Diplomamunka

EGÉSZSÉGES ÉS KÓROS ÖREGEDÉSI FOLYAMATOK MRI-ALAPÚ BIOMARKEREINEK KUTATÁSA GÉPI TANULÁS SEGÍTSÉGÉVEL

Szegedi Domonkos

Fizikus MSc, Orvosi fizika szakirány

II. évfolyam

Témavezető:

Dr. Légrády Dávid

(Egyetemi docens)

Nukleáris Technika Intézet

BME

2018

ide kerül a témakiírás

Tartalomjegyzék

1. Bevezetés

Az agy sokat változik az emberi élet során. Az elmúlt évtizedekben jelentős figyelem összpontosult az agykutatásban az emberi agy hálózatának, azaz angol nyelven connectome feltérképezésére akár strukturális akár funkcionális értelemben. Ennek a konnektivitásnak a megváltozása az emberi élet velejárója. Mértéke és üteme azonban indikátora lehet az agy kóros öregedési folyamatainak, s így számos korunkban gyógyíthatatlan, illetve nehezen kezelhető neurodegeneratív betegségnek, mint például az Alzheimer- vagy a Parkinson-kórnak. Az agyi öregedést jellemző biomarkerek kutatása ezért döntő jelentőségű ezen betegségek korai diagnózisához, melyek a páciens terápiáját könnyítik meg, valamint magasabb életminőséget tesznek lehetővé [1].

Az utóbbi időben legdinamikusabban fejlődő orvosi képalkotó eljárás a mágneses rezonancia képalkotás (MRI), melynek nagy előnye a többféle anatómiai, valamint funkcionális információ, mely különböző kontrasztok és mérési eljárások segítségével nyerhető az agyról [2][3]. Továbbá igen jó térbeli felbontás érhető el MRI technikával, ami az agyról készült kép minőségét jelentősen befolyásolja, ezért is előszeretettel használt képalkotási eljárás az agykutatásban [4].

A növekvő alanyszámú MRI kísérletek mára már lehetővé teszik gépi tanulás alkalmazását MRI képek kiértékelésére is. Az úgynevezett mély tanulás (deep learning) egyik módszere a konvolúciós neurális hálók, melyek eredetileg 2D-s képek klasszifikációjára lettek kidolgozva, és kiterjeszthetőek 3D-s MRI képeken történő alkalmazásra is. A 3D-s konvolúciós hálók módszerével a közelmúltban is foglalkozott egy kutatócsoport pédául neurodegeneratív betegségek diagnózisa kapcsán [5].

Egy potenciális biomarker lehet az agy kóros öregedése szempontjából a gépi tanulással kapott modellünk által becsült életkor, valamint a páciens valós életkora közötti különbség. Ezen biomarker pontos definiálása és alkalmazhatósága jelenleg is kutatott terület, T1 súlyozott MRI képeken történt már erre vonatkozó vizsgálat a közelmúltban [1][6][7][8].

Diplomamunkám során a Magyar Tudományos Akadémiai Természettudományi Kutatóközpont (MTA TTK) agyi öregedés vizsgálatával is foglalkozó Agyi Képalkotó Központban mért adatbázison alkalmazom a gépi tanulás módszereit, mely nyugalmi állapotú

funkcionális MRI (resting state fMRI, vagy RS-fMRI) felvételeket tartalmaz egészséges fiatal illetve idős személyekről. Emellett nyilvánosan elérhető RS-fMRI adatokon is végzek vizsgálatokat.

2. A dolgozat és munka célja

A diplomamunkám során a 3D-s konvolúciós neurális hálók módszerével foglalkoztam. Célom volt, hogy a vonatkozó irodalom széleskörű áttekintése, s így az elméleti háttér és módszerek megismerése után létrehozzak olyan konvolúciós neurális hálót, melyet a rendelkezésre álló in-house adatokon tanítva idős és fiatal életkori kategóriák alapján csoportosítani tudjam a mintáimat a véletlen találgatásnál szignifikánsan jobb pontossággal. Továbbá a hiperparaméterek vizsgálatával megkeresni azokat a beállításokat, melyek esetén az elért osztályozási pontosság optimális. Ezek után a külső forrásból származó adatokon való előzetes tanítás után a transfer learning módszerrel megvizsgálni, hogy tudom-e növelni a nagyobb elemszámmal rendelkező publikus adatokon előtanított hálóval a saját adatokon mérhető pontosságot. Végül egy regressziós modellel kapcsolatban is végzek kezdeti illesztéseket, hasonlóan a transfer learning hatását is vizsgálva a regressziós háló teljesítményére.

3. Elméleti háttér és módszerek

3.1. Nyugalmi állapotú fMRI képek

Munkám során tehát részben az MTA TTK Agyi Kutatóközpont adatain végeztem gépi tanítást, részben külső forrásból származó felvételeken, melyek minden esetben RS-fMRI felvételek voltak. Az agynak nincsenek cukor és oxigénraktárai, ezeket a vérkeringés juttatja a megfelelő régiókba. Az egyes agyi területeken a vér oxigén ellátottságának mérésén alapuló képalkotó eljárás az úgynevezett BOLD (blood-oxygen-level dependent imaging – a vér oxigén szintjétől függő képalkotás) jel mérése, ami egy kontrasztanyag nélküli fMRI technika. Nyugalmi állapotú fMRI mérésről beszélünk, amikor a páciensek nem végeznek kognitív tevékenységet a mérés során, nyitott vagy csukott szemmel, de éberen fekszenek a mérőberendezésben. Ekkor a BOLD jelben bekövetkező spontán, alacsonyfrekvenciás (0,01-0,1 Hz) fluktuációkat tudjuk mérni [9]. Amennyiben az egyes voxelekben a BOLD jel változásának korrelációit figyeljük a felvételek idősoros adatain, úgy a képek az agy nyugalmi állapotú funkcionális konnektivitásáról adnak információt [10]. Esetemben az idősor adatokból nem a BOLD jel fluktuációit vizsgáltam, hanem az egyes időpillanatokban felvett képeket, mint anatómiai információkat használtam, ugyanis az RS-fMRI képek alapjele az anatómiát tükrözi.

Az RS-fMRI technika T2* súlyozott MRI képet eredményez szemben az anatómiai képalkotásban gyakrabban használt, és a bevezetőben említett T1 súlyozású képekkel. Emellett a funkcionális képekre jellemző módon a felbontás tipikusan rosszabb, mint az anatómiai esetben, míg utóbbiban a gyakori felbontás az 1x1x1 mm^3, funkcionális képalkotásban ez az adat a duplája szokott lenni irányonként, vagy rosszabb (2x2x2 mm^3). Ez azt eredményezi, hogy kevesebb voxeladatunk keletkezik egy mérésben az agyról (nyolcad annyi).

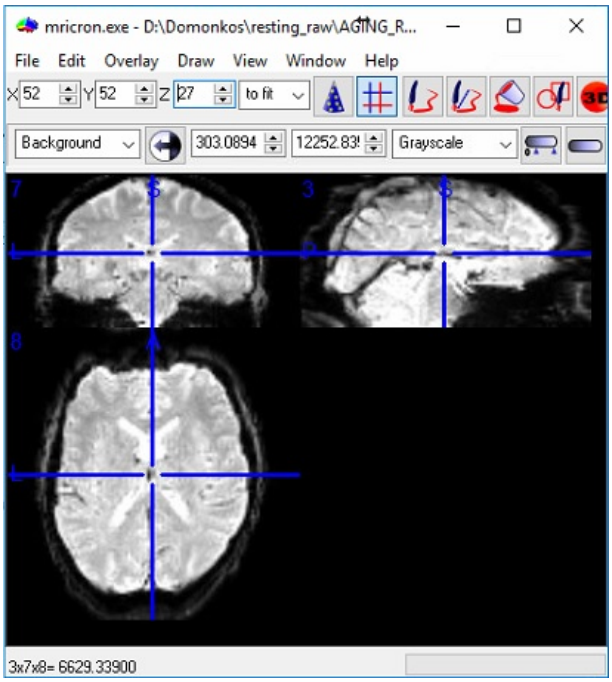
Másrészt a szuszceptibilitás mérésén alapuló technika és a T2* súlyozás eredményeképpen artefactumként jelentősebb jelkiesést kapunk a folyadékkal vagy levegővel telt régiók, így például az agykamrák és a homloküreg környékén [11]. Ez segíti az életkorral kapcsolatos gépi tanítást, mivel az agykamrák és a koponyában található egyéb üregek mérete és pontos alakja változik az emberi élet során, így az életkorral kapcsolatos becslések egyik

alapját képezhetik az ezekben való minél pontosabb distinkció [12].

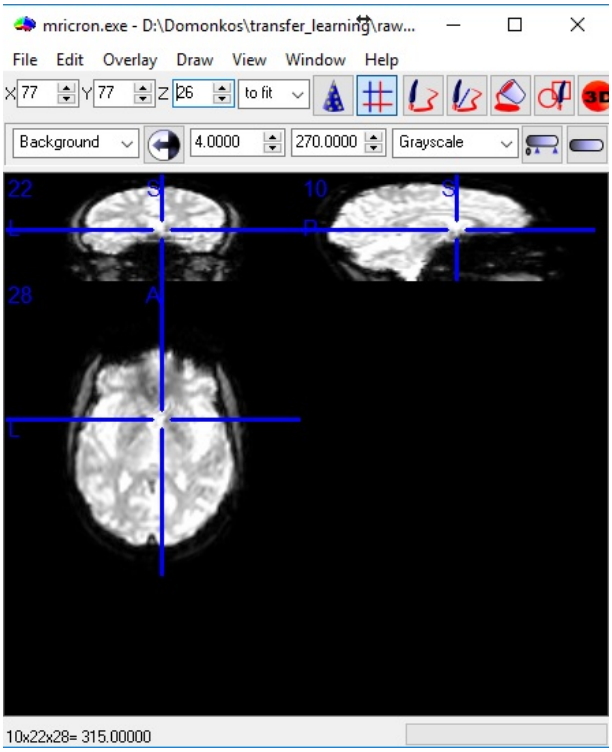
Tehát a nyugalmi állapotú fMRI mérés során számos 3 dimenziós felvétel készül a páciensről a mérési idő folyamán. Ezen képek között tapasztalható eltérés oka többek között a páciens esetleges mozgása, de egyszerűen a BOLD jel időben bekövetkező változása is. Amikor a funkcionális konnektivitás feltérképezése érdekében történik képalkotás, akkor ezen képek összefésülése komoly figyelmet és korrekciók alkalmazását igényelheti, azonban esetemben azt jelenti, hogy a gépi tanításhoz egy pacienstől több képem is rendelkezésre állt, noha ezek csak kis mértékben térnek el.

Az általam használt, az MTA kutatócsoportjának belső adatbázisának adatai olyan RS-fMRI képek, melyek 2x2x2 mm^3-ös felbontással készültek, 846 különböző időpillanatban rögzített 3D-s képek, 104x104x54 szürkeskálás voxelt tartalmaznak. A képek a szakterületen használt .nii formátumban voltak elérhetőek, mely fejlécet és metaadatokat is tartalmazó adatformátum, python környezetben elérhetőek azok a függvénycsomagok (nibabel), melyekkel a beolvasás, és további adatfeldolgozás lehetséges. Emellett a fájlokat MRIcroN szoftver segítségével könnyen vizualizálhatóvá tudtam tenni, az idő, és térbeli szeletek között egyszerűen navigálva.

A külső forrásból származó adataim 3 csoportban voltak elérhetőek a Consortium for Reliability and Reproducibility (CORR) internetes felületén ([13][14][15]), a Ludwig Maximilian University méréseiből 3x3x4.4 mm^3-es (LMU_3), 3x3x4 mm^3-es (LMU_2), valamint 1.65x1.65x3 mm^3-es (LMU_1) felbontásban. Itt rendre 120, 120, illetve 180 különböző időpillanatban elkészített 3D-s kép tartozott egy páciensheznek egy mérési alkalom kapcsán, ugyanis ebben az adatbázisban egy alanyhoz több mérés is tartozott, tipikusan 2-6. A 3D-s képek rendre 64x64x28, 64x64x36, valamint 144x144x52 szürkeskálás voxelt tartalmaztak.



1. ábra. Példa az in-house adatokból



2. ábra. Példa a publikus adatokból

3.2. Gépi tanulás

A gépi tanulás a számítástechnika egy területe. Azt mondhatjuk, hogy a számítógépes program tapasztalat (experience = E) alapján képes tanulni egy teljesítmény mérővel jellemezhető módon valamely feladat (task = T) kapcsán, ha a feladat elvégzéséhez tartozó teljesítménye (performance = P) a tapasztalattal nő [16]. A feladat, a teljesítménymérő és a tapasztalat problémától függően lehetnek különbözőek, a gépi tanulás során kialakultak a tipikus csoportosításai a fenti kategóriáknak. Esetemben például klasszifikációs feladatról beszélünk, amennyiben az a célunk, hogy a program egy beadott mintára (példa adat, vagy example) kimenetként meg tudja mondani, hogy az általunk felállított életkori kategóriák közül melyikbe esik az adott minta (például idősről, vagy fiatalról készült a felvétel).

Az imént említett minták azok, melyeket az algoritmusnak fel kell dolgoznia a feladat végzése közben. Ezeket a mintákat különféle tulajdonságokkal (feature) tudjuk jellemezni. Az algoritmizálhatóság érdekében ezen tulajdonságok számszerűsítettek, így egy adott minta egy tömbbel lesz reprezentálható ($\mathbf{x}$). Esetemben a páciensekhez tartozó MRI felvételek szürkeskálás voxelértékei lesznek a mintákat jellemző tulajdonságok, melyeket egy tömbben tárolok el. Egyéb tulajdonság lehet például egy karakter és annak környezetének kódja az írott szövegben, vagy éppen meteorológiai idősor adatok.

Felügyelt tanításról beszélünk akkor, amikor a bemeneti adatokból történő predikciónkat össze tudjuk vetni a mintához tartozó valamely valós adattal, vagy címkével (label). Így például klasszifikációs problémánkban az általunk becsült életkori kategóriát a páciens tényleges életkorával (fiatal, vagy idős a kétosztályos esetben). Ezen összevetés számszerűsítése révén juthatunk el a problémánk teljesítménymérőjéhez, mely legegyszerűbb esetben a pontosság. A helyesen kategorizált mintáink aránya az összes vizsgált mintához képest ugyanis jó mérőszáma lehet annak, hogy mennyire volt sikeres a klasszifikáció. A teljesítménymérőnk javítása a tanítás során optimalizálással valósítható meg, ehhez definiálnunk kell egy optimizálandó mennyiséget, mely nem feltétlen maga a teljesítménymérőnk.

Míg regressziós feladat kapcsán alapesetben a célunk, hogy az átlagos négyzetes hibát

(mean-squared-error vagy MSE) csökkentsük, addig a klasszifikációs problémákhoz definiált veszteségfüggvényünk, melyet minimalizálni szeretnénk legtöbbször a kereszt-entrópia. Ebből az információelméletben is ismert mennyiségből [17] az alábbi alakban írható fel a minimalizálandó függvényünk diszkrét valószínűségi változókra:

$$H(p,q) = -\Sigma_x p(x) log(q(x)) \tag{1}$$

ahol p és q a klasszifikációra vonatkozó diszkrét valószínűségeink (x itt a diszkrét valószínűségi változó, például a kategóriák). Esetünkben a valós életkori kategória ismert, így kétosztályos becslőnél egyik vagy másik osztály értéke lesz 1, míg a másiké 0. A becsült kategóriák esetében pedig minden osztály valószínűsége egy a modell által adott érték (felösszegezve 1-et adva). Gépi tanítás során tehát a szakirodalomban sokszor veszteség-, költség, vagy hibafüggvénynek (loss/cost/error function) is nevezett mennyiség szélsőértékét próbálja az algoritmus megkeresni optimalizálással a későbbiekben részletezett paramétereink sokdimenziós terében, a mintákon számolt költségfüggvény által meghatározott hiperfelületen.

Ehhez tekintsük az egyenes illesztés egyszerű példáját. Ekkor adva vannak mintáink egy többdimenziós térben, ahol parametrizálunk egy egyenest (síkon két paraméterrel: például a meredekséggel és a tengelymetszettel), majd a paramétereinket változóként kezelve végzünk optimalizálást a négyzetes eltérés csökkentése érdekében a paraméterek értékeit változtatva, miközben a mintáinkat (adatpontok) konstansnak tartjuk. A gépi tanulás során is a költségfüggvényünk gradiensét számoljuk ki a modellünk paramétereinek függvényében, majd ezeket a paramétereket aszerint változtatjuk, hogy a költségfüggvényünk értéke csökkenjen, ezt az eljárást nevezzük gradiens módszernek (gradient descent) [18].

$$w = w - \frac{\delta loss}{\delta w} * lr \tag{2}$$

ahol w a modellünk egy paramétere (w = weight – vagy súly), loss a költségfüggvényünk, lr pedig a tanulási rátánk (learning rate), az optimalizációs lépésünk nagysága. Ez a folyamat gyakorlatilag egy minimum keresést jelent, mely szerencsés esetben, illetve különféle optimalizációs módszerekkel globális minimumhely megtalálását eredményezi. A mini-

mumkereséshez kapcsolódó paraméterbecslésünk röviden az alábbi összefüggéssel írható le, melyet a fenti képlet használatával algoritmizálni tudunk:

$$\mathbf{w} = argmin(loss(\mathbf{w})) \tag{3}$$

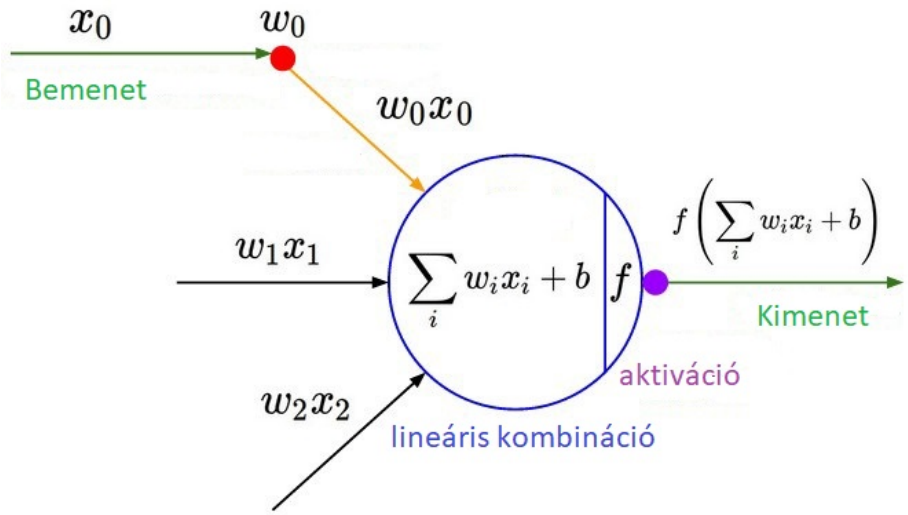
ahol $\mathbf{w}$ tartalmazza a modellünk valamennyi paraméterét.

Az opimalizáció során több lehetőség is rendelkezésemre állt, meghívható függvények formájában is a TensorFlow környezetben [19][20][21]. A fenti gradiens módszernek, az úgynevezett gradient descentnek az egyik továbbfejlesztése a szochasztikus gradiens módszer, mely a gradiens értékét a mintáknak csak egy alhalmaza alapján számolja, ez gyorsítja egy tanítási lépés idejét, azonban lassabb konvergenciát eredményez. További lehetőség a momentum módszer, mely a lokális minimumokból való kijutást tudja segíteni azzal, hogy a korábban kapott gradiensek értékét is használja az aktuális lépésben számolt gradiens mellett a súlyok frissítésére. Ezek a módszerek kombinálhatóak is, a szakirodalomban megtalálható részletes leírásuk [19][20][21].

3.3. Mély neurális hálók

A fenti hipersík illesztéses példán láthatjuk, hogy a költségfüggvényünk a paramétereinknek lineáris függvénye. Ez igaz még a magasabb fokú polinom illesztés esetén is, hiszen változóink együtthatói szintén lineárisan szerepelnek az adatokra illesztendő modell-függvényünkben. Ez azt jelenti, hogy a modellünk a mintánk bemeneti változói (adataink) között lineáris kapcsolatokat tud feltérképezni. Ezzel szemben egy úgynevezett aktivációs függvény bevezetése a bemeneti adataink közötti nemlineáris kapcsolatokat is képes felderíteni. Gyakran alkalmazott aktiváció az úgynevezett rectified linear unit, vagy ReLU, mely az $f(x) = max(0, x)$ alakban írható fel [22]. Az aktivációs függvény adja ezért modellünk erejét, és ennek a révén jutunk el a neurális hálók alapegységét képező neuronhoz. A neuron esetében nem csak a bemeneti adatainknak a paraméterek (súlyaink) szerinti lineáris kombinációja lesz a kimenet, hanem erre az értékre még alkalmazunk egy nem-lineáris függvényt is. Aktivációként számos függvényt használhatunk, az általam említett ReLU mind klasszifikáció, mind regressziós problémák esetén is jól teljesít. A módszer elne-

vezése onnan ered, hogy agyunk neuronjai is a dendriteken kapott jelek szuperpozícióját követően egy ingerküszöb eléréséhez kötötten (aktiváció) továbbítanak jelet axonjaikon keresztül.

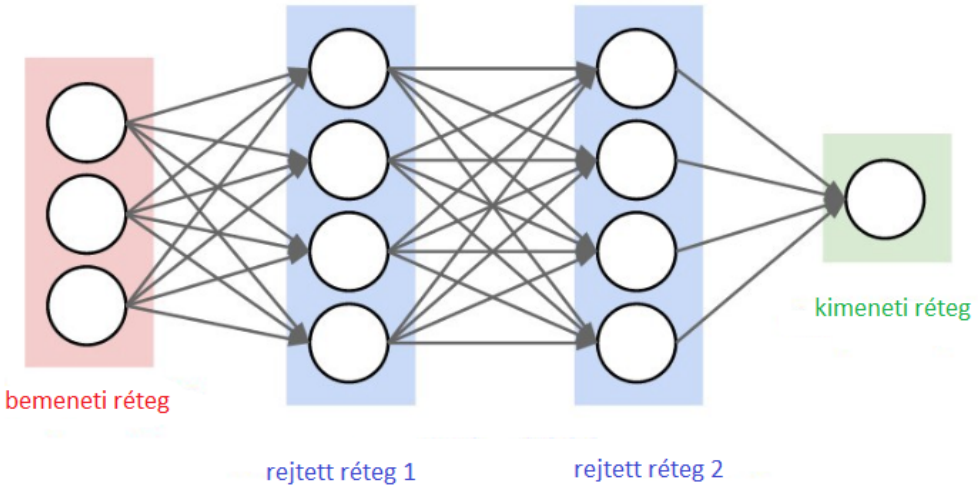


3. ábra. Neuron egység a neurális hálóból (átdolgozva [36]-ból)

Az idegrendszerrel való összevetés juttat el minket a neurális hálók és a mély neurális hálók ötletéhez. A bemeneti adatainkból ugyanis nem csak egyfajta lineárkombinációt és az arra történő aktivációt számolhatjuk ki, hanem akár többet is. Ez több neuron bevezetését eredményezi, melyek persze a paramétereink számának növekedésével is járnak. Továbbá ezen neuronok kimeneteiből nem kell a modellünknek közvetlenül regresszió esetén a kimeneti értéket, klasszifikáció esetén pedig a becsült osztályt, illetve az adott osztályokba való tartozás valószínűségét megadnia, hanem az értékek lehetnek egy következő neuron bemeneti adatai. Azon neuronok számítanak tehát az első rejtett rétegbe, melyek a bemeneti adatokból számolják kimenetüket. További rejtett rétegekbe tartoznak azok a neuronok, melyek korábbi rétegeket neuronjainak kimeneteiből számolják saját kimenetüket, ahol ezen kimenetek egy következő réteg bemeneteit adják. Ezt szemlélteti a 4. ábra, valamint az alábbi, két rejtett réteget tartalmazó neurális hálót leíró képlet:

$$y = f(\mathbf{W_3} f(\mathbf{W_2} f(\mathbf{W_1}\mathbf{x} + \mathbf{b_1}) + \mathbf{b_2}) + b_3) \tag{4}$$

ahol **x** a bemeneti adatainkat y a kimenetet jelentik (ebben az esetben a kimenet egyetlen szám, mint például egy regressziós feladatban lehet), f az aktivációs függvény. **W** mátrixok súlyokat tartalmaznak, a **b** bias értékekkel közösen a modell paramétereit alkotják, számukat a mátrixok méreteiből kaphatjuk meg, melyet a rejtett rétegekben a neuronok száma, valamint a bemenet és a kimenet dimenziója határoz meg.



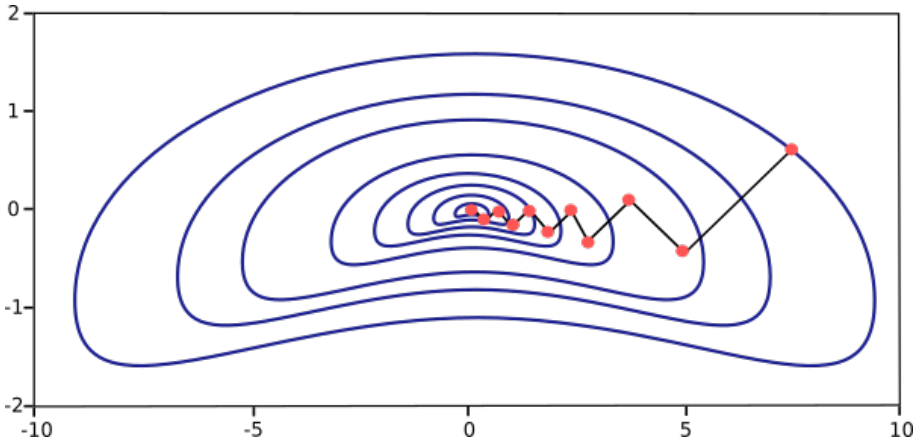
4. ábra. Neurális háló 2 rejtett réteggel (átdolgozva [36]-ból)

Felügyelt tanítás esetén tehát a költségfüggvényünk a modellünk által becsült kimenetekből és a valódi címkékből a korábban említett módon számolható problémaspecifikusan. Emellett a paraméterek is a már látott gradiens-módszerrel optimalizálhatóak, mely folyamatot többrétegű hálók esetén backpropagation-nak nevezzük.

A neurális háló a rejtett rétegek számának növelésével lesz mély neurális háló. Minél több réteget használunk ugyanis, annál bonyolultabb kapcsolatokat tudunk feltérképezni a bemeneti adataink között annak érdekében, hogy a kimenetre minél jobb becslést adhassunk. Amennyiben az összes neuronunk két réteg között össze van kötve, úgy teljesen összekapcsolt (FC - fully connected) hálóról beszélünk, vagy multilayer perceptronról.

3.4. Normalizálás és kimenet számítása

A gradiens keresésen alapuló optimalizáció tulajdonságaiból fakad, hogy a konvergencia gyorsabb, illetve stabilabb lehet nagyobb tanulási ráták esetén is, ha a bemeneti adatok normalizálva vannak 0 átlagúra és egységnyi szórásúra [23]. Gondolhatunk arra az esetre, ha minden bemeneti adat pozitív (képnél a bemeneti adatok nem negatív volta teljesül), a gradiens módszer miatt a minimumkeresés cikk-cakkossá válhat (5. ábra. Megj. az optimalizáció más okokból is cikk-cakkossá válhat), ami lassítja a konvergenciát, ezért a képadataimat minden esetben normáltam a fenti feltételeknek megfelelően.



5. ábra. Példa az optimumkeresés cikk-cakkosságára (átdolgozva [38]-ból)

A kimeneti réteg számítása ki kell elégítse a rétegre vonatkozó elvárásunkat. Osztályozás esetén a legtöbbször alkalmazott technika, hogy a kimeneti réteg minden neuronja egy skalár, melyet egy olyan tömbbé tudunk összerakni, aminek annyi eleme van, ahány osztályunk, s ennek megfelelően felügyelt tanításnál a címkéink is hasonló tömbök 0 elemekkel a valódi osztályt jelölő elemet kivéve, ahol ez az érték 1 (one-hot-encoding). Azt várhatjuk el a becslőnktől, hogy a kimeneten minden osztályra megadja a háló szerint az adott osztályba való tartozás valószínűségét. Ez vezet minket oda, hogy a kimeneti tömbünk értékei 0 és 1 közé kell legyenek korlátozva, felösszegezve pedig egyet kell adjanak. Klasszifikációhoz ezért gyakran használt osztályozó a softmax függvény, melyhez az utolsó rétegből számolt értékeinket 0 és 1 közé kell skáláznunk. A tensorflow softmax osztályozóján alapuló hibaszámító függvény ezt a skálázást automatikusan végzi el, ezért

az utolsó rétegből számolt értékeket aktiváció nélkül kell az osztályozónak továbbítani, ahol a nemlineáris softmax függvény lesz a tulajdonképpeni aktiváció.

Általánosan is elmondható, hogy az aktivációs függvény alkalmazása, valamint a súlyok inicializálása skálázási feltételt írhat elő felügyelt tanítás során a címkék értékeire. Regressziós probléma esetén, amennyiben konkrét életkor becslőt szeretnénk kapni, akkor a kimenet egy adott érték (skalár). Az aktivációtól függően kellhet skálázást alkalmaznunk itt is a címkéken (valós adatokon), ReLU aktiváció esetén ez a probléma nem áll fenn, hiszen az életkor csak pozitív érték lehet, ahogy a ReLU kimeneti értéke is. Mégis a kimeneti rétegben az átlagos négyzetes hibán alapuló költségfüggvény számításhoz az utolsó rétegből számolt lineárkombinációkra sok esetben tisztán lineáris aktivációt alkalmaznak (azaz nem használnak aktivációt).

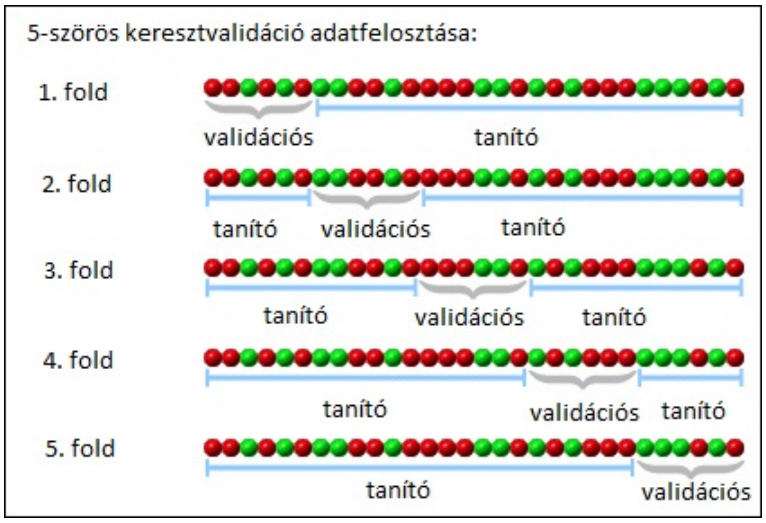
3.5. Tanító és teszt adatok, keresztvalidáció

A gépi tanítás során a rendelkezésünkre álló adatbázist felosztjuk részekre. Az adatok egy részén végezzük csak a tanítást, ezt nevezzük tanító adatoknak. Ezek után a tesztadatokon nézzük meg a modellünk teljesítőképességét. A modellünket a tanító (training) adatokon tudjuk illeszteni, miközben a mintákon mért hiba vagy költség értéke csökken (training error). Minél kisebb ez az érték, annál jobban illeszkedik a modellünk a tanítóadatokra. A teszt adatokon mért hiba és annak viszonya a tanító minták hibájához megadja, hogy mennyire képes a modellünk általánosítani (ezért a teszt mintákon mért hibát generalizációs hibának is nevezik) [16]. Hiába illeszkedik ugyanis a modellünk nagyon pontosan a tanító adatokra, ha a korábban nem látott tesztadatokra szinte semennyire.

Definiálhatunk még egy harmadik kategóriát is a validációs adatokat. Ezek lényege, hogy a tanítás során ezen minták alapján nem történik paraméter frissítés, de a pontosságot folyamatosan mérjük rajtuk, amivel a modell generalizációs képességét tudjuk monitorozni. Amikor a validációs adatokon mért hiba elkezdene nőni, akkor a tanítást nem érdemes folytatni, mert noha a tanító adatokra a modell jobban is tudna illeszkedni, de általánosító képessége csökkenne, ezt nevezzük korai megállásnak (early stopping). A validációs adathalmazt a megfelelő hiperparaméterek kiválasztására is alkalmazhatjuk.

Validációs adatok használata mellett a pontosságot a tanítás végén számoljuk csak ki a teszt adatokra.

Esetemben kevés minta állt rendelkezésre, ezért az ebben a helyzetben alkalmazott technikát használtam. E szerint az adatokat tanító és validációs csoportra osztottam, relatíve kevés validációs adattal, hogy a tanítás minél több adaton történhessen. A tanítást követően az addig tanító adatokból választok validációra használt mintákat, és a többi adaton végzem a tanítást. Ezt megismétlem addig, míg minden minta nem lesz egyszer validációs adat, ahogy az 6. ábra is szemlélteti. Az eljárást keresztvalidációnak nevezik ([16] 118-120.o).



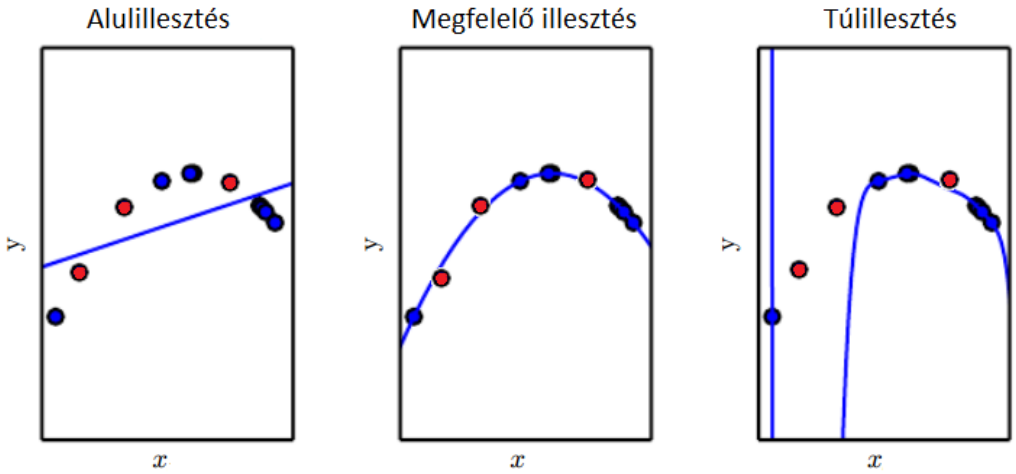
6. ábra. 5-szörös keresztvalidáció adatfelosztása (átdolgozva [37]-ból)

3.6. Alul és túlillesztés, hiperparaméterek

Gépi tanítás során tehát két alapvető célunk van, egyrészt a tanító adatokon mért hibát minél kisebb értékre csökkenteni, másrészt a tanító és a validációs adatokon mért hibák közötti különbséget szintén kis értéken tartani. Az alul és túlillesztés jelensége ezen célok megvalósulásával kapcsolatos. Alulillesztés esetén ugyanis a modellünk nem képes a tanító adatokra illeszkedni kellő mértékben, így a tanító adatokon mért hiba nagy marad. Példa lehet erre, ha egy adott polinommal jellemezhető hiperfelületre illeszkedő adatpontokra egy jóval alacsonyabb fokú polinomot próbálunk illeszteni (például egy parabola mentén

található pontokra az egyenes nem illeszkedik megfelelően).

Túlillesztés esetén a modellünk a tanító adatokra ugyan jól illeszkedik, de a validációs adatokon mért hiba ehhez képest igen nagy. Az előző példánál maradva, ha egy adott polinommal jellemezhető hiperfelület környezetében található adatpontokra egy jóval magasabb fokú polinomot illesztünk, noha a tanító adatokon mért hiba még kisebb is lehet, mint egy alacsonyabb fokú esetben, a polinómunk által meghatározott hiperfelület az adatpontokat jobban megközelíti, de a korábban nem látott validációs adatokon a mért hiba nagyobb.



7. ábra. Alul- (balra), helyes (középen) és túlillesztés (jobbra) példái (átdolgozva [16]-ból)

Modellünk alul-, illetve felülilleszkedését döntően befolyásolják annak hiperparaméterei. Ezek olyan paraméterek, melyeket a tanítás során nem frissítünk, de a modell viselkedését tudjuk velük befolyásolni. Ilyenek a korábbi bekezdésekben szereplő tanulási ráta vagy a rejtett rétegek, valamint a rétegeken belüli neuronok száma is. A hiperparaméterek alkalmas megválasztása segít elkerülni az alul-, illetve túlilleszkedést ezért munkám során a tanítás sikerességét különféle hiperparaméter értékek esetén is megvizsgáltam, hogy ki tudjam választani a modellem számára alkalmas beállításokat.

Túltanítás ellen további segítséget jelenthetnek különböző regularizációs módszerek. Standardizált bemeneti adatok és átskálázott kimeneti adatok esetén a túltanulás ellen használhat a minimalizálandó költségfüggvény kiegészítése a súlyparaméterek egyes,

vagy kettes normájával egy szorzófaktor használatával, mely hiperparaméter lesz [24]. Amennyiben a bemeneti adatokat normalizáljuk, és a kimenet is át van skálázva (osztályozásnál a valószínűségekkel 0 és 1 közé például), akkor a várakozásunk az, hogy a súlyok is kis értékűek lesznek. Nullához közeli súlyokkal legalábbis nem tévedhet nagyot (a mérhető hibafüggvény nem tud elszállni) a háló. A kiugróan magas súly a túlillesztés jele lehet, ezért használhatjuk például osztályozás esetén az alábbi veszteségfüggvényt, melynek első tagja a hagyományos veszteségfüggvény, amit kiegészítünk a súlyokra vonatkozó mellékfeltételünkkel, amit szintén beveszünk az optimalizálásba.

$$L(p, q, \mathbf{w}|\lambda) = H(p, q) + \lambda \mathbf{w}^{\mathbf{T}} \mathbf{w} \tag{5}$$

Az optimalizálás kezdetén a súlyok kis értékét feltéve, statisztikai úton is megkaphatjuk az L2 normával kiegészített tagot a minimalizálandó függvényben. Ehhez a súlyparaméterekre vonatkozó maximum aposteriori becslésünk priorjába egy nullára centrált gauss függvény beírásával juthatunk el [16].

További lehetőség a dropout alkalmazása [26]. Ennek a módszernek a lényege, hogy a tanítás során az egyes rétegekben a neuronok aktivációinak számítása után bizonyos valószínűséggel kinullázzuk az értéküket, a következő réteg neuronjaira így ezek a neuronok nem lesznek hatással. Felfoghatjuk a módszert úgy is, mintha alacsonyabb neuronszámú alhálókat tanítanánk párhuzamosan, és ezek aggregált tudását kapnánk végeredményként. A validálás és tesztelés során nem alkalmazunk dropout-ot. Az eljárás hiperparamétere a neuronok elhagyásának valószínűsége.

3.7. 3D-s konvolúciós neurális hálók

Abban az esetben, amikor a tanítást képeken végezzük, ki tudunk használni olyan lehetőségeket, melyek a képi információ tulajdonságaiból adódnak. Ilyen lehet, hogy amennyiben képeket szeretnénk osztályozni, akkor ebben segít, ha a neurális hálónk képes felismerni tipikus alakzatokat, mintázatokat a képen, mint például egyszerű esetben éleket. Egy teljesen összekapcsolt háló is képes lehet hosszas iteráció után az éleket alkotó voxelek közötti kapcsolatot felismerni, azonban a konvolúciós hálók esetén ezt a tudást, azaz

a képen a voxelek közötti lokális változások fontosságát belekódolhatjuk a modellünkbe, ezzel jelentősen gyorsítva a tanítást. Távoli voxelek közötti kapcsolatok tehát nem fontosak annyira, ezért a neurális hálóknál látott súlymátrixok ritka mátrixokká válnak, hiszen távoli voxelek közötti kapcsolatokat kifejező súlyok kinullázódnak és a szomszédos, illetve közeli voxeleket összekapcsoló súlyok maradnak véges értékűek.

A fenti felismerésből adódik, hogy egy, a képfeldolgozásban már hosszabb ideje ismert műveletet használjunk a neurális háló rétegeinek számítása során. Ez a képek esetében a rejtett rétegeknek az előző rétegekből konvolúcióval való számítását jelenti, hiszen a konvolúció pont a voxelek közötti ritka konnektivitáson alapszik. A konvolvált képen ugyanis egy voxel értéke az eredeti kép egy voxelének csupán a konvolúciós kernel méretével arányos szomszédságában található voxelek súlyozott összegéből számítódik, ahogy az alábbi összefüggés is mutatja (2D-s eset):

$$S(i,j) = (K * I)(i,j) = \Sigma_m \Sigma_n I(i-m, j-n) K(m,n)[16] \tag{6}$$

Az képlet könnyen kiterjeszthető magasabb dimenziókba, esetemben a 3 dimenziós képekre vonatkozó összefüggés a releváns. A gyakorlatban (noha konvolúciónak nevezik) a keresztkorrelációt számolják a számítógépes algoritmusok, melynek a konvolúcióval rokon tulajdonságai vannak, és a 3D-s esetre felírható hasonló kifejezése:

$$S(i,j,k) = (I * K)(i,j,k) = \Sigma_m \Sigma_n \Sigma_p I(i+m, j+n, k+p) K(m,n,p) \tag{7}$$

Az eljárás további előnye a paraméter megosztás és a művelet eltolási invarianciája. Előbbi azt jelenti, hogy ugyanazt a kernelmátrixok alkalmazzuk a teljes képen, így ugyanazokat a súlyokat használjuk a következő réteg számításához. Ez a modellünknek a problémához való illeszkedése miatt a paraméterek számának csökkenése ellenére nem vezet alulillesztéshez, ugyanakkor az elvégzendő számításokat drasztikusan csökkenti. Az eltolási invariancia tulajdonság arra utal, hogy a keresztkorreláció eredménye egy eltolt kép esetén az eredeti kép keresztkorrelációs képének az eltoltját eredményezi.

A konvolúció két réteg között a neuronok számát nem, vagy csak kis mértékben csökkenti (amennyiben a képünk széleit nem egészítjük ki nullákkal, akkor a konvolúciós

kernelünk méretének felével fogja csökkenteni a képek méretét a művelet, ez tipikusan néhány voxelt jelent). A dimenziók csökkentése érdekében ezért alul mintavételezést alkalmazhatunk a konvolúció művelete után [27]. Ennek egyik módja, amit munkám során használtam is, a maxpool művelet [28], ami egy adott kernelmérettel és kernelléptetéssel végigpásztázik a képen, és a méretnek megfelelő voxelek közül a maximális értéket tartja csak meg. Így például egy 100x100-as képen a 2x2-es maxpool művelet x és y irányban is 2 lépésközzel felére csökkenti a képméreteket. Az új kép tömbjének (0,0)-ás értéke pedig a (0,0), (0,1), (1,0) és (1,1) értékek közül a legnagyobb lesz.

3.8. Transfer learning

Kis adathalmazon való tanítás esetén fennáll a veszélye, hogy noha validációs pontosság értelmében nem történik túlillesztés, mégis az adott mintahalmaz esetleges specifikus tulajdonságaira túlzott mértékben optimalizáljuk a neurális hálónkat. Amennyiben rendelkezésre állnak külső forrásból származó adatok is, akkor indokolt lehet a transfer learning módszer alkalmazása. Én is így jártam el az adatok részletezésénél leírt külső forrásból származó minták használatával.

A transfer learning módszere [29][30] egyfajta tudásátvitelt valósít meg azzal, hogy egy nagyobb forrástartomány adatbázisán tanított rendszer optimalizált jellemzőit használja fel egy kisebb céltartomány adatain történő tanításhoz. Ez praktikusan a tanítás során kapott súlyok használatát jelenti más adatokon való tanításhoz. A konvolúciós rétegben a képek alternatív reprezentációit kapjuk meg, s ezen rétegek nagyobb adatbázison történő tanításától azt várhatjuk, hogy általánosabb reprezentációit adják a bemeneti mintáknak (gondolhatunk itt például a szkennerfüggetlenségre). Így ezen rétegek súlyainak transzferálásával a kisebb mintahalmazon történő tanítás során az osztályozás az általánosabb értelemben releváns jellemzők alapján történhet, az adathiányból adódó potenciális túlillesztés nagyobb eséllyel kerülhető el. Ez várakozásomban a saját adatokon való tanítás során elérhető maximális pontosság növekedését eredményezheti.

A transfer learning során vizsgálható esetek lehetnek többfélék, a konvolúciós, valamint a fully connected rétegek súlyait együtt, illetve külön-külön is felhasználhatjuk rögzítetten, vagy csak inicializálásra a saját adatainkon történő tanítás során. Ezeket az

eseteket mind vizsgáltam, s megnéztem, hogy az elért pontosságra milyen hatással vannak. A tensorflow környezetben definiált hálóm modelljének súlyait a különféle szcenáriók alapján hoztam tehát létre konstansként vagy változóként, és használtam a korábbi tanítás végeredményeként kapott és elmentett súlyok tömbjeit ezen paraméterek bemeneti értékeként, vagy csak inicializálásához. Vizsgáltam, hogy a saját adatokon hangolt hálóval elért pontosság hogyan viszonyul a transfer learning során elérhető eredményhez. A transfer leraning fordított alkalmazása esetén láthatjuk ugyanis, hogy a saját mintáinkra hangolt tanítás hogyan tud teljesíteni más adatokon.

4. Végzett vizsgálatok

4.1. Random találgatáshoz tartozó szignifikancia számítása

A random találgatásnál, azaz véletlenszerű osztályválasztáshoz tartozó eredménynél szignifikánsan jobb eredmény megnevezéséhez szükségünk van az ehhez tartozó alapvonalat kiszámolni a pontosságban, illetve a helyesen osztályozott minták számában. Jól lehet ugyanis, hogy 30-30 minta esetén 50% eséllyel tudjuk eltalálni egy mintának a kategóriáját, ami 60 minta esetén várhatóan 30 sikeres találatot jelent, de ahhoz, hogy egy osztályozó kapott pontossága szignifikánsan jobb legyen ennél, ahhoz ki kell számolnunk az ezen szignifikanciaszinthez tartozó maximális véletlen találati számot [31]. Ezen szám alatti sikeres találatok valószínűségeit felösszegezve az eredmény szignifikanciaszintjét kapjuk meg. Két osztály esetén ehhez a binomiális eloszlást kell használnunk. N minta, p találati esély, és maximálisan k sikeres találatra ez a valószínűség a következőképpen számolható:

$$F(N,k,p) = \sum_{i=0}^{k} \binom{N}{i} p^i (1-p)^{n-i} \tag{8}$$

A fenti képlettel tehát meghatározhatjuk, hogy kívánt szignifikancia szinthez mekkora k érték tartozik, aminél nagyobb számú sikeresen osztályozott minta esetén a kérdéses osztályozó a random találgatásnál szignifikánsan jobb.

50%-os találati eséllyel a 60 (in-house), illetve a 92 mintás (public) adathalmazra 95%-os szignifikanciaszintet választva az alábbi szignifikanciához tartozó sikeres találati számokat kapjuk:

mintaszám (N)	**maximális sikeres találat (k)**	**valószínűség**	**pontosság**
60	36	0,954	60%
92	54	0,962	58,7%

1. táblázat

Azaz vizsgálataim során a random találgatásnál szinginfikánsan jobb eredményhez 36, illetve 54 sikeres osztályozás szükséges a saját, illetve publikus adatok esetén, ami rendre 60 és 58,7 %-os pontosságot jelent.

4.2. Globális mérőszámok vizsgálata

4.2.1. Intenzitások átlagértéke és szórása

A tanítás során az alkalmazott modell kapcsán szempont az egyszerűség. Minél egyszerűbb modellt szeretnénk azonban használni, annál világosabban kell meghatároznunk azokat a jellemzőket, melyeken a tanítást végezni kívánjuk. Ezt a folyamatot feature extractingnak is nevezik az angol szakirodalomban [32]. Digitális kép esetén a primer adataink a képek voxeljeinek értéke. A konvolúciós hálók nagy erőssége, hogy ezen nyers adatból a tanítás során az egyes rétegekben a kép olyan reprezentációit hozza létre, mely az osztályozást adott esetben megkönnyítik. Így a neurális háló egyszerre végzi el a feature extracting valamint a klasszifikáció műveletét a tanítás során. Amennyiben a képpel kapcsolatban meg tudnunk határozni általunk definiált jellemzőket, egyszerűbb modell használata is szóba jöhet.

Első, talán alapvető ötletünk is lehet, hogy a képek átlagintenzitását, vagy a voxelértékek szórását tekintsük globális jellemzőknek. A 8. ábra szemlélteti in-house adataimban szereplő 60 páciens felvételeinek adott időszeletben vett 3D-s fMRI képének átlagintenzitását.

A két alcsoporton (30-30 elemszámú) végzett F-próba nem mutatott ki szignifikáns különbséget a minták szórásai között ($p=0.32$). Ezek alapján a számolt kétszélű T-próba a két alcsoport intenzitás átlagértékei között nem adott szignifikáns eltérést ($p=0.44$).

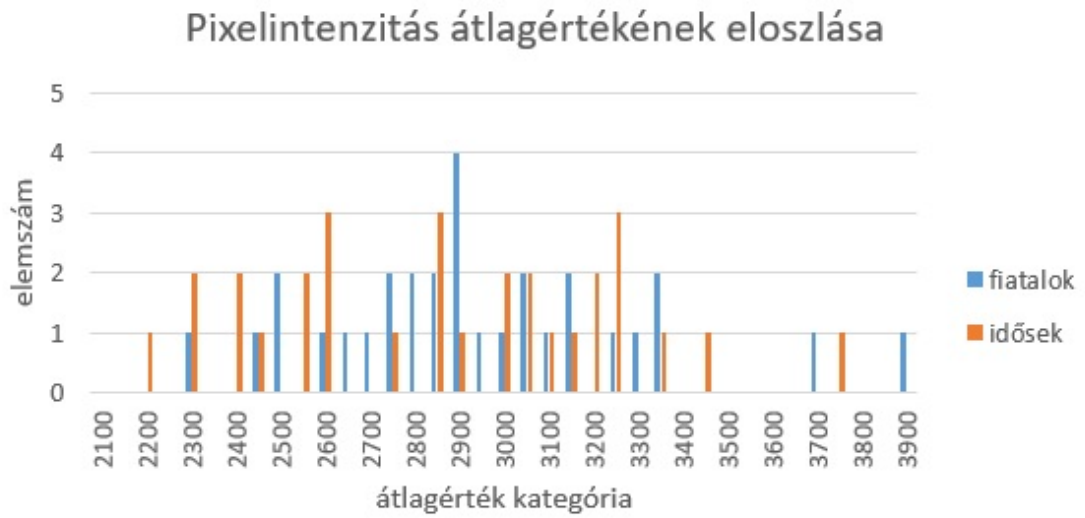
Hasonlóan látható a képek intenzitásainak szórásai a 9. ábra a két alcsoportra.

Az értékek szórása F-próba alapján ebben az esetben sem tért el szignifikánsan ($p=0.054$), így a két alcsoportot kétszélű T-próbával is megvizsgáltam. A próba alapján nincs szignifikáns eltérés a két alcsoport adatai között ($p=0.54$).

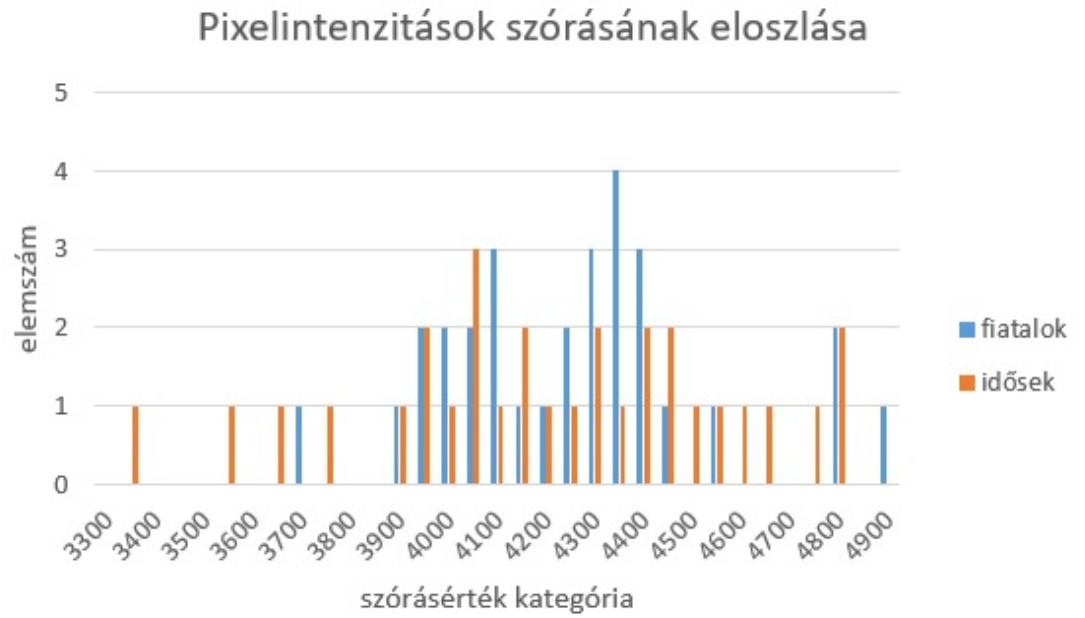
Tehát a képek voxelintenzitásainak átlagértéke és szórása sem lehet a gépi tanulás alapját képező globális jellemző, ezen értékek alapján az életkori kategorizálás nem lehetséges. A későbbiekben, a konvolúciós hálókkal való tanításhoz ezeket az értékeket standardizáljuk.

Megjegyzés, hogy az átlagértékeknél többször nagyobb szórás oka, hogy az átlagszámításban természetesen az agyon kívüli, közel nulla intenzitású voxelek is beletartoztak, melyek a

kép egy jelentős hányadát tették ki.



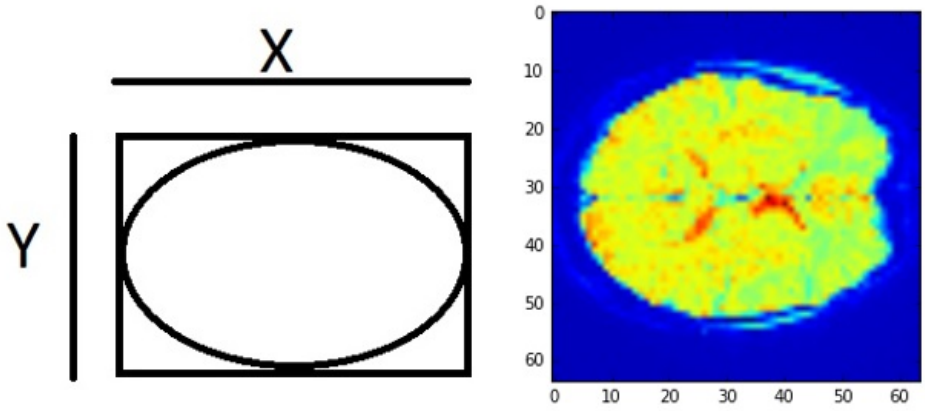
8. ábra. Voxelintenzitások értékének eloszlása n=60 alanyra



9. ábra. Voxelintenzitások szórásának eloszlása n=60 alanyra

4.2.2. Az agytérfogat méretei

Az MRIcron szoftver segítségével vizuálisan vizsgálva az in-house adatokat merült fel bennem további globális mérőszám bevezetése, az agy szürkeállománya által határolt térfogat, azaz az agy külső széleinek mérete. Első közelítésben gondolhatunk az agyra, mint transzverzális síkban egy ellipszis. Ekkor a hossztengely, valamint a szélesség, esetleg a kettő aránya jól jellemzi az ellipszis és így az agy lapítottságát.

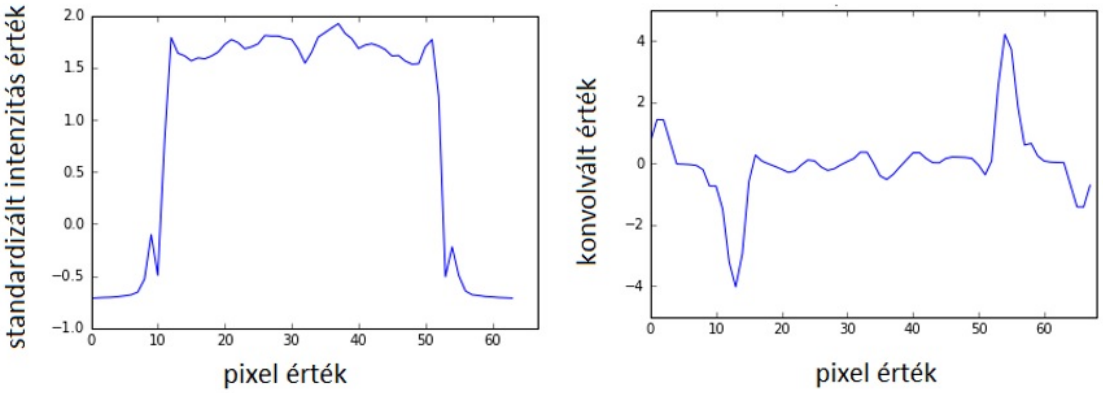


10. ábra. Az agy transzverzális síkmetszetének közelítése ellipszissel (balra), valamint egy páciens transzverzális agyszeletének alul mintavételezett képe (jobbra)

Ez alapján vezettem be az alábbi mérőszámokat: az agy hosszirányú, valamint széltében vett maximális mérete, illetve ennek a két értéknek a hányadosa. Az alábbi mérési eljárást állítottam fel ezen mennyiségek algoritmizálható méréséhez.

A 3D-s képen végigiteráltam az egyes x, valamint y irányú sorokon, és kis környezetükre kiszámoltam a vonalak mentén az átlagértékeket. (a kis környezet x irányban az alul mintavételezett képeken – 64x64x28 voxel – az y=iteráció_y:(iteráció_y+7) valamint a z=iteráció_z:(iteráció_z+5) értékeket jelentette).

A kapott egydimenziós tömböt élkeresővel (v=[-1,-1,0,1,1]) konvolváltam, majd az argumentum szerinti minimum és maximum helyeket elmentettem egy változóba. A képen történő végig iterálás után a maximum és minimumhelyeket tartalmazó vektorok maximumát és minimumát véve rendre, ezek különbsége adta az adott irányban a keresett méretet.



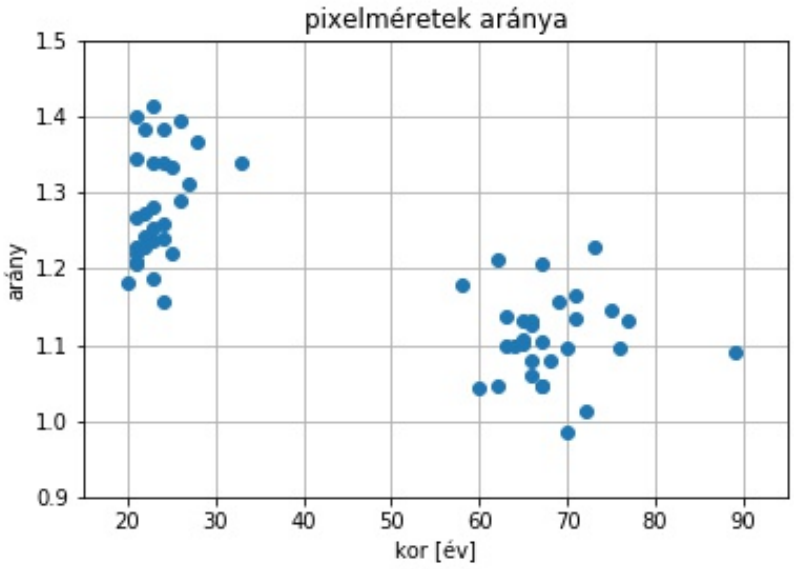
11. ábra. Egy vonal kis környezetére vett átlagos voxelérték (balra), valamint ennek konvolváltja az élkereső vektorral (jobbra)

Először F-próbával megvizsgáltam, hogy az idősek és fiatalok esetén ezen mennyiségek szórása megegyezőnek tekinthető-e, és annak függvényében végeztem T-próbát az adatokon. Több esetben is szignifikáns eltérést kaptam az idősek és fiatalok között a fenti mérőszámok esetén mind saját in-house, mind pedig a publikus adatokra. Ezeket a különbségeket (p értékeket) foglalja össze az alábbi táblázat.

P értékek	x méret [voxel]	y méret [voxel]	y/x arány
in-house	$\mathbf{2 * 10^{-5}}$	**0,004**	$\mathbf{4 * 10^{-10}}$
public	**0,0004**	0,84	**0,009**

2. táblázat

Látható, hogy a legtöbb esetben szignifikáns volt az eltérés, noha publikus adatokon kevésbé. A 12. ábrán az in-house adatok esetén a méretek arányának eltérése van szemléltetve fiatal és idős páciensekre. Az in-house adatok multiband felvételi módja érzékenyebb a szuszceptibilitás artefaktumra, ez lehet az egyik oka annak, hogy a fenti paraméterekben nagyobb eltérés mutatkozik a public adatokhoz képest.



12. ábra. x és y irányú méretek aránya in-house adatok esetén

4.2.3. Multilayer perceptron alkalmazása az agytérfogat méreteire

Bár az eltérés okát lehet keresni individuális variabilitásban, a nemek közötti eltérésben, mégis az idősek és fiatalok közötti szignifikáns különbség miatt megpróbáltam csak ezen néhány bemenő paraméteren (x méret, y méret, y/x arány) felügyelt tanítást végezni. Két rejtett réteget tartalmazó (64 és 32 neuronnal) perceptront használtam ReLU aktivációval, a kimeneten pedig softmax() függvénnyel számolva kaptam az adott életkori kategóriába tartozás valószínűségét.

Keresztvalidációnál érdemes annyiszoros értéket választani, ami osztója a teljes mintaszámnak, mert akkor minden körben ugyanannyi validációs és tanító adatunk lesz, így az egyes esetekben mért pontosságok átlaga lesz a végleges pontosság, további súlyozás bevezetése nem szükséges. Figyelembe véve, hogy kellő számú minta legyen a validációs halmazban is, de maradjon elegendő a tanító adatok között is, ezért in-house adatok (60 minta) esetében 10- és 6-szoros keresztvaliációt, míg public adatokra (92 minta) 4-szereset használtam. A tanítást a keresztvalidáció minden foldjára lefuttattam, majd megismételtem 10-szer más keresztvalidációs felosztás esetén. Így kaphattam meg a tanításokban elért pontosság szórását a 10 különböző futtatásra. Az egyes foldokban

elért pontosság kvantáltságát a foldban szereplő validációs adatok száma adja meg, így a foldok pontossága közötti szórást nem érdemes számolni, mert adott esetben pusztán a validációs adatok kis száma növelni tudja. Ezek alapján az in-house és public adatokra az agyméretek alapján a fenti multilayer perceptronnal az alábbi pontosságokat tudtam elérni (egész százalékra kerekítve).

adatok	**keresztvalidáció**	**elért pontosság**
in-house	10-szeres	85%
in-house	6-szoros	85%
public	4-szeres	70%

3. táblázat

In-house adatok esetén tehát a random találgatásnál szignifikánsan jobb pontosság 60%, míg a public adatok esetén 58,7 %. Látható, hogy ezeknél a véletlen tippeléseknél mindkét esetben szignifikánsan jobb eredményt tudtunk elérni pusztán a bevezetett három mérőszám segítségével. Ennek lehetséges magyarázata, hogy az x irányban az agy mérete a homloklebeny környékén a legnagyobb. Ezen rész előtt található a homloküreg, mely idősekben nagyobb méretű, így nagyobb árnyék fog megjelenni artefaktként az fMRI felvételen, kitakarva egy részt a homloklebenyből. A funkcionális MRI képeken ugyanis a hirtelen szuszceptibilitás változások torzítási és jelkiesés műtermékeket okoznak, így a levegővel telt térfogatok határán is jelentősebb árnyékot kaphatunk [11].

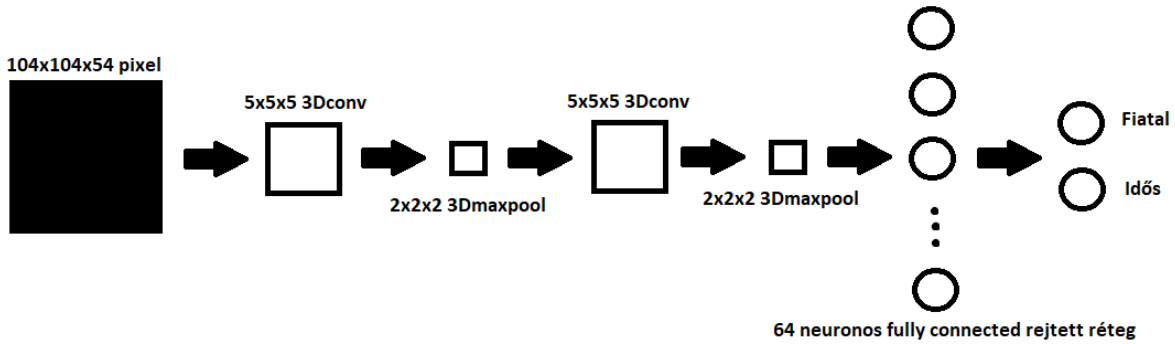
A definiált méretek alapján tehát lehet osztályozást végezni saját adatokon nagyobb pontossággal, publikus adatokon kisebbel, de még mindig jobbal, mint a random találgatás. A konvolúcós hálók módszere ezt a jellemzőt implicite módon ismeri fel továbbiak mellett, így nem igényli az általam végzett feature extractingot, valamint a nyers adatok további jellemzőit is figyelembe tudja venni a konvolúciós rétegekben létrehozott reprezentációin a képnek. Ennek pontosság javító hatását a következő fejezetben fogjuk látni. Előzetesen megjegyzem, hogy mivel az agy méretének meghatározásához kvázi a nagyobb intenzitásértékű voxelek alkotta terület határait, azaz éleket kerestem, ez gyakorlatilag egy fix súlyokkal való konvolúciós háló adta feature extracting, ami az elért magas pontosság miatt alátámasztja a konvolúciós hálók használatának indokoltságát, mely több

rétegben is akár, s tetszőleges irányokban képes hasonló tulajdonságok felismerésére a fix súlyokkal szemben, ezáltal is javítva a várható pontosságot.

4.3. Konvolúciós neurális háló tanítása klasszifikációra

A konvolúciós neurális hálóval is az idős, valamint fiatal életkori kategóriákban történő predikciót tűztem ki első célként. A háló felépítésénél Yann LeChun LeNet nevű neurális hálóját vettem alapul [27]. Ezek alapján a háló felépítése a következő volt: egy konvolúciós réteg ReLU aktivációval, majd egy maxpool réteg a dimenzióredukció érdekében. Ezután ismét egy konvolúciós réteg következett ReLU aktivációval, majd még egy maxpool művelet. A kapott réteget egydimenziós többé átformálva ezután az osztályozáshoz egy fully connected rejtett réteg következett ugyancsak ReLU aktivációval, majd a kimeneti két osztályhoz tartozó fully connected réteg a keresztentrópia számításához alkalmas softmax aktivációval. A háló felépítése a 13. ábrán is látható.

Kiindulási értékként mindkét konvolúciós rétegben 5 darab 5 voxel oldalhosszúságú kocka alakú tömböt használtam kernelnek, az eredeti kép széleit 0-ákkal úgy kiegészítve, hogy a konvolúció után is az eredetivel megegyező méretű tömböket kapjak a következő rétegben. A maxpool művelethez 2x2x2-es kernelt használtam 2-es lépésközzel, így minden irányban felezni tudtam a tömbök méretét. A második dimenzióredukálás és átalakítás után ezáltal az egydimenziós tömböm 9464 méretű volt. A fully connected rétegben 64 neuront, míg a kimeneti osztályozó réteg esetén a két életkori kategóriához két neuron használtam.



13. ábra. Klasszifikációhoz használt konvolúciós neurális háló vázlata

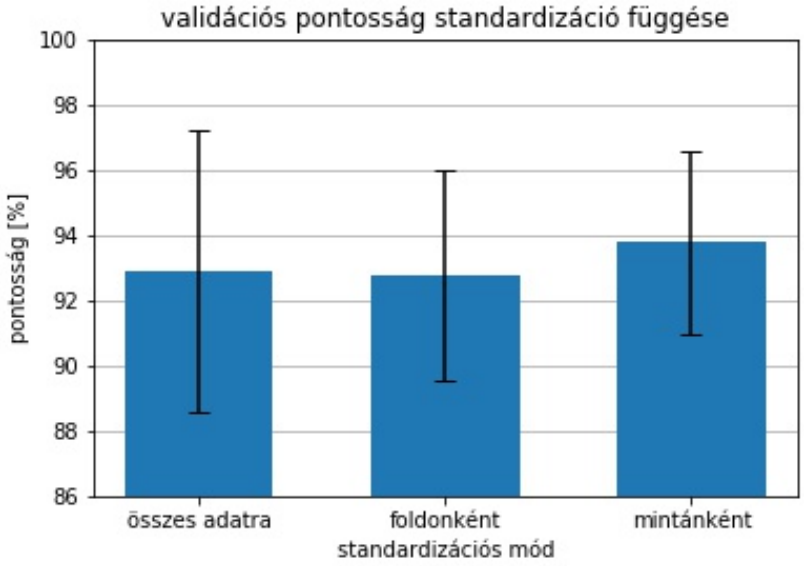
Az in house adatok esetén minden pácienstől 846 időpillanatban állt rendelkezésemre 3D-s felvétel. Minden nyolcadikat numpy tömbbé konvertálva 100 mintám volt egy alanytól. A nagy adatmátrixok és korábbi megfontolás alapján a tanítást a sztochasztikus gradiens módszerrel kezdtem, minden batch 4 mintát tartalmazott. A keresztvalidációhoz a pácienseket minden foldban tanító és validációs csoportba osztottam, majd az adott lépésben aktuális batch generálásához a tanító adatok következő 4 páciensének a 100 rendelkezésre álló mintájából sorsoltam ki egyet-egyet. Így minden batch különböző alanyok adatait tartalmazta, további variabilitást kapva a különböző időszeletek alkalmazása révén. Emellett csak tanító halmazból használtam mintákat a tanításhoz értelem szerűen, hasonlóan a validációs esetben.

A bemenő adatokat ahogy korábban láttuk, standardizálni kell. A voxelintenzitásokból kivont átlag és a leosztáshoz használt szórásértéket úgy kell meghatározni, hogy a validációs, vagy tesztadatok ne legyenek rájuk befolyással, hiszen akkor a tanítás során felhasználná a háló ezt az információt a validációs adatokról, amire elvileg képes lenne tanulni. A képek átlagintenzitásának és szórásának korábban látott eloszlása miatt ez a hatás kérdéses, hogy egyáltalán kimutatható-e. Másrészt a keresztvalidáció miatt egy tanítás során változik a tanító és validációs adatok összetétele, ezért amennyiben csak a tanító adatok alapján szeretnénk számolni a standardizáláshoz használt átlagot és szórást, akkor ezt el kellene végeznünk minden keresztvalidációs fold előtt, minden tanítás során. Ez a futási időt drasztikusan növeli. Ehelyett szeretnénk, hogy valamiféleképpen előre normált tömböket kelljen betöltenie a programunknak.

Az egyik lehetőségem az volt, hogy az összes adat alapján számoltam a standardizáláshoz használt átlagot és szórást, a másik, hogy minden egyes mintát az adott kép voxeljeinek átlagintenzitásával, valamint az intenzitások szórásával standardizálom. Az előbbi éppen az az eset, amikor validációs adatok információit használnánk fel a tanító adatok,s ezáltal a tanítás befolyásolására. Ezt a lehetőséget éppen ezért csak összevetettem a többivel. A korábbi foldonkénti, valamint ezen két standardizációs módszer használatával is megnéztem, hogy milyen eredményt kapok pontosság tekintetében.

4-4 tanítást végeztem mind a három eljárás esetében, a kapott végső pontosságok és azok tanítások közötti szórásai láthatóak a 14. ábrán. Szignifikáns eltérés nem mutatató

ki a módszerek között, így a futási idő, és az elkerülni kívánt előzetes tudás transzfer miatt a mintánkénti standardizálást is használható, ahol a numpy tömböket előre standardizálva tudjuk elmenteni, tanításnál csak ezeket a tömböket kell betölteni.



14. ábra. Pontosság különböző standardizációs módok esetén

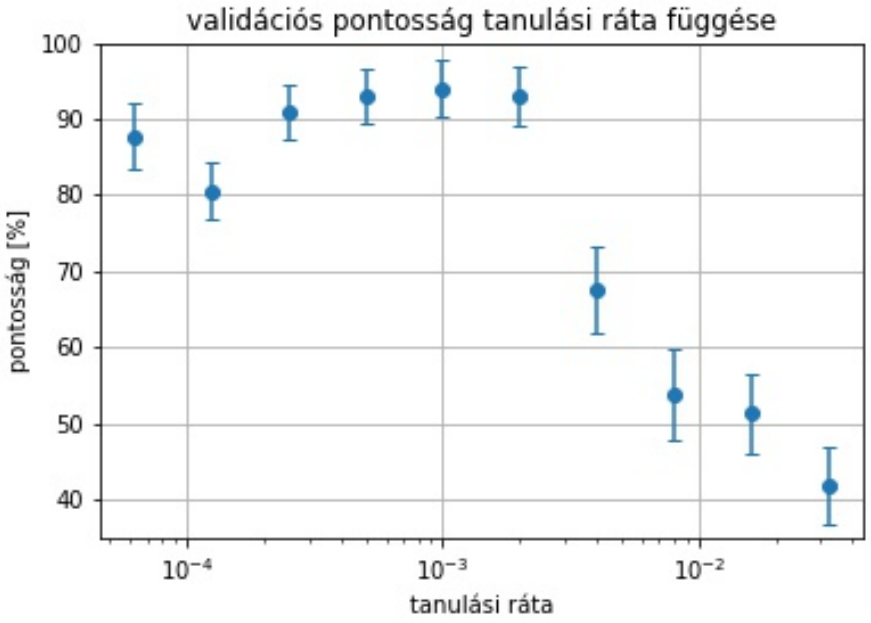
Míg az előre elvégzett standardizálás mellett mindkét esetben a tanítás lépései és a keresztvalidációs foldok közötti váltás is 1 másodpercen belül történt, addig a foldonkénti standardizálásnál a tanítás lépései között noha szintén 1 másodperc alatti idő telt el, azonban a foldok között már több, mint 5 perc.

4.4. Hiperparaméterek vizsgálata

4.4.1. Tanulási ráta

A hiperparaméterek különböző értékei kapcsán vizsgáltam, hogy milyen pontosságot tudok elérni a validációs adatokon. Először a sztochasztikus gradiens módszer tanulási rátáját néztem meg, mint a legalapvetőbb hiperparamétert. Ennek értékét 0,032 és 0,0000625 között változtattam mindig a korábbi értéket felezve, összesen 10 esetet vizsgálva. A tanítást addig végeztem, míg a validációs adatokon mért pontosság már nem változott trendszerűen, ekkor az utolsó 50 értékre vett átlaggal becsültem az elért pontosságot.

Az eredményeket a 15. ábrán foglaltam össze. A bizonytalanságokat a standardizációs tanítás során a tanítások között számolt szórás és a tanulási ráta számolásához használt értékek szórásának négyzetösszegéből vont gyökkel közelítettem.



15. ábra. Validációs pontosság függése a tanulási rátától

Látható, hogy 0,002 értéknél nagyobb tanulási rátákra a validációs pontosság drasztikusan lecsökken a sztochasztikus gradiens módszer esetén. Továbbá a 0,0005 érték alatt is tapasztalható egy visszaesés, aminek az oka lokális minimumban történő megakadás lehet, amiből a kis ráta miatt az optimalizáció során a program nem tud kijutni. Az eredmények fényében a 0,001-es értéket választottam a továbbiakban. Adaptív tanulási rátát használó optimalizációs módszer esetén a későbbiekben pedig ezen értékkel inicializáltam ezt a hiperparamétert.

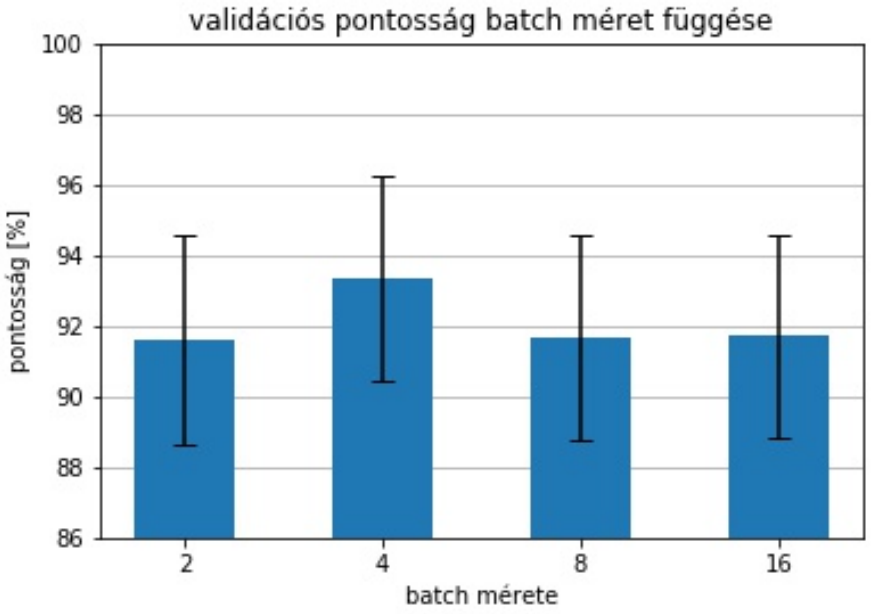
4.4.2. Batch méret

A sztochasztikus gradiens módszer egyik hiperparamétere az egy lépében használt minták száma, azaz a batch mérete. Ezen adatok alapján számoljuk ugyanis egy lépésben a gradiens értékét, és frissítjük a súlyokat. A párhuzamosíthatóság érdekében javasolt a batch méretét 2 valamely hatványaként megválasztani. Én a [2,4,8,16] értékekre

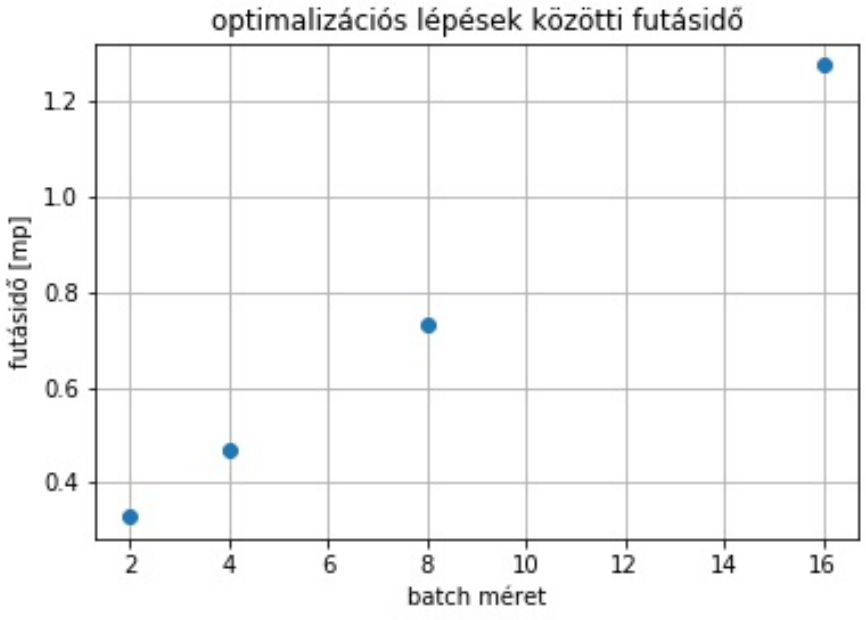
vizsgáltam meg, hogy miként változik a validációs adatokon mérhető pontosság 0,001-es tanulási rátával 1001 lépés után (16.ábra), valamint az eltérő batchméret okozta futásidő-különbség (17. ábra).

A pontosságok bizonytalanságát a futtatás során az utolsó 50 lépés értékeinek szórásából (ez nagyon közel volt 0-hoz minden esetben), valamint egy független 10 tanításból álló sorozatban elért pontosságok szórásából becsültem a négyzetösszegből vont gyök értékével.

Ilyen bizonytalanságok esetén a pontosságértékek között szignifikáns eltérés nem mutatható ki. A futásidő mérésén az iterációs lépések között eltelő időt értettem, ez közel lineárisan változott, de nem egységnyi meredekséggel. A végső pontossághoz konvergálás gyorsasága a batch méretének növelésével kis mértékben csökkent. A pontosság közel méretfüggetlensége miatt, és a futásidő csökkentésének kis értéken tartásának figyelembe vételével végül a 4-es mérettel dolgoztam tovább.



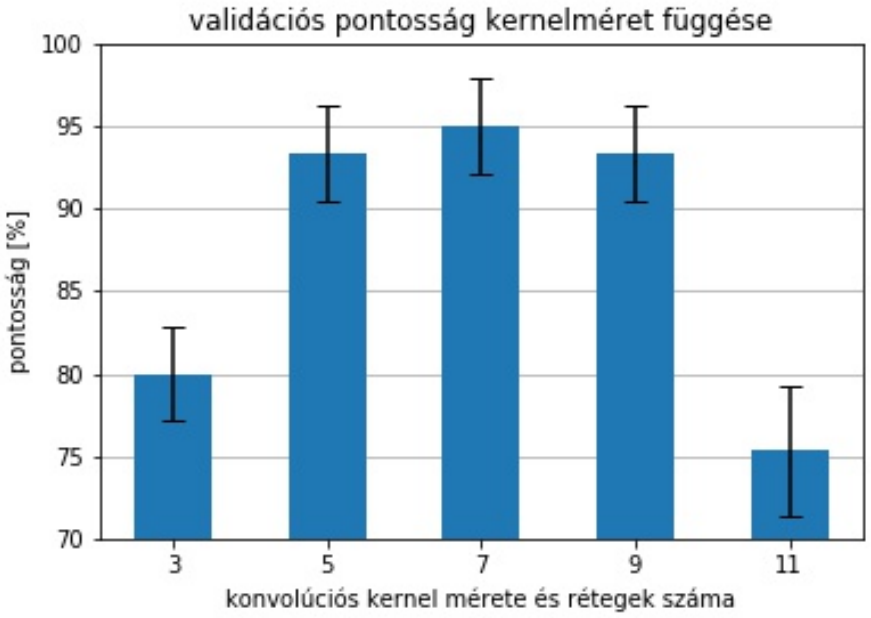
16. ábra. Validációs pontosság különböző batch méretekre



17. ábra. Iterációs lépések közötti futásidő különböző batch méretekre

4.4.3. Konvolúciós kernelek mérete és száma

A konvolúciós kernel méretének és számának növelésével a hálónk súlyainak számát növeljük, ezáltal a modellünk komplexitását. Ez adott esetben túlillesztéshez vezethet. Másrészt nagyobb kernelméret még távolabbi szomszédsági kapcsolatokat is fel tud térképezni. Ezen potenciális trade-off miatt érdemes vizsgálni, hogy mekkora mérettel dolgozzunk. Olyan vizsgálatot végeztem, ahol a kernelméretet és a rejtett rétegben a kernelek számát egységesen változtattam 3,5,7,9, és 11 értékek között. A nagyobb kernelméretek miatt 1500 iterációs lépést futtattam, hogy a nagyobb neurális hálók estén is konvergáljanak a pontosság értékek. Az ily módon kapott validációs pontosságok láthatóak a 18. ábrán. A pontosságok bizonytalanságát ebben az esetben is a futtatás utolsó 50 lépésében a pontosságadatok szórásából (ez csak a 11-es méretre volt jelentős), valamint a független 10 futtatásban elért végső pontosságok szórásának négyzetösszegéből vont gyökkel becsültem. A kernelek méretének és számának együttes változtatása hatványszerű növekedést eredményezett a futásidőben, ezért közel azonos pontosság esetén a kisebb háló használata az előnyösebb.



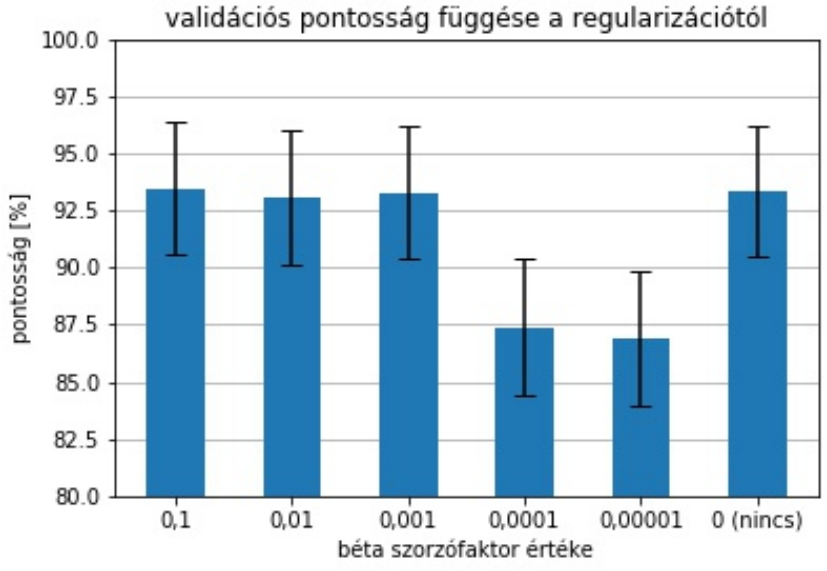
18. ábra. Valdiációs pontosság különböző konvolúciós kernel különböző méretei és számosságai esetén

A 3-as méret esetén a költségfüggvény a 400. lépéstől kezdve egy 0,23 és 0,24 közötti sávban változott, azaz közel konstans maradt. Hasonlóan a validációs pontosság sem változott, a tanító batch-on mérthető, keresztvalidációs foldokra átlagolt pontosság viszont folyamatosan oszcillált 70 és 95 % között, arra lehetett következtetni, hogy a háló nem képes illeszkedni megfelelően még a tanító adatokra sem, alulillesztjük a problémát. 5, 7, 9-es kernelméret esetén a tanító batch-eken mért pontosság a tanítás második felére a 100 %-hoz konvergált, hasonlóan a validációs pontosság is a fenti ábrán látható értékekhez, az utolsó 50 lépésben ezek az értékek már nem változtak. 11-es méretnél a validációs pontosság nem egy értékhez konvergált, hanem oszcilláció maradt 72 és 82 % közötti értékekkel.

4.4.4. Regularizáció a súlyok L2 normájával és dropout

Standardizált bemenet és skálázott kimenet esetén a súlyok négyzetösszegének a költségfüggvényünkhöz való hozzáadása egy szorzófaktor segítségével tehát segíthet elkerülni a túlillesztést. A módszer hiperparamétere a szorzófaktor, melyet béta értéknek nevez-

tem (a szakirodalomban gyakran λ-val jelölt mennyiség [16][24][25]). Ennek a bétának az értékét változtatva (0,1;0,01;0,001;0,0001 és 0,00001 értékek között) vizsgáltam a validációs pontosságra gyakorolt hatást. Az eredményeket a 19. ábrán foglaltam össze, ahol az elért pontosság bizonytalanságát a korábbiakhoz hasonlóan az ismételt tanításokban kapott közös bizonytalanságból és az egyes béta értékekkel való tanítás utolsó 50 iterációs lépésében kapott pontosság szórásából becsültem.

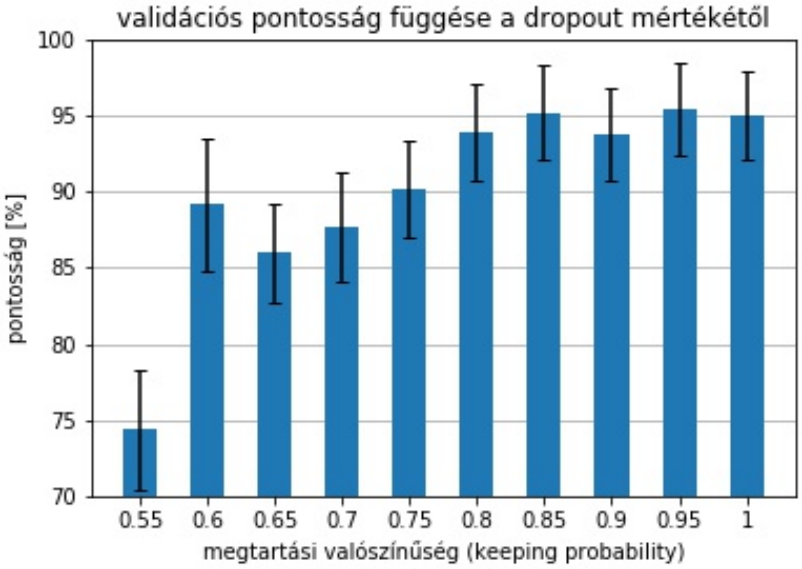


19. ábra. Az L2-es normával való regularizáció hatása a validációs pontosságra

Látható, hogy nagyobb béta értékek mellett a regularizáció nélküli esethez közeli pontosságot értünk el, míg a 0-hoz közelebbi bétákra marad el a teljesítménymutató. Ennek lehetséges magyarázata, hogy kis béták használatakor a háló a béta nélküli esethez hasonló megoldást talál, azonban az optimalizációba bevett extra tag miatt az elért pontosság enyhén csökken. Eközben nagyobb bétákra a háló egy esetlegesen más megoldásra tanul rá, így a költségfüggvényben megadott nagyobb regularizáció ellenére is a béta nélküli eset pontossága megközelíthető. A várakozásunknak nem teljesen megfelelő eredmény oka lehet a regularizáció iránti szükség hiánya ebben az esetben, a kis bétákra vonatkozó bizonytalan eredmény miatt ezért a súlyparaméterek L2-es normájával való regularizációt a továbbiakban nem alkalmaztam.

Dropout esetén az egyes neuronokat csak adott valószínűséggel tartjuk meg a költségfüggvény

számításánál, egyébként pedig lenullázzuk őket. Azt vizsgáltam, hogy adott megtartási valószínűség mellett (keeping probability – keep_prob), hogyan alakul az elért pontosság (20. ábra). Ehhez 5-ös kernelméretet használtam.

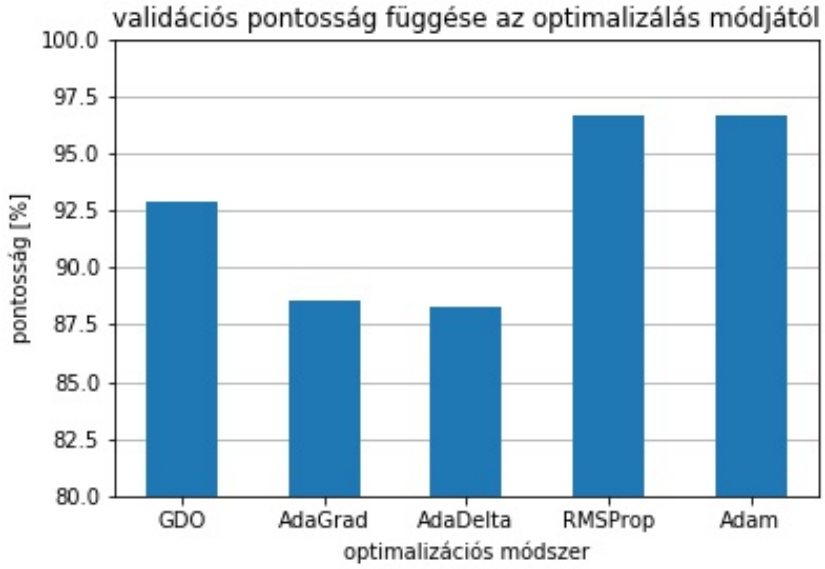


20. ábra. Dropout hatása a validációs pontosságra

Látható, hogy a drasztikusabb elvetés esetén a pontosság csökken, a végső pontosság bizonytalansága pedig nő. A pontosság csökkenését a kernelméret csökkentése esetén is láthattuk a 3-as kernelméretre, a 0,55-ös megtartási valószínűség mellett a megtartott neuronok száma közelít a 3-as kernelméret neuronjainak számához (fontos különbség, hogy minden lépésben más neuronokat tartunk jelen esetben meg). A szakirodalom alapján mondható, hogy kis elemű tanító adatbázison való tanításhoz alkalmazott dropout esetében az elért pontosság javulásának hatása nem garantált [26], egyes esetekben kontraproduktív is lehet. Noha a modell robosztusságát növelheti a módszer, de a pontosságot nem tudta, ezért jelen helyzetben nem láttam indokoltnak a dropout használatát az adott hálóméret mellett.

4.4.5. Optimalizáló algoritmus

Tensorflow környezetben elérthetőek különböző optimalizáló algoritmusok a súlyok frissítéséhez, ahogy ezt korábban írtam, ezek célja, hogy az alapvető gradiens módszerben rejlő veszélyeket, mint például a lokális minimumba való jutást elkerüljék, és gyorsabb konvergenciát tegyenek lehetővé. Megvizsgáltam néhányat ezek közül, hogy a melyik a leghatékonyabb a problémám kapcsán. Minden esetben a korábban használt 0,001 értékű tanulási rátával inicializáltam az optimalizáló függvényeket, a többi inicializálandó paraméternél, amennyiben volt, az alapbeállítást használtam. A validációs adatokon elért pontosságokat tüntettem fel a 21. ábrán.



21. ábra. Validációs pontosság különböző optimalizáló algoritmusokra

Az utolsó kettő közül az Adam módszer kombinálja az Adagrad és az RMSProp előnyös tulajdonságait [33], ezért az Adam optimalizáló használata lenne indokolt. A magas pontosságértékek esetén különböző futások közötti szórás értelmezése problémás lehet, hiszen a mérhető pontosságnak van egy felső határa. Különböző módszerek szóráson alapuló t-próbás összehasonlításának relevanciája ezért kérdéses, így más statisztikai összevetéshez érdemes folyamodni. Az egyes esetekben az eltérő, valamint ezeken belül a másik eljárással szemben sikeres osztályozások számain alapuló módszerrel vizsgáltam ezért a GDO és

RMSProp, valamint a GDO és Adam optimalizálók közötti különbséget [34].

Ennek érdekében közös 10-szeres keresztvalidációs adatfelosztás mellett végeztem tanítást 1000 lépésig mindhárom optimalizáló esetén. Ezúttal a pontosság mellett az validációs minták osztályokba tartozásának becsült valószínűségét, és ezáltal a becsült osztály adatait is exportálva. A hivatkozásban említett cikk módszertanával ezek alapján a GDO és az RMSProp közötti különbség szignifikanciájára p=0,16, míg a GDO és az Adam optimalizáló között p=0,015 értéket kaptam. Mivel a mintaszámom kicsi, és ezen belül az eltérő osztályozások száma töredék volt, ezért a kiértékelés nagyon érzékeny az egymáshoz képesti sikeres osztályozások számaira. A tanítás eredményeként az RMSProp optimalizáló 2 esetben tévedett, de ezekben az esetekben a GDO helyesen osztályozott, míg Adam esetében egy helytelen osztályozást kaptam, amit a GDO is elvétett. Így jöhetett ki a közel azonos pontosság esetén is a szignifikanciában látható jelentős különbség.

Ezek fényében nagyobb mintaszámú vizsgálat is fontos lenne, de a meglevő eredmények ismeretében az Adam optimalizálót választottam a további munkához.

Az előzőek alapján tehát az in-house adatokon való tanításhoz az alábbi hiperparaméterek használata optimális. Tanulási rátának a 0,001-es érték, illetve adaptív optimalizáció esetén az ezzel történő inicializálás. A tanulási ráta kisebb értéke is megfelelő pontosságot eredményezhet a 15. ábráról leolvasható mértékig, a nagyobb ráta a gyorsabb kezdeti konvergenciát segíti. Batch méretnek a 4-es mintaszámot javaslom, noha ebben az esetben a nagyobb elemszám ellen pusztán a futásidő szól, pontosságcsökkentő hatását nem tudtam megfigyelni. Kernelméret és kernelszámok esetén az 5,7,9 értékek közel azonos pontosságot értek el, a futásidő itt is a kisebb érték használatát indokolja. Optimalizációnak az Adam módszer ajánlott, kiegészítő dropout és súlyértékek L2 normájával történő regularizáció alkalmazása nem szükséges.

4.5. Transfer Learning klasszifikációhoz

Transzfer learning módszer [29] egyik esete, mikor egy adott adatdoménen megtanult paraméterekkel kapcsolatos tudást átviszünk egy másik adatdoménen történő tanítás paramétereire. Jelentheti ez azt, hogy a második adatcsoporton való tanításhoz a paramétereink inicializálásához a korábban megtanult paraméterértékeket használjuk, de

akár közvetlen a klasszifikációhoz is használhatjuk rögzítetten ezeket az értékeket. Emellett a paramétereink több csoportba osztása esetén az egyes csoportokban a random, vagy korábban tanult paraméterekkel történő inicializálás, valamint a transfer learning során átvett súlyok rögzített értéken tartását tetszőleges kombinációban alkalmazhatjuk [30], és vizsgálhatjuk, hogy melyik vezet nagyobb pontosságra a saját adatokon való klasszifikáció esetében. Ezáltal megnézhetjük, hogy a praktikusan nagyobb adatokon megtanult információ milyen módon, és mértékben tud hozzájárulni a saját adatokon történő tanításhoz.

Mivel a saját adatokon is magas pontosságot tudtam elérni, ezek után látványos javulást már nem vártam, azonban megvizsgáltam, hogy milyen pontosságot tud elérni a transfer learning módszere.

Hogy a public és in-house adatok egységesek legyenek, a gyengébb felbontású kép adatai alapján alul mintavételeztem a többi tömböt a numpy csomag beépített zoom() függvényével. Továbbá a legrosszabb felbontáshoz tartozó legkisebb voxelméretű mintákhoz méreteztem a többit azáltal, hogy a széleken elhagytam a megfelelő számú voxelt szimmetrikusan. Így jutottam (3x3x4,4) mm3-es felbontáshoz és 64x64x28-as tömbmérethez. A felbontások alapján, és nem a tömbméretek alapján történő alul mintavételezés azért volt fontos, mert előbbi művelet után a voxelszámokban az eltérés oka (ez z irányban jelentkezett) az volt, hogy a public adatok egy részénél kisebb látómezőt alkalmaztak. A voxelszámok alapján történő mintavételezés ezért torzította volna a képeket különböző mértékben, ami szisztematikus változtatást jelentett volna a képek egy alcsoportjának struktúráiban, ez meghamisíthatta volna a tanítás kimenetelét.

Először megnéztem, hogy az alul mintavételezett adataimon mekkora pontosságot tudok elérni a korábban optimálisnak talált hiperparaméterekkel. Ezek alapján az előzőekben bemutatott hálófelépítést használtam, 0,001-es kezdeti tanulási rátával, a konvolúcióhoz 5-ös kernelméretet (az alul mintavételezés ellenére – struktúrák méreteinek csökkenése – nem volt indokolt kisebb kernel használata, hisz nagyobb méret esetén a 7-es kernelméret is megfelelő volt). Dropoutot tehát nem használtam, valamit 4-es batch mérettel és 10-szeres keresztvalidációval futtattam a tanítást 1001 epochon keresztül. Eredményként 94 %-os pontosságot kaptam a szochasztikus gradiens módszerrel, ami közel megegyezik a na-

szcenárió	**konvolúciós réteg saját adatokon**	**fully connected réteg saját adatokon**
kons_konst	public adatokon kapott súlyokkal	public adatokon kapott súlyokkal
konst_tanít	public adatokon kapott súlyokkal	tanítás random inicializálással
konst_inic	public adatokon kapott súlyokkal	public adatokon kapott súlyokkal inicializálva
inic_inic	public adatokon kapott súlyokkal inicializálva	public adatokon kapott súlyokkal inicializálva
inic_tanít	public adatokon kapott súlyokkal inicializálva	tanítás random inicializálással
tanít_tanít	tanítás random inicializálással	tanítás random inicializálással

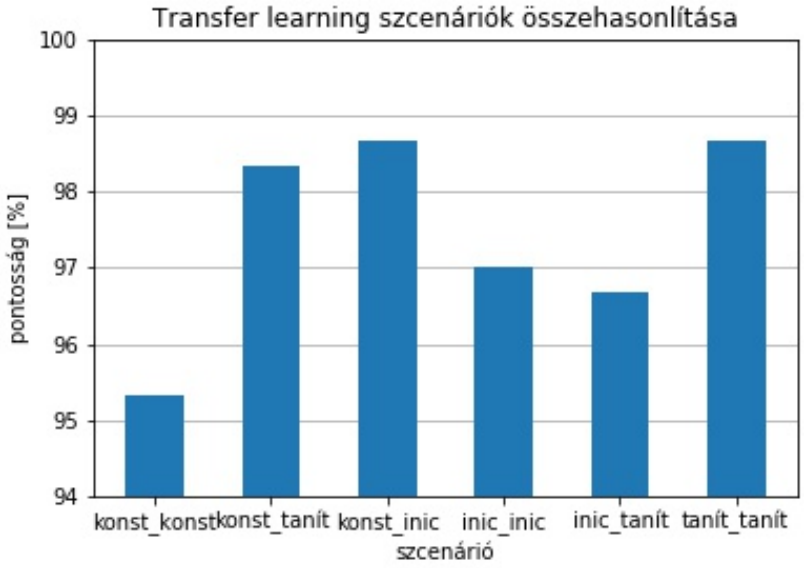
4. táblázat

gyobb méretű képen elért pontossággal. Az összevehetőség érdekében legeneráltam előre 5 különböző keresztvalidációs felosztást, majd ezeket használva végeztem vizsgálatokat. Az egyszerű tanítást megismételve Adam optimalizációval ez alkalommal 98,3 %-os pontosságot tudtam elérni.

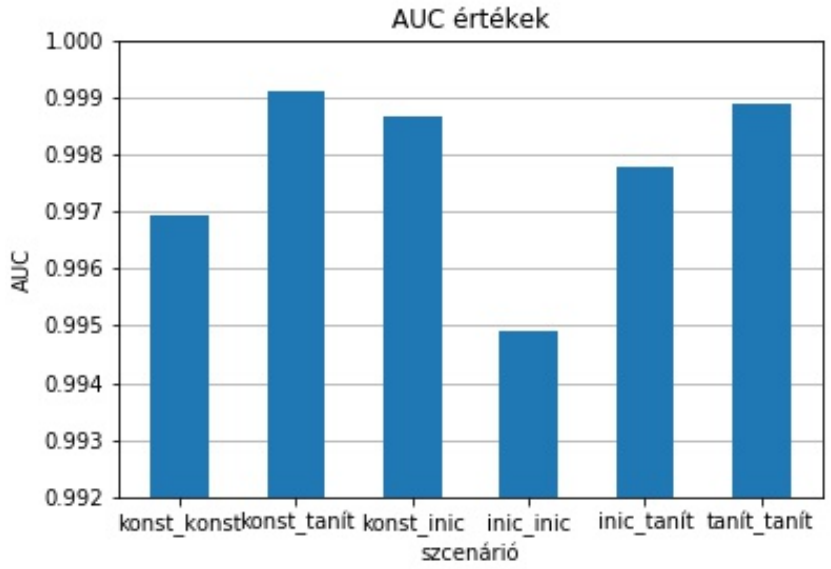
Ezek után vizsgáltam a különféle transfer learning lehetőségeket. Ebben Vakli et al. módszertanát követtem [35]. Minden esetben a 92 alanyú public adatokon végeztem előbb tanítást, majd a konvolúciós és fully connected rétegek súlyait konstansokként (konst), vagy inicializáláshoz (inic) transzferáltam a saját adatokon történő tanításhoz. Vizsgáltam olyan esetet is, amikor a háló egy részét random inicializálás után tanítottam (tanít). Egységesen Adam optimalizációt használtam 0,001-es tanulási rátával, 4 elemű batchmérettel, a korábbi hálófelépítéssel, dropout nélkül, az előtanítás esetén 1501 lépéssel, a transzferálás után 151 lépéssel. A 3. táblázatban összefoglalt eseteket vizsgáltam tehát.

Az első eset a fent említett egyszerű tanítás a saját adatokon. A szcenáriókban elért pontosságot tehát 5 alkalommal számoltam minden esetben. A pontosság mellett egy másik teljesítménymérőt is számoltam a szcenáriókban kapott predikciókra, ez az úgynevezett AUC érték (Area under the receiver operating characteristics). A one-hot kódolt címkék esetén az idős kategóriát véve pozitívnak, a predikció és a címkék tömbjeinek második oszlopából számolható a mennyiség. Én a sklearn csomag metrics.roc_auc_score() függvényt használtam az AUC érték számításához. Ebben az esetben a töredékvalószínűségek is javíthatják, vagy épp gyengíthetik a tanítás értékelését,

hiszen a pontosság csak a nagyobb valószínűséghez tartozó osztályra 1 et, a másikra 0-át véve számolható, míg az AUC számításánál azt is figyelembe vesszük, hogy még ha nem is találtuk el az osztályt, mekkora bizonytalansággal tippeltünk a helytelen címkére. A különböző futtatásokra vett átlagos validációs pontosság és AUC értékeket foglaltam össze a 22. és a 23. ábrán.



22. ábra. Transfer learning során elért pontosságok



23. ábra. A számolt AUC értékek

Látható, hogy a publikus adatokon való tanításban kapott súlyok konstansként való átvétele maradt a legalacsonyabb, noha még mindig 95 % feletti pontosságot eredményezett, a további esetekben még ennél is jobb eredmény volt elérhető, megközelítve a saját adatokon történő kizárólagos tanítást a megfelelő paraméterekkel. Bár a pontosságot a transfer learninges módszerek nem tudták tovább növelni, az AUC érték kis mértékben nőtt, azonban az optimalizátorok közötti összehasonlítás módszerével itt nem mutatható ki szignifikáns különbség a transfer learninges és a csak saját adatokon való tanítás között. Ennek oka, hogy gyakorlatilag a tanítások futtatása között egy vagy még kevesebb volt az átlagos hibázás a konst_tanít és konst_inic esetekben, a többiben is csak néhány hibázás történt, valamint az AUC értékek is igen közeliek mind egymáshoz, mint az 1-es értékhez.

Megvizsgáltam, hogy a saját adatokon tanított háló súlyainak konstansként történő alkalmazása a publikus adatokra milyen pontosságot eredményez, erre 66,1 % adódott, ami ugyan szignifikánsan jobb ezen adatok esetén a nagyobb alcsoporttal történő random találgatásnál (58,7 %), de jóval elmarad a fordított esethez képest (95,3 %). Ez mutatja a saját adatokon tanított háló gyengébb általánosító képességét.

Szemléletben a neurális hálónkat szétválasztva a konvolúciós és FC részekre, mint a feature extracting-ot és az osztályozást végő részekre tehát a konst_inic szcenárió volt

talán, amelyikben a leginkább meg tudtam közelíteni a szimpla tanít_tanít esetet mind a pontosság, mind AUC értelemben, a másik nagyon hasonló eset a konst_tanít volt. A közös bennük, hogy a konvolúciós háló súlyai változás nélkül lettek alkalmazva a saját adatokon való tanításhoz. Tehát a feature extracting kizárólag a nyilvános adatokon történt, ennek ellenére volt olyan sikeres az osztályozásra való tanítás a saját adatokon, mintha mindkét lépést a saját adatokon optimalizált hálón tanítottuk volna meg. Hogy az ellenkező irányú transfer learninget jobban össze tudjuk vetni ezzel az esettel, 3 tanítást végeztem, mely során a saját adatokon történt először a konvolúciós és FC réteg tanítása, majd a konvolúciós súlyokat konstansként áthozva, az FC súlyait pedig inicializáláshoz használva tanítottam a hálót a nyilvános adatokon osztályozni, a korábban használt hiperparaméterekkel. A tanítások során elért pontosság az előző bekezdésben bemutatott esethez képest emelkedett $(83,6 \pm 6,5)$ % értékre. Noha az eredmény szórása a korábbi vizsgálatokhoz képest magasabb volt, de a pontosság növekedése ellenére egyértelműen alacsonyabb eredményt értünk el, mint az eredeti irányban alkalmazott transfer learning esetén.

Ez azt indikálja, hogy a nyilvános adatokon megtanult jellemzők (feature-ök) jobb generalizációs képességgel rendelkeznek, mint saját adatok esetén. Tehát a transfer learning indokolt, mert a pontosság tekintetében sem teljesít rosszabbul a pusztán saját adatokon való tanításhoz képest, noha a pontosságot szignifikánsan már nem tudta javítani a saját adatokon hangolt konvolúciós hálóval történő tanításhoz képest.

4.6. Regressziós vizsgálatok

Regressziós becslő esetén a páciens életkorára tudunk direkt becslést adni. A távlati cél ennek minél pontosabb elvégzése. Korábbi vizsgálatban, mely T1 súlyozott anatómiai képeken történt, egy 2000 fős mintán, a nyers adatokon végzett tanítással 4,65 éves átlagos abszolút hibával (mean absolute error - MAE) sikerült regressziós becslőt létrehozni [1]. Megvizsgáltam, hogy a jelenleg rendelkezésemre álló adatokon, a saját konvolúciós hálómmal milyen eredményt tudok elérni. Mivel az egyébként is alacsonyabb elemszámú mintám alanyainak életkora nem egyenletesen oszlott el, hanem a középkorúaktól egy adat sem állt rendelkezésemre, ezért előzetes várakozásom szerényebb pontosság volt.

A regresszióhoz a klasszifikációra használt konvolúciós neurális hálót módosítottam oly módon, hogy a kimeneti rétegben csupán egy neuront hagytam, erre aktivációt már nem alkalmaztam. A költség függvényem az átlagos négyzetes hiba volt (MSE). A páciensek életkorát 90-es értékkel normáltam 0 és 1 közé, hisz a kimeneti rétegben ekkora nagyságrendű értéket vártam, így az optimalizáció az első néhány lépésben sem okozott túl nagy költség függvény értéket, ezáltal a minimumkeresés sikeres lehetett. Mivel a regresszió bonyolultabb feladatnak tekinthető a klasszifikációnál ebben az esetben, ha csak arra gondolunk, hogy a kimenet 0 és 1 közötti értéke alapján klasszifikációnál egy osztályhoz tartozás valószínűségét értettük, ami még nagyobb tartományon is ugyanazt az becsült osztályt adja, addig regressziónál a 90 évvel való normálás miatt a kimenet 0,011 értékkel való változása már egy évvel eltolja a becsült kort.

Vizsgálataim során több alkalommal a tanítás kezdetén a költségfüggvényem nagyon magas értékeket is felvett, a háló nehezen tudta az optimalizáció elején az optimumot közelíteni. Ennek lehet oka, hogy ehhez a feladathoz az inicializáláshoz korábban használt tanulási ráta az ideálisnál valamivel nagyobb, valamint a batch-ek elemszáma túl kicsi, és a kevés mintára számolt gradiens nem közelíti kellően az optimizáció számára ideális irányt. Éppen ezért a tanítás során a korábbiaknál valamivel kisebb, 0,0005-ös kezdeti tanulási rátával használtam az Adam optimalizálót, ezzel egy időben az iterációk számát enyhén megemeltem 2500-ra, hogy az optimumba való konvergencia megvalósulhasson. Dropoutot és más regularizációt ez alkalommal sem használtam. A batch méretét 8-nak választottam, amitől azt vártam, hogy kevésbé fluktuál a költségfüggvényünk a tanítási lépések között, mert nagyobb alhamazon számoljuk a gradienseket. A tanítás végeztével a teljesítmény jellemzésére az átlagos abszolút eltérést (MAE) számoltam, valamint a becsült és valódi életkor közötti R^2 értéket. A saját és a nyilvános adatokon is végeztem tanítást, ezeket az eredményeket foglaltam össze a 5. táblázatban.

A transfer learning hatását regresszió esetén is vizsgáltam, noha csak egy esetben. Láttuk, hogy a konvolúciós háló nyilvános adatokon történő tanítása során kapott értékeinek puszta átvételével is az FC rétegek random vagy az előzetesen kapott súlyokkal való inicializálását követő rövid tanítás után is elérhető a pusztán saját adatokon kapott pontosság. Ebből kiindulva regresszió esetén azt az esetet vizsgáltam, amikor a hálót a nyilvános

adatokon tanítva a konvolúciós súlyokat konstansként átvéve (az FC réteg súlyait pedig inicializáláshoz) a hálót a saját adatainkon tanítjuk tovább. Három tanítás átlagából kaptam eredményként a MAE-re 8,54 értéket, míg R^2-re 0,82.

adathalmaz	**MAE [év]**	$\mathbf{R^2}$
in-house	8,92	0,8
public	10,31	0,69
konst_inic (transfer learning)	8,54	0,83

5. táblázat

Látható tehát, hogy a transfer learning regresszió kapcsán is jól teljesít és pusztán saját adatokon történő tanítás során elért eredményt javítani tudja. A statisztikai szignifikancia érdekében további futtatások szükségesek.

5. Konklúziók és jövőbeni tervek

Munkám során tehát RS-fMRI képeken dolgoztam fel gépi tanulás módszerével. Célom volt, hogy a számítógép által az életkorra történő klasszifikációs, illetve regressziós becslő révén vizsgálatokat végezzek a páciens becsülhető életkorára, mint az agyi öregedéssel kapcsolatos potenciális biomerkerre vonatkozóan. Sikerült néhány alapvető globális képjellemző gépi tanulásra való alkalmasságának kizárása után találni továbbiakat, melyek az általam felállított képfeldolgozási módszerrel meghatározva egy kisebb méretű FC hálón a fiatal és idős egyének osztályozása kapcsán saját és nyilvános adatokon is a random találgatásnál szignifikánsan magasabb pontosságot tesznek lehetővé. Saját adatok esetén 85 %-os pontosság volt elérhető. A módszerhez szükséges volt az általam kidolgozott előfeldolgozás a fontos képjellemzők kinyeréséhez (feature extracting). Ezt a konvolúciós neurális hálók inherensen képesek elvégezni.

Ezek után a LeNet konvolúciós neurális hálóból kiindulva létre hoztam egy saját neurális hálót, mely a kép releváns tulajdonságait tartalmazó reprezentációk kinyeréséhez 3D-s konvolúciókat és dimenzió csökkentő lépéseket tartalmaz, majd az osztályozáshoz két FC réteget használ. Ezzel a hálóval a korábbi, pusztán FC hálóhoz képest magasabb pontosságot tudtam elérni az osztályozás kapcsán. A hiperparaméterek finomhangolásával egészen 95,6%-os pontosság, illetve a transfer learninghez alulmintavételezett képeken 98,3%-os pontosság volt elérhető.

Ezen folyamat során különféle hiperparaméterek validációs pontosságra gyakorolt hatását vizsgáltam. A tanulási rátára, háló paramétereinek számosságára (kernelméret és kernelek száma) és az optimalizációs módszerre javaslatot tudtam tenni a paraméterek adott értékére vonatkozóan a további munkához. A dropout, regularizáció és a batch mérete kapcsán a paraméterre vonatkozó javaslatra nem kaptam szignifikánsan konklúzív eredményt, azonban a vizsgálatok és a szakirodalom alapján tudtam ezen hiperparaméterekre is ajánlást tenni, vagy a használatát elhagyni.

Külső adatok bevonásával transfer learning módszeren is végeztem vizsgálatokat. Ebből azt kaptam, hogy a nyilvános adatokon tanult konvolúciós súlyok konstans átvétele mellett a saját adatokon magasabb osztályozási pontosság érhető el, mint abban az esetben,

ha a saját adatokon tanított konvolúciós súlyokat használnánk konstansként a nyilvános adatokon való további tanításhoz. Azaz a nyilvános adatokon a hálónk általánosabb jellemzőket tudott megtanulni, ezért a konvolúciós súlyok tanítása ezen az adatbázison javallott, hiszen a pusztán saját adatokon való tanítás pontosságát eléri ez a módszer is. Emellett a saját adataink túlságosan partikuláris jellemzők megtanulásának a veszélye is fenn állhat, ami lehet például a koponyaméretekben tapasztalt, a nyilvános adatokon mérhető értéknél jelentősebb tendencia. Ennek oka lehet az eltérő mérési paraméterek használata, így erre a tulajdonságra való túlzott rátanulás a saját tesztadatokon mérhető pontosságra nem feltétlen negatív hatású, amennyiben hasonló beállításokkal zajlik a további adatfelvétel. Más beállítások mellet, illetve más mérésekből származó adatokon az így optimalizált háló a túlzott rátanulás miatt a tesztadatok osztályozási pontosságát csökkentheti.

A diplomamunka záró szakaszában további MRI mérések történtek, ezért a jövőben ezeket az adatokat tesztadatként tervezem használni a jelenleg tanított hálómhoz. Emellett a transfer learnng módszer javításához további nyilvánosan elérhető adatot kívánok bevonni.

Regresszió esetén is azt kaptam, hogy a transfer learning módszere konstansként átvitt konvolúciós súlyok esetén legalább olyan jól teljesít, mint a kizárólag saját adatokon történő tanítás. A javulás szignifikanciájának kimutatásához további vizsgálatok szükségesek. Továbbá a dolgozat megírását követően végzett MRI mérések adatai alapján a becslő pontossága javítható, tervben van ezért a köztes korkategóriákba eső pácienssek felvétetele.

Általánosságban elmondható, hogy a saját adatok kis elemszáma megnehezítette a gépi tanulás alkalmazását, ennek ellenére sikerült elérni konklúziv eredményeket. A konvolúciós háló alkalmazása a tanítási időt jóval megnövelte a feature extractingon alapuló esethez képest. Pontosság növelő hatása mellett fontos hangsúlyozni, hogy későbbi minták kategorizálása (további tesztadatok) már töredékidő alatt lefuttatható, itt ugyanis csak predikciót ad a meglévő háló. Ezzel szemben utóbbi esetben nem csak a tanítás során kell a feature extractingot elvégezni, hanem minden jövőbeni tesztmintára is. Ezek a tények indokolttá teszik a konvolúciós hálók használatát a biomarker kutatásokban.

6. Köszönetnyilvánítás

Szeretnék köszönetet mondani ez úton mindazoknak, akik segítettek a diplomamunkám elkészítése során. Dr. Légrády Dávidnak, aki témavezetőként felügyelte a munkámat és tanácsokkal látott el.

Külön köszönet Deák-Meszlényi Reginának a szakmai tanácsokért, a módszertannal való megismertetésért, irodalmi anyagok rendelkezésemre bocsátásáért, valamint a mindig precíz és gyors visszajelzéseiért a munkámmal kapcsolatban.

Köszönöm az MTA TTK Agyi Képalkotó Központnak és munkatársainak a mérési adatokhoz és távoli vezérléssel a számítógépes kapacitáshoz való hozzáférés biztosítását.

Köszönöm családomnak és barátaimnak a diploma megírása során nyújtott támogatásukat.

7. GitHub repository

A diplomamunkám során Tensorflow környezetben programoztam a neurális hálókat, melyekkel a vizsgálatokat végeztem. A https://github.com/szegedomi/3d_conv_net_brain_age/tree/master repositoryban megtalálható feltöltve ezen python scripek közül néhány alapvető, melyek módosításaival tudtam elérni az egyes hiperparaméterek vizsgálatához használt, illetve további transfer learning szcenáriókban alkalamzott kódokat.

Hivatkozások

[1] Cole, James H., et al. "Predicting brain age with deep learning from raw imaging data results in a reliable and heritable biomarker." *NeuroImage 163* (2017) 115-124.

[2] Bernstein, Matt A., Kevin F. King, and Xiaohong Joe Zhou. "Handbook of MRI pulse sequences." *Burlington MA Elsevier Google Scholar* (2004.) 243-281.

[3] Brown, Robert W., et al. "Magnetic resonance imaging: physical principles and sequence design." *John Wiley & Sons*, (2014.) 113-136.

[4] Good, Catriona D., et al. "A voxel-based morphometric study of ageing in 465 normal adult human brains." *Biomedical Imaging*, (2002), 5th IEEE EMBS International Summer School on. IEEE

[5] Payan, Adrien, and Giovanni Montana. "Predicting Alzheimer's disease: a neuroimaging study with 3D convolutional neural networks." *arXiv preprint arXiv:1502.02506* (2015)

[6] Dosenbach, Nico UF, et al. "Prediction of individual brain maturity using fMRI." *Science 329.5997* (2010) 1358-1361.

[7] Franke, Katja, et al. "Estimating the age of healthy subjects from T1-weighted MRI scans using kernel methods: exploring the influence of various parameters." *Neuroimage 50.3* (2010) 883-892.

[8] Gaser, Christian, et al. "BrainAGE in mild cognitive impaired patients: predicting the conversion to Alzheimer's disease." *PloS one 8.6* (2013) e67346.

[9] Biswal, Bharat, et al. "Functional connectivity in the motor cortex of resting human brain using echo-planar mri." *Magnetic resonance in medicine 34.4* (1995) 537-541.

[10] Lee, Megan H., Christopher D. Smyser, and Joshua S. Shimony. "Resting-state fMRI: a review of methods and clinical applications." *American Journal of Neuroradiology 34.10* (2013)

[11] Smith, Stephen M., et al. "Resting-state fMRI in the human connectome project." *Neuroimage 80* (2013) 144-168.

[12] Yue, Nancy Chang, et al. "Sulcal, ventricular, and white matter changes at MR imaging in the aging brain: data from the Cardiovascular Health Study." *Radiology 202.1* (1997) 33-39.

[13] LMU_1 dataset http://fcon_1000.projects.nitrc.org/indi/CoRR/html/lmu_1.html [Internet] (Utolsó letöltés: 2018.05.28.)

[14] LMU_2 dataset http://fcon_1000.projects.nitrc.org/indi/CoRR/html/lmu_2.html [Internet] (Utolsó letöltés: 2018.05.28.)

[15] LMU_3 dataset http://fcon_1000.projects.nitrc.org/indi/CoRR/html/lmu_3.html [Internet] (Utolsó letöltés: 2018.05.28.)

[16] Ian Goodfellow and Yoshua Bengio and Aaron Courville, "Deep Learning". *MIT Press* (2016) http://www.deeplearningbook.org [Internet] (Utolsó letöltés: 2018.05.28.) 96-137.

[17] Pal, Nikhil R. "On minimum cross-entropy thresholding." *Pattern Recognition 29.4* (1996) 575-580.

[18] Bottou, Léon. "Large-scale machine learning with stochastic gradient descent." *Proceedings of COMPSTAT'2010. Physica-Verlag HD* (2010) 177-186.

[19] Sutskever, Ilya, et al. "On the importance of initialization and momentum in deep learning." *International conference on machine learning* (2013)

[20] Duchi, John, and Yoram Singer. "Efficient online and batch learning using forward backward splitting." *Journal of Machine Learning Research* (10.Dec 2009) 2899-2934

[21] Zeiler, Matthew D. "ADADELTA: an adaptive learning rate method." *arXiv preprint arXiv:1212.5701* (2012)

[22] Xu, Bing, et al. "Empirical evaluation of rectified activations in convolutional network." *arXiv preprint arXiv:1505.00853* (2015)

[23] Zhuge, Ying, and Jayaram K. Udupa "Intensity standardization simplifies brain MR image segmentation." *Computer vision and image understanding 113.10* (2009) 1095-1103.

[24] Krogh, Anders, and John A. Hertz. "A simple weight decay can improve generalization." *Advances in neural information processing systems* (1992)

[25] Moody, John E. "The effective number of parameters: An analysis of generalization and regularization in nonlinear learning systems." *Advances in neural information processing systems.* (1992)

[26] Srivastava, Nitish, et al. "Dropout: A simple way to prevent neural networks from overfitting." *The Journal of Machine Learning Research 15.1* (2014) 1929-1958.

[27] LeCun, Yann, et al. "Learning algorithms for classification: A comparison on handwritten digit recognition." *Neural networks: the statistical mechanics perspective 261* (1995) 276.

[28] Scherer, Dominik, Andreas Müller, and Sven Behnke. "Evaluation of pooling operations in convolutional architectures for object recognition." *International conference on artificial neural networks. Springer, Berlin, Heidelberg* (2010)

[29] Pan, Sinno Jialin, James T. Kwok, and Qiang Yang. "Transfer learning via dimensionality reduction." *AAAI. Vol. 8.* (2008)

[30] Pan, Sinno Jialin, and Qiang Yang. "A survey on transfer learning." *IEEE Transactions on knowledge and data engineering 22.10* (2010) 1345-1359.

[31] Pereira, Francisco, Tom Mitchell, and Matthew Botvinick. "Machine learning classifiers and fMRI: a tutorial overview." *Neuroimage 45.1* (2009) S199-S209.

[32] Guyon, Isabelle, and André Elisseeff. "An introduction to feature extraction." *Feature extraction. Springer, Berlin, Heidelberg* (2006) 1-25.

[33] Kingma, Diederik P., and Jimmy Ba. "Adam: A method for stochastic optimization." *arXiv preprint arXiv:1412.6980* (2014)

[34] Salzberg, Steven L. "On comparing classifiers: Pitfalls to avoid and a recommended approach." *Data mining and knowledge discovery 1.3* (1997) 317-328.

[35] Pál Vakli, Regina J. Deák-Meszlényi, Petra Hermann and Zoltán Vidnyánszky "Transfer learning improves resting-state functional connectivity pattern analysis using convolutional neural networks" *Under Review*

[36] ÁBRA. Andrej, Karpathy. "CS231n Convolutional Neural Networks for Visual Recognition" http://cs231n.github.io/neural-networks-1/ [Internet] Utolsó letöltés: 2018.06.04.

[37] ÁBRA. ProClassify User's Guide "Cross-Validation Explained" *Institute for Genomics and Bioinformatics - Graz University of Technology, Department of Information Design - FH JOANNEUM - Graz University of Applied Sciences* http://genome.tugraz.at/proclassify/help/pages/XV.html [Internet] Utolsó letöltés: 2018.06.04.

[38] ÁBRA. Alberto, Quesada. "5 algorithms to train a neural network" https://www.neuraldesigner.com/blog/5_algorithms_to_train_a_neural_network [Internet] Utolsó letöltés: 2018.06.04.

Ábrák jegyzéke