazhoz bináris címkéket társítani úgy, hogy meghatároz egy hipersíkot, ami elválasztja egymástól a két osztályt (3.3. ábra).



3.3. ábra. Lináris osztályozó

Amennyiben egy ilyen lineáris modellt határozunk meg érdemes megkeresni azt a hipersíkot, ami a legközelebbi pontoktól egyenlő távolságra van, hogy maximalizáljuk a rést a klasszifikálási határ, illetve a tanító adatok között. Ezen hipersík és a tanító pontok közti eltolást hívjuk szupport vektornak.



3.4. ábra. Szupport vektorok [25]

Sok esetben fennáll az az eset, hogy nem lehet lineárisan elszeparálni egymástól ezeket az adatokat: ebben az esetben alkalmazzuk a feature leképzéseket. Ebben az esetben áttranszformáljuk egy magasabb dimenziós térbe az adatokat, hogy ott aztán már lineáris klasszifikációt tudjunk rajta végrehajtani. Matematikailag a hipersík leírható a következőképpen:

$$w^T \Phi(x_i) + b. \tag{3.8}$$

Ezen szupport vektorokat felfoghatjuk egy optimalizálási problémának is, miszerint a távolságot szeretnénk maximalizálni a tanító adatok és a döntési küszöb között [26, 27]:

$$min_{\alpha,w,b} \; L_p = \frac{\|w\|^2}{2} - \sum_{i \in T} \alpha_i [y_i (w^T \Phi(x_i) + b) - 1], \tag{3.9}$$

ahol α_i a Lagrange-multiplikátorok, y_i a címkék. Ehelyett azonban érdemesebb bevezetni a probléma duálisát:

$$max_\alpha \;\; L_D(\alpha) = \sum_{i \in T} \alpha_i - \frac{1}{2} \sum_{i,j \in T} y_i y_j \alpha_i \alpha_j \Phi(x_i)^T \Phi(x_j). \tag{3.10}$$

Ez a felírás azért célszerűbb, hiszen a címkéken és a multiplikátorokon kívül a tanító állapotokból egyfajta kernel mátrix hozható létre, melynek segítségével magasabb térben tudunk lineárisan klasszifikálni, ami lényegében egy nemlineáris klasszifikációt jelent az eredeti térben:

$$K_{i,j} = K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j). \tag{3.11}$$

4. fejezet

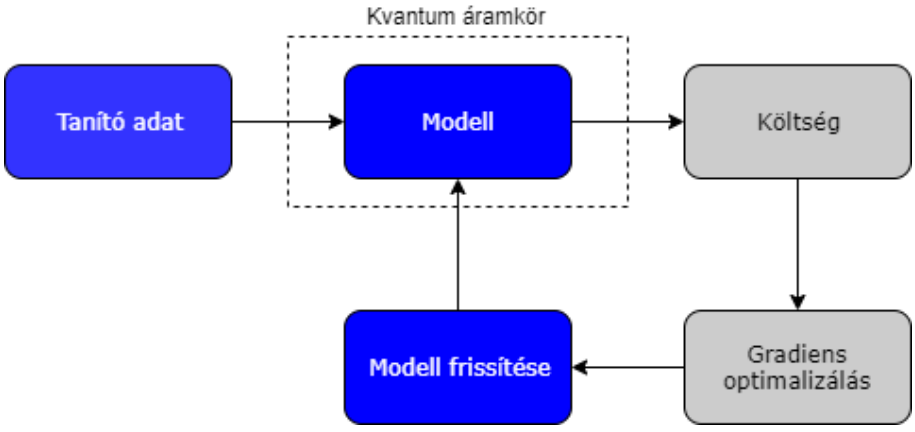
Gépi tanulás módszerek a kvantuminformatikában

A gépi tanulás és a kvantuminformatika négy különböző módon képes összefonódni, melyeket a következő táblázat szemlélteti:

	Adatok típusa	
Adatfeldolgozás	CC	CQ
	QC	QQ

4.1. táblázat. Kvantum gépi tanulás fajtái

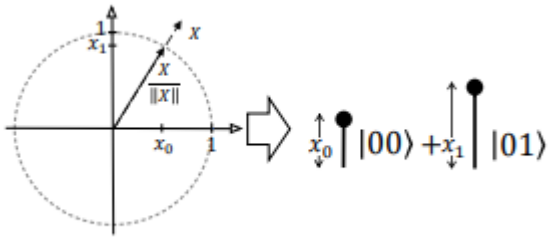
Ezen táblázatban az első karakter az adat, míg a második a feldolgozás típusát jelöli. A dolgozatom témája a klasszikus adatok tisztán kvantumos feldolgozása, tehát a CQ típusú kvantum gépi tanulás, ezesetben a 3.2. ábra a következőképpen módosul:



4.1. ábra. Tanulási folyamat kvantumos modellel

Eszerint a taníttandó modellünk egy kvantumkapukból álló áramkör lesz, ahol a változtatható paraméterek az adott kvantumkapuk paraméterei lesznek.

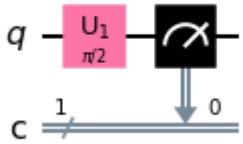
Első lépésként szükség van a klasszikus adatok kvantumos adatokra történő leképzésére, mely többféle módon is történhet. A legegyszerűbb módja ennek nyilvánvalóan az, hogyha a klasszikus állapotokat (legyen x_1, x_2) közvetlenül képezzük le bázisállapotokká. Alkalmazhatunk amplitúdó leképzést (másnéven hullámfüggvény leképzést [28]), mely során a $|0\rangle$ állapotokból egy unitáris amplitúdótranszformáció segítségével vagyunk képesek előállítani a klasszikus állapotokat [29].



4.2. ábra. Amplitúdó leképezés[29]

A következő leképezési technika a szögleképezés, ahol az előzőhöz hasonlóan egy alapállapotból indulunk ki, majd ebből Pauli X,Y, vagy Z kvantumkapukkal hozzuk létre a leképzett állapotokat. Ezen felül alkalmazhatunk még magasabb rendű összetettebbek leképezéseket is.

Az adatok kódolása után alkalmazhatunk egy variációs modellt, mint egy parametrizálható kvantum áramkört.



4.3. ábra. Kvantum áramkör

Ezen áramkör általában jóval bonyolultabb ennél (példát erre később láthatunk is), annak érdekében, hogy pontosabban legyen képes a tulajdonságok tanulására. Végezetül ki kell számítanunk a hibafüggvényt a címkék segítségével, majd gradiens optimalizálással megváltoztatni a modellünk paramétereit.

4.1. Kvantumos becslés optimalizációs algoritmus (QAOA)

Dolgozatom első témájaként a QAOA algoritmust vizsgálom. Egy optimalizálási probléma költségfüggvénye megadható a következő alakban:

$$H\left|z\right\rangle = f(z)\left|z\right\rangle, \tag{4.1}$$

ahol $|z\rangle$ a sajátvektort, $f(z)$ a sajátértékeket jelenti. Feladatunk, hogy létrehozzuk az állapotok szuperpozíciójából $|\phi\rangle$ állapotot, melyből kiszámíhatjuk $\langle H|H\rangle = \langle z|\, C\,|z\rangle$, melyet maximalizálnunk kell a folyamat során. Előbbi könnyedén kiszámítható klasszikusan is, amennyiben ismerjük az állapotot, azonban a maximalizálás egzakt megoldásához $O(2^N)$ lépésre lenne szükségünk. Azonban, ha kvantum változókat használunk a teljes eseményteret egyidőben be tudjuk járni, majd a kvantumállapotok mérésével egy pontos eredményt kaphatunk.

A QAOA algoritmus lényege, hogy a létrehozott áramkör forgatásokat alkalmaz, melyek megváltoztatják a kvantumbitek értékét úgy, hogy a mérés során a legvalószínűbb előálló állapot a helyes állapot legyen. Ezen forgatások lényegében egy költség réteg és egy keverő réteg alternálásából alkothatók meg a szuperozíció előállítása után [30, 31, 32, 33]. Ez a folyamat egyébként nagyban kötődik a "Kvantuminformatika" fejezetben megismert kvantum-lehűtéshez.

4.1.1. Szuperpozícionáló réteg

A kvantum áramkör modellek esetén a szuperpozíció megalkotásához Hadamard kapukat használunk. Eszerint ezen réteg kimenetén előálló állapot megadható a következő módon:

$$H\left|\phi\right\rangle = \sum_{x\in\{0,1\}^n} \frac{1}{\sqrt{2^n}}\left|x\right\rangle. \tag{4.2}$$

4.1.2. Költség réteg

Ebben az esetben egy fáziskaput alkalmazhatunk, melynek paraméterei a költségfüggvényen (Hamilton függvényen) alapulnak:

$$U(H,\gamma) = e^{-i\gamma H} = \prod_{\alpha=1}^{m} e^{-i\gamma H_\alpha}. \tag{4.3}$$

Az alábbi egyenletek mutatnak két példát a költségfüggvény helyes implementálására:

$$U(H_\alpha(z),\gamma) = 1 \qquad \text{,ha } H_\alpha(z) = 0, \tag{4.4}$$

$$U(H_\alpha(z),\gamma) = e^{-i\gamma} \qquad \text{,ha } H_\alpha(z) = 1. \tag{4.5}$$

4.1.3. Keverő réteg

A költségfüggvény implementálásakor észrevehetjük, hogy bizonyos bázisokban a költségréteg nem módosít az adott kvantumbit bázisán mért valószínűségeken, ezért használunk $U(B,\beta)$ forgatástranszformációt.

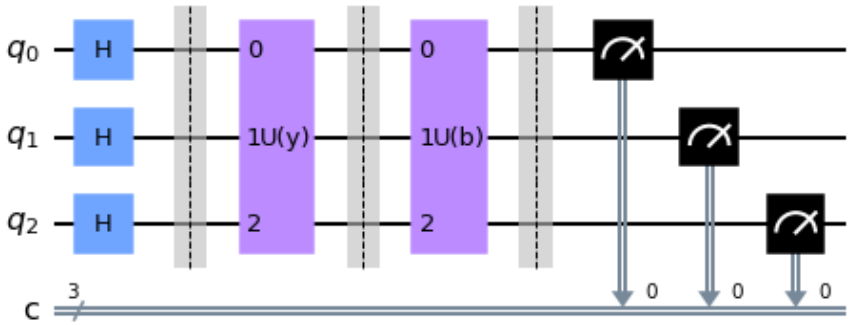
$$B = \sum_{j=1}^{n} \sigma_j^x, \tag{4.6}$$

ahol σ_j egybites operátor.

$$U(B,\beta) = \prod_{j=1}^{n} e^{-i\beta\sigma_j^x}. \tag{4.7}$$

4.1.4. QAOA áramkör

A QAOA áramkör tehát a fenti rétegekből áll 4.4. A rétegek közül a költség és keverő rétegek többször is előfordulhatnak.



4.4. ábra. QAOA áramkör

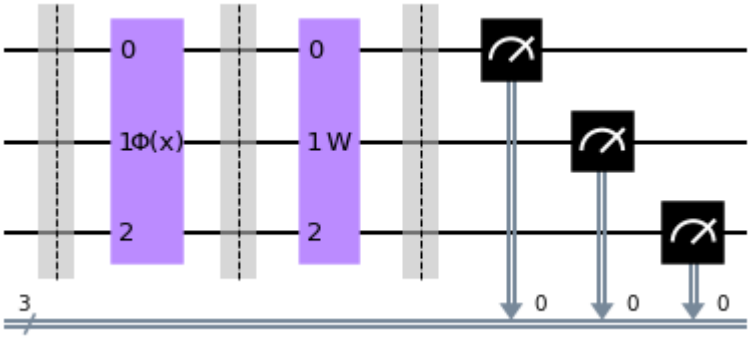
A QAOA megoldásához szükség van tehát egy adott optimaliázációs probléma QUBO vagy Ising-modelljének előállítására, majd ezen modellt optimalizálhatjuk már klasszikus úton a 4.1. ábra szerint.

4.2. Variációs kvantum osztályozás

A variációs kvantum osztályozás négy lépésből áll. Elsőként szükséges leképeznünk az adatokat ($x \in \Omega$) kvantum állapotokká feature terek segítségével ($\Phi(x)$). Ezeket a $\Phi(x)$) leképezéseket meghatározhatjuk különféle eljárásokkal, lényege, hogy előálljanak a különböző kvantumállapotok a klasszikus bitekből és megvalósuljon a $\Omega \longrightarrow H$ leképezés.

Ezután alkalmazhatunk egy k rétegből álló kvantumáramkört $\Theta \in \mathbb{R}^{2^n(1+k))}$, melynek paraméterhalmazát optimalizáljuk a tanítás során. Ezen paraméterhalmaz határozza meg a klasszifikáló hipersík paraméterezését, hiszen a VQC egy lineáris osztályozás lesz [27].

Ezután szükséges a számítási bázisba történő forgatás, végezetül pedig egy címkét társítunk az adatokhoz a mérés folyamatánál. A VQC áramkör a 4.5. ábrán látható.



4.5. ábra. Variációs kvantum áramkör

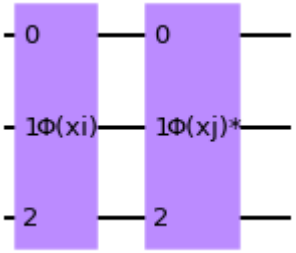
Fontos észrevennünk, hogy a kvantum variációs osztályozás a következő műveleteket valósítja meg:

$$\langle \Phi(x)| W^\dagger Z W |\Phi(x)\rangle . \tag{4.8}$$

, ahol Z a mérési operátor.

4.2.1. Kvantum kernel becslés

A kernel becslés (Quantum Kernel Estimation, QKE) során lecseréljük a fenti hálózatunkat (4.5 a következővel:



4.6. ábra. Klasszifikáció kvantum kernel segítségével

Amennyiben ezen hálózat bemenetére a $|0\rangle$ változót tesszük, akkor $P(|0\rangle)$, azaz a $|0\rangle$ állapot valószínűsége a kimeneten megadható a következőképpen:

$$P(|0\rangle) = \left|\langle 0| \, \Phi(x)\Phi(x)^\dagger \, |0\rangle\right|^2. \tag{4.9}$$

Amennyiben a kernel függvényünk tehát előáll, a 3.10 egyenletből könnyedén meghatározhatjuk α értékeit, amiből már minden adott a klasszifikáláshoz. Ezen kernelfüggvényt pedig a feature terek Hilbert-Schmidt belső szorzatából tudjuk meghatározni [27].

5. fejezet

Kvantum becslés optimalizációs algoritmus vizsgálata

Első feladatként a QAOA-t vizsgáltam meg gráfszínezés problémán. Pusztán annyira voltam kíváncsi, hogy mekkora méretű (azaz hány kvantumbitet igénylő, milyen bonyolultságú probléma oldható meg pontosan az IBM kvantumos elvű számítási egységeit használva. A fejezet első részeként bemutatom a gráfszínezés QUBO modelljét.

5.1. Gráfszínezés QUBO modellje

Gráfszínezés esetén a szomszédos csúcsokhoz különböző színt kell rendelnünk. A K-színezés problémája megpróbálja kiszínezni a gráfot pontosan K színnel. A probléma megoldásához $N \cdot K$ változóra van szükségünk, melyeket x_{ij}-vel fogok jelölni. Ezeket kvantumos esetben kezelhetjük logikai kvantumbitként [16]. Legyen $x_{ij} = 1$ ha a j. szín van társítva az i. csúcs, ellenkező esetben 0. Megkötés a feladat során, hogy minden csúcsot ki kell színeznünk.

$$\sum_{j=1}^{K} x_{i,j} = 1 \quad i = 1, 2..N. \tag{5.1}$$

Ezen megkötés QUBO formája a következő módon írható le:

$$P \sum_{i} (\sum_{j} x_{i,j} - 1)^2 \quad i = 1, 2...N \; j = 1, 2...K. \tag{5.2}$$

Fontos azt a megkötést is beletenni, hogy minden szomszédos csúcs más és más színnel legyen kiszínezve.

$$x_{i,p} + x_{i,p} \leq 1 \quad p = 1, 2..K \text{ és i. és j. csúcsok szomszédosak.} \tag{5.3}$$

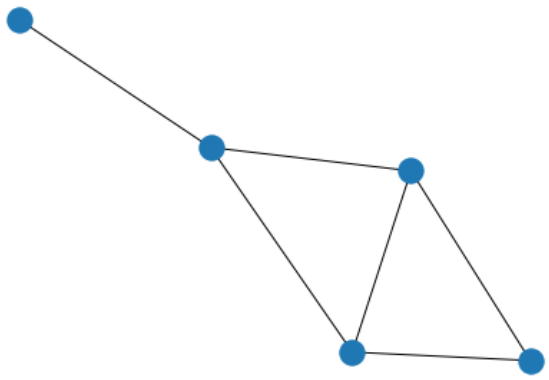
Mely megkötés QUBO formája a következő:

$$P \sum_{i,j} x_i x_j \quad i = 1, 2...N \; j = 1, 2...K. \tag{5.4}$$

5.2. Gráfgenerálás Erdős-Rényi módszerrel

A gráfot mindig véletlenszerűen generáltam Erdős-Rényi gráfként. A modell 1959-ben lett bemutatva Erdős Pál és Rényi Alfréd által. A modellben adott egy fix csúcshalmaz és egy fix valószínűség minden élhez, ami azt mondja meg, hogy az adott él benne van-e a

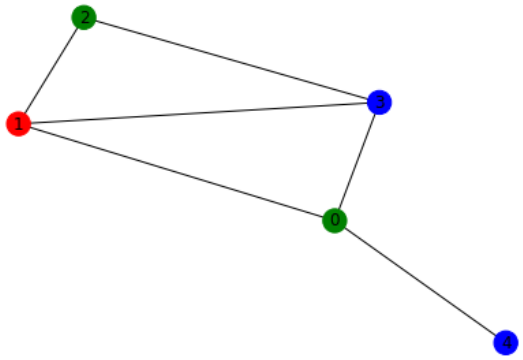
gráfban vagy sem. Az adott élek valószínűsége egymástól független [34]. A megalkotott gráf az alábbi ábrán látható:



5.1. ábra. Generált Erdős-Rényi gráf

5.3. A probléma megoldása klasszikusan

A klasszikus megoldás során a Qiskit egzakt megoldóját használjuk: ennek a lényege, hogy a megalkotott Ising/QUBO modellen kipróbálja az összes lehetséges megoldást, majd a legkisebb energiájút adja vissza. Ez nyilván egy erőforrásigényes feladat, viszont mindig a helyes megoldást szolgáltatja. A megoldás egzakt megoldóval az alábbi ábrán látható:

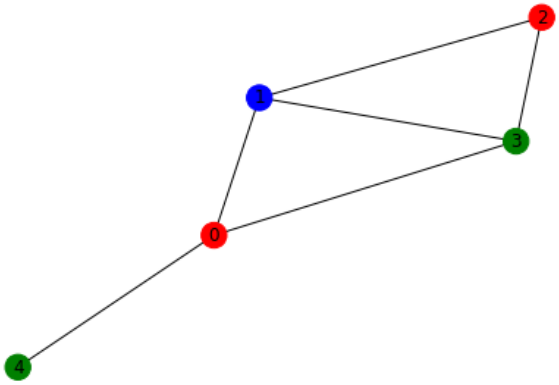


5.2. ábra. Színezés egzakt megoldóval

Tizenöt logikai változónál nagyobb problémát már nem tudtam megoldani klasszikusan, mert túl sok memóriát kellett volna allokálnia.

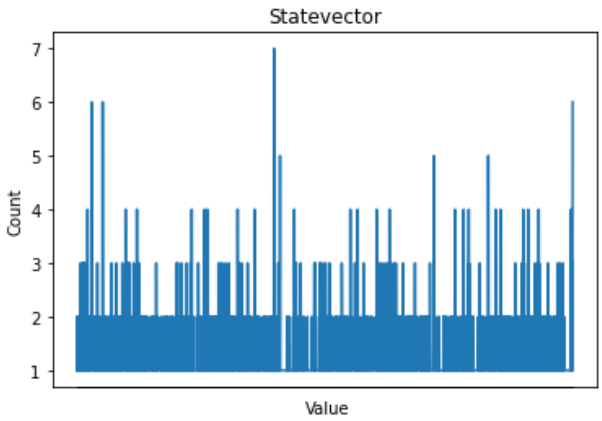
5.4. A probléma megoldása QAOA-vel

A probléma megoldása állapotvektor szimulátorral:



5.3. ábra. Megoldás QAOA segítségével

A kapott megoldást többszöri futtatás, paraméteroptimalizáció segítségével sikerült generálni. A számítást több indításra futtattam, az így generált állapotok:



5.4. ábra. A megoldások 4000 futásra

Látható, hogy a helyes állapot nem sokkal jött ki többször, mint néhány helytelen. A magasabb csúcsokhoz tartozó állapotok nem feltétlenül helytelen megoldások, hiszen egy ilyen problémának sokféle különböző megoldása is lehet.

5.5. Konklúzió a QAOA-vel kapcsolatban

Ami érdekesség volt QAOA esetében, hogy a 4000 futtatást elvégezte kevesebb mint 100 ms alatt, míg az egzakt megoldó néhány tized másodpercig futott. Ami még érdekesség, hogy nehéz volt az optimális megoldást kihozni, emiatt ekkora problémaméretnél még érdemesebb az egzakt megoldót használni.

Nagyobb problémaméret esetén valóban jobb döntés a QAOA algoritmust használni, azonban fontos a paraméteroptimalizáció. Többször is megpróbálkoztam nagyobb problémákat megoldani, azonban kevés sikerrel jártam: ennek az az oka, hogy rendkívül kis méretű szimulátorokat bocsát rendelkezésre az IBM, valamint érdemes azt megjegyezni, hogy nagyon nehéz megfelelő paraméterhalmazt találni hozzá.

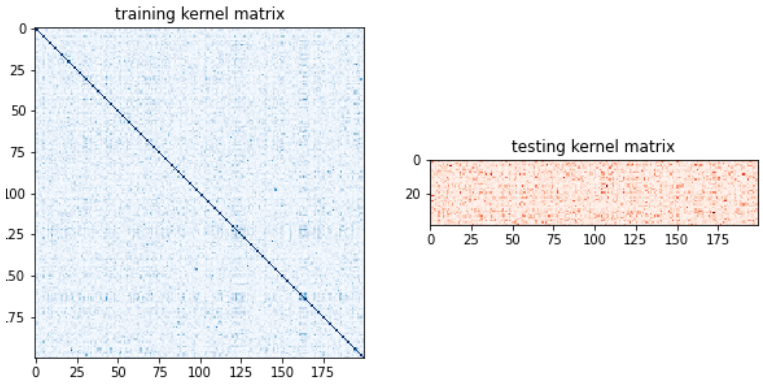
6. fejezet

Kvantum kernel vizsgálata

Ebben a fejezetben szeretném bemutatni a kernel metódusokkal elért eredményeket. Elsőként szerettem volna az MNIST [35] adatbázishoz hasonló, kézzel írt számokon vizsgálatokat végezni, a scikit-learn package használatával. Ezt abból a célból tettem meg, hogy reprodukáljam az IBM eredményeit úgy, hogy kiterjesztem több osztályra az általuk vizsgált, csak 0-ból és 1-ből álló osztályokat, ahol ők $95\% - os$ pontosságot értek el [36]. Ezután a következőleg vizsgált adatbázis esetén 2D-ben elszórt pontokat vizsgáltam, szintén az sklearn segítségével.

6.1. Kézzel írott számok vizsgálata

Az általam vizsgált 9 osztályból álló adathalmazon $22.5\% - os$ pontosságot sikerült elérni a következő kernellel:

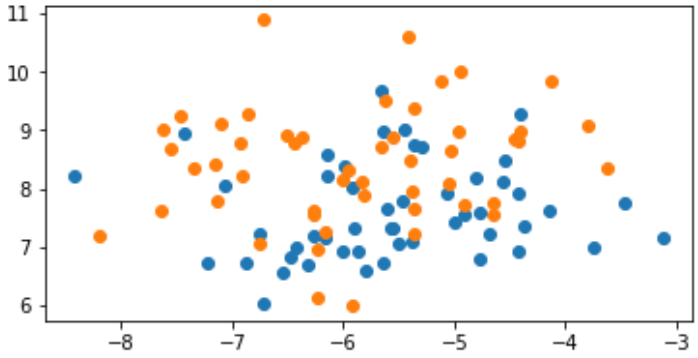


6.1. ábra. Kvantum kernel számfelismerés esetén

Ezt semmilyen optimalizálással nem tudtam növelni, ezért megvizsgáltam, hogy lehetséges-e a többosztályos osztályozás kvantumos módon egy könnyebb adatbázissal.

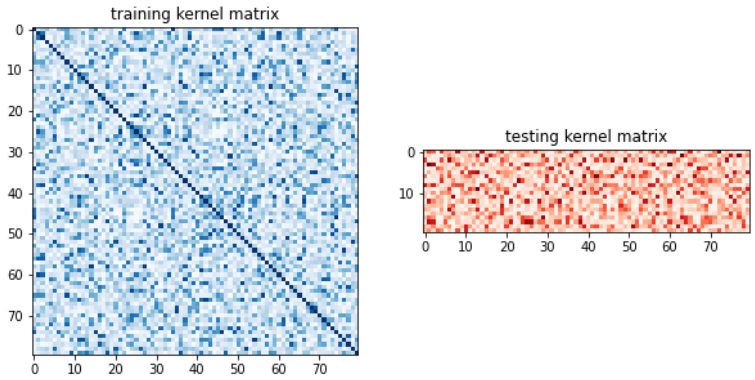
6.2. 2D ponthalmaz vizsgálata

Az első vizsgálatom során csak két osztályt vizsgáltam meg a validáció szempontjából. A vizsgált adatbázisom a következő volt:



6.2. ábra. 2D ponthalmaz vizsgálata

Ezen az adatbázison a kapott kernel függvényem a következőre adódott:



6.3. ábra. Kernel 2D ponthalmaz vizsgálata esetén

Érdekesség, hogy a kiélesedő csík minden szimuláció esetén ugyanott volt, azonban itt 75%-os pontosságot sikerült elérni, ami viszonylag elfogadható eredmény. Ellenben, ezekután megpróbáltam itt is kiterjeszteni az eredményemet több, jelen esetben négy osztályra. Az így kapott kernellel 35%-os pontosságot sikerült elérni, ami szintén nem megfelelő eredmény, így arra a következtetésre jutottam, hogy ez egyelőre csak két osztályra megvalósítható.

6.3. Összegzés

A vizsgálat eredményét több különböző paraméterhalmaz, futtatás, illetve a fentebb leírt adatbázisok variálása sem javította. Valószínűleg a többosztályos kiegészítés nem működik megfelelően, hiszen ott volt hatalmas visszaesés a teszteknél. Érdekesség az is, hogy folyamatosan ugyanazt a kernelmátrixot volt képes meghatározni tanulás esetén. Ez valószínűleg amiatt van, hogy az állapotvektor szimulálásánál nem adódnak egyértelműen az amplitúdók. Ezt a hibát kvantumzaj, illetve a feladat bonyolultsága is okozhatja. A kernelkiszámítás azonban néhány ms-ot igényelt csak. Ez mindenképpen előny, azonban az eredmények néhol kiábrándítóak voltak.

7. fejezet

Kép- és jelfeldolgozás kvantum konvolúciós hálóval

Jelen fejezetben szeretném bemutatni a klasszikus és kvantumos konvolúciós neurális hálózatokkal végzett vizsgálatokat. Elsőként a különböző kvantumos rétegekről, illetve azok implementációjáról olvashatunk, a második részben pedig az elért eredményeket fogom bemutatni.

7.1. Kvantum rétegek

Ez a rész prezentálja az adott kvantumos rétegek felépítését, illetve implementációs megközelítéseit.

7.1.1. Kvantum konvolúciós réteg

A kvantum konvolúciós réteg lényegében ugyanazt valósítja meg, mint a klasszikus konvolúciós réteg, a működési elve azonban teljesen más. Az adott képen kijelöl egy NxN- es szegmenst, melyet egyszerre képes feldolgozni, azonban nem mátrixszorzásokat használ, hanem egy kvantumáramkört alkalmaz az adott szegmens értékein. Elsőként ezeket a biteket át kell kódolnunk kvantumbitekké egy adott függvény segítségével (ezt a részt korábban már megismerhettük a 4. fejezetben), majd alkalmazzuk a megadott kvantumáramkört. Végül az adott kvantumáramkör kimenetét mérjük, ezzel klasszikus biteket kapva. Ezt az N^2 kvantumbittel rendelkező unitér transzformációt kell a megfelelő módon optimalizálni. Fontos megjegyezni, hogy ez jóval gyorsabb működésre képes, mint az $O(N^2)$-es mátrixszorzás klasszikus konvolúciós háló esetén, hiszen egy adott időpontban tudja végezni a transzformációkat [37].

7.1.2. Kvantum osztályozó

Az elméleti részben megismert variációs kvantum osztályozás gyakorlatban a következőképpen néz ki: lényegében egy olyan kvantumáramkört kell megalkotnunk, amely a bemeneti paraméterek összes lehetséges állapotát előállítja, majd egy θ paraméterrel jellemezhető transzformációt hajt végre. Végül egy kimeneti méréssel meghatározhatjuk az adott állapotok valószínűségét. Többosztályos osztályozásnál ezt használjuk fel a kimeneti címke meghatározására. Nyilvánvalóan ez azt jelenti, hogy nekünk egy $\Theta = (\theta_1, \theta_2 ... \theta_N)$ paraméterhalmazunk lesz, melyet optimalizálnunk kell.

7.2. MNIST adatbázison elért eredmények

Elsőként az MNIST [35] adatbázissal kezdtem a feldolgozást, ami kézzel írott számokat tartalmaz. Ez egy relatíve egyszerű adatbázis, a célom ezzel kizárólag a neurális hálózatok tesztelése volt, olyan szempontból, hogy sikerül-e egyáltalán konvergálást elérni vele mind hiba, mind pontosság szempontjából, ugyanis az eljárás annyira újszerű, hogy az is jelentős valószínűséggel következhet be, hogy nem megfelelő az alkalmazása ilyen célokra. Sok más kutató, köztük Yann Le Cun, a konvolúciós neurális hálózat megalkotója [38] is ezen kísérletezett. A validáláshoz tehát ezen adatbázist használtam.

7.1. ábra. MNIST adatbázis

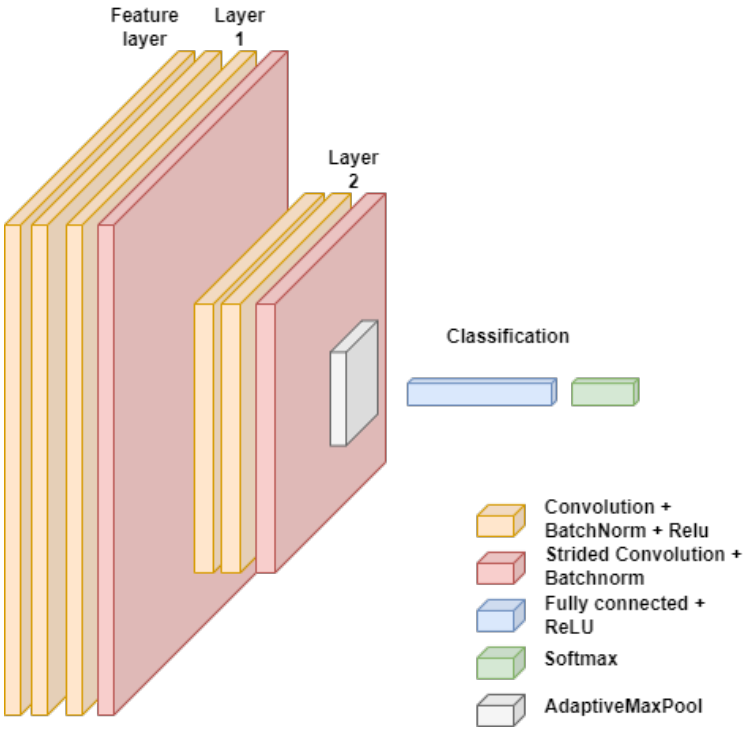
7.2.1. Klasszikus konvolúciós neurális hálózat

Elsőként egy parametrizálható konvolóciós neurális hálózattal kezdtem a feldolgozást. A neurális hálózat paraméterei a következők: réteg, szint, kernel méret, feature vektorok száma és csatornaszám. Az optimalizáláshoz Bayes-optimalizációt hajtottam végre, melynek során nem véletlenszerűen választjuk a hiperparamétereket, hanem a következő paraméterhalmazt az adott kimeneti érték (esetemben tanítási pontosság) követésével találja meg [39]. Fontos megjegyezni, hogy csak egyszer, konvolúciós neurális háló optimalizálásánál futtattam le ezt, utána minden hibrid megoldásnál hasonló hálót használtam, annak érdekében, hogy a lehető legjobban ki tudjam hangsúlyozni a különbségeket az eljárások között. A továbbiakban tehát a következő paramétereket használtam:

szintek száma	rétegek száma	kernel méret	feature vektorok száma
2	2	3	8

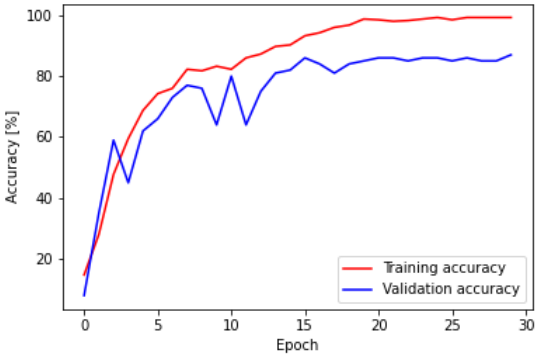
7.1. táblázat. Modellparaméterek összefoglalása

A paraméterek közül a szintek és a rétegek száma nem egyértelmű: a rétegek száma egy adott képméret vizsgálatához használt konvolúciós rétegek számát, a szintek száma pedig az ilyen rétegek számát jelöli. A tanítási paramétereket (tanulási ráta, súlyzaj, tanítási ütemező lineáris felosztással) változatlanul hagytam az eljárások között. Az alkalmazott neurális hálózatom az alábbi képen látható:

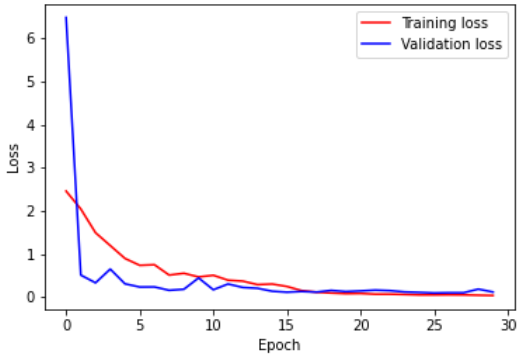


7.2. ábra. Klasszikus konvolúciós hálózat

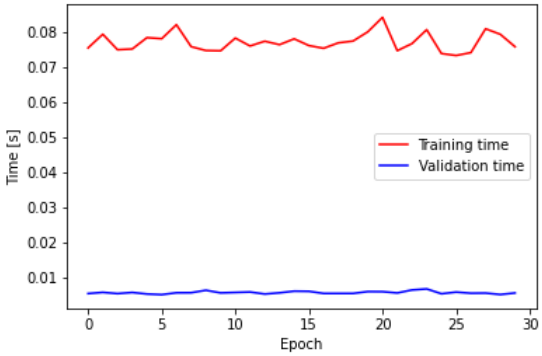
Az adott háló esetén a következő eredmények adódtak:



7.3. ábra. Tanítási és validációs pontosság



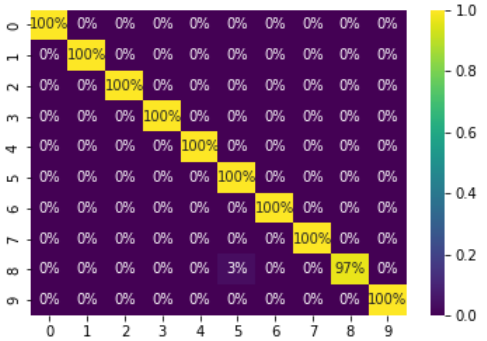
7.4. ábra. Tanítási és validációs hiba



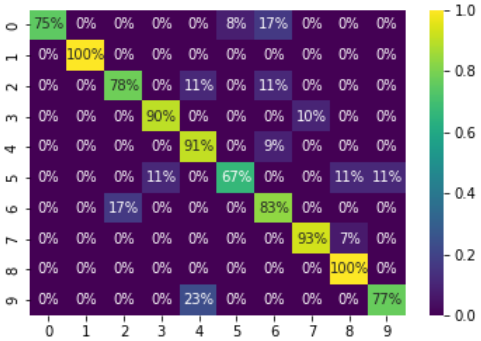
7.5. ábra. Tanításhoz és validációhoz szükséges idő

A pontosság és a hiba megjelenítésével láthatjuk, hogy a hálónk konvergál a majdnem tökéletes pontosság felé. Fontos megjegyezni, hogy ehhez nagyon kevés epoch-ra volt szükség, illetve a szükséges idő is nagyon kevés volt. A vizsgálat előtt ezt is vártuk, hiszen egy viszonylag egyszerű adatbázisról van szó.

Az alábbi ábrákon megfigyelhetjük a tanítási és validációs konfúziós mátrixokat. Látható, hogy a főátlóban vannak a leggyakoribban az elemek, ami azt jelenti, hogy a predikált és valódi címkeértékek megegyeznek.



7.6. ábra. Tanítási konfúziós mátrix



7.7. ábra. Validációs konfúziós mátrix

Az eredmények javítása érdekében további augmentációt, illetve tanítást hajthattam volna végre, azonban ahogy már fentebb említettem, ebben az esetben az összehasonlítás és a validálás a cél, nem pedig a feladat tökéletes megoldása.

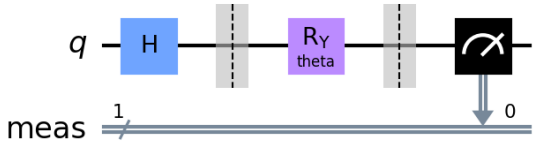
7.2.2. Hibrid konvolúciós neurális hálózat

Ebben a részben több klasszikus és kvantumos neurális hálózat elem együttes használatát vizsgálom.

7.2.2.1. Klasszikus konvolúciós hálózat kvantumos elvű klasszifikálóval

Ebben az eljárásban lecseréltem a klasszikus teljesen összekötött klasszifikáló réteget egy kvantum-áramkörre. Ehhez szükség volt egy kvantumos áramkör létrehozására, ahhoz az előre és hátraterjesztés implementálására, majd ezt az egészet összekapcsolni a jelenlegi neurális hálózattal.

Elsőként tehát megalkottam a kvantumáramkört, mely az osztályozásért lesz felelős. Az implementálások során nagy segítségemre voltak a Qiskit által kiadott, illetve a Qiskit versenyén használt példákódok [40, 41, 42]. Az alábbi kvantumáramkört használom tehát osztályozóként:



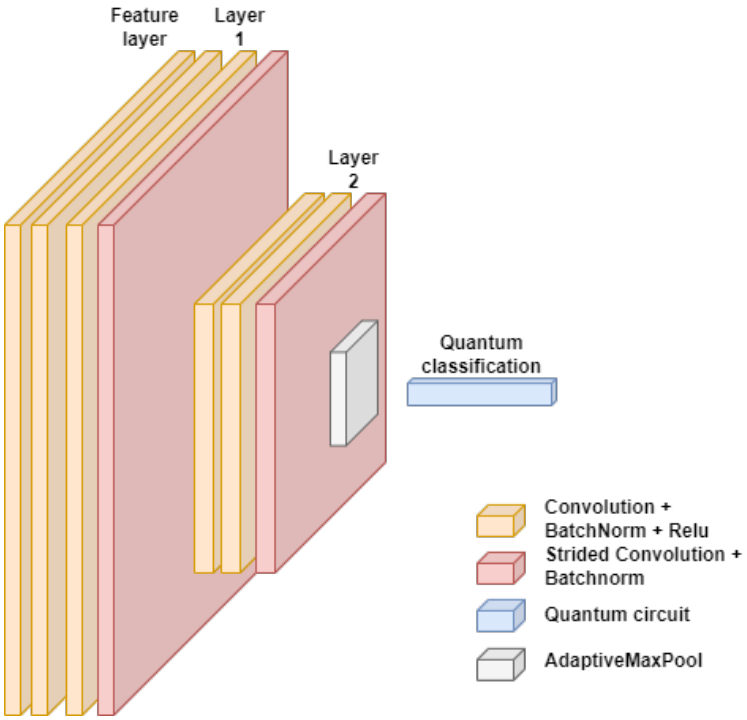
7.8. ábra. Klasszifikáló kvantumáramkört

Mivel többosztályos osztályozást hajtok végre, emiatt ezen kvantumáramkört meg kell ismételni annyiszor, ahány kimeneti csatornát szeretnék előállítani (pontosabban minden jelen lévő csatorna végére kell egy áramkör, de ezt egy lineáris osztályozó réteggel egyenlővé teszem).

Az előterjesztés során az adott kvantumáramkört kell alkalmazni, a visszaterjesztés azonban nem ennyire egyértelmű: szükséges a két egymást követő érték előállítása, majd ezek differenciájának képzése:

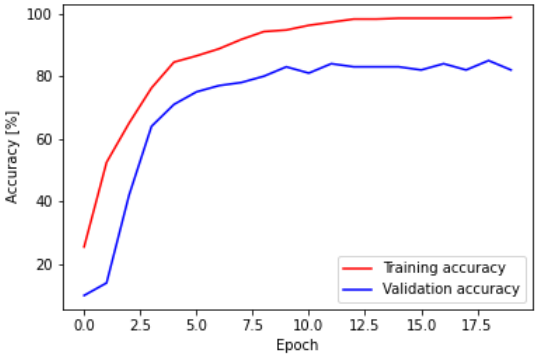
$$grad(\theta) = sQ(\theta) - sQ(\theta), \tag{7.1}$$

ahol s egy infinitezimálisan kicsi lépést, Q pedig a kvantumáramkör függvénnyel történő helyettesítést jelenti. Ezután gradiens visszaterjesztéssel optimalizálhatjuk a hálót. A neurális hálózatom többi részét változatlanul hagytam.

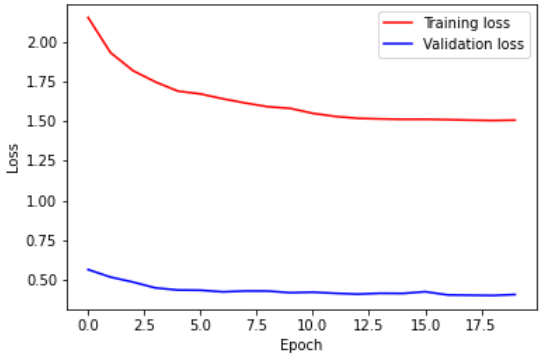


7.9. ábra. Klasszikus konvolúciós neurális hálózat kvantumos osztályozóval

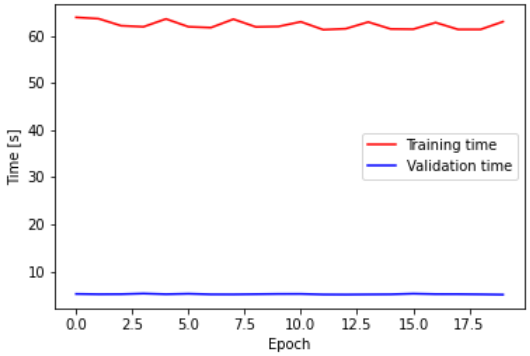
Az így elért tanítási és validálási eredményeket az alábbi ábrákon foglaltam össze:



7.10. ábra. Tanítási és validációs pontosság

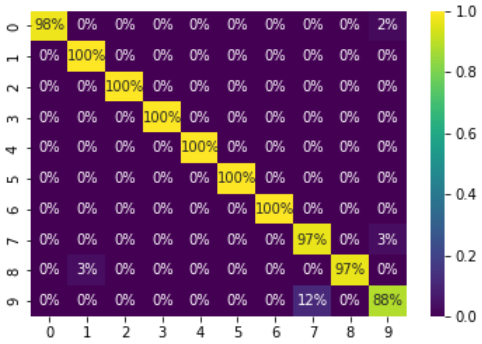


7.11. ábra. Tanítási és validációs hiba

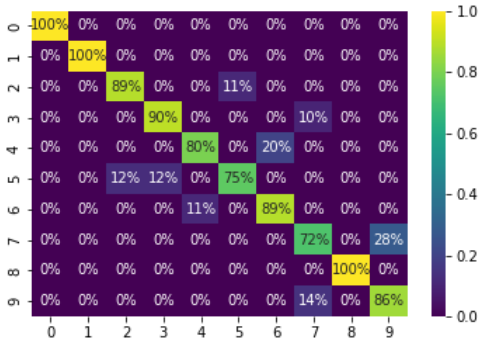


7.12. ábra. Tanításhoz és validációhoz szükséges idő

A tanítási pontosságon és a hibaértékeken látszik, hogy a háló rendkívül gyorsan konvergált és nagy pontosságot lehetett vele elérni. Látható, hogy a tanítási idő rendkívül megnőtt az előző esethez képest: ennek az oka az, hogy szimulátort használok a teszteléshez, mely rendkívül lassú a kvantumprocesszoros futtatáshoz képest, azonban így lokálisan is lehetséges volt a tanítás. A másik megoldás további hátránya az, hogy az elérési idő még ennél is többre adódik, emiatt érdemesebb az általam is preferált szimulátort használni. Ami érdekes, hogy a hiba a tanítás során nagyobbra adódott, mint a validáció során, hiába pontosabb.



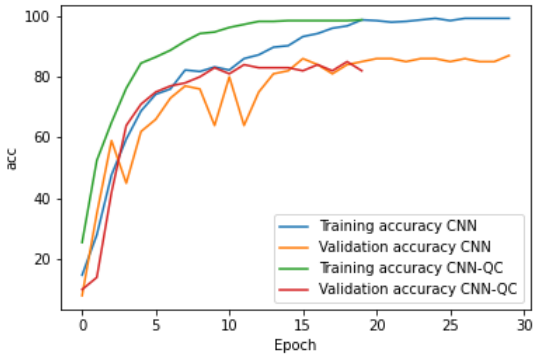
7.13. ábra. Tanítási konfúziós mátrix



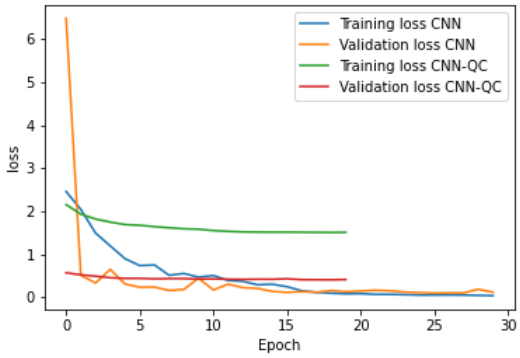
7.14. ábra. Validációs konfúziós mátrix

A konfúziós mátrixokon látható, hogy valóban jól működött a kvantum klasszifikáló, ugyanis itt is szinte csak a főátló tartalmaz elemek.

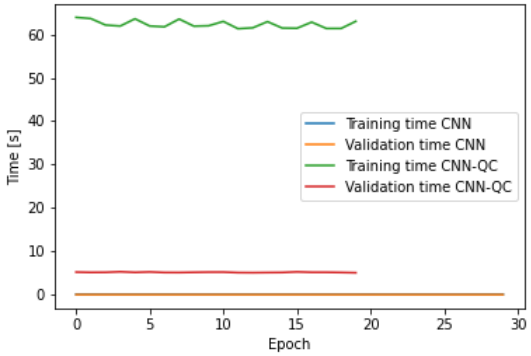
Az alábbi ábrákon összefoglaltam a különbségeket a klasszikus megoldással.



7.15. ábra. Pontosság összehasonlítása



7.16. ábra. Hibák összehasonlítása



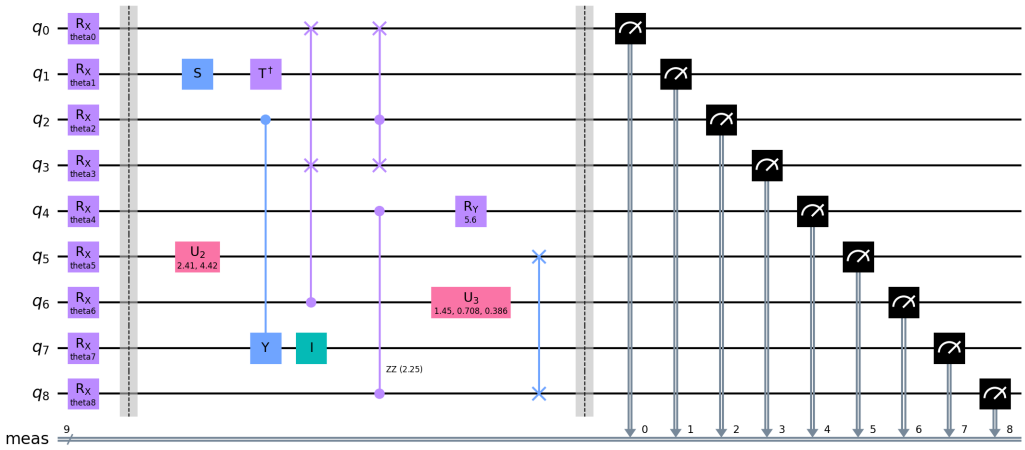
7.17. ábra. Szükséges idők összehasonlítása

Látható, hogy a kvantumos elvű klasszifikálóval vizsgált eredmények előbb kezdtek konvergálni, azonban a hibában valamiért nagyobbra adódtak a tanítás során. Az időket összehasonlítva a fentebb leírtak érvényesülnek, azaz jóval lassabban tudunk kvantumo-

san szimulálni. Ez a helyzet a kvantumprocesszorok elterjedésével javulna, ahogyan az a kvantumos elvű kernelbecslésnél is látszódott (néhány ms alatt képes volt meghatározni a kernelt, míg klasszikusan ez több időt venne igénybe).

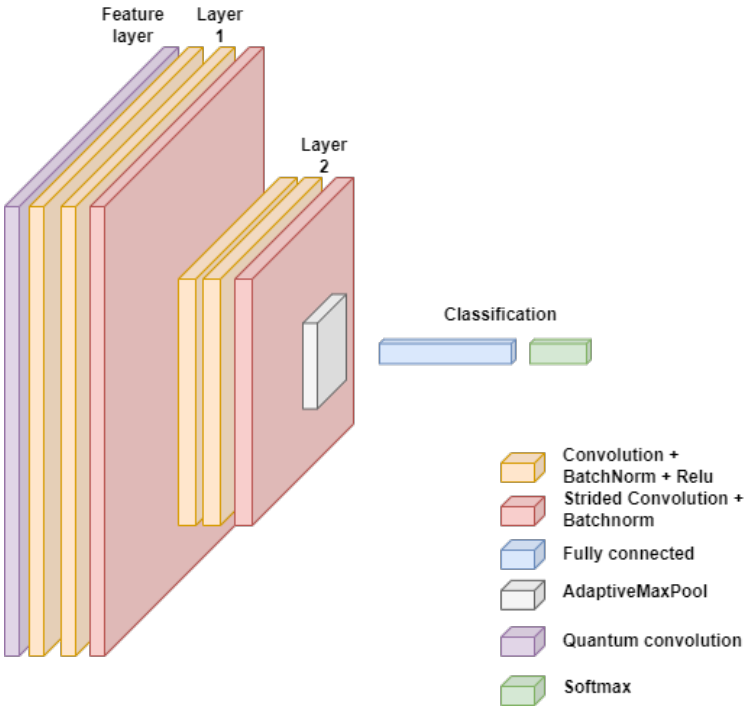
7.2.2.2. Klasszikus-kvantumos konvolúciós hálózat

Elsőként létre kell hozni egy véletlen paraméterhalmazzal rendelkező kvantumáramkört, mely N^2 kvantumbittel rendelkezik, ahol N a kernel mérete. A többi részben annyiban különbözik a klasszifikálótól, hogy ebben az esetben a paraméterenkódolást úgy végeztem, hogy egy adott küszöbszintnél nagyobb paraméterekhez π, míg fordított esetben 0 forgatást társítottam. Az implementáció során nagy segítségemre voltak az alábbi leírások: [41, 42, 43]



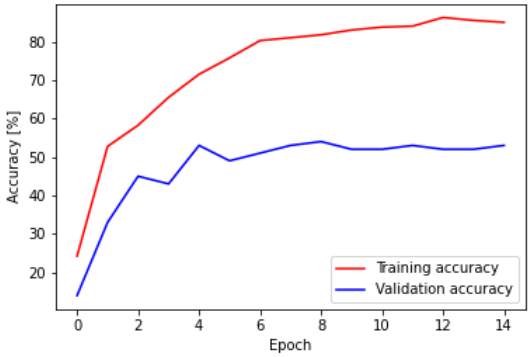
7.18. ábra. "Quanvolúciós" áramkör

A vizsgálat során ugyanazt a korábban megismert hálót alkalmaztam, annyi különbséggel, hogy a feature vektorok kialakításához a fenti konvolúciós réteget használtam. Ezen háló volt a konvolúciós neurális hálóm bemenete.

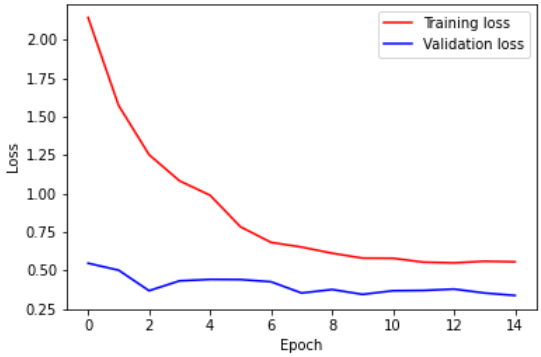


7.19. ábra. Klasszikus-kvantumos konvolúciós neurális hálózat

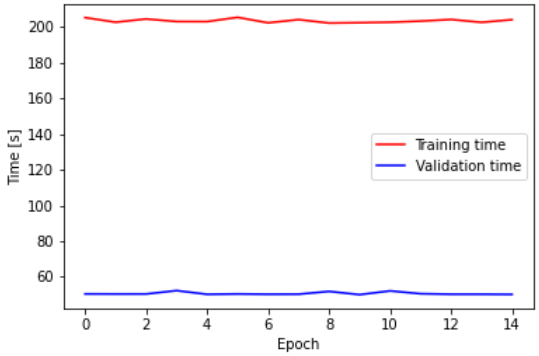
Ebben az esetben is az MNIST adatbázissal kezdtem tehát a vizsgálatot, melyre a következő eredmények adódtak:



7.20. ábra. Pontosság

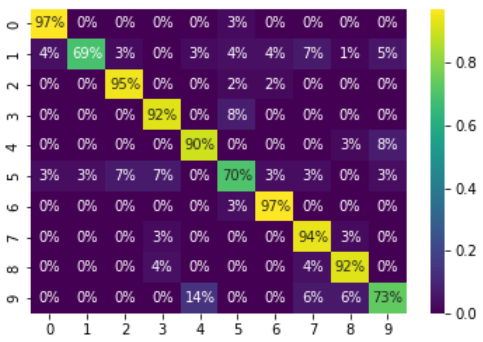


7.21. ábra. Hibák

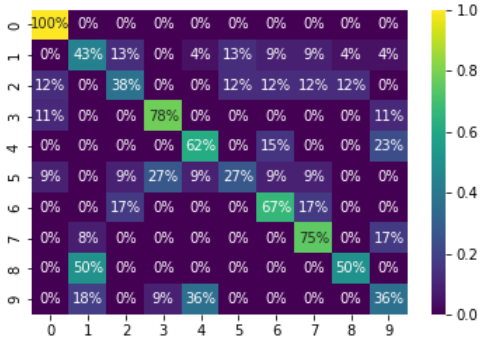


7.22. ábra. Szükséges idők

Itt is megfigyelhető, hogy a háló konvergált, azonban sem tanítási, sem validálási pontosságban nem sikerült az előzőekhez hasonló eredményeket elérni. Ez hosszabb futtatásokra sem volt jellemző, viszont a tanítási és validálási idők nagyon nagyra adódtak (200 ms, illetve 50 ms).



7.23. ábra. Tanítási konfúziós mátrix

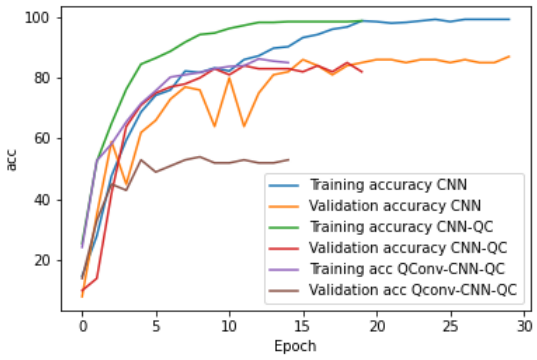


7.24. ábra. Validációs konfúziós mátrix

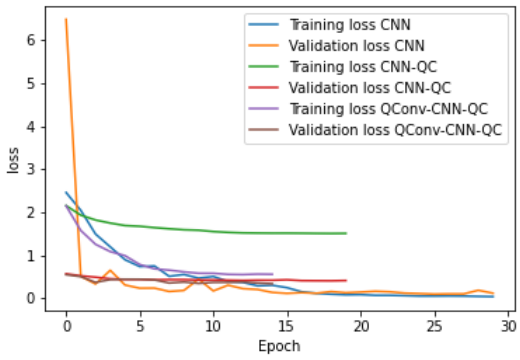
A konfúziós mátrixokon megfigyelhetjük, hogy a tanítást szinte hibátlanul meg tudta tenni, azonban a validálásnál már hibásan működött a háló. Próbáltam dropout rétegekkel,

illetve különféle normálizálással, hálócsökkentéssel redukálni a túltanulást, de nem jártam sikerrel.

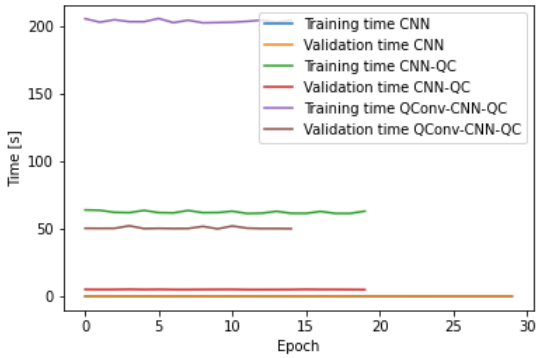
Összehasonlítva a korábbi eljárásokkal látható, hogy jóval kisebb pontosságot és jóval nagyobb túltanulást értünk el ezzel a hálózattal. Ezt semmilyen optimalizálással sem sikerült javítani. A legnagyobb problémát azonban a tanulás és validálás sebessége okozta: ez annak tudható be, hogy egy $3x3$-as kernel lefuttatása 9 kvantumbites áramkört igényel, mely szimulátoron rendkívül időigényes.



7.25. ábra. Pontosságok összehasonlítása



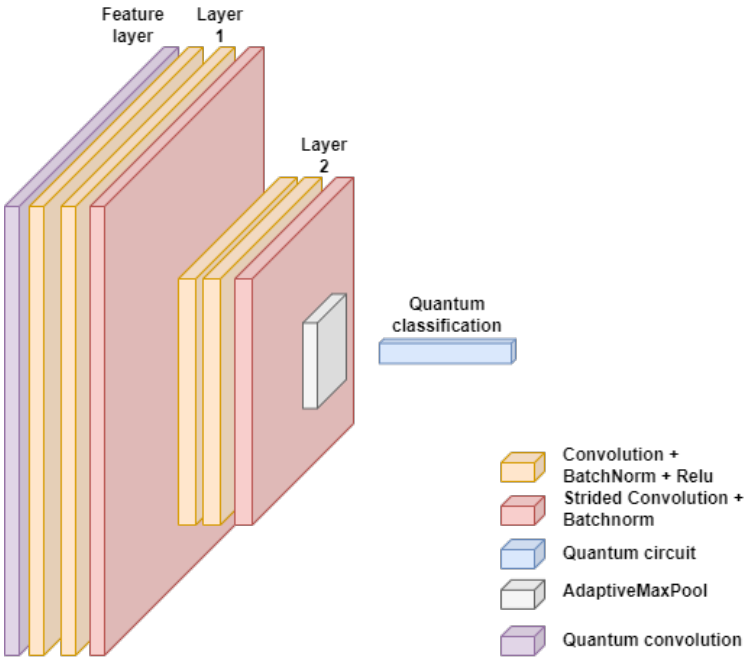
7.26. ábra. Hibák összehasonlítása



7.27. ábra. Szükséges idők összehasonlítása

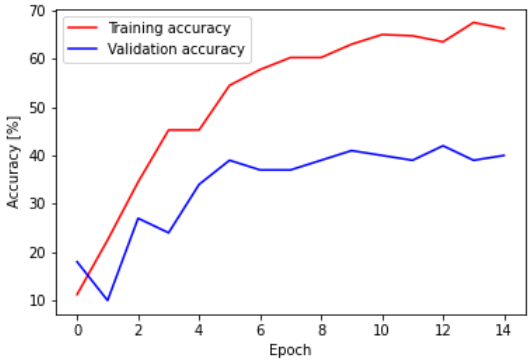
7.2.2.3. Klasszikus-kvantumos konvolúciós hálózat kvantumos klasszifikálóval

Végül megpróbáltam együtt használni a kvantumos konvolúciós kvantum konvolúciós réteget, illetve a kvantum klasszifikáló áramkört. Eszerint a hálóm az alábbi módon alakult.

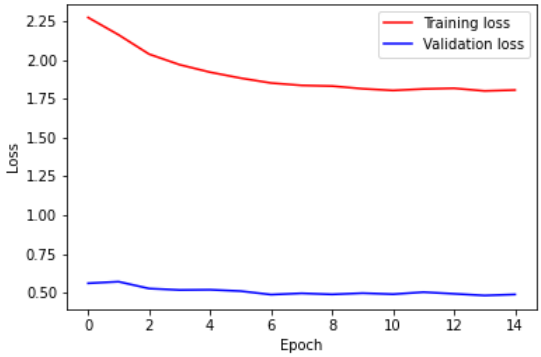


7.28. ábra. Klasszikus-kvantumos konvolúciós neurális hálózat kvantumos osztályozó réteggel

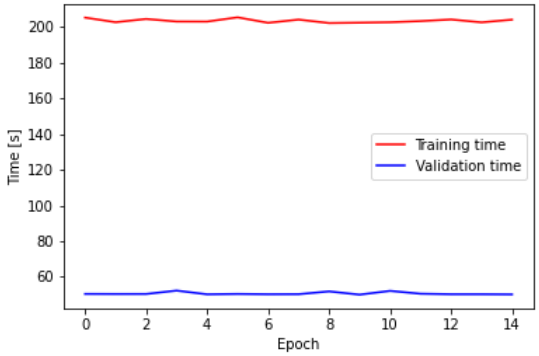
A következő ábrákon megfigyelhető a konvergencia, azonban nagyon leesett a tanítási, illetve a validációs pontosság is. Az is megfigyelhető, hogy a tanítás és a validáció még az előzőeknél is jóval több időt igényelt.



7.29. ábra. Pontosság

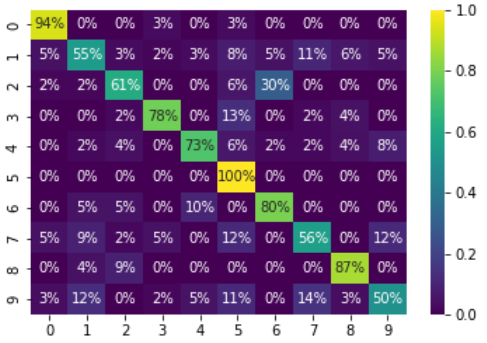


7.30. ábra. Hiba

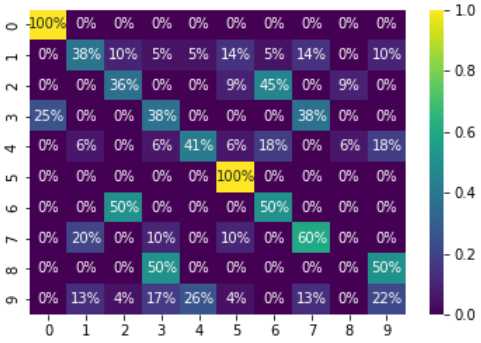


7.31. ábra. Szükséges idő

A tanítási konfúziós mátrixon megfigyelhetjük, hogy még mindig a főátlóban vannak a leggyakrabban elemek, azonban már jóval több helyen is megjelennek jelentős százalékkal predikciók. A validációs konfúziós mátrix ellenben már nem mutat pontosságot.



7.32. ábra. Tanítási konfúziós mátrix



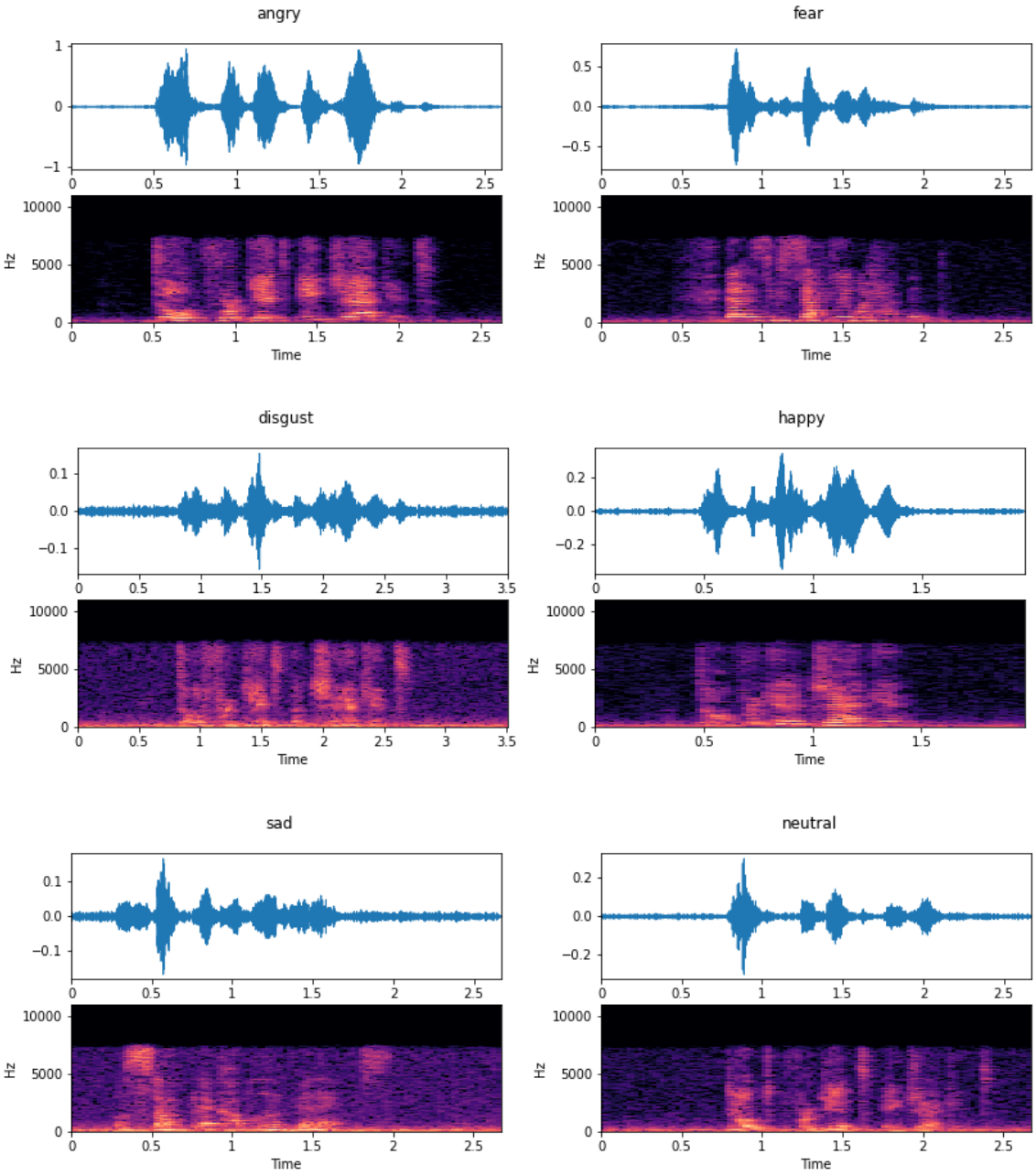
7.33. ábra. Validációs konfúziós mátrix

7.2.2.4. Összegzés

A fentiekből talán a legkézenfekvőbb megállapítás az, hogy ameddig nem áll rendelkezésre számunkra valódi QPU, addig nem érdemes kvantumos hálózatot alkotni, ugyanis hiába futtattam a szimulátorokat GPU-n, a tanítás és a validáció még így többszöröse (legrosszabb esetben 2500-szorosa) a klasszikus eljárásnál tapasztaltnak. A kvantumos konvolúciós layer alkalmazása emiatt csak bemeneti rétegként volt releváns, azonban a pontosság és hibaértékek miatt a későbbi vizsgálatok során nem alkalmazom, mert amellett, hogy kisebb pontosságot értem el vele, a paraméterkészletre is jóval érzékenyebb. Ellenben a kvantumos klasszifikáló réteg alkalmazása bíztató eredményekkel szolgált, ugyanis kevesebb epoch is elég volt ugyanakkora pontosság eléréséhez. A jelfeldolgozást tehát ezen két háló segítségével fogom megvalósítani.

7.2.3. Crema-D adatbázis [1]

A CREMA-D adatbázis egy 7442 rövid hangfájlból álló adatbázis, mely hangok 91 embertől származnak. Ezen hangfájlok különböző érzelemhez csatolhatóak, melyek a düh, a félelem, az undor, a boldogság, a szomorúság, illetve a semleges érzés [1]. Az alábbi ábrán szemléltetem az érzelmeket és a hozzájuk tartozó mel spektogrammot.



7.34. ábra. A különböző érzések időtartománybeli jelének és mel spektrogramjának szemléltetése

7.2.3.1. Augmentáció

A feldolgozás során szeretném a legtöbb információt átadni a hálóknak a jelekről, emiatt adataugmentációt végzek. Az augmentáció során az egydimenziós adatsorból háromcsatornás képeket készítek, azonban a módszer leírásához először néhány fogalmat mutatok be.

Elsőként STFT(**S**hort-**T**ime **F**ourier **T**ransorm)-t hajtok végre a jelen, melynek segítségével időtartománybeli analízis helyett frekvenciatartománybeli analízist tudok vég-

rehajtani. Az STFT képlete a következőképpen adható meg [44]:

$$X(m,\omega) = \sum_n x(n)w(n-m)e^{-j\omega n}, \tag{7.2}$$

ahol ω a frekvencia, w pedig az alkalmazott ablak. Az így kapott amplitúdók értékének abszolútérték-négyzetéből megkaphatjuk a spektogrammot.

$$S(m,\omega) = |X(m,\omega)|^2, \tag{7.3}$$

Az emberek nem lineáris skálán érzékelik a frekvenciákat, jobban tudják észlelni az alacsonyabb frekvenciákon lévő különbségeket, mint a magasabbakon lévőt. Emiatt 1937-ben Stevens, Volkmann és Newmann egy olyan hangmagasság-mértékegységet javasoltak, hogy az egyenlő hangmagasság távolságok ugyanolyan távoliak legyenek a hallgató számára. Ezt hívjuk **mel**-skálának hívják[45]. Az STFT-vel megalkotott spektogrammunkról könnyedén áttérhetünk a következő módon[46]:

$$m = 2595 \cdot \log_{10}(1 + f/700) \tag{7.4}$$

Az augmentáció során a 3 csatornás képek lényegében ezen mel-spektogrammokat tartalmazzák. Az első csatorna megalkotásakor a nyers adatokból a fent leírt módon meghatározzuk a mel-spektogrammot. A második és harmadik csatorna szintén hasonló módon alakul, annyi különbséggel, hogy egyik esetben fehér zajjal terheljük a mintákat, majd azokból alkotjuk meg a mel-spektogrammot, míg másik esetben a hangmagasságot változtattam és így határoztam meg a mel-spektogrammot, így kapva tehát háromcsatornás képeket.

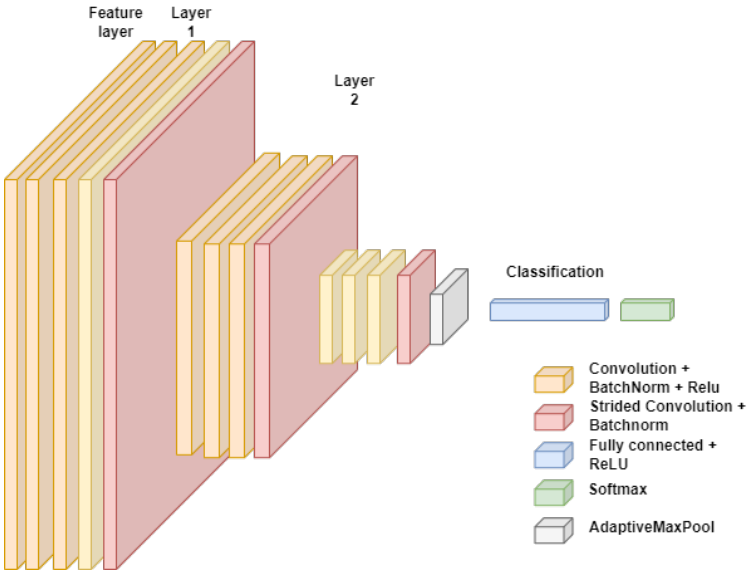
7.2.4. Klasszikus konvolúciós neurális hálózat

Elsőként a fentebb már megismert klasszikus konvolúciós neurális hálózatomat vizsgáltam meg itt is a referenciaeredmények érdekében. Később ezen neurális hálózatot módosítottam. A modellparaméterek a következőképpen módosultak:

szintek száma	rétegek száma	kernel méret	feature vektorok száma
3	3	3	8

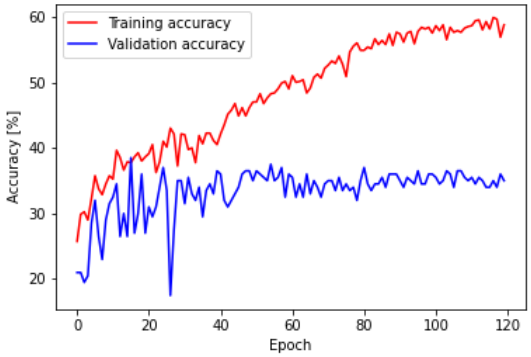
7.2. táblázat. Modellparaméterek összefoglalása

Látható, hogy ebben az esetben egy jóval nagyobb konvolúciós hálóval dolgoztam, mely több optimalizálandó paraméterrel rendelkezik. Erre azért volt szükség, mert a feladat jóval bonyolultabb, mint a korábban megismert MNIST adatbázisnál volt.

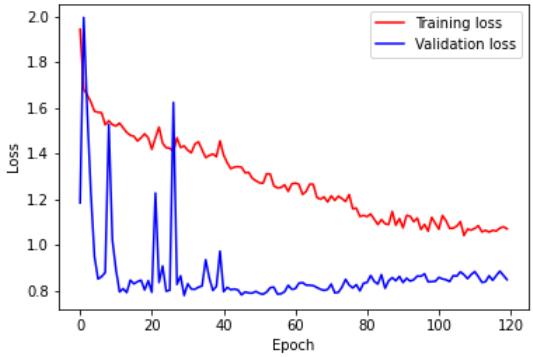


7.35. ábra. Klasszikus konvolúciós hálózat Crema adatbázis esetén

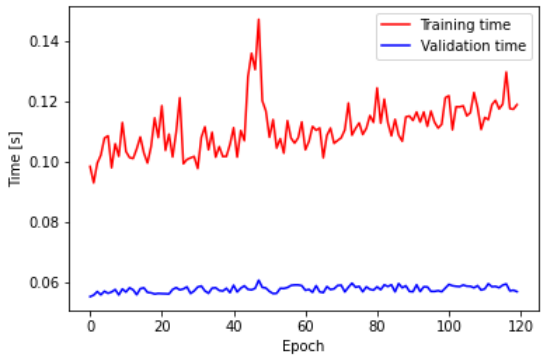
Korábbi munkák ezen adatbázison ezzel az eljárással 60% körüli pontosságot értek el hatosztályos osztályozás esetén [47]. Több hasonló cikk csak néhány osztályt választott ki a hat elérhető osztály közül, így közel 100%-os eredményt elérve [48]. Ezek a megoldások általában egy túl nagy konvolúciós neurális hálózatot alkalmaztak, melyeket én elérhető erőforrások hiányában nem tudtam megvizsgálni. Néhány publikációban egydimenziós konvolúciós hálózatot alkalmaztak arra, hogy ezt a feladatot tökéletesen meg tudják oldani [49]. Mind a hat osztályt megvizsgáltam a korábban megalkotott 2D-s konvolúciós neurális hálózatommal. Az eredményeket a következő ábrákon foglaltam össze.



7.36. ábra. Pontosság

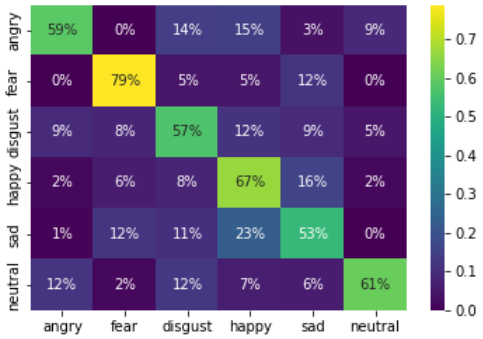


7.37. ábra. Hiba

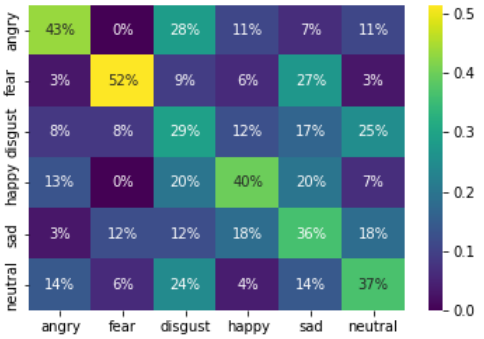


7.38. ábra. Szükséges idő

Látható, hogy 120 epoch-ot tanítva is 60%-ra adódott a tanítási, 40%-ra a validálási pontosság. A hálót továbbra is az eddig megismert optimalizálási paraméterekkel tanítottam. Megfigyelhető, hogy a tanítás egyes epoch-okra nem igényelt sokkal több időt, mint MNIST adatbázis esetén.



7.39. ábra. Tanítási konfúziós mátrix

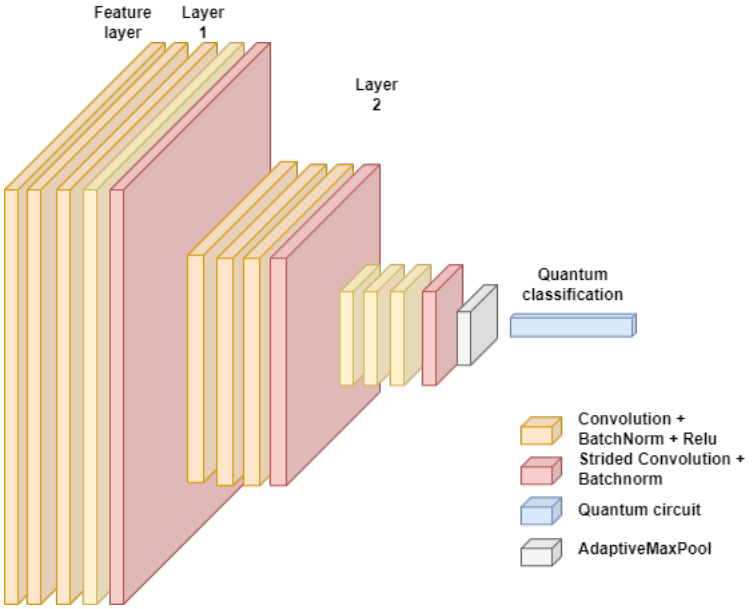


7.40. ábra. Validációs konfúziós mátrix

A konfúziós mátrixokon megfigyelhető, hogy a főátlóban vannak a leggyakoribban elemek, azonban azok gyakorisága meg sem közelíti a 100%-ot. A validációs konfúziós mátrixok jóval pontatlanabb értékeket mutatnak.

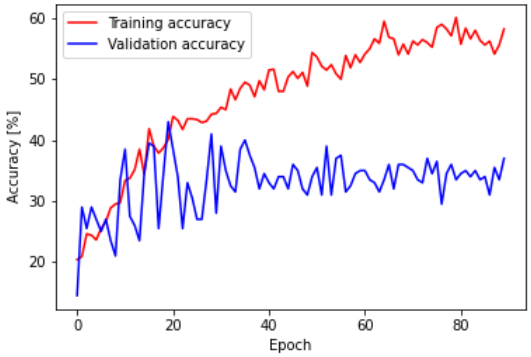
7.2.5. Klasszikus konvolúciós neurális hálózat kvantumos osztályozó réteggel

Ezen részben szintén a korábban megismert klasszikus konvolúciós neurális hálózatomat használtam fel, kvantumos osztályozóval.

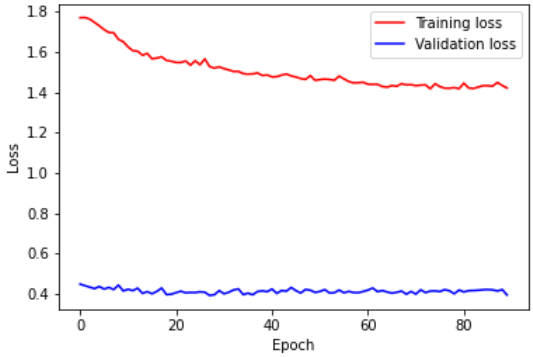


7.41. ábra. Klasszikus konvolúciós hálózat kvantumos osztályozóval Crema adatbázis esetén

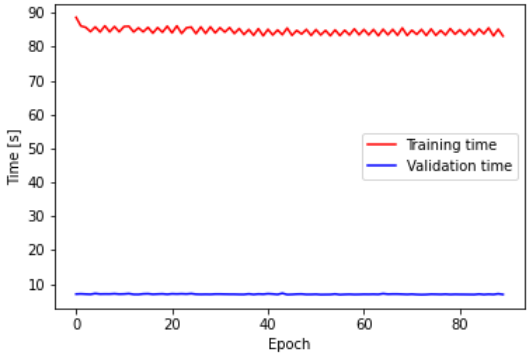
Az ezzel a hálóval produkált tanítási és validálási eredmények ugyanazok, mint amit a klasszikus osztályozóval sikerült produkálni, az ezekhez szükséges idő azonban jelentősen megnőtt.



7.42. ábra. Pontosság

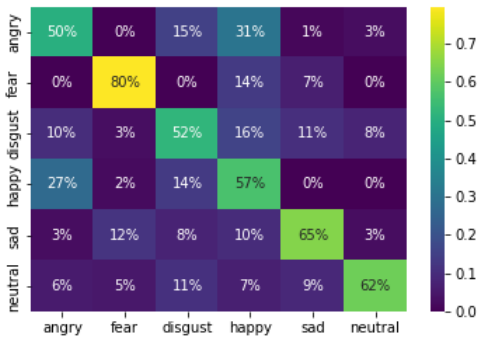


7.43. ábra. Hiba

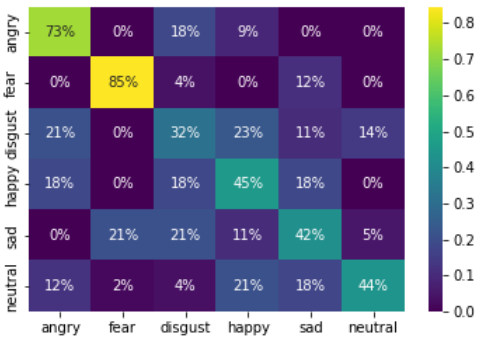


7.44. ábra. Szükséges idő

A konfúziós mátrixok szintén nem adnak megfelelő pontosságot, azonban az eredmények jóval bíztatóbbak, mint klasszikus osztályozóval.



7.45. ábra. Tanítási konfúziós mátrix

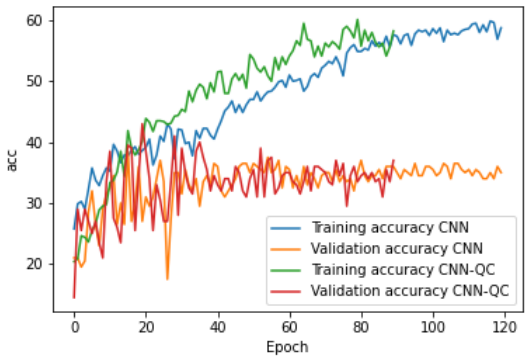


7.46. ábra. Validációs konfúziós mátrix

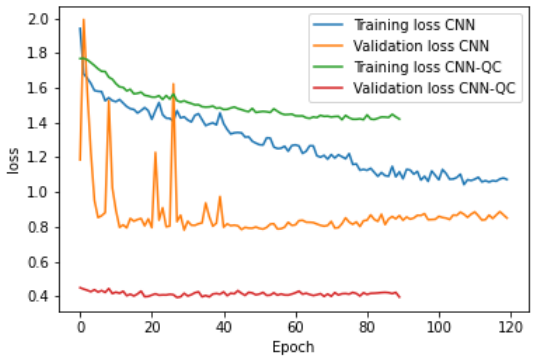
7.2.5.1. Összegzés

A fenti ábráról leolvasható, hogy nem sikerült optimális megoldást adni elsőként erre a problémára, azonban az MNIST adatbázisnál megismert különbséget a klasszikus és kvan-

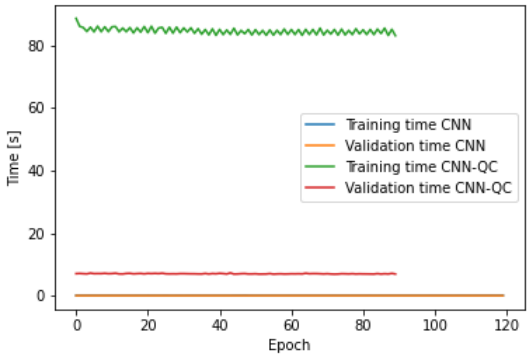
tumos osztályozó között jól kihangsúlyozza. Az eredményekből a következő összefoglaló ábrákat készítettem:



7.47. ábra. Pontosság összehasonlítása



7.48. ábra. Hibák összehasonlítása



7.49. ábra. Szükséges idők összehasonlítása

Észrevehető, hogy jóval kevesebb epoch-ra volt szükség kvantumos osztályozó esetén ugyanahhoz a tanítási és validációs pontossághoz. Látható, hogy a validáció 20%-al pontatlanabb, mint a tanítás, ezt ennyi osztály esetén nem sikerült kiküszöbölnöm. A szimulátor futtatásához szükséges időtartam az előzőekben, megismert eljárásokkal egyetemben, sokszorosa a klasszikus konvolúciós neurális hálózathoz képest. Látható, hogy ez egy jóval bonyolultabb feladat, mint az MNIST adatbázis volt, emiatt a konvolúciós réteggel rendelkező hálót nem tudtam működésre bírni (illetve a tanítás is rengeteg időt ölelt fel). Összességében kijelenthetjük, hogy érdemesebb kvantumos osztályozót használni, amennyiben rendelkezésünkre áll QPU, vagy gyors hozzáférésű felhő alapú QPU. Ez az IBM esetében előfizetést igényel, emiatt esetünkben a klasszikus osztályozó jobban teljesít, de kizárólag a tanítási idő miatt.

A konfúziós mátrixokról leolvashatjuk, hogy vannak olyan típusok, melyeken jóval nagyobb pontossággal sikerül a helyes predikció-címke párosítás, mint más típusokon. Ez azért van, mert valószínűleg két érzelem között spektrálisan nincs elég nagy különbség. Könnyen jósolhatónak számít például a félelem és a düh, nehezen megkülönböztethető például a többi érzelem. Ezért sikerülhetett más kutatóknak csökkentett adatbázison közel 100%-os eredményt produkálni [48].

8. fejezet

Összegzés

Munkám során újra felidéztem a korábbi tanulmányaim során megismert kvantuminformatika alapvető összefüggéseit, posztulátumait, majd megismerkedtem a kvantum-kapus és a lehűtésen alapuló kvantumprocesszorokkal. Ezek közül főleg az IBM kvantumszámítógépeit, szimulátorait használtam, valamint az IBM Qiskit könyvtár segítségével végeztem szimulációkat. Sajnos ezen kvantumprocesszorok csak korlátozott mértékekben elérhetőek (néhány kvantumbites valódi processzorok, illetve szimulátorok) és az elérési idő (mely magában foglalja a hálózati kommunikációt, a token validálását. sorbanállást) rendkívül nagy, emiatt lokálisan (pontosabban a Google számítógépén, ahol a projektet fejlesztem) kellett a szimulátorokat futtatni.

Ezután megismerkedtem részletesebben a gépi tanulás alapjaival (azokat eddig csak gyakorlati úton ismertem), egzakt matematikai leírást adtam rájuk. Ezen felül az újonnan alkalmazott modellem, az SVM modell alapos leírását gyűjtöttem össze.

A következő lépésként megvizsgáltam, hogyan lehetséges összekötni a gépi tanulást a kvantuminformatikával: elsőként itt is az alapvető összefüggéseket ismertettem, megvizsgáltam azt, hogyan lehetséges tanítani egy kvantumáramkört, illetve milyen teljesen kvantumos eljárások léteznek és azokat a gyakorlatban hogyan lehetséges használni. Ezen eljárások még nem elegendően pontosak, emiatt megismerkedtem különböző hibrid kvantum-klasszikus metódusokkal.

A munkám első részében a talán leggyakrabban alkalmazott QAOA algoritmus elméleti háttének feltárásával foglalkoztam. Ezen algoritmus három rétegből áll (szuperpozíciónáló, költség és keverő), ezek segítségével képes NP-teljes problémákra megoldást adni. A konklúzió ebben az esetben az lett, hogy rendkívül gyorsan képes lefutni még nagyobb problémaméret esetén is, azonban nagy paraméteroptimalizációt igényel. Mivel a végső célom nem ezen algoritmus futtatása volt, ezért ezzel nem foglalkoztam részletesebben. Célom az IBM kvantumszámítógépének, szimulátorainak kipróbálása volt azáltal, hogy alkalmazom korábbi munkáim során megszerzett gyakorlati tapasztalataimat.

Ezután variációs kvantum osztályozással és a kvantum kernel becsléssel foglalkoztam. A variációs osztályozás során egy előállított feature vektor segítségével előállítjuk a kvantumállapotakat, majd egy parametrizált kvantum áramkört alkalmazunk modellként, végül pedig méréssel kapjuk meg a klasszikus eredményeinket. Ezen modellt helyettesíthetjük kvantum kernelekkel is, ami ezeket a műveleteket mind magában foglalja, és pusztán a feature vektor szükséges a meghatározásához. Ebben az esetben arra jutottam, hogy a Qiskit többosztályos kiégeszítése SVM architektúráknál valamiért nem működik megfelelően. Úgy gondolom, hogy (habár ez az IBM által kiadott dokumentációk egyik legjellemzőbb feladata) a kvantumos eljárás a kvantumzaj jelenléte miatt nem képes ezt a feladatot jelenleg megfelelő módon megoldani, emiatt nem folytattam a többosztályos osztályozás további vizsgálatait. Itt érdemes megemlíteni, hogy habár a futási idő az IBM

szimulátorain nagyon alacsony (néhány 10-100 ms, ami a kernelszámítás során rendkívül jónak mondható), sajnos a hozzáférési idő perceket ölelt fel. Ez abból a szempontból jelent problémát, hogy ezek a szimulátorok a különböző futásokat különböző folyamatokként értelmezik, emiatt ezt többször is ki kellett várni.

Munkám jelentős részében kvantumos konvolúciós neurális hálózatokkal foglalkoztam: elsőként az MNIST adatbázison folytattam vizsgálatokat, melyek során négy különböző hibrid neurális hálózatot vizsgáltam. Elsőként a klasszikus konvolúciós neurális hálózatot, mely a vártaknak megfelelően tökéletesen megoldotta a feladatot, majd ugyanezen hálón megváltoztattam az osztályozó réteget egy kvantumáramkörre, amivel gyorsabb konvergenciát sikerült elérnem. Ezután bemeneti rétegként alkalmaztam kvantum-konvolúciós réteget, melyhez mind a klasszikus, mind a kvantumos osztályozóréteget hozzátársítottam. Ezzel nem sikerült elérnem megfelelő pontosságot.

Végezetül kipróbáltam ezen hálót jelfeldolgozási feladatokra, melyeken korábbi munkákhoz hasonló eredményeket sikerült produkálni. A kvantumos osztályozó itt is gyorsabb konvergenciát mutatott, mint klasszikus társa.

További célok Mivel a feldolgozott téma rendkívül újszerű, ezért ezen eredmények rendkívül sok potenciált tartalmaznak. A következőkben szeretném kipróbálni 1D-s konvolúcióval a jelfeldolgozást, mind klasszikus, mind kvantumos úton. Ezen felül szeretném LSTM (Long Short-Term Memory) és QLSTM (Quantum Long Short-Term Memory) hálók segítségével a jelfeldolgozás időszerűségét kihasználni.

A végső célom egy olyan kvantumos elvű mély neurális háló megalkotása, hogy az egy jelfeldolgozási feladatot megfelelő pontossággal el tudjon látni. Ezen háló korábbi kutatásaim alapján egy hibrid kvantum-klasszikus neurális hálózat lesz.

Köszönetnyilvánítás

Szeretném megköszönni konzulensem, Dr. Szegletes Luca munkáját, aki támogatott a témaválasztáskor (annak ellenére, hogy annak jelentős része neki is rendkívül újszerű), valamint rendszeres konzultációinkkor segítséget nyújtott felmerülő kérdéseimben.

Irodalomjegyzék

[1] Crema-D database. https://paperswithcode.com/dataset/crema-d.

[2] David Elieser Deutsch. Quantum computational networks. *Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences*, 425(1868):73–90, 1989.

[3] Michael A Nielsen and Isaac Chuang. Quantum computation and quantum information, 2002.

[4] Chetan Warke. Introduction to quantum computing part -1 Representation of qubit using Bloch sphere, 2019. [Online; accessed 20-Mar-2019].

[5] Wenjie Liu, Yinsong Xu, Maojun Zhang, Junxiu Chen, and Ching-Nung Yang. A novel quantum visual secret sharing scheme. *IEEE Access*, 7:114374–114384, 2019.

[6] IBM. Heavy-hex-lattice. https://research.ibm.com/blog/heavy-hex-lattice, 2021. [Online; accessed 07-Jul-2021].

[7] IBM. IBM Unveils Breakthrough 127-Qubit Quantum Processor. https://newsroom.ibm.com/2021-11-16-IBM-Unveils-Breakthrough-127-Qubit-Quantum-Processor, 2021. [Online; accessed 16-Nov-2021].

[8] IBM. Quantum computing. https://www.ibm.com/topics/quantum-computing. [Online].

[9] Ernst Ising. Contribution to the theory of ferromagnetism. *Z. Phys*, 31(1):253–258, 1925.

[10] David Sherrington and Scott Kirkpatrick. Solvable model of a spin-glass. *Physical review letters*, 35(26):1792, 1975.

[11] Ralph Baierlein. Thermal physics, 1999.

[12] Rudolf Peierls. On ising's model of ferromagnetism. In *Mathematical Proceedings of the Cambridge Philosophical Society*, volume 32, pages 477–481. Cambridge University Press, 1936.

[13] John A Mydosh. *Spin glasses: an experimental introduction*. CRC Press, 1993.

[14] Natalia N Vtyurina, David Dulin, Margreet W Docter, Anne S Meyer, Nynke H Dekker, and Elio A Abbondanzieri. Hysteresis in dna compaction by dps is described by an ising model. *Proceedings of the National Academy of Sciences*, 113(18):4982–4987, 2016.

[15] UCD. Quantum-Mechanical Tunneling. https://chem.libretexts.org/@go/page/20297, 2016. [Online; accessed 27-Apr-2016].

[16] Fred Glover, Gary Kochenberger, and Yu Du. A tutorial on formulating and using qubo models. *arXiv preprint arXiv:1811.11538*, 2018.

[17] Andrew Lucas. Ising formulations of many np problems. *Frontiers in physics*, page 5, 2014.

[18] Di Wang and Robert Kleinberg. Analyzing quadratic unconstrained binary optimization problems via multicommodity flows. *Discrete Applied Mathematics*, 157(18):3746–3753, 2009.

[19] Phys.org Bob Yirka. D-Wave announces launch of new Advantage quantum computer for business use. `https://phys.org/news/2020-09-d-wave-advantage-quantum-business.html`, 2020. [Online, accessed 30-Sep-2020].

[20] Altrichter Márta, Horváth Gábor, Pataki Béla, Strausz György, Takács Gábor, and Valyon József. *Neurális hálózatok*. Panem, 2006.

[21] John Hertz, Anders Krogh, and Richard G Palmer. *Introduction to the theory of neural computation*. CRC Press, 2018.

[22] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[23] Peter I Frazier. A tutorial on bayesian optimization. *arXiv preprint arXiv:1807.02811*, 2018.

[24] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[25] Satya Mallick. Support Vector Machines (SVM). `https://learnopencv.com/support-vector-machines-svm/`, 2018. [Online; accessed 11-Jul-2018].

[26] Andrew Ng. Cs229 lecture notes, part v: Support vector machines. *Technical report*, 2012.

[27] Vojtěch Havlíček, Antonio D Córcoles, Kristan Temme, Aram W Harrow, Abhinav Kandala, Jerry M Chow, and Jay M Gambetta. Supervised learning with quantum-enhanced feature spaces. *Nature*, 567(7747):209–212, 2019.

[28] Ryan LaRose and Brian Coyle. Robust data encodings for quantum classifiers. *Physical Review A*, 102(3):032420, 2020.

[29] Manuela Weigold, Johanna Barzen, Frank Leymann, and Marie Salm. Data encoding patterns for quantum computing. In *Proceedings of the 27th Conference on Pattern Languages of Programs*, pages 1–11, 2020.

[30] Edward Farhi, Jeffrey Goldstone, and Sam Gutmann. A quantum approximate optimization algorithm. *arXiv preprint arXiv:1411.4028*, 2014.

[31] Edward Farhi and Aram W Harrow. Quantum supremacy through the quantum approximate optimization algorithm. *arXiv preprint arXiv:1602.07674*, 2016.

[32] Zhihui Wang, Stuart Hadfield, Zhang Jiang, and Eleanor G Rieffel. Quantum approximate optimization algorithm for maxcut: A fermionic view. *Physical Review A*, 97(2):022304, 2018.

[33] Qingfeng Wang and Tauqir Abdullah. An introduction to quantum optimization approximation algorithm, 2018.

[34] Paul Erdős, Alfréd Rényi, et al. On the evolution of random graphs. *Publ. Math. Inst. Hung. Acad. Sci*, 5(1):17–60, 1960.

[35] Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

[36] Qiskit SVM. `https://qiskit.org/documentation/stable/0.26/tutorials/machine_learning/01_qsvm_classification.html`.

[37] Maxwell Henderson, Samriddhi Shakya, Shashindra Pradhan, and Tristan Cook. Quanvolutional neural networks: powering image recognition with quantum circuits. *Quantum Machine Intelligence*, 2(1):1–9, 2020.

[38] Yann LeCun, Koray Kavukcuoglu, and Clément Farabet. Convolutional networks and applications in vision. In *Proceedings of 2010 IEEE international symposium on circuits and systems*, pages 253–256. IEEE, 2010.

[39] Will Koehrsen. A Conceptual Explanation of Bayesian Hyperparameter Optimization for Machine Learning.

[40] Qiskit. Qiskit tutorial. `https://qiskit.org/documentation/tutorials.html`.

[41] Qiskit. Hybrid quantum-classical Neural Networks with PyTorch and Qiskit. `https://qiskit.org/textbook/ch-machine-learning/machine-learning-qiskit-pytorch.html`.

[42] Quantum neural networks. `https://github.com/yh08037/quantum-neural-network`.

[43] Quantum neural networks. `https://pennylane.ai/qml/demos/tutorial_quanvolution.html`.

[44] Short-time Fourier transform. `https://musicinformationretrieval.com/stft.html`.

[45] Stanley Smith Stevens, John Volkmann, and Edwin Broomell Newman. A scale for the measurement of the psychological magnitude pitch. *The journal of the acoustical society of america*, 8(3):185–190, 1937.

[46] Douglas O'Shaughnessy. Speech communication, human and machine addison wesley. *Reading MA*, page 40, 1987.

[47] Shivam Burnwal. Speech Emotion Recognition. `https://www.kaggle.com/code/shivamburnwal/speech-emotion-recognition`, 2020. [Online; accessed 2020].

[48] Pavol Harár, Radim Burget, and Malay Kishore Dutta. Speech emotion recognition with deep learning. In *2017 4th International Conference on Signal Processing and Integrated Networks (SPIN)*, pages 137–140, 2017.

[49] Sakorn Mekruksavanich, Anuchit Jitpattanakul, and Narit Hnoohom. Negative emotion recognition using deep learning for thai language. In *2020 joint international conference on digital arts, media and technology with ECTI northern section conference on electrical, electronics, computer and telecommunications engineering (ECTI DAMT & NCON)*, pages 71–74. IEEE, 2020.