Emberi tevékenység felismerés viselhető szenzorok adataiból gépi-tanulással

Egyetemi doktori (PhD) értekezés

Sütő József

Témavezető: Dr. Oniga István

DEBRECENI EGYETEM

Természettudományi Doktori Tanács

Informatikai Tudományok Doktori Iskola

Debrecen, 2018.

Ezen értekezést a Debreceni Egyetem Természettudományi Doktori Tanács Informatikai Tudományok Doktori Iskola Az Informatika Ipari és Tudományos Alkalmazásai programja keretében készítettem a Debreceni Egyetem természettudományi doktori (PhD) fokozatának elnyerése céljából.
Nyilatkozom arról, hogy a tézisekben leírt eredmények nem képezik más PhD disszertáció részét.

Debrecen, 2018.

.....
Sütő József

Tanúsítom, hogy Sütő József doktorjelölt 2014-2017 között a fent megnevezett Doktori Iskola Az Informatika Ipari és Tudományos Alkalmazásai programjának keretében irányításommal végezte munkáját. Az értekezésben foglalt eredményekhez a jelölt önálló alkotó tevékenységével meghatározóan hozzájárult. Nyilatkozom továbbá arról, hogy a tézisben leírt eredmények nem képezik más PhD disszertáció részét.
Az értekezés elfogadását javaslom.

Debrecen, 2018.

.....
Dr. Oniga István

Emberi tevékenység felismerés viselhető szenzorok adataiból gépi-tanulással

Értekezés a doktori (Ph.D.) fokozat megszerzése érdekében az informatika tudományágban

Írta: Sütő József okleveles programtervező informatikus

Készült a Debreceni Egyetem Informatikai Tudományok doktori iskolája
Az Informatika Ipari és Tudományos Alkalmazásai keretében

Témavezető: Dr. Oniga István

A doktori szigorlati bizottság:

elnök:	Dr. Szrik János	……
tagok:	Dr. Györök György	……
	Dr. Gál Zoltán	……

A doktori szigorlat időpontja: 2017. szeptember 15.

Az értekezés bírálói:

	Dr……..	……..
	Dr……..	……..

A bírálóbizottság:

elnök:	Dr……..	……..
tagok:	Dr……..	……..
	Dr……..	……..
	Dr……..	……..
	Dr..……	……..

Az értekezés védésének időpontja: 2018. …… … .

Tartalomjegyzék

1. Bevezetés 1
2. Szakirodalmi áttekintés és előzmények 5
3. Adat elő-feldolgozás 13
3.1. Szegmentálás 13
3.2. Adatgyűjtők optimális kombinációja és az adatleírók használatának hatékonyság vizsgálata a WARD adatbázison 13
3.2.1. Inerciális szenzorok adatainak fúziója 16
3.2.2. A köztes rétegen elhelyezett neuronok számának hatása a felismerési arányra 17
3.3. Normalizált nyers adat használata a tanuló algoritmus bemeneteként 19
3.4. Adatleírók 22
3.5. Adatleírók kiválasztása 26
3.6. Módosított gyors Fourier transzformáció 38
4. Sekély neurális hálózatok hatékonyság elemzése 46
4.1. Sekély neurális hálózatok tervezése és tesztelése 46
4.2. Az eltérő neurális háló architektúrák hatékonyság vizsgálata 49
4.2.1. Hatékonyság vizsgálat négy széles körben használt adatleíróval 49
4.2.2. Hatékonyság vizsgálat kibővített adatleíró halmazzal 50
4.2.3. Hatékonyság vizsgálat válogatott adatleírókkal 55
4.2.4. Kibővített neurális hálózat hatékonyság vizsgálata 58
5. Összetett és mély neurális hálózatok hatékonyság elemzése 61
5.1. Neurális háló együttesek 61
5.2. Bináris neurális hálózatok 63
5.3. Konvolúciós neurális hálózatok 66
6. Offline eredmények tesztelése valós-időben 74

6.1. Az applikáció működése 77
6.2. Valós idejű tesztek 78
7. Összefoglalás 85
Irodalomjegyzék 88
A. Függelék 96
B. Függelék: Publikációs Lista 100
Idegen nyelvű folyóiratcikkek 100
Idegen nyelvű konferenciacikkek 101

Köszönetnyilvánítás

Köszönöm témavezetőmnek, **Dr. Oniga Istvánnak** a szakmai és a mindennapi életben nyújtott segítségét és támogatását.

Köszönöm **Dr. Végh Jánosnak**, hogy az egyetemi éveim elején és azt követően is szakmailag támogatott. Ő motivált a kutatói munka elkezdésére.

Továbbá, szeretnék köszönetet mondani korábbi két munkatársamnak, **Hegyesi Gyulának** és **Kertész Zsoltnak,** akik nemcsak szakmailag, hanem az élet egyéb területein is mindig készségesen segítettek.

Végül szeretném megköszönni **Máté Ákosnak** az egyetemi éveim elején nyújtott segítségét.

Rövidítések

ANN	Mesterséges neurális hálózat
ANN1	1. számú neurális hálózat struktúra
ANN2	2. számú neurális hálózat struktúra
ANN3	3. számú neurális hálózat struktúra
AR	Autoregresszív együttható
CFS	Correlation-based adatleíró kiválasztó
CHI	Chi square adatleíró kiválasztó
CNN	Konvolúciós neurális hálózat
CNN1	1. számú konvolúciós hálózat struktúra
CNN2	2. számú konvolúciós hálózat struktúra
CORR	Tengelyek közötti korreláció
DFT	Diszkrét Fourier transzformáció
DT	Döntési fa
E	Spektrum entrópia
E3	3 neurális hálózatot tartalmazó együttes
E4	4 neurális hálózatot tartalmazó együttes
E5	5 neurális hálózatot tartalmazó együttes
E6	6 neurális hálózatot tartalmazó együttes
FCBF	Fast correlation-based filter adatleíró kiválasztó
FFT	Gyors Fourier transzformáció
FIS	Fisher score adatleíró kiválasztó
HAPT	2. számú adatbázis
IG	Information gain adatleíró kiválasztó
IQR	Interkvartilis tartomány
kNN	k-legközelebbi szomszéd

KS	Kurtózis
KW	Kruskal-Wallis adatleíró kiválasztó
M	Várható érték
MAD	Átlagos abszolút eltérés
MAX	Maximum konfidencia stratégia
MM	Minimum-maximum különbség
MRMR	Minimum redundancy maximum relevance adatleíró kiválasztó
NB	Naiv Bayes
OVA	One-versus-all stratégia
OVO	One-versus-one stratégia
PE	75. percentilis
PF	Alap frekvencia
Q	Kvartilis
RF	Véletlen erdő
RMS	Négyzetes közép
SC	Spektrum közép
SE	Spektrum energia
SMA	Jel nagyságterület
SVM	Support vector machine
TA	Dőlésszög
V	Szórásnégyzet
WARD	1. számú adatbázis
ZCR	Nulla átmenet ráta

1. Bevezetés

Ahogyan az köztudott, a rendszeres fizikai aktivitás az egyik kulcs eleme az egészséges életnek. Korábbi tanulmányok megmutatták, hogy rendszeres fizikai aktivitással (sétálás, kocogás, futás, stb.) az egészségi állapot nemcsak megőrizhető, de javítható is [1].

Az adatbányászat megjelenése egy mérföldkőnek számított a modern orvosbiológiai alkalmazásokban. Az elmúlt évtizedek során az orvosbiológiai kutatások folyamatosan tárták fel a mindennapi mozgás pozitív hatásait, valamint rávilágítottak arra, hogy a mozgásszegény életmód az egyik fő oka néhány meglehetősen elterjedt népbetegségnek, mint például a hátfájás, elhízás valamint a szív és érrendszeri problémák.

A mozgásszegény életmód kialakulása visszavezethető a munkakörülmények átalakulására is – folyamatos ülő munka, ami számos iparágban bekövetkezett. A fizikailag aktív személyek esetében, az ilyen jellegű betegségek kialakulásának kisebb a kockázata, mint azoknál, akik nem végeznek rendszeres mozgást [1]. Ezen a témakörön belül az emberi tevékenység felismerés egyre fontosabb szerepet tölt be, ahol az egyes emberi cselekvések felismerése a cél. A cselekvések meghatározása és kategorizálása fontos információval szolgál egy adott személy életmódjáról és funkcionális képességeiről. Egy megfigyelt személy cselekvéseinek a felismerése az egészségügyi szolgáltatásokon felül, más ágazatokban is fontos szerepet tölthet be, mint például az egészségtudatos életmód vagy a rehabilitáció [3].

Az elmúlt évtizedben számos fejlett országban súlyos problémává vált elöregedő társadalmuk, ami ezzel egyidejűleg nagymértékben befolyásolta az egészségügyi szolgáltatások fejlődését. Ezzel párhuzamosan a cselekvés felismerés fontossága is rohamosan növekszik és egyre több kutató csoport kapcsolódik be ebbe a témakörbe.

A tevékenység felismeréshez általában két adatgyűjtési technikát használnak. Az egyik esetben kamerák képeit dolgozzák fel, míg a másik esetben viselhető szenzorokat alkalmaznak. A kamera alapú megfigyelő rendszerek hátrányai és korlátai miatt (adatvédelmi problémák, háttér és megvilágítás változás, speciális környezet, stb.), ebben a kutatási témában a viselhető szenzorok váltak elsődleges adatgyűjtő eszközzé. A modern MEMS technológia lehetővé tette a megfigyelt személyek számára, hogy

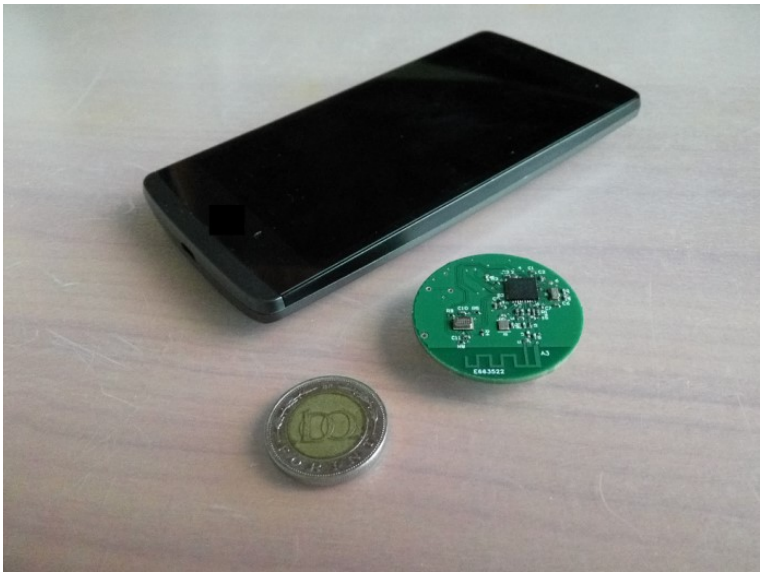
folyamatosan viselhessenek adatgyűjtő eszközöket a testük egyes előre meghatározott részein. A viselhető eszközök előnyei, (kis méret, alacsony költség, hosszú idejű és folyamatos adatgyűjtés) gyorsan növelte a népszerűségüket. Ezek többnyire egy vagy két inerciális szenzort foglalnak magukba: gyorsulásszenzort és giroszkópot. A viselhető eszközök népszerűsége számos kutatót sarkalt saját adatgyűjtő rendszer kifejlesztésére és az inerciális szenzor alapú adatgyűjtésre [4-6]. Mindezek mellett, széles körben terjedt el a nagyszámú integrált szenzort tartalmazó okostelefonok használata is adatgyűjtésre. Az okostelefonok egy alternatívát nyújtanak az emberi tevékenység felismeréshez, mivel ezek egy komplett platformot biztosítanak, amely magába foglalhatja a szenzort és az adatkiértékelő szoftvert. Az okostelefonok talán legfőbb hátránya a speciális adatgyűjtőkkel szemben a méretük, viszont ez a méretbeli különbség a legtöbb esetben nem okoz problémát. Szemléltetésként az 1. ábrán látható a méretbeli eltérés egy okostelefon és egy viselhető szenzor között.

A kereskedelmi forgalomban lévő okostelefonok fejlődésével folyamatosan jelentek meg olyan alkalmazások, amelyek az emberi cselekvések felismerésére irányulnak, és az egészségmegőrzést segítik elő [7, 8]. Ezek közül valószínűleg a lépésszámláló szoftverek a legismertebbek, amik egyre szélesebb körben terjednek el. Ezek a technológiai újítások és a kutatási eredmények hasznosulása megalapozta a nyilvános adatbázisok megjelenését, amik lehetővé teszik az egyes tevékenység felismerő módszerek hatékonyság elemzését.

Annak ellenére, hogy egyre több és jobb adatgyűjtő eszköz jelent meg, a tevékenység felismerés nem egyszerű feladat. Az adatgyűjtő eszközökön túl, egy hatékony cselekvés osztályozó algoritmus is szükséges a begyűjtött adathalmaz feldolgozásához és értelmezéséhez. Az adatgyűjtést befolyásoló zajok és a hiányos adatkészlet néhány esetben megakadályozza a helyes felismerést. Annak érdekében, hogy a felismerési arány a lehető legnagyobb legyen, a kutatók különböző robosztus gépi-tanulási stratégiákat használnak. Nyilvánvalóan ennek a megoldásnak a kritikus pontja az adat elő-feldolgozó és a felismerő algoritmusokban rejlik, amelyeknek képesnek kell lennie a tanulásra és a zajok toleranciájára. A felismerést végző szoftverrel szembeni általános elvárások, hogy a tanulási és a döntési ideje is relatív rövid legyen, valamint ne igényeljen emberi beavatkozást vagy szakértői ismeretet.

A Tárgyak Internete (IoT) és a felhő számítások integrációja három adat elemzési módszert hozott létre [9]: *teljesen felhő alapú, teljesen lokális* és *elosztott*. A teljes és a részleges felhő alapú szolgáltatások esetében, az adatgyűjtő eszközök csak mérést végeznek, míg az adatkiértékelés főként a felhőben történik. Erre jó példa lehet az okos városok koncepcióján belül kidolgozott döntéshozó rendszerek, ahol a különböző szenzoroktól származó, valós idejű adatfolyamok, késleltetetten lesznek feldolgozva [10]. Ezzel szemben a teljes lokális elemzésnél, az adatgyűjtést és a kiértékelést is ugyan az az eszköz végzi el. A tevékenység felismerés esetében, a viselhető szenzorok dinamikus fejlődése lehetővé teszi a teljes körű lokális adatfeldolgozást, mivel az új eszközök már figyelemreméltó számítási és tárolási kapacitással rendelkeznek [11]. Továbbá, ha azt feltételezzük, hogy a közeljövőben a cselekvés felismerés általánosan elterjedté válik, akkor a teljes és a részleges felhő alapú adatelemzés hatalmas kommunikációs forgalmat generálna. Ebből kiindulva, hosszútávon a lokális kiértékelés sokkal jobb megoldásnak tűnik.

Az adatfeldolgozás és az elemzés elhelyezésének kérdésen túl, számos egyéb kérdés is megválaszolatlan volt az emberi tevékenység felismerés problémakörében. Ezek közül az egyik legkiemelkedőbb a tanuló algoritmussal szembeni bizonytalanság. Vajon melyik tanuló algoritmus alkalmazható a leghatékonyabban erre a célra. Valamelyik sekély tanuló módszer, esetleg a sekély módszereknek valamilyen együttese, vagy a napjainkban egyre nagyobb teret hódító mély tanulási technikák? A tanuló algoritmusok hatékonyságának növelésével kapcsolatban egy másik fontos kérdés, hogy milyen elő-feldolgozáson essen át az az adathalmaz, amely a tanuló algoritmust fogja táplálni. Esetleg elég a szenzortól érkező normalizált adatok használata, vagy célszerűbb lenne az így kapott adatokat jellemző úgynevezett adatleírók (*features*) alkalmazása a tanuló algoritmus bemeneteként? Amennyiben adatleírókat szeretnénk használni, akkor milyen adatleírókat kell kiszámítanunk? Végül az utolsó fontos kérdés, hogy mennyire megbízhatóan alkalmazhatóak az úgynevezett offline módon kapott eredmények a valós életben? A doktori dolgozatomban főként ezekre a kérdésekre keresetem a válaszokat. A fent említett kérdések mögötti szakirodalmi háttér a következő fejezetben lesz részletesen ismertetve.



1. ábra. Méretbeli különbség egy okostelefon és egy viselhető szenzor között

2. Szakirodalmi áttekintés és előzmények

A tevékenység felismerés azon a feltételezésen alapul, hogy mozgás közben az emberi test jellegzetes szenzorjelet, más szóval mintát generál, amelyek digitális jelfeldolgozás és/vagy gépi-tanulás segítségével felismerhető. Erre a feladatra számos tanuló algoritmus alkalmazható.

A korábbi cikkek egy részében a kutatók összetett algoritmusokat definiáltak a tevékenység és esés felismeréshez [12, 13, 14]. A témához kapcsolódó első cikkünkben mi is egy általunk kidolgozott algoritmussal próbáltuk elvégezni a cselekvések beazonosítását, amelynek az alapötlete az az észrevétel volt, hogy az egyes elemi cselekvések, mint például a sétálás, futás, ülés, stb. szemmel is viszonylag jól felismerhető mintákat generálnak [J1]. Ebből kiindulva minden egyes cselekvéshez egy jellegzetes mintát definiáltunk, amit csúszóablak formájában „úsztattunk" végig a szenzortól beérkező adatsoron és a korreláció (1), illetve a négyzetes hiba számítás alapján (2) vizsgáltuk a kettő közötti hasonlóságot. A két egyenletben *x* jelöli a szenzortól begyűjtött adatsort, míg *h* az *N* elemű előre definiált minta. Annak ellenére, hogy a négyzetes hibán alapuló módszer hatékonynak bizonyult, az ilyen összetett algoritmusok nagy hátránya a rugalmasság hiánya.

$$y[i] = \sum_{j=0}^{N-1} h[j]\,x[i+j] \tag{1}$$

$$SSE(x,h)[i] = \sum_{j=0}^{N-1} \left(h[j] - x[i+j]\right)^2 \tag{2}$$

Abban az esetben, ha újabb cselekvéseket szeretnénk bevinni a modellbe, akkor ez sok esetben a modell gyökeres megváltoztatását igényli. Feltehetően ebből a megfontolásból kiindulva a kutatók többsége a felismerést valamelyik jól ismert, „sekély", paraméteres (naiv Bayes, előrecsatolt mesterséges neurális hálózatok) és nem paraméteres (k-legközelebbi szomszéd, döntési fa, support vector machine) gépi-tanuló módszerrel végezte [15].

Annak ellenére, hogy számos cikk született már ezen a témakörön belül az egyes gépi-tanuló algoritmusok egymással szembeni hatékonysága nem volt egyértelmű. Például a [7] cikk szerzői 6 sekély algoritmus közül a neurális hálózatot találták a leghatékonyabbnak. Ezzel szemben a [16] cikk szerzői azt

állították, hogy a support vector machine hatékonyabban alkalmazható tevékenység felismerésre, mint a neurális hálózatok. A [17] tanulmány mérései alapján pedig, a neurális hálózatok bizonyultak az egyik leggyengébben teljesítő tanuló algoritmusnak.

A mély tanulás megjelenése számos gépi-tanulási problémában hozott áttöréseket. Az egyre mélyebb neurális hálózatok és az új típusú rétegek használatának ötlete egyre népszerűbbé válik a kutatók körében. Ez annak is köszönhető, hogy ezek az új tanuló módszerek automatikusan egy magasabb szintű reprezentációt képeznek a nyers adathalmazból. Ebből adódóan a mély tanulás egy általánosabb megoldást nyújt, mivel az adatleírók kinyerését automatizálja, szemben a sekély módszerekkel, ahol az adatleíró-kinyerése statikusan történik (erről a folyamatról bővebben a 3.4 fejezetben lesz szó).

A mély algoritmusok már több kutatási területen felülmúlták a korábbi leghatékonyabb sekély módszereket, mint például a szövegelemzésben, a természetes nyelv feldolgozásban, de legfőképp az objektum felismerésben [18-21]. Ez nagymértékben motiválta a tevékenység felismerést végző kutatókat is az összetettebb módszerek használatára, mint például a konvolúciós neurális hálózatok, amely a legáltalánosabban elterjedt, mély algoritmus lett ebben a kutatási témában. Például Sheng és mások, valamint Yiang és Yin már konvolúciós hálókat használtak és megközelítőleg 95% felismerési arányt értek el nyilvános adatbázisokon [22, 23].

A mély tanulást támogatók köre azt állította, hogy a statikus adatleíró-kinyerés helyettesíthető a konvolúciós rétegekkel [24-26]. Mivel ez a módszer nem igényli a statikus adat elő-feldolgozást, ezért a használata is kényelmesebb [27]. Ezen felül, a skálafüggetlenség és a térbeli kapcsolatok detektálása tovább növelte népszerűségüket.

Mindezek ellenére a mély algoritmusok hatékonysága a sekély módszerekkel szemben nem egyértelmű a cselekvés felismerés témakörén belül. A [28-30] tanulmányok jól tükrözi ezt a bizonytalanságot. A [28] cikk szerzői összehasonlították a konvolúciós háló (CNN), véletlen erdő (RF), döntési fa (DT), naiv Bayes, k-legközelebbi szomszéd (kNN) és a support vector machine (SVM) algoritmusok hatékonyságát két adatbázison és a méréseik alapján, a CNN és az RF felismerési arányai között a különbség meglehetősen csekély volt. A [29] cikk eredményei is azt mutatták, hogy a hatékonyságbeli eltérés a CNN, SVM, és KNN között minimális. A három

cikk közül, kizárólag a [30] felmérésben használtak neurális hálózatot (ANN) is az összehasonlítás során. Ezekben a mérésekben is a CNN (egyetlen konvolúciós réteggel) csak egy kevéssel tűnt hatékonyabbnak az ANN-el szemben.

Annak érdekében, hogy a gépi-tanuló algoritmusokkal kapcsolatos bizonytalanságot szemléltessem, az 1. táblázatba összegyűjtöttem azokat az ismertebb cikkeket, ahol valamilyen tanuló algoritmust használnak a cselekvések felismeréséhez. Ahogy azt a táblázat is mutatja, az egyes kutatók eltérő módszerekkel próbálják meg felismerni a cselekvéseket. Szerencsére a szakirodalomban már található néhány cikk, amely leszűkíti a sekély algoritmusok körét. A [4, 31] tanulmányok eredményei alapján az ANN és a kNN bizonyult a leghatékonyabb sekély módszernek az emberi tevékenység felismerésre. Valójában az ANN esetében ez az eredmény nem is olyan meglepő, mivel elméletileg az ANN sokkal komplexebb döntési határokat képes generálni az n-dimenziós térben, mint más sekély technikák. Ez egyúttal azt a kérdést is felveti, hogy mi lehet az oka a korábbi eltérő eredményeknek?

Az imént ismertetett bizonytalanság motivált bennünket arra, hogy megvizsgáljunk különböző neurális hálózati architektúrák hatékonyságát eltérő rétegszámmal és bemenő adattal. Ezen felül, a korábbi tanulmányokban, ahol ANN-t használtak tanuló algoritmusként, az úgynevezett *hiper-paraméterek* a szerzők tapasztalatain alapultak vagy egy szűk rácson végzett keresésből származtak [4, 16, 17, 32, 33]. Azonban ez a megközelítés a legtöbb esetben nem megbízható. Továbbá, az eddigi ismereteink azt mutatták, hogy a neurális háló együttesek (*ensemble*) és a bináris neurális hálózatok modellje nem volt kellőképpen tesztelve a cselekvés felismerésben, ezért ezeknek a módszereknek a vizsgálatát is elvégeztük.

A tevékenység felismerés kezdeti szakaszában, az egyes kutatók és kutató csoportok saját adathalmazokat gyűjtöttek össze, amelyek nyilvánosan nem elérhetőek. Ez megfigyelhető a 1. táblázat adatai alapján is. Például, a [32, 33] cikkek szerzői ANN-t használtak a tevékenységek osztályozásához és 97.9%, illetve 95% felismerési arányt mértek a saját adathalmazaikon. A [8, 35] tanulmányok szerzői 95% és 97.8% hatékonyságot mértek a kNN módszerrel, míg a [4, 5] tanulmányokban a DT is hasonló eredményeket

produkált: 92.8%, 96.4%. Ezekben az esetekben az eredmények összehasonlítása nagyon nehéz, mivel az egyes kutatók eltérő adathalmazokat használtak, valamint a legtöbb esetben az elvégzett cselekvések is többé-kevésbé eltérőek voltak. Később a modern viselhető szenzorok nagymértékű elterjedésének köszönhetően egyre több nyilvános adatbázis lett elérhető a kutatók számára. Napjainkban több különböző adathalmaz is szabadon felhasználható köztük az Opportunity, Pamap2, Skoda, stb. adathalmazok a legismertebbek [26, 27, 36]. Ezeknek az adatbázisoknak a használata jó lehetőséget biztosít az eltérő cselekvés felismerési technikák minőségi összehasonlítására és elemzésére. Erre jó példa a [34] tanulmány, ahol a szerzők egy nyilvános adatbázison dolgoztak és egy neurális hálózat segítségével felülmúlták az adott adatbázison elért korábbi eredményeket.

1. táblázat. Leggyakrabban használt tanuló algoritmusok és felismerési arányaik

Tanuló algoritmus	**Referencia**	**Adatforrás**	**Adatgyűjtésben résztvevő önkéntesek száma**	**Felismerési arány**
kNN	[8]	nem nyilvános	5	98.0%
	[35]	nem nyilvános	20	97.0%
	[31]	nem nyilvános	8	99.7%
DT	[45]	nem nyilvános	10	97.3%
	[5]	nem nyilvános	6	92.8%
SVM	[40]	nyilvános	20	98.5%
	[38]	nyilvános	30	96.0%
	[16]	nem nyilvános	12	90.2%
ANN	[33]	nem nyilvános	6	97.9%
	[32]	nem nyilvános	7	95.0%
	[4]	nem nyilvános	8	96.8%
CNN	[22]	nyilvános	20	95.9%
	[23]	nyilvános	30	95.2%
	[44]	nyilvános	30	95.8%

Annak érdekében, hogy az eredményeink összehasonlíthatóak legyen mások munkáival, sok esetben mi is nyilvános adatbázisokat használtunk adatforrásként. Az általunk felhasznált adatbázisok kiválasztásakor több tényező is fontos szerepet játszott. Olyan adatbázisokon akartunk dolgozni, amelyek eltérő számú és típusú adatgyűjtő eszközzel lettek begyűjtve, többnyire hasonló cselekvéseket tartalmaznak, és elegendő referencia kapcsolódik hozzájuk. Ezek alapján munkánk tetemes része két nyilvános adatbázison folyt. Ezek közül az első a *wearable action recognition database (WARD)*[1], amelyet a Berkeley Egyetem kutatói gyűjtöttek össze, a felismerő algoritmusok hatékonyságának tesztelése céljából. Az adatbázis gyorsulás és giroszkóp szenzorok adatait tartalmazza, ami 20 önkéntes 13 eltérő cselekvése során lett felvéve. Az adatgyűjtés során az önkéntesek (13 férfi és 7 nő) minden cselekvést 5 alkalommal végezték el. A cselekvések listája a következő:

1. Állás
2. Ülés
3. Fekvés
4. Séta előre
5. Séta közben fordulás balra
6. Séta közben fordulás jobbra
7. Jobbra fordulás
8. Balra fordulás
9. Séta lépcsőn felfelé
10. Séta lépcsőn lefelé
11. Kocogás
12. Ugrás
13. Tolószék görgetés

Az adatgyűjtés során az önkéntesek 5 adatgyűjtő eszközt viseltek a testük különböző pontjain: jobb és bal felkar, derék, jobb és bal boka. Az adatgyűjtés mintavételi sebessége minden szenzor estén megközelítőleg 20 Hz volt. Minden adatgyűjtő eszköz egy 3 tengelyű gyorsulásszenzort és egy 2 tengelyű giroszkópot foglalt magába. Ennek alapján a *k.* szenzor *t.* mintáját egy 5 elemű vektorral írhatjuk le (3), amelynek az első három eleme a gyorsulásszenzor *x*, *y*, és *z* tengelyétől származik, míg az utolsó két eleme a

[1] https://people.eecs.berkeley.edu/~yang/software/WAR/

giroszkóp θ és ρ tengelyeitől. Az adatbázisról további információ a [37] cikkben található. Méréseink során, amikor ez az adatbázis volt az adatforrás, a felhasznált adatok 60% volt a tanító halmaz, 20% a validációs halmaz és 20% a teszt halmaz.

$$s_k(t) \coloneqq (x_k(t), y_k(t), z_k(t), \theta_k(t), \rho_k(t)) \in \Re^5 \tag{3}$$

A második adatbázis az UCI machine learning repository[2] egyik legtöbbet letöltött adathalmaza, amely 30 résztvevő adatait tartalmazza, akik 6 általános cselekvést végeztek:

1. Séta előre
2. Séta lépcsőn felfelé
3. Séta lépcsőn lefelé
4. Ülés
5. Állás
6. Fekvés

Ebben az esetben az adatgyűjtés egy Android alapú okostelefonnal történt, ami a derékra volt erősítve. A nyers adatokat a telefonban lévő 3-tengelyű gyorsulás és giroszkóp szenzorok szolgáltatták. A dolgozat hátralévő részében, erre az adatbázisra HAPT rövidítéssel fogok hivatkozni. Ellentétben a WARD adatbázissal, ez az adathalmaz nyers és a nyers adatból származtatott adatleírókat is tartalmaz. Továbbá, a tanításra és a tesztelésre szánt adatok is már előre el vannak különítve. További információ a HAPT adatbázisról a [38] referenciában olvasható.

Mindkét adatbázis jól ismert a cselekvés felismeréssel foglalkozó kutatói közösség körében és számos korábbi kutatásban használták adatforrásként. Éppen ezért már néhány sekély és mély tanuló algoritmust is teszteltek mindkét adathalmazon. Az 2. táblázatba összegyűjtöttem a legrelevánsabb korábbi munkákat a szakirodalomból, amelyek a két adatbázis valamelyikéhez kapcsolódnak.

Annak ellenére, hogy egyre több cikk jelenik meg ebben a kutatási témakörben, néhány kérdés még nyitott volt. Ezek közül az egyik legfontosabb, hogy az összegyűjtött adat milyen elő-feldolgozást követően képes maximalizálni a tanuló algoritmus hatékonyságát? A korábbi

[2] http://archive.ics.uci.edu/ml/datasets/Human+Activity+Recognition+Using+Smartphones

cikkekben, a szerzők normalizált nyers adatot vagy különböző adatleírókat használtak (főként az idő és frekvenciatartományból), mivel egy általánosan elfogadott adatleíró halmaz nem létezik a szakirodalomban [15]. Ezzel szemben a kutatók egy kisebb csoportja a hullám transzformációt választotta adatleíró információ kinyerésre, a nyers, időtérbeli adathalmazból [47]. Ezek alapján fontosnak tartottuk egy tanulmány elvégzését, amelyben összegyűjtjük a legnépszerűbb adatleíró-kinyerési módszereket a szakirodalomból, majd ezt követően a hatékonyságukat részben adatleíró-kiválasztó (*feature selection*) algoritmusok segítségével, részben a tanuló algoritmus hatékonyságának figyelésével vizsgáljuk meg.

Ezen felül, arra is kerestük a választ, hogy létezik olyan általánosan használható adatleíró halmaz, amely a megfigyelt személytől függetlenül, általánosan és hatékonyan alkalmazható?

2. táblázat. Korábbi eredmények a WARD és HAPT adatbázisokon

Adatbázis	**Referencia**	**Tanuló algoritmus**	**Felismerési arány**
WARD	[37]	Distributed sparsity	93.5%
	[22]	CNN	95.92%
	[39]	Lexikai megközelítés	97.8%
	[40]	SVM	98.5%
HAPT	[41]	Conf-AdaBoost	94.33%
	[23]	CNN	95.18%
	[38]	SVM	96%
	[42]	Tanuló vektor kvantálás	96.23%
	[43]	OVO SVM többségi döntéssel	96.4%
	[44]	CNN	95.75%

Végül a kutatás lezárásaként, az adott témakör aktuális „*offline*” eredményei és a saját eredményeink alapján elkészítettünk egy Android alapú applikációt, aminek segítségével megvizsgáltuk, hogy az offline, utólagos adatkiértékelések eredményei mennyire állják meg a helyüket valós környezetben.

Ahogy a bevezetésben említettem, a modern okostelefonok használhatók úgy, mint egy komplett cselekvés felismerő rendszer és már néhány kutató használt telefonokat a cselekvés felismeréshez. A többségük a telefont adatgyűjtő eszközként használta, míg az adatok kiértékelése elkülönítetten történt valamilyen jól ismert matematikai vagy adatbányász szoftverrel, mint például a Weka, Python vagy Matlab [7, 46]. Ma már a legtöbb ember rendelkezik okostelefonnal, ami elegendő számítási kapacitást és üzemidőt biztosít. Éppen ezért ezek az eszközök egy új lehetőséget nyújtanak az emberi tevékenység felismerésre. Az okos eszközök adta előnyök, mint a magas szintű programozás, különböző megjelenítési, kommunikációs és adattárolási lehetőségek, nagymértékben megkönnyítik és meggyorsítják a fejlesztés folyamatát. Ebből adódóan a valós idejű és az offline mérési eredmények összehasonlításához mi is egy okostelefont használtunk adatgyűjtőként és kiértékelőként egyaránt. Az adatbegyűjtést és kiértékelést végző applikációt, kizárólag ehhez a kutatáshoz fejlesztettünk ki. Ismereteink szerint ez volt az első cikk az irodalomban, amely ilyen módon hasonlítja össze az offline és a valós idejű mérési eredményeket.

3. Adat elő-feldolgozás

A tipikus gépi-tanulási problémák egy úgynevezett tanulási lánccal írhatóak le, ami adat begyűjtésből, (a legtöbb esetben) adatleíró-kinyerésből, esetenként adat leíró kiválasztásból, tanításból és kiértékelésből áll. Az emberi tevékenység felismerés is tekinthető egy gépi-tanulási problémának, amely sajátos adat elő-feldolgozást igényel.

3.1. Szegmentálás

Általában a nyers adat, ami az inerciális szenzoroktól származik, egy diszkrét adatfolyamnak tekinthető. A tanítási és a felismerési fázisoknál, ezt az adatfolyamot kisebb darabokra kell tördelni (szegmentálás), amelyet *ablakoknak* neveznek.

Leggyakrabban egy ablak néhány másodpercnyi időintervallumot fed le és a mérete a mintavételezési frekvenciától függ. Például, a [4, 48] cikkek szerzői 1 másodperc széles ablakméretet használtak, az ablakok közötti átfedés nélkül. A [16, 35] tanulmányokban az ablak méret 3.88 és 2 másodperces volt, 50% átfedéssel a szomszédos ablakok között. A WARD adatbázis esetén mi megközelítőleg 1.6 másodperc széles ablakkal dolgoztunk (32 minta), amik között 50% átfedés volt. Ezt a döntést főként Lara és Labrador [15] irodalmi áttekintése motiválta, amelyben a szerzők röviden bemutatták a tipikus ablak méreteket és a kisebb-nagyobb méretek előnyeit és hátrányait. Keskeny ablak esetén a kis frekvenciás cselekvések által generált minták nem férnek bele egy ablakba, míg a széles ablakok több eltérő mintát is tartalmazhatnak. A HAPT adatbázis készítői is ennek alapján szegmentálták az adataikat, annyi eltéréssel, hogy ők 2.56 másodperc széles ablakkal dolgoztak, viszont hasonlóan 50% átfedéssel.

3.2. Adatgyűjtők optimális kombinációja és az adatleírók használatának hatékonyság vizsgálata a WARD adatbázison

A [J2] munkánknak több célkitűzése is volt. Az első az, hogy megvizsgáljuk különböző szenzor kombinációk hatékonyságát a WARD adatbázis 13 cselekvésének felismerésére. A második, az adatleírók nélküli és az adatleírókkal kibővített normalizált nyers adatok hatékonyság elemzése.

Végül munkánk kiterjedt az adatgyűjtőkben lévő inerciális szenzorok és a neurális háló köztes rétegén lévő neuronjai számának vizsgálatára is. Ez a tanulmány az első lépés volt a valós idejű cselekvés felismerő rendszer tervezése felé, amelynek az eredményei meghatározóak voltak a későbbi munkáink során is. Ebben a tanulmányban én a mérési eredmények kiértékelésével foglalkoztam. Mivel munkánk során több mint 1000 szimulációt kellett lefuttatnunk, ezért a WARD adatbázis egyes személyeihez tartozó adathalmazoknak csak a felét használtuk fel.

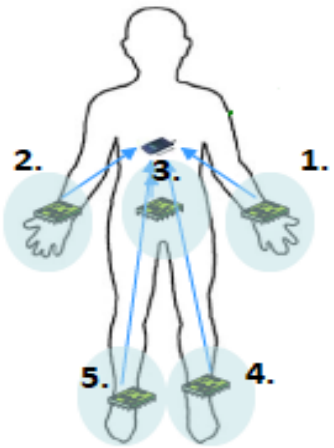
A WARD adatbázis esetén, minden alany 5 szenzort viselt, amely egy 3 tengelyű gyorsulásszenzort és egy 2 tengelyű giroszkópot foglal magába. 5 szenzor folyamatos viselése nem túl kényelmes, ezen felül 5 szenzor használata költséges és az adat elő-feldolgozás idejét is meghosszabbítja. Az 5 szenzor közül kíváncsiak voltunk arra, hogy az egyes adatgyűjtők, valamint a bennük elhelyezett inerciális szenzorok mennyire hatékonyak és hogyan befolyásolják a felismerési arányt.

Tanuló algoritmusként előre csatolt ANN-t használtunk, egy köztes réteggel. A hálót Matlabban hoztuk létre az alapbeállítások használatával. A köztes rétegen elhelyezkedő neuronok számát 10, 20 és 30-ra állítottuk be 3 különböző tesztfázisban. A köztes rétegen *szigmoid* aktivációs függvényt használtunk, míg a kimeneti rétegen három különböző függvényt teszteltünk: lineáris, versengő és küszöb (0.5 küszöbértékkel).

Összesen 31 szenzor konfigurációt vizsgáltunk meg, három, részben eltérő bemenő adathalmazzal – az ablakban lévő normalizált nyers adattal, a normalizált adat és a jel nagyságterületét megadó adatleíró (SMA) használatával és végül ezekhez hozzávettük az ablak szórásnégyzetét (V) is. A mérési eredmények a 3. táblázatban láthatóak, ami a különböző neurális háló stratégiák eredményei közül a legjobbat tartalmazza. A WARD adatbázishoz használt szenzorok elhelyezését és sorszámait a 2. ábra illusztrálja.

Ahogy várható volt, a legjobb felismerési arányt mind az 5 szenzor felhasználásával kaptuk, valamint a gyorsulás és giroszkóp szenzorok együttes figyelembevételével. Azt is megfigyeltük, hogy ha a nyers adatokhoz hozzávesszük a két adatleírót, akkor a felismerési arány számottevően növekszik. Amennyiben jobban megfigyeljük a kapott eredményeket, akkor az is jól látható, hogy az adatleírók bevonásával, kevés

szenzor esetén, a hatékonyságnövekedés sokkal jelentősebb, mint több szenzorral. A javulás két szenzorral 10-20% közötti. Viszont több szenzor bevonásával ez egyre inkább csökken. Például mind az 5 szenzor használatakor az adatleírók hozzáadása a nyers adathoz már kevesebb, mint 2%-os javulást hozott.



2. ábra. A WARD adatbázis szenzorjainak elhelyezése az emberi testen

Az is jól megfigyelhető, hogy mindössze két adatgyűjtő felhasználásával már elég magas, 95% feletti felismerési arányt értünk el. Tehát az alkalmazás jellegétől függően, a WARD adatbázis 13 cselekvésének a felismeréséhez elegendő lehet 2 adatgyűjtő szenzor is az eredeti 5 helyett, mivel ez a választás mindössze 3.67% felismerés csökkenést okoz. Fontos kihangsúlyozni, hogy ebben az esetben az adatleírókat is felhasználtuk, valamint az adatgyűjtő eszközök mindkét inerciális szenzorját. Az itt kapott eredmények a [J2] munkánkban találhatóak és ennek alapján fogalmazódott meg az első tézispont:

1. tézis: *A WARD adatbázis adatainak felhasználásával, méréseink során megmutattuk, hogy már két adatleíró felhasználásával a neurális háló felismerési aránya egyértelműen növelhető. Egy, két és öt adatgyűjtővel a legjobb felismerési arányok 87.88%, 95.51% és 99.18% voltak és ezekben az esetekben, az adatleírók bevonása 15.65%, 9% és 1.11%-os javulást hozott. Ennek alapján azt állapítottuk meg, hogy kevés adatgyűjtő eszköz használatakor az adatleírók alkalmazása jelentősen növeli a felismerési arányt, viszont az*

adatgyűjtők számának növelésével, az adatleírók pozitív hatása a felismerési arányra, egyre inkább csökken.

3.2.1. Inerciális szenzorok adatainak fúziója

Ahogy korábban említettem, a [J2]-es cikkünkben azt is vizsgáltuk, hogy az adatgyűjtőkben elhelyezett gyorsulásszenzor önmagában, illetve a giroszkóppal együtt, hogyan befolyásolja a felismerési arányt. Hasonlóan az előzőhöz, itt is több különböző szenzor kombinációt próbáltunk ki.

Ennek az eredménye a 3. ábrán látható, ahol kék vonallal ábrázoltuk a felismerési arányt abban az esetben, amikor csak a gyorsulásszenzor adatait használtuk fel, míg a piros görbe azt a felismerési arányt mutatja (mindkét esetben százalékos arányban megadva), amikor mindkét inerciális szenzor adatait felhasználtuk. Összesen 270 szimulációt futattunk és az így kapott eredményeket ábrázoltuk. Az ábra alapján jól megfigyelhető, hogy mindkét inerciális szenzor felhasználásával jobb felismerési arányokat kaptunk. Továbbá itt is észrevehető, hogy ez az arány a szenzorok kombinációjától is függ.

3. táblázat. Felismerési arányok (százalékosan) a szenzorok számának és elhelyezésének függvényében

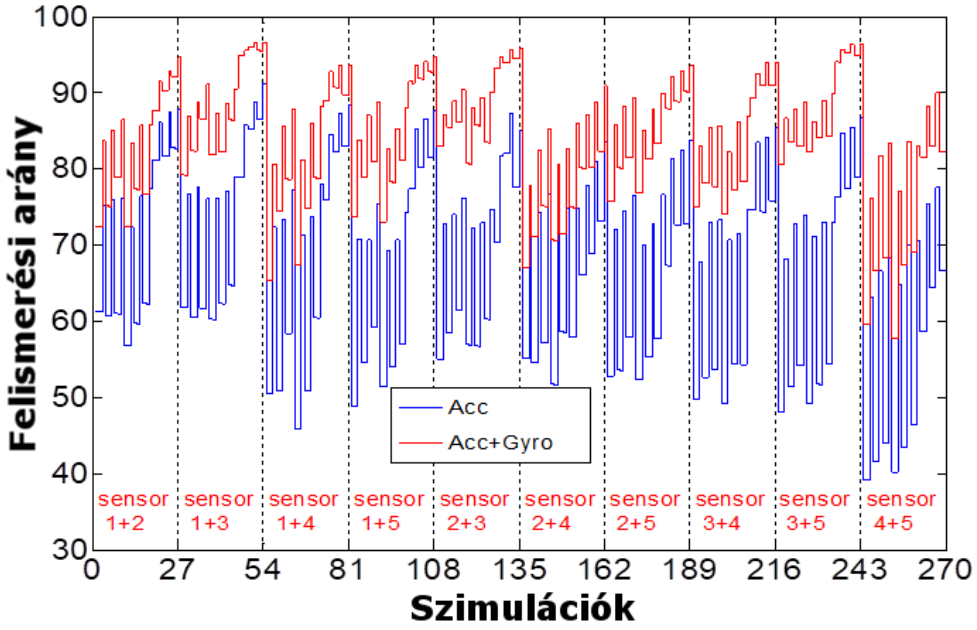
Szenzor konfiguráció	**Bemenő adat**		
	Nyers adat	Nyers adat + SMA	Nyers adat + SMA + V
1	65.96	69.70	83.64
2	68.96	70.33	74.38
3	72.23	74.05	87.88
4	49.23	48.83	60.41
5	52.50	53.80	60.64
1-2	78.87	76.55	92.21
1-3	86.51	86.41	95.51
1-4	78.62	78.82	89.71
1-5	81.02	81.20	92.73
2-3	86.12	83.54	94.43
2-4	75.12	75.02	82.35
2-5	81.55	83.32	90.23

3-4	77.70	78.27	90.96
3-5	82.97	84.32	94.95
4-5	68.38	69.06	82.22
1-2-3	89.98	88.76	97.93
1-2-4	87.32	87.86	95.38
1-2-5	87.14	87.89	96.25
1-3-4	88.99	88.24	97.33
1-3-5	90.33	90.04	97.45
1-4-5	88.46	87.06	95.55
2-3-4	88.54	86.39	95.66
2-3-5	91.41	92.53	98.17
2-4-5	85.91	89.19	93.58
3-4-5	88.26	89.63	97.08
1-2-3-4	94.60	94.78	98.82
1-2-3-5	96.48	96.30	99.03
1-2-4-5	95.33	95.18	98.73
1-3-4-5	96.38	96.33	98.88
2-3-4-5	96.83	95.70	99.07
1-2-3-4-5	98.07	98.38	99.18

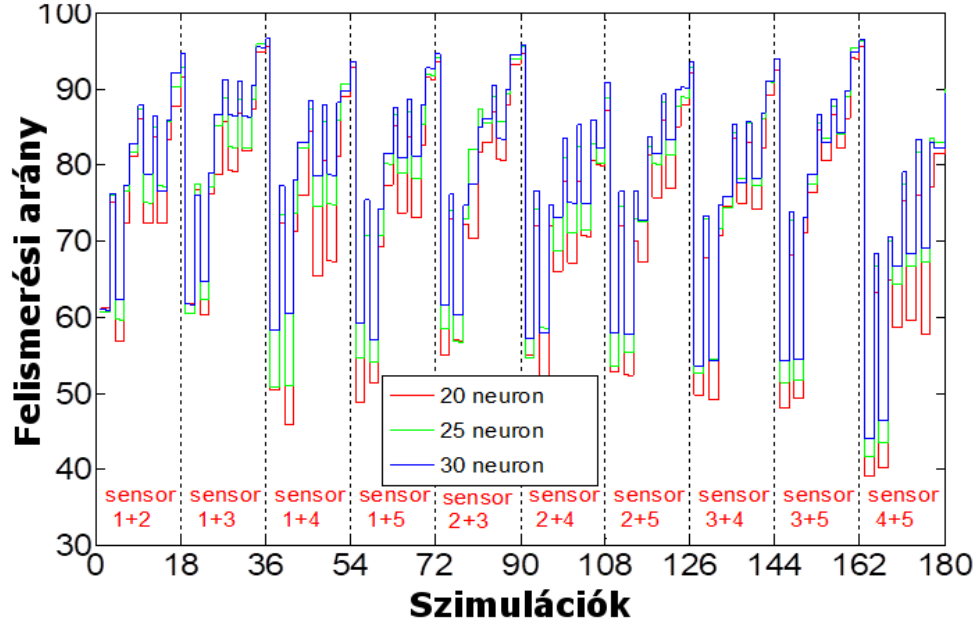
3.2.2. A köztes rétegen elhelyezett neuronok számának hatása a felismerési arányra

A [J2] tanulmányunk harmadik részében, a köztes rétegen elhelyezett neuronok számának a hatását vizsgáltuk a felismerési arányon. Ennek az eredménye a 4. ábrán tekinthető meg.

Ahogyan az látható 20, 25 és 30 neuronnal teszteltük a hatékonyságot, amelyek felismerési görbéjét különböző színekkel ábrázoltuk. A szimulációs környezet megegyezett az iménti kísérletével, tehát ugyan azokat az adatgyűjtő kombinációkat használtuk és itt is több szimulációt futtatunk minden kombinációval. Az eredmények alapján megfigyelhető, hogy a neuronok számának növelése csak csekély felismerési arány növekedést okoz. Ennek a megfigyelésnek az eredményét is alkalmaztuk a későbbi munkáink során, ahol a neurális hálók paramétereit vizsgáltuk.



3. ábra. Felismerési arányok a gyorsulásszenzor adataival, illetve a gyorsulásszenzor és a giroszkóp adatainak kombinációjával



4. ábra. A köztes szinten elhelyezett neuronok számának hatása a felismerési arányra

3.3. Normalizált nyers adat használata a tanuló algoritmus bemeneteként

Első pillantásra kézenfekvőnek tűnhet a normalizált és esetleg digitálisan szűrt nyers adatok használata a tanuló algoritmus bemeneteként. Ráadásul az adatleírók kinyerése nélkül a döntési folyamat is gyorsabb lehetne. Ezt a megközelítést választották az [49] cikk szerzői is, akik 3-tengelyű gyorsulásszenzor adatait fűzték egybe és ezt használták egy hidden Markov modell bemenő adataként.

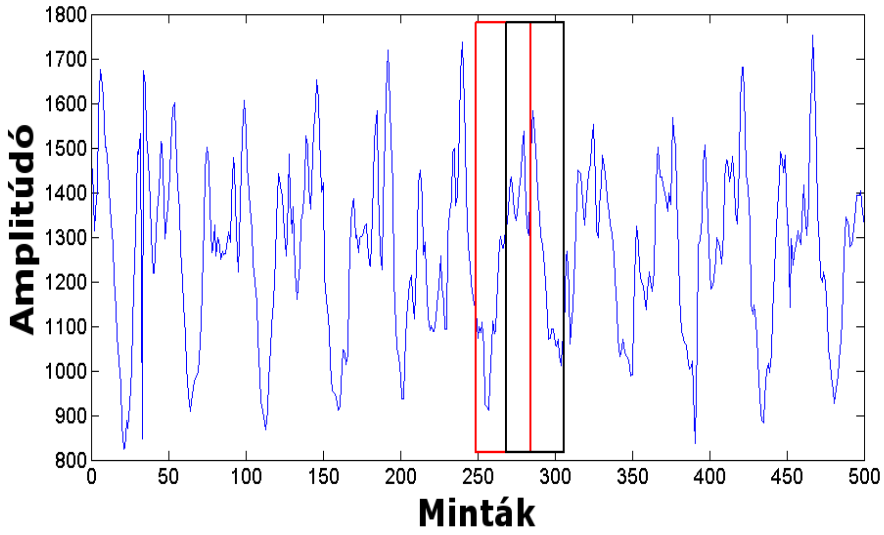
A nyers adatok használata más gépi-tanulási problémában sem ritka. Például az [50] referenciában a szerzők képek pixeleit módosítatlanul használták egy sokrétegű neurális háló bemeneteként objektum felismerésre és hasonlóan jó eredményt értek el, mint mások konvolúciós hálók használatával. Ez arra motivált minket, hogy megvizsgáljuk a normalizált nyers adatok hatékonyságát neurális hálózatok bemeneteként. Ebből kiindulva a korábban ismertetett [J2]-es cikkünkben és a későbbi [J4]-es munkánkban is használtunk normalizált nyers adatokat. Eredeti állapotukban az adatok és az adatleírók skálafüggők lehetnek, ezért normalizációra van szükséges. A [J2], [J4] és a későbbi munkáink során is a normalizálás a standard-normalizáló módszer (4) alapján történt, ahol $\mathbf{X}_{\text{norm}}$ és $\mathbf{X}_{\text{raw}}$ a nyers és normalizált adatmátrix, míg $\boldsymbol{\mu}$ és $\boldsymbol{\sigma}$ a nyers adatmátrix egyes adatvektorainak a szórása és várható értéke.

$$\mathbf{X}_{\text{norm}}(i, j) = \frac{\mathbf{X}_{\text{raw}}(i, j) - \boldsymbol{\mu}(j)}{\boldsymbol{\sigma}(j)} \tag{4}$$

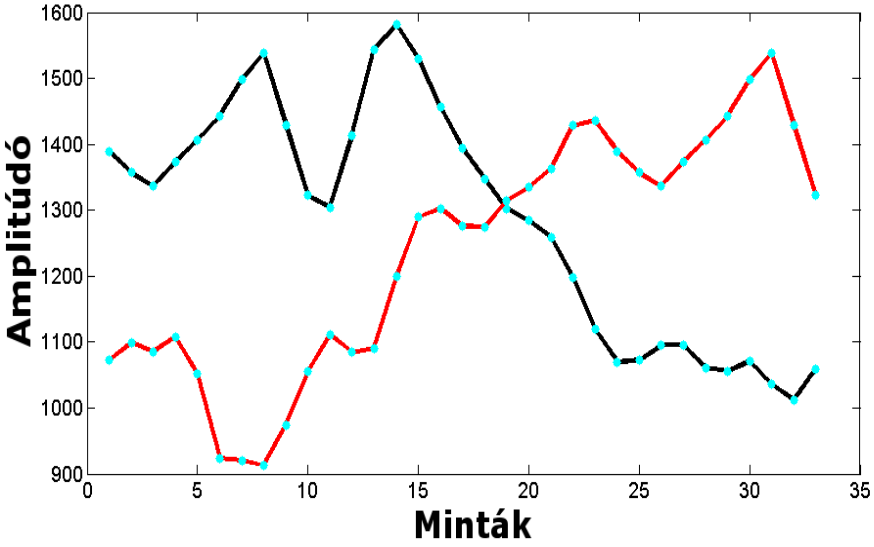
A [J4]-es munkánkban azt vizsgáltuk, hogy két illetve három réteggel és a rétegeken eltérő neuron számmal rendelkező neurális hálózatok milyen felismerési arányt adnak, ha a bementük az ablakokban lévő normalizált nyers adatok, az adatok összege valamint azok szórása. Eltérően a [J2]-es cikktől, itt egy általunk létrehozott adathalmazon dolgoztunk, viszont a végső következtetéseink itt is megegyeztek a [J2]-es cikk tapasztalataival. Ezt követően egy másik tanulmányban [J5] már a teljes WARD adatbázist használtuk adatforrásként, ahol kizárólag csak a normalizált nyers adathalmaz volt a bemenete egy általunk definiált neurális hálónak (*ANN1* architektúra - részletes leírása a 4.1 fejezetben olvasható). 100 futtatás után, amelynek során a háló főbb paraméterei (erről bővebben szintén a 4.1 fejezetben) véletlenszerűen egy exponenciális skála mentén lettek kiválasztva, a 7. ábrán

látható eredményeket kaptuk. Ezen az ábrán és a dolgozat további ábráin is (ahol releváns) a felismerést valószínűségi arányban adom meg. A legmagasabb felismerési arány 91.8% volt, ami alacsonyabb, mint néhány korábbi eredmény ugyan ezen az adatbázison (2. táblázat). Ez azt észrevétel rámutatott arra, hogy a tanuló algoritmus hatékonyság növelésének egyik kulcsa az adatleírókban rejlik, mivel a normalizált nyers adat (adatleírók nélkül vagy helyett) nem képes maximalizálni a tanuló algoritmus hatékonyságát. Véleményünk szerint ennek az egyik oka a cselekvést jellemző minta, váltakozó eloszlása a csúszó ablakon belül.

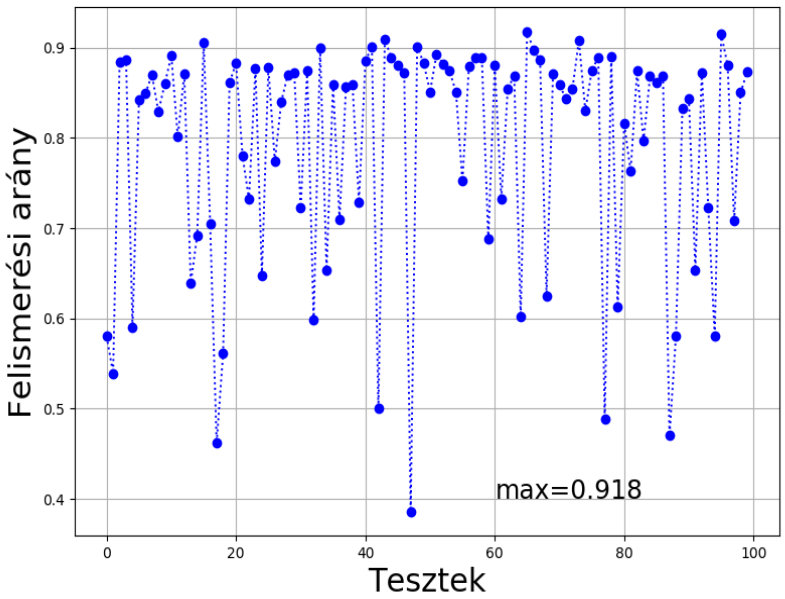
Vegyük alapul az 5. ábrán látható két szomszédos ablakot (piros, fekete) egy adott gyorsulásszenzor jelén (normalizálatlan nyers értékek), ami séta közben lett rögzítve. Az első ablakban az egyes értékek amplitúdója szakaszosan növekszik balról jobbra ezzel szemben a második ablakban ez a tendencia fordított. A szemléltetés érdekében a két ablakban lévő adatok a 6. ábrán kinagyítottan is megtekinthetőek. Annak ellenére, hogy mindkét ablak ugyan abból a cselekvésből származik, és hasonló paraméterekkel rendelkezik (pl.: átlag), eltérő módon fogják aktiválni a háló bemeneti neuronjait. Ebben az esetben a neuronok optimális súlyának a megtalálása sokkal nehezebb.



5. ábra. Két szomszédos ablak elhelyezkedése egy gyorsulásszenzor jelén



6. ábra. Az ablakokban elhelyezkedő adatpontok kinagyított formában



7. ábra. A neurális háló felismerési arányai normalizált nyers adattal a WARD adatbázison

3.4. Adatleírók

A fent bemutatott ablakok képzik az adatleíró származtató algoritmusok bemenetét, amelyeknek az a célja, hogy olyan jellegzetes információt nyerjenek ki a nyers adathalmazból, ami az adott ablakot jellemzi és független az ablakon belüli minta elhelyezkedésétől. Egy megfelelő adatleíró halmaz figyelemre méltóan képes növelni a felismerés hatékonyságát, valamint a tanuló algoritmus bemenő adatainak a számát is csökkentheti. Mivel a szakirodalomban nincs általánosan elfogadott adatleíró halmaz, ezért első lépésként összegyűjtöttük a leggyakrabban használt módszereket, amelyek a 4. táblázatban láthatóak [C1, J6].

4. táblázat. A leggyakrabban használt adatleírók

Tartomány	Adatleíró	Rövidítés	Referencia
Idő	Várható érték	M	[4, 35]
	Szórásnégyzet	V	[4, 35]
	Átlagos abszolút eltérés	MAD	[5, 15]
	Négyzetes közép	RMS	[5, 15]
	Nulla átmenet ráta	ZCR	[4, 5]
	Interkvartilis tartomány	IQR	[5, 15]
	75. percentilis	PE	[5, 35]
	Kurtózis	KS	[2, 15]
	Jel nagyságterület	SMA	[4, 15]
	Maximum-minimum különbség	MM	[7, 51]
	Tengelyek közötti korreláció	CORR	[4, 7]
	Autoregresszív együttható	AR1, AR2	[11, 33]
	Dőlésszög	TA	[33, 51]
Frekvencia	Spektrum energia	SE	[4, 35]
	Spektrum entrópia	E	[15, 35]
	Spektrum közép	SC	[4, 51]
	Alap frekvencia	PF	[14, 35]

A táblázat első négy eleme jól ismert statisztikai paraméterek, amelynek a bemutatására nem térek ki, viszont a többi adatleírót röviden ismertetem. A következő egyenletekben T jelöli az ablak méretét, F a frekvencia komponensek száma, $a_i(t)$ jelöli az $i.$ idősor (a szenzor $i.$ tengelye vagy más szóval dimenziója) $t.$ elemét és $A_i(f)$ az $f.$ frekvencia komponens.

- A nulla átmenet ráta az idősorban bekövetkezett előjelváltozásokat számolja. A lenti képletben az $I\{x\}$ függvény 1-el tér vissza, ha az argumentuma igaz, különben 0-val.

$$\mathrm{ZCR}_i = \frac{1}{\mathrm{T}\text{-}1}\sum_{t=1}^{\mathrm{T}} \mathrm{I}\left\{\alpha_i(t)\,\alpha_i(t+1) < 0\right\} \tag{5}$$

- A kvartilisek ($Q1$, $Q2$, és $Q3$) az a három pont, amely negyedekre bontja a rendezett idősort. IQR a felső és az alsó kvartilisek közötti különbség: $IQR = Q3 - Q1$. Ez a mérőszám az adatok szétszórtságáról ad információt egy adott tartományon belül. A percentilisek hasonlóak a kvartilisekhez, azzal a különbséggel, hogy ebben az esetben az idősor tetszőleges részekre osztható (százalékosan megadva). Tehát, a 25. percentilis megegyezik $Q1$-el, míg a 75. percentilis egyenlő $Q3$-al.

- A kurtózis arról ad információt, hogy az adathalmaz eloszlásának a szélei milyen lecsengésűek a normál eloszlás görbéjéhez viszonyítottan.

$$\mathrm{KS}_i = \frac{\frac{1}{\mathrm{T}}\sum_{t=1}^{\mathrm{T}}\left(\alpha_i(t) - \mathrm{M}_i\right)^4}{\left(\frac{1}{\mathrm{T}}\sum_{t=1}^{\mathrm{T}}\left(\alpha_i(t) - \mathrm{M}_i\right)^2\right)^2} \tag{6}$$

- A jel nagyságterület, a normalizált összege, egy adott szenzor dimenziói abszolút értékének. Például egy háromtengelyes gyorsulásszenzor (x, y, z) esetén:

$$\mathrm{SMA} = \frac{1}{\mathrm{T}}\sum_{t=1}^{\mathrm{T}} \left|\alpha_{\mathrm{x}}(t)\right| + \left|\alpha_{\mathrm{y}}(t)\right| + \left|\alpha_{\mathrm{z}}(t)\right| \tag{7}$$

- A maximum-minimum különbség egyszerűen az adatsor legkisebb és legnagyobb elemei közötti eltérés.

- A tengelyek közötti korreláció, két tengely közötti lineáris kapcsolatot méri. Tehát az *i.* és *j.* tengely esetén a korrelációt $CORR_{i,j}$-vel jelöljük, ami a $-1 \leq CORR_{i,j} \leq 1$ intervallumba esik, ahol az előjel a korrelációs kapcsolat irányát jelzi.

$$\mathrm{CORR}_{i,j} = \frac{\sum_{t=1}^{\mathrm{T}} \left(a_i(t) - \mathrm{M}_i\right)\left(a_j(t) - \mathrm{M}_j\right)}{\sqrt{\sum_{t=1}^{\mathrm{T}} \left(a_i(t) - \mathrm{M}_i\right)^2 \left(a_j(t) - \mathrm{M}_j\right)^2}} \tag{8}$$

- Az autoregresszív együtthatók kiszámítása és felhasználása, lehetővé teszi az idősor egy másik reprezentációjának előállítását (9), ahol ϕ_k jelöli az együtthatókat és $\varepsilon(t)$ a fehér zajt. Ebben a modellben az idősor egy eleme közelíthető az azt megelőző elemek lineárisan súlyozott összegével, ahol a súlyokat az autoregresszív együtthatók adják. Ezeknek a meghatározása a legtöbb esetben a *Yule-Walker* egyenletek alapján történik (10), (11), ahol r_k jelöli az autokorrelációs együtthatókat. Munkánk során mi az első két autoregresszív együtthatót használtuk.

$$\mathrm{AR}(\rho) \rightarrow \alpha_i(t) = \sum_{k=1}^{\rho} \phi_k \alpha_i(t-k) + \varepsilon(t) \tag{9}$$

$$\mathrm{r}_k = \frac{\frac{1}{\mathrm{T}} \sum_{t=1}^{\mathrm{T}-k} \left(\alpha_i(t) - \mathrm{M}_i\right)\left(\alpha_i(t-k) - \mathrm{M}_i\right)}{\mathrm{V}_i} \tag{10}$$

$$\boldsymbol{\phi} = \begin{bmatrix} 1 & \mathrm{r}_1 & \mathrm{r}_2 & \cdots & \\ \mathrm{r}_1 & 1 & \mathrm{r}_1 & \cdots & \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathrm{r}_{\rho-1} & \mathrm{r}_{\rho-2} & \mathrm{r}_{\rho-3} & \cdots & \end{bmatrix}^{-1} \begin{bmatrix} \mathrm{r}_1 \\ \mathrm{r}_2 \\ \\ \mathrm{r}_\rho \end{bmatrix} \tag{11}$$

- A dőlésszög egy háromtengelyű gyorsulásszenzor esetén megadja a *z* tengely és a gravitációs vektor közötti szöget.

$$\mathrm{TA} = \arccos\left(\frac{\alpha_z(t)}{\sqrt{\alpha_x(t)^2 + \alpha_y(t)^2 + \alpha_z(t)^2}}\right) \tag{12}$$

- Általánosan egy adott jel energiáján az elemei abszolút értékének négyzete által alkotott görbe alatti területet értjük. Tehát a spektrum energia a frekvencia komponensek abszolút értékének négyzetösszegével közelíthető.

$$\mathrm{SE}_i = \sum_{f=1}^{\mathrm{F}} \left|\mathrm{A}_i(f)\right|^2 \tag{13}$$

- Az entrópia egy adott rendszer rendezettségi fokát határozza meg. Az információ elméletben ezt a mérőszámot, a rendszer információtartalmának meghatározásához használják (Shannon entrópia), ami a rendszer lehetséges állapotainak valószínűségeiből számítható. Erre támaszkodva egy jel spektrumának az entrópiája a következő képletekkel adható meg:

$$\mathrm{p}_f = \frac{\left|\mathrm{A}_i(f)\right|^2}{\sum_{j=1}^{\mathrm{F}} \left|\mathrm{A}_i(j)\right|^2} \tag{14}$$

$$\mathrm{E}_i = -\sum_{f=1}^{\mathrm{F}} \mathrm{p}_f \log_2\left(\mathrm{p}_f\right) \tag{15}$$

- A spektrum közép kiszámításakor, a spektrumot egy valószínűségi eloszlásnak tekintjük, és ennek határozzuk meg a várható értékét.

$$\mathrm{SC}_i = \frac{\sum_{f=1}^{\mathrm{F}} f\left|\mathrm{A}_i(f)\right|}{\sum_{j=1}^{\mathrm{F}} \left|\mathrm{A}_i(j)\right|} \tag{16}$$

- Az alap frekvencia azt a frekvencia komponenst jelenti, amely a legnagyobb amplitúdóval rendelkezik (a DC komponenst figyelmen kívül hagyjuk).

$$\mathrm{PF}_i = \max_f\left(\mathrm{A}_i(f)\right) \qquad f \neq 0 \tag{17}$$

Mivel a folytonos hullám transzformáció kapcsolatot nyújt egy adott jel idő és frekvenciatérbeli reprezentációja között, ezért széleskörűen használják számos jelfeldolgozó alkalmazásban. A transzformáció képlete (18) gyakorlatilag azt mondja, hogy az adatsorunkon *x(t)* végigúsztatunk (a τ paraméter használatával, ami egyben az idő skálát adja) egy úgynevezett „anya hullám" ψ skálázott módosulásait, amit az *s* paraméter határoz meg (frekvencia skála).

A gyakorlatban a transzformáció gyorsabb elvégzése érdekében a hullám transzformációt digitális szűrők hierarchikus struktúrájával számíthatjuk ki. Ebből kiindulva néhány kutató számára vonzónak tűnt ennek a transzformációnak a használata a cselekvés felismeréshez. Valójában egy jel idő és frekvenciatérbeli ábrázolására a rövid idejű Fourier transzformáció is alkalmazható (STFT). Viszont ezzel szemben a hullám transzformáció előnye az, hogy jobb felbontást biztosít az idő és térbeli ábrázoláshoz. Ez annyit jelent, hogy a nagy frekvenciás komponensek nagyobb felbontással rendelkeznek az idő tengelyen, míg az alacsony frekvenciásak a frekvencia tengelyen [52].

Munkánk során mi nem használtunk hullám transzformációt, egyrészt azért, mert a [35] tanulmány eredményei azt mutatták, hogy az idő és frekvenciatérbeli adatleírók hatékonyabban alkalmazhatók a hullám transzformáción alapuló megoldásoknál. Másrészt, a gépi-tanuláson alapuló, automatizált cselekvés felismerés esetén, a hullám transzformációnak nincs sok haszna, mivel egy adott időintervallum alatt begyűjtött adatokra (egy ablakra) akarunk döntést hozni, így valójában nem is kell tudni azt, hogy a frekvencia komponensek az ablakon belül, időben hol helyezkednek el.

$$\mathrm{CWT}(\tau, s) = \frac{1}{\sqrt{|s|}} \int_{-\infty}^{+\infty} \mathrm{x}(t) \psi^* \left(\frac{t - \tau}{s} \right) \mathrm{dt} \tag{18}$$

3.5. Adatleírók kiválasztása

Néha az adatleírók száma túl sok lehet, amelyek közül néhány szükségtelen. Az adatleíró kiválasztás során megpróbáljuk meghatározni a kiinduló adatleírók halmazának azt a részhalmazát, amely nem tartalmaz redundanciát és irreleváns adatleírókat. A kiválasztás folyamata a legtöbb esetben az adatleíró halmazok eloszlásán vagy korrelációs kapcsolataik alapján zajlik. Ezt gyakran használják dimenziócsökkentésre, mivel

egyszerűsítheti és gyorsíthatja a tanuló modellt. Ellentétben más gyakran alkalmazott dimenziócsökkentési technikákkal, mint például az *Linear discriminant analysis* vagy *Principal component analysis*, amik az adathalmaz projekcióján alapszanak, az adatleíró kiválasztás nem módosítja az adatleíró halmaz eredeti reprezentációját [53].

Annak függvényében, hogy az adathalmaz címkézett, részlegesen címkézett vagy nem címkézett, ahol a címke azt az osztályt jelöli, amelyből az adatelem származik, három algoritmus kategória létezik: felügyelt, részben felügyelt vagy felügyelet nélküli. Ezen felül, az adatleíró osztályozási technika alapján, a kiválasztó algoritmusok további három csoportba sorolhatóak: szűrő (*filter*), keretező (*wrapper*) és beágyazott (*embedded*). A szűrő algoritmusok minden egyes adatleíróhoz jósági tényezőt számítanak ki és ez alapján rangsorolják őket. A keretező módszerek az adatleírók osztályozását a kiválasztott osztályozó vagy tanuló algoritmus bevonásával végzik és ezek teljesítménye alapján kategorizálják az adatleírókat. A beágyazott modellekben az adatleíró kiválasztás a tanulási folyamatba van beágyazva. Munkánk során mi kizárólag a felügyelt kategóriára koncentráltunk azon belül is főként a szűrő és részben a keretező algoritmusokra.

Napjainkban már gazdag szakirodalmi háttér van a kiválasztó algoritmusok mögött. A korábbi évtizedekben számos szűrő módszert dolgoztak ki, amelyek statisztikai, információ elméleti és egyéb megközelítéseken alapulnak. Ezeket az algoritmusokat különböző célokkal alkották meg különböző problémákhoz. Például, a [62] cikkben a szerzők egy algoritmust javasoltak a digitális képek hátterének manipulálásához, a [63] tanulmányban a *Laplace pontszám* hatékonyságát mutatták be az Iris és a PIE arc adatbázisokon, míg a [64] cikk szerzői egy kiválasztó algoritmust fejlesztettek digitális képekhez, amelyek az emberi mozgás pillanatfelvételeit ábrázolják. Az utolsó évtizedben könyvek és felmérések is megjelentek ehhez a témához kapcsolódódóan, amelyekben a legnépszerűbb algoritmusokat és azok alkalmazási területeit ismertették. Az [55] tanulmány erre mutat példát, amiben kiválasztó módszerek alkalmazási lehetőségeit mutatják be a bioinformatikában.

Ezen felül egy pár, az algoritmusokat összehasonlító cikk is létezik a szakirodalomban, ahol néhány módszer hatékonyságát mérték össze egy adott

adathalmazon. A [65] cikkben a *Gait ratio* és néhány korreláció vizsgálat alapú technika, eltérő tanuló algoritmusokkal kombinálva volt tesztelve, cukorbetegséghez kapcsolódó adatbázison. A [66] cikk szerzői 5 szűrő módszert teszteltek szövegosztályozási problémán, ahol a szerzők kapcsolatot figyeltek meg egyes módszerek között.

A szűrő algoritmusokkal ellentétben a keretező algoritmusokhoz kapcsolódó cikkek száma jóval kevesebb. Az egyik legismertebb ide kapcsolódó cikk a [67]. Ebben a cikkben, Kohavi és John egy átfogó leírást ad a keretező technikák erősségeiről, gyengeségeiről és a keresési stratégiákról. Erre támaszkodtak a [68] tanulmány szerzői is, ahol a *Correlation-based adatleíró kiválasztó* (CFS) algoritmust hasonlították össze keretező technikákkal.

A tanuló algoritmuson felül, a keretező módszerek más paraméterektől is függenek, mint például a keresési stratégia, megállási feltétel, stb. Sajnos a keretező algoritmusok pontos leírása nem elérhető vagy hiányos a cikkekben [69]. Ez adta az okot arra, hogy létrehozzunk egy saját keretező módszert, amelyet felhasználtunk munkánk során.

A valós idejű alkalmazásokban a szűrő algoritmusok előnyben részesítettek a keretezővel szemben, mivel ezek az osztályozó algoritmustól függetlenül alkalmazhatóak és sokkal gyorsabbak, mint a keretező módszerek. Azonban, a keretező technikák hatékonyabbak lehetnek, mivel figyelembe veszik az osztályozó működését és egyben az adatleírók közötti függőségeket is kezelik [54]. Tehát mindkét módszernek megvannak az előnyei és hátrányai is egyaránt. A kiválasztó algoritmus kategóriákhoz kapcsolódó további információk a [61] referenciában találhatóak. Valójában függetlenül az algoritmus kategóriáktól és csoportoktól, a cél minden esetben az, hogy a kiindulási „adatleíró térben” egy olyan alteret (adatleírók részhalmazát) találjanak, ahol az egyes osztályokhoz tartozó adathalmazok jól elkülöníthetőek.

Annak ellenére, hogy elég sok tanulmány született már a tevékenység felismerés témakörében, azoknak a tanulmányoknak a száma, ahol kiválasztó algoritmussal vizsgálták az adatleírók hasznosságát, meglehetősen kevés. A kutatók többsége nem végzett semmilyen kapcsolatvizsgálatot az általuk használt adatleírók között. Összesen 3 cikket találtam, amelyben a szerzők kiválasztó módszert is használtak. Az itt alkalmazott algoritmusok szintén a

felügyelt kategóriába tartoznak. Az [5] cikk szerzői az adatleírók számának csökkentését a CFS algoritmussal végezték. A [56] tanulmány szerzői az *Information gain* (IG) algoritmusra támaszkodtak. Végül a [57] referenciában az *Minimum redundancy maximum relevance* (MRMR) módszer volt kiválasztásra használva. Mindhárom technika szűrő típusú, ami egyben azt is tükrözi, hogy az adott témakörben a keretező technikák teljesen figyelmen kívül maradtak. Ez volt az egyik motivációja az [J6] munkáinknak, amelyekben 8 szűrő és egy keretező algoritmust teszteltünk 3 tanuló algoritmussal köztük a k-legközelebbi szomszéddal és neurális hálóval a WARD adatbázison. Ehhez a tanulmányhoz én készítettem el a Matlabos teszt környezetet és én végeztem el a méréseket. A szerzőtársaim a kapott eredmények értelmezésében nyújtottak segítséget. A tesztkörnyezetben a szűrő algoritmusokat az [58] referenciában megadott nyilvános adattárból használtam fel, amit az Arizonai Egyetem egyik kutatócsoportja hozott létre. Az adattárból, az algoritmusok Matlab csomagként tölthetőek le. Név szerint a következő 8 szűrő algoritmust foglalta magába a tanulmány:

- CFS
- MRMR
- IG
- CHI (Chi square)
- FCBF (Fast correlation-based filter)
- FIS (Fisher score)
- KW (Kruskal-Wallis)
- T-teszt

Ezekhez társult egy általunk kidolgozott keretező algoritmus is, ami a naiv Bayes (NB) tanuló algoritmust használta fel. Az algoritmus részletes bemutatására a [C1] cikkünkben került sor. Azt feltételeztük, hogy azok az adatleírók, amikkel az NB hatékonyan teljesít, azok hatékonyak lesznek más tanuló algoritmusok esetében is. Az NB választása abból eredt, hogy a keretező módszer önmagában is egy lassú folyamat, és ha ezt egy időigényes tanuló algoritmussal ötvözöm, mint a neurális háló (tanulási fázis) vagy a k-legközelebbi szomszéd (döntési fázis), akkor ez egy valós idejű alkalmazásban, nagy adathalmaznál használhatatlanná válna.

Ahogy a neve is jelzi, az NB a *Bayes'* szabályon alapszik (19), amely az utólagos valószínűséget (*posterior)* az előzetes (*prior*) és az osztályfüggő (*class-conditional*) valószínűségekből határozza meg.

$$P(c_i|\, x_j) = \frac{P(x_j \mid c_i)\, P(c_i)}{P(x_j)} \tag{19}$$

A fenti képletben, $P(c_i)$ adja meg, hogy egy adott minta mekkora valószínűséggel esik az *i.* osztályba, $P(x_j)$ mutatja, hogy egy minta milyen valószínűséggel esik a *j.* intervallumba, $P(x_j|c_i)$ annak a feltételes valószínűsége, hogy a *j.* intervallumban lévő minta az *i.* osztályba tartozik. A mi esetünkben a *j.* intervallum alatt egy adatleíró értékkészletének egy kis részét értjük (hasonlóan egy hisztogramhoz), valamint a minta egy adatleíró vektor, amely egy adott osztályhoz sorolható. Ennek alapján az osztály-függő valószínűség a következő formában írható le, ahol *n* a vektor méretét adja meg.

$$P(\mathbf{x}_j|c_i) \;=\; P(\mathbf{x}_j^1, \mathbf{x}_j^2, \ldots\, \mathbf{x} \tag{20}$$

Amennyiben azt feltételezzük, hogy az adatleírók függetlenek, a (20) képlet gyakorlatilag az egyes komponensek szorzatával írható le.

$$\prod_{k=1}^{n} P(\mathbf{x}_j^k|c_i) \tag{21}$$

Mivel, $P(x_j)$ a (19) képletben csak egy normalizáló faktor, így az utólagos valószínűség közelíthető a (22) képlettel,

$$P(c_i|\mathbf{x}_j)^* \;=\; P(c_i) \prod_{k=1}^{n} P(\mathbf{x}_j^k|c_i) \tag{22}$$

Tehát az utólagos valószínűség alapján egy új mintát a következő egyenlőtlenség alapján rendelhetünk egy adott osztályhoz,

$$P(c_i|\mathbf{x}_j)^* \;>\; P(c_k|\mathbf{x}_j)^* \qquad \forall i \neq k \tag{23}$$

Ez az egyenlőtlenség a *maximum posteriori hipotézis*, amely a legvalószínűbb osztályt rendeli minden mintához. Az imént leírt műveletsor az NB tanuló algoritmus alapja [60].

Tehát a kidolgozott keretező program a fent vázolt NB-n alapuló mohó algoritmus, ami előre kiválasztó stratégiát használ két megállási feltétellel: *n*

<= i és $felismerés_{i+1}$ <= $felismerés_{i,}$ ahol i az iterációk, míg n a maximális kiválasztható adatleírók számát jelöli. Az algoritmus pszeudokódjai a 8. 9. és 10. ábrákon látható.

Az algoritmus paraméterei a maximálisan kiválasztható adatleírók száma és az **F** adatleíró mátrix (24). A visszatérési értéke pedig a kiválasztott adatleírók indexét tartalmazó vektor (**Sf**). Az egyszerűség kedvéért a kiválasztó algoritmusban azt feltételeztük, hogy az **F** egy harmadrendű tenzor (adatleírók, minták, osztályok), amelyben minden osztályhoz ugyan annyi minta tartozik. A pszeudokódban a kettőspont egy adott dimenzió összes elemére utal, hasonlóan, mint a Matlabban.

Require: F
Ensure: Sf
  $F \leftarrow normalize(F)$
  $Pc \leftarrow calculatePc(F)$
  $Pxc \leftarrow calculatePxc(F)$
  for f : 1 **to** features **do**
    $Fs(f) \leftarrow bayesScore(F(f,:,:), Pxc(f,:,:), Pc)$
  end for
  $Sf(1) \leftarrow maxIndex(Fs)$
  for a : 1 **to** features - 1 **do**
    for f : 1 **to** features **do**
      if f was not selected yet **then**
        for sf : 1 **to** a **do**
          $F\_temp(sf,:,:) \leftarrow F(Sf(sf),:,:)$
          $Pxc\_temp(sf,:,:) \leftarrow Pxc(Sf(sf),:,:)$
        end for
        $F\_temp(a+1,:,:) \leftarrow F(f,:,:)$
        $Pxc\_temp(a+1,:,:) \leftarrow Pxc(f,:,:)$
        $Fs(f) \leftarrow bayesScore(F\_temp, Pxc\_temp, Pc)$
      else
        $Fs(f) \leftarrow 0$
      end if
    end for
    $findex \leftarrow maxIndex(Fs)$
    $recognition\_rate \leftarrow max(Fs)$
    $Sf(a+1) \leftarrow findex$
    if recognition_rate $=$ 100 **then**
      break loop
    end if
  end for

8. ábra. Az NB alapú keretező, kiválasztó algoritmus

```
Require: F
Ensure: Pxc
  dx ← 2/bins
  for c : 1 to classes do
    for f : 1 to features do
      for s : 1 to samples do
        for b : 0 to bins - 1 do
          if (-1 + b*dx) ≤ F(f,s,c) and (-1 + (b+1)*dx) >
          F(f,s,c) then
            Pxc(f, b+1, c) ← Pxc(f, b+1, c) + 1
          end if
        end for
      end for
      for b : 1 to bins do
        Pxc(f, b, c) ← Pxc(f, b, c)/(features * samples)
      end for
    end for
  end for
```

9. ábra. A Pxc valószínűségi tenzor számítása

```
Require: F, Pc, Pxc
Ensure: recognition_rate
  for c : 1 to classes do
    for s : 1 to samples do
      for f : 1 to features do
        for b : 0 to bins - 1 do
          if (-1 + b*dx) ≤ F(f,s,c) and (-1 + (b+1)*dx) >
          F(f,s,c) then
            bin(f) ← b + 1
          end if
        end for
      end for
      for c1 : 1 to classes do
        Pcx(c1) ← 1
        for f : 1 to features do
          Pcx(c1) ← Pcx(c1) * Pxc(f, bin(f), c1)
        end for
        Pcx(c1) ← Pcx(c1) * Pc(c1)
      end for
      cindex ← maxIndex(Pcx)
      predicted_class(c, s) ← cindex
    end for
  end for
  recognition_rate ← calculate_rate(predicted_class)
```

10. ábra. A *bayesScore* függvény

Az algoritmus első lépésben normalizálja az adatleírókat, ami egyszerűen annyi jelent, hogy minden adatleíró elemet beoszt a hozzá tartozó legnagyobb abszolút értékű elem plusz eggyel. A második lépésben meghatározza az osztály valószínűségeket $P(c_i)$. Ezt követően, az osztály-függő valószínűségek kiszámítása következik a teljes adathalmazból. Ezeket a valószínűségeket a *Pxc* tenzor tartalmazza (adatleírók, intervallumok, osztályok), amelynek elemei megmutatják, hogy az *i.* osztályban egy adatleíró milyen valószínűséggel esik a *j.* intervallumba. Ez után, az algoritmus, az eddigiek felhasználásával meghatározza a megfelelő intervallumot a vektor minden eleméhez és azt az osztályt választja, amelyik teljesíti a (23) egyenlőtlenséget (*bayesianScore* függvény). A *calculate_rate* függvény a kód végén a jósolt és a valós osztályok aránya alapján számítja ki a felismerési százalékot és ennek alapján lesz felvéve a leghatékonyabb adatleíró a kiválasztottak listájába.

Röviden összefoglalva az algoritmus működését, ez első lépésben kiválasztja azt az adatleírót, amellyel az NB a legjobban teljesít, majd ezt követően a még ki nem választott adatleírók közül mindig egyet hozzácsatol a már kiválasztottakhoz és tárolja az így kapott felismerési arányát az NB-nek. A végén az az adatleíró lesz a listára felvéve, amelyik a már addig kiválasztottakkal együtt a legjobb eredményt produkálta. Ennek a módszernek az egyetlen gyengepontja, hogy az adatleíró értékek besorolásához meg kell találni a részintervallumok számát ((-1, 1) tartományt hány részre osztjuk fel), amivel az algoritmus jól működik. Néhány teszt elvégzése után mi fixen 100 részintervallumot használtunk.

Ehhez a tanulmányhoz 7 személy adatait választottuk ki a WARD adatbázisból és ezeken végeztük el az adatleíró-kinyerést. A kiválasztás során fontos kritérium volt, hogy a kiválasztott adathalmazok eltérő korú személyektől származzanak. Az adatbázisban használt sorszámokkal jelölten, a következő 7 alany adatait használtuk:

- Alany 1: 19 éves
- Alany 2: 75 éves
- Alany 3: 27 éves
- Alany 4: 29 éves
- Alany 5: 20 éves
- Alany 6: 29 éves

- Alany 8: 34 éves

A 4. táblázatban két adatleíró (SMA, TA) a szenzorok tengelyeiről származó információt összevontan használják fel, míg a többiek egy adott értéket számítanak ki minden tengelyhez, vagy a korreláció esetén adott tengely párokhoz. Ennek alapján a 4. táblázatban lévő 17 adatleíró-kiszámító módszer 77 adatleíró értéket generált a szenzorok tengelyeinek adataiból (5 tengely) egy ablak méretű időszelethez. Ezáltal minden egyes személy adathalmazát egy **F** adatleíró mátrixszal lehetett reprezentál, ahol $f_{i,j}$ az *i.* ablakhoz kiszámított *j.* adatleírót jelöli.

$$\mathbf{F} = \begin{bmatrix} f_{1,1} & f_{1,2} & f_{1,3} & \cdots & \\ f_{2,1} & f_{2,2} & f_{2,3} & \cdots & ; \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ f_{\omega,1} & f_{\omega,2} & f_{\omega,3} & \cdots & ; \end{bmatrix} \tag{24}$$

Ez a mátrix volt a bemenő adata minden egyes kiválasztó algoritmusnak és az általuk kiválasztott 5, illetve 6 leghatékonyabbnak vélt adatleíró vektor adta a vizsgálatunkban használt tanuló algoritmusok bemenetét. Az így kapott adatleíró vektorok felhasználásával mértük a felismerési arányokat és ez alapján határoztuk meg a kiválasztó algoritmusok és a kiválasztott adatleírók hatékonyságát. Az egyes algoritmusok által kiválasztott adatleírók az 5-13 táblázatokban tekinthetők meg. A táblázatokban használt rövidítések a 4. táblázatból származnak, míg az alsó index arra a tengelyre utal, amelyikhez az adatleíró tartozik (ahol ez releváns).

5. táblázat. A CFS által kiválasztott adatleírók

Alany	**Adatleírók**
1	Mx, Mz, MADx, RMSx, RMSy, RMSz
2	Mx, My, MADz, RMSx, RMSy, RMSz
3	Mx, My, Vz, MADy, RMSx, RMSz
4	Mx, My, Mz, RMSy, RMSz, IQRx
5	Mx, My, Mz, Vx, Vz, RMSx
6	Mx, My, Mz, MADy, RMSx, RMSz
8	Mx, My, Mz, MADy, RMSx, IQRy

6. táblázat. A CHI által kiválasztott adatleírók

Alany	Adatleírók
1	PEy, PEz, MMz, MMy, PEx, SMA
2	PEy, MMx, PEx, MMy, MMz, RMSx
3	PEy, MMx, MMz, MMy, PEx, PEz
4	PEy, MMx, MMy, PEz, MMz, PEx
5	MMy, PEy, MMx, MMz, PEz, PEx
6	MMy, PEy, MMz, MMx, PEx, PEz
8	PEy, MMx, MMy, PEx, MMz, PEz

7. táblázat. A FCBF által kiválasztott adatleírók

Alany	Adatleírók
1	RMSx, PEy, RMSy, Mz, RMSz, SCx
2	SEx, PEx, SEz, RMSy, My, SMA
3	PEy, SMA, RMSx, My, RMSz, MMx
4	PEy, SEy, SEx, Mz, SCx, MADx
5	RMSx, PEy, Mx, TA, Ey, SCx
6	SMA, RMSz, PEx, PEy, Mx, MADz
8	PEy, RMSx, Mz, MADy, SCx, IQRx

8. táblázat. A FIS által kiválasztott adatleírók

Alany	Adatleírók
1	RMSx, MADx, Mx, PEx, SCz, Ey
2	SEz, RMSx, Mx, TA, PEx, Mz
3	RMSx, PEx, Mx, MADx, Ey, RMSz
4	MMx, MADx, MADy, MMy, SCx, SCz
5	Ey, SCz, RMSx, SCy, Ez, Mx
6	RMSx, PEx, Mx, SEx, SCy, MMz
8	MMx, MADy, RMSx, IQRy, Vy, SEx

9. táblázat. Az IG által kiválasztott adatleírók

Alany	Adatleírók
1	ZCRz, ZCRx, MMx, Ex, Ez, AR2x
2	AR2z, KSz, PFz, AR1y, ZCRy, CORRy
3	ZCRz, IQRz, Ex, AR1x, AR2x, AR2y
4	ZCRy, ZCRz, Ex, Ey, AR1x, AR2x
5	KSz, ZCRx, ZCRz, ZCRy, Ex, AR1x
6	AR1z, AR2y, PFz, AR2z, ZCRy, ZCRz
8	AR2y, ZCRz, Ex, Ey, AR1x, AR2x

10. táblázat. A KW által kiválasztott adatleírók

Alany	Adatleírók
1	Ey, SCy, SCz, SCx, IQRx, MADx
2	SCz, SCy, Ey, IQRy, KSz, MADy
3	SCy, SCz, IQRx, IQRy, MMy, MADx
4	SCx, SCy, SCz, IQRx, IQRz, MADx
5	Ey, Ez, SCy, SCz, SCx, IQRy
6	SCy, SCz, Ey, Ez, MADx, PEy
8	SCy, SCz, SCx, My, IQRy, IQRz

11. táblázat. Az MRMR által kiválasztott adatleírók

Alany	Adatleírók
1	PEy, TA, AR2y, AR2z, AR2x, AR1z
2	PEy, TA, SCz, SCy, Ey, Ez
3	MMx, TA, AR2z, AR1z, AR2x, AR2y
4	PEx, TA, AR2y, AR2z, AR2x, AR1z
5	MMy, TA, AR2z, AR2y, AR2x, AR1z
6	MMy, TA, AR2z, AR2y, SCz, Ez
8	PEy, TA, AR2z, AR2y, AR2x, AR1z

12. táblázat. A T-test által kiválasztott adatleírók

Alany	Adatleírók
1	MMx, Ex, Ez, AR2x, SEz, RMSz
2	ZCRz, RMSz, TA, SEy, SEz, RMSy
3	IQRz, Ex, AR1x, AR2x, AR2y, RMSz
4	Ex, Ey, AR1x, AR2x, ZCRy, ZCRz
5	Ex, AR1x, RMSy, RMSz, SEy, SEz
6	TA, SEz, RMSz, ZCRy, AR2x, RMSy
8	Ey, Ez, AR1x, AR2x, ZCRz, ZCRy

13. táblázat. A keretező módszer által kiválasztott adatleírók

Alany	Adatleírók
1	PEz, SMA, CORRy, KSx, CORRx, MMy
2	RMSy, My, MMz, KSx, PEz, PEx
3	PEy, MMx, SMA, PFy, IQRx, Ey
4	PEy, MMx, PEz, MMy, Mx, PFy
5	PEy, TA, CORRy, MMx, PFy, MMz
6	PEx, PEy, PFy, PEz, MMx, MMy
8	PEy, PEx, MMx, IQRx, SMA, PFx

Az adatleírók kiválasztása után történt a hatékonyságuk vizsgálata a tanuló algoritmusok felismerési arányaik alapján. Az így kapott eredményeket, minden egyes alanyhoz, az A. függelékben található táblázatok tartalmazzák. A tanulmány eredményei alapján a következő konklúziókat vontuk le:

- Az 5-13 táblázatok jól tükrözik, hogy nincs egy általánosan elfogadható adatleíró halmaz, amely az egyes személyektől függetlenül, hatékonyan alkalmazható.
- Az is jól megfigyelhető, hogy ugyan az a kiválasztó módszer csak részben átfedő adatleíró halmazokat választ ki az eltérő személyekhez. Ez az eltérő mozgási mintákkal is magyarázható, ami a korral együtt is változik.

- Az összesített eredmények alapján a CFS, CHI és a keretező módszer tűnt hatékonynak.
- A kiválasztó és a gépi-tanuló algoritmusok között is összefüggés figyelhető meg. Például: a neurális háló a keretező módszerrel volt a leghatékonyabb.

A [J6]-os munkánk célkitűzése és az elért eredményeink alapján fogalmazódott meg a 2. tézispont:

2. tézis: *A 8 szűrő és egy keretező algoritmussal végzett adatleíró kiválasztás eredményei, amit a WARD adatbázison kaptunk, azt mutatták, hogy nincs egy általánosan elfogadható adatleíró halmaz, amely az egyes személyektől függetlenül, hatékonyan alkalmazható lenne, mivel az egyes módszerek csak részben átfedő adatleíró halmazokat választottak ki a különböző személyekhez.*

3.6. Módosított gyors Fourier transzformáció

Ahogyan a 4. táblázatban látható, néhány adatleíró-kinyerése a frekvenciatartományban történik. Nyilvánvalóan ehhez az első lépés a frekvenciatartományba való áttérés, amit a *gyors Fourier transzformáció (FFT)* segítségével tehetünk meg. A transzformáció hatékony végrehajtásának fontos szerepe van, mivel a tanítási és döntési fázisokban is használni kell.

Az FFT egy hatékony módszere a diszkrét Fourier transzformáció (DFT) kiszámításának, amely a digitális jelfeldolgozás egyik alapja [70]. A DFT képletében (25) $X(k)$ jelöli a frekvencia komponenseket, $x(n)$ az időtartománybeli jel és W_N^{nk} a szorzó vagy más néven a „twiddle" faktor (26).

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{nk} \quad \text{k} = 0,\ldots,\text{N} - 1 \tag{25}$$

$$W_N^{nk} = e^{-i\frac{2\pi}{N}nk} \tag{26}$$

Az FFT felhasználásával a DFT N^2 művelete helyett elegendő $N\log_2(N)$ műveletet elvégezni, ahol N az időtartománybeli jel elemeinek számát jelöli. Az FFT esetében az egyetlen plusz kritérium az, hogy az algoritmus típusától függően N-nek egy hatvány értéknek kell lennie. Az emberi tevékenység

felismerés témakörében ez annyit jelent, hogy az ablak méretét ennek a kikötésnek megfelelően kell megválasztani.

Valójában az FFT algoritmusnak több módosulata létezi, mint például a *radix-2*, *radix-4* és *split-radix* módszerek és ezek szabják meg az időtartománybeli jel méretére vonatkozó korlátozásokat. Az általunk javasolt FFT egy módosított radix-2 algoritmus, amely a hagyományoshoz képest sokkal kevesebb szorzófaktor kiszámítását és tárolását igényli. Mivel a viselhető szenzorok esetében az eszközök számítási és tárolási kapacitása meglehetősen limitált, ezért mindkét erőforrással hatékonyan kell gazdálkodni.

Függetlenül az FFT algoritmus típusától, mindegyik az „oszd meg és uralkodj" elven alapszik, ami annyit jelent, hogy a kiindulási *x(n)* jelet hierarchikusan részekre kell bontani és az egyes részek DFT transzformációja adja majd a végső eredményt. A radix-2 algoritmus szétbontja az időtartománybeli jelet a páros és páratlan komponenseire, ezáltal a DFT a következő módon írható le,

$$\begin{aligned} X(k) &= \sum_{n=0}^{N/2-1} x(2n) W_N^{2nk} + \sum_{n=0}^{N/2-1} x(2n+1) W_N^{(2n+1)k} \\ &= \sum_{n=0}^{N/2-1} x(2n) W_{N/2}^{nk} + W_N^k \sum_{n=0}^{N/2-1} x(2n+1) W_{N/2}^{nk} \end{aligned} \tag{27}$$

A fenti képletben W_N^k az 1 komplex gyökei, amelyek a komplex egységkörön helyezkednek el. Amennyiben kihasználjuk a komplex gyökök közötti első összefüggést (28), akkor a (27) egyenlet egy egyszerűbb formában is kifejezhető (29), ahol *n = 0, 1, 2, ... N/2-1*.

$$W_N^k = W_N^{k+N/2} \tag{28}$$

$$\begin{aligned} X(k) &= DFT_{even}(n) + W_N^k DFT_{odd}(n) \quad \text{k = 0,...N/2-1} \\ X(k) &= DFT_{even}(n) - W_N^k DFT_{odd}(n) \quad \text{k = N/2,...N-1} \end{aligned} \tag{29}$$

Azokban az esetekben, amikor az időbeli jel elemeinek száma négy hatványa ($N = 4^m$), a radix-4 algoritmus (30) is alkalmazható, ami kissé gyorsabb is a radix-2-es változatnál. A radix-4 és split-radix (a 2-es és a 4-es kombinációja) módszerek már a szorzó faktorok közötti második összefüggést is kihasználják (31) [71].

$$X(k) = \sum_{n=0}^{N/4-1} \left(\sum_{l=0}^{3} x(n + l\frac{N}{4}) W_N^{N/4lk} \right) W_N^{nk}$$
$$= \sum_{n=0}^{N/4-1} \left(\sum_{l=0}^{3} x(n + l\frac{N}{4}) (-i)^{lk} \right) W_N^{nk} \quad (30)$$

$$W_N^{k+N/4} = -iW_N^k \quad (31)$$

A (31) összefüggés valójában megadható a komplex számok valós és képzetes részei közötti kapcsolatokkal is (32), ahol *Re* és *Im* a valós és képzetes részekre utal.

$$W_{N_{\mathrm{Im}}}^{k+N/4} = -W_{N_{\mathrm{Re}}}^{k}$$
$$W_{N_{\mathrm{Re}}}^{k+N/4} = W_{N_{\mathrm{Im}}}^{k} \quad (32)$$

A szorzótényezők közötti első két kapcsolat felhasználásával elegendő, mindössze az első N/4 szorzótényezőt kiszámítani, mivel a többi ezekből származtatható.

Az imént említett összefüggéseken kívül vagy egy harmadik is, amelyet sem a radix-2, sem a radix-4 módszer nem használ fel és közel sem annyira ismert, mint az előző kettő. Ez az összefüggés a következő képlettel írható le, ahol $N \geq 16$, $0 \leq k \leq N/4$ és $k \neq N/4 - k$.

$$W_{N_{\mathrm{Im}}}^{k} = -W_{N_{\mathrm{Re}}}^{N/4-k}$$
$$W_{N_{\mathrm{Re}}}^{k} = -W_{N_{\mathrm{Im}}}^{N/4-k} \quad (33)$$

A harmadik összefüggés bemutatásához vegyünk egy egyszerű példát, ahol N = 16. Ekkor az első 4 szorzótényező a következő lesz (ahol felhasználtuk az Euler formulát (34) is):

$$\mathrm{e}^{ix} = \cos(x) + i\sin(x) \quad (34)$$

$$e^{-i2\pi 0/16} = \cos(1) - i\sin(1)$$
$$e^{-i2\pi 1/16} = \cos(\frac{\pi}{8}) - i\sin(\frac{\pi}{8})$$
$$e^{-i2\pi 2/16} = \cos(\frac{\pi}{4}) - i\sin(\frac{\pi}{4}) \quad (35)$$
$$e^{-i2\pi 3/16} = \cos(\frac{3\pi}{8}) - i\sin(\frac{3\pi}{8})$$

Ha megvizsgáljuk a fenti szorzótényezőket, akkor a második és harmadik komponensek között további összefüggést fedezhetünk fel,

$$\begin{aligned}\cos(\frac{\pi}{8}) &= \sin(\frac{3\pi}{8}) \\ \cos(\frac{3\pi}{8}) &= \sin(\frac{\pi}{8})\end{aligned} \tag{36}$$

Ez a szinusz és koszinusz trigonometrikus függvények tulajdonságaiból adódik (37, 38), amik azt mondják, hogy a két függvény között π/2 eltolás van.

$$\cos(\theta) = \sin(\frac{\pi}{2} - \theta) \tag{37}$$

$$\sin(\theta) = \cos(\frac{\pi}{2} - \theta) \tag{38}$$

Az így kapott összefüggés természetesen nagyszámú N esetén is megfigyelhető. Általánosan ezt a következő képletekkel írhatjuk le,

$$\cos(\frac{\pi}{2^{n-1}}) = \sin(\frac{(2^{n-2}-1)\pi}{2^{n-1}}) \tag{39}$$

$$\cos(\frac{(2^{n-2}-1)\pi}{2^{n-1}}) = \sin(\frac{\pi}{2^{n-1}}) \tag{40}$$

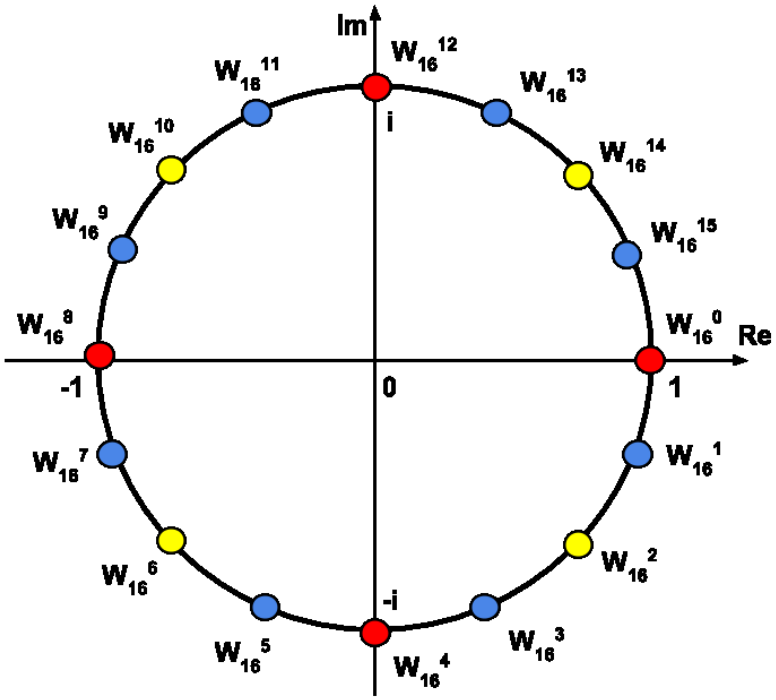
A (39) képlet bizonyítása,

$$\begin{aligned}&\sin(\frac{(2^{n-2}-1)\pi}{2^{n-1}}) = \sin(\frac{2^{n-2}\pi - \pi}{2^{n-1}}) = \\ &\cos(\frac{\pi}{2} - (\frac{2^{n-2}\pi - \pi}{2^{n-1}})) = \cos(\frac{\pi}{2} - \frac{2^{n-2}\pi}{2^{n-1}} + \frac{\pi}{2^{n-1}}) \\ &= \cos(\frac{2^{n-2}\pi}{2^{n-2}} - \frac{2^{n-2}\pi}{2^{n-1}} + \frac{\pi}{2^{n-1}}) = \cos(\frac{\pi}{2^{n-1}})\end{aligned} \tag{41}$$

Az előző alapján a (40) is hasonlóan bizonyítható,

$$\begin{aligned}&\sin(\frac{\pi}{2^{n-1}}) = \cos(\frac{\pi}{2} - \frac{\pi}{2^{n-1}}) = \\ &\cos(\frac{2^{n-2}\pi}{2^{n-1}} - \frac{\pi}{2^{n-1}}) = \cos(\frac{2^{n-2}\pi - \pi}{2^{n-1}}) \\ &= \cos(\frac{(2^{n-2}-1)\pi}{2^{n-1}})\end{aligned} \tag{42}$$

Ha a példánkban mindhárom összefüggést felhasználjuk, akkor az egyes szorzófaktorok közötti kapcsolatokat a 11. ábrával illusztrálhatjuk, ahol az azonos színnel jelölt pontok állnak egymással kapcsolatban.



11. ábra. Összefüggések az egyes szorzófaktorok között

Ez azt is jól szemlélteti, hogy a kapcsolatok felhasználásával elegendő összesen az első N/8 + 1 szorzófaktor kiszámítása. A [J3] cikkünkben ismertettük a fent bemutatott három összefüggést a szorzófaktorok között és itt világítottunk rá arra, hogy ezek egyszerűen beépíthetőek egy radix-2 algoritmusba.

Tehát az általunk javasolt módosítás azt jelenti, hogy kevesebb konstans szorzófaktor kiszámítására és tárolására van szükség az FFT futtatása előtt, mivel ezek egymásból származtathatók. Viszont a transzformáció futtatásakor, a műveletek száma ez által nem csökken. A fent leírt összefüggések kihasználásához egy egyszerű függvényt dolgoztunk ki, amit beépítettünk egy radix-2-es algoritmusba. A módosított algoritmus pszeudokódját a 12. ábrán tekinthető meg.

Algorithm 1 Modified FFT

Require: $x[N]$
Ensure: $X[N]$
  if $is2N(N)$ **then**
    $bitRevSort(x)x$
    for $s \leftarrow 1$ to $stages$ **do**
      $d2 \leftarrow 1 << s$
      $d1 \leftarrow d2 >> 1$
      for $dfts \leftarrow 0$ to $(N >> s) - 1$ **do**
        for $k \leftarrow 0$ to $N/(N >> (s-1))$ **do**
          $a_i \leftarrow k + dfts * d2$
          $b_i \leftarrow a_i + d1$
          $w_i \leftarrow k * (N >> s)$
          $Wi \leftarrow getWi(w_i)$
          $fftButterfly(X[a_i], X[b_i], Wi)$
        end for
      end for
    end for
  else
    return $size\_error$
  end if

12. ábra. A módosított radix-2 FFT pszeudokódja

Az algoritmusban az *is2n(N)* függvény igazat ad vissza, ha N kettő hatványa, különben hamisat. A *bitRevSort(x)* eljárás a bit fordított rendezést végzi el az időtartománybeli jelen – decimálás időben. Az algoritmus külső ciklusa a számítási folyamat szintjeit járja be, ahol a szintek száma log_2N. A középső ciklus az adott szinten elvégzendő DFT-k számát írja le, míg a belső ciklus a DFT-n belül elvégzendő *pillangó* számításokat adja meg. A középső ciklusban elhelyezett *d1* és *d2* paraméterek a DFT-k pillangó számításaiban résztvevő két komponens indexének a meghatározásához szükséges. A belső ciklusban a_i és b_i a pillangó művelet két tagjának indexei, míg w_i az aktuális szorzófaktor indexe, ami ha N/8 + 1-nél nagyobb, akkor a fent leírt összefüggések alapján lesz származtatva a *getWi(*w_i*)* függvényben. Végül az *fftButterfy(X[ai], X[bi], Wi)* függvény végzi el a jól ismert pillangó műveletet:

$$\begin{aligned} X[a_i] &= X[a_i] + Wi * X[b_i] \\ X[b_i] &= X[a_i] - Wi * X[b_i] \end{aligned} \tag{43}$$

Az algoritmus leírásából kiderül, hogy a hagyományos radix-2 számításhoz képest, a módosítást az általunk létrehozott *getWi()* függvény jelenti, ami a szorzófaktorok származtatását végzi a (32) és (33)-as képletek felhasználásával. Ennek a függvénynek a pszeudokódját a 13. ábrán tekinthetjük meg. A kódban feltételezzük azt, hogy a *W[]* vektor már tartalmazza az időtartománybeli jel mérete alapján meghatározott N/8 + 1 szorzófaktort.

Algorithm 1 getWi

Require: w_i
Ensure: Wi
  if $w_i \geq N >> 2$ **then**
    $w_i \leftarrow w_i - N/4$
    if $w_i < (N >> 3) + 1$ **then**
      $Wi_{Re} \leftarrow W[w_i]_{Im}$
      $Wi_{Im} \leftarrow -W[w_i]_{Re}$
    else
      $w_i \leftarrow (N >> 2) - w_i$
      $Wi_{Re} \leftarrow -W[w_i]_{Re}$
      $Wi_{Im} \leftarrow W[w_i]_{Im}$
    end if
  else
    if $w_i < (N >> 3) + 1$ **then**
      $Wi \leftarrow W[w_i]$
    else
      $w_i \leftarrow (N >> 2) - w_i$
      $Wi_{Re} \leftarrow -W[w_i]_{Im}$
      $Wi_{Im} \leftarrow -W[w_i]_{Re}$
    end if
  end if

13. ábra. A szorzófaktorok meghatározását végző függvény pszeudokódja

A leírt algoritmus helyben számol, tehát ugyan azt az eredmény vektort használja a számítás során (minden szintjén), ami majd a végső eredményt is tartalmazza.

Valójában, ennek a módosításnak elsődlegesen azokban az alkalmazásokban lehet jelentősége, ahol az időtartománybeli jel elemeinek a száma nagy, mivel ekkor a szorzófaktorok letárolásához szükséges memória

igény a töredékére csökkenthető. A fent leírt észrevétel adja a dolgozat 3. tézispontját:

3. tézis: *Létrehoztunk egy kiegészítő függvényt a radix-2-es gyors Fourier transzformáció algoritmushoz, aminek segítségével kihasználható a szorzófaktorok közötti mindhárom összefüggés. Ennek felhasználásával elegendő az első N/8 + 1 szorzófaktort kiszámítani és letárolni a transzformáció futtatása előtt, mert a többi szorzófaktor ebből származtatható. Ez gyakorlatilag azt jelenti, hogy kevesebb memóriaterületet kell lefoglalnunk az előre kiszámított szorzófaktoroknak.*

4. Sekély neurális hálózatok hatékonyság elemzése

A tevékenység felismerés témakörében a neurális hálókkal kapcsolatos bizonytalanság, amit a 2. fejezetben ismertettem, arra motivált minket, hogy egy átfogó tanulmány keretében megvizsgáljuk a különböző neurális hálózat alapú stratégiák hatékonyságát a mi problémakörünkön. A következő fejezetekben leírt eredményeket az [J5] cikkben publikáltuk. A tanulmányban a mérési környezetet és az ott használt szoftvereket is én készítettem el témavezetőm útmutatása alapján. Ebben és az ezt követő fejezetekben is, az összes mérést ugyan azon a gépen futtattuk, ami i5-2.3GHz processzorral és 8GB memóriával rendelkezett.

4.1. Sekély neurális hálózatok tervezése és tesztelése

A neurális hálók szakirodalmában számos hasznos tanács és útmutatás található, amire a háló tervezője támaszkodhat, azonban a legtöbb esetben a háló megtervezése probléma és erőforrás függő. Annak ellenére, hogy a sekély hálók állítható paramétereinek száma, a mély módszerekéhez képest kevesebb, mégis több, eltérő hálózat konstrukció alakítható ki. Ebből fakadóan a vizsgálat első lépésében az eltérő sekély neurális hálózati architektúrák hatékonyság elemzését végeztük el.

A témakör aktuális állása alapján tipikusan három költség (vagy hiba) függvényt használnak eltérő aktivációs függvényekkel a köztes és a kimeneti szinteken [72]. Ennek alapján három különböző neurális háló architektúrát hoztunk létre. Mindhárom esetében egy köztes és egy kimeneti szinttel. Az első esetben (*ANN1*) *négyzetes hibafüggvényt* használtunk (44) tangens (45) és lineáris (46) aktivációs függvényekkel a köztes és a kimeneti szinteken.

$$\mathrm{C}_1 = \sum_{j}^{\mathrm{M}} \left(y_j - a_j^L \right)^2 + \frac{\lambda}{2\mathrm{N}} \sum_{\omega} \omega^2 \tag{44}$$

$$\sigma_1(\eta) = \frac{\mathrm{e}^{\eta} - \mathrm{e}^{-\eta}}{\mathrm{e}^{\eta} + \mathrm{e}^{-\eta}} \tag{45}$$

$$\sigma_2(\eta) = \eta \tag{46}$$

A második hálózat (*ANN2*) költségfüggvénye *cross-entropy* (47), tangens és szigmoid (48) aktivációs függvényekkel az egyes rétegeken.

$$C_2 = -\sum_j^M \left[y_j \ln a_j^L + (1 - y_j) \ln \left(1 - a_j^L\right) \right] + \frac{\lambda}{2N} \sum_\omega \omega^2 \tag{47}$$

$$\sigma_3(\eta) = \frac{1}{1 + e^{-\eta}} \tag{48}$$

Végül a harmadik háló esetében (*ANN3*) a hibafüggvény *log-likelihood* (49) volt tangens és soft-max (50) aktivációs függvényekkel.

$$C_3 = -\ln a_y^L + \frac{\lambda}{2N} \sum_\omega \omega^2 \tag{49}$$

$$\sigma_4(\eta_i) = \frac{e^{\eta_i}}{\sum_j^M e^{\eta_j}} \tag{50}$$

Mindhárom költségfüggvény (44), (47), (49) *L2* normalizációt tartalmaz, ahol M a kimeneti neuronok száma (ami megegyezik az osztályok számával), N a minták száma, y_j a valós kimeneti érték vagy osztály (attól függően, hogy regresszióra vagy osztályozásra használjuk a hálót), a_j a kimeneti neuron aktivációs szintje, míg az ω és λ a súlyokra és a súlyok normalizációs erősségére utalnak. Az aktivációs függvényekben η a neuron összegzett bemenete, ami az (51) képlet alapján számítható, ahol b^m az adott neuron eltolási értékére, ω^m az aktuális szint (m.) súlyaira, míg S^{m-1} és α^{m-1} az előző szint neuronjainak a számára és azok kimeneti értékeire utal.

$$\eta_i^m = \sum_{j=1}^{S^{m-1}} \omega_{i,j}^m \alpha_j^{m-1} + b_i^m \tag{51}$$

Meg kell említenem, hogy míg a tangens, szigmoid és lineáris aktivációs függvényeknél a kimenet csak a neuron összegzett bemenetétől függ (η), addig a soft-max függvény kimenete az adott szinten elhelyezkedő összes többi neuron kimenetétől is.

Az aktivációs és költség függvényeken felül számos más paramétere is létezik a neurális hálóknak, ami a hálók tervezése során az egyik fő hibaforrás lehet. A legtöbb esetben a hálók úgynevezett hiper-paramétereinek a helyes beállítása időigényes folyamat, ami teszteket igényel. A hatékony paraméterek megtalálásához mi véletlen keresést használtunk, mivel a [73]

cikk szerzői megmutatták, hogy a véletlen keresés hatékonyabb a hálós (*grid*) keresésnél. Azonban a véletlen keresés kiterjesztése a háló összes paraméterére egy meglehetősen lassú keresési folyamatot eredményezne. Éppen ezért a gyakorlatban csak a főbb paraméterek keresése történik ezen vagy ehhez hasonló elven, míg a többi paraméter esetén „bevált" konstans értékeket használnak.

A háló főbb paraméterei a kezdeti tanulási arány (α_0), a tanulási arányt csökkentő faktor (φ), a normalizálás erőssége (λ) és a köztes rétegen vagy rétegeken elhelyezett neuronok száma. Ezek közül mi, munkánk során csak az első háromra alkalmaztunk véletlen keresést, mivel a 3.2.3-as fejezetben bemutatott eredményeink és későbbi tapasztalataink is azt mutatták, hogy néhány tíz neuron a köztes rétegen már elegendő. Ebből kiindulva mindhárom hálózati architektúra esetén, fixen 40 neuront használtunk.

A három főbb paraméter α_0, φ, és a λ keresése tízes alapú exponenciális skálán történt, ahol az exponens egyenletes eloszlásból származik, az (52) képletnek megfelelően.

$$\alpha_0, \varphi, \lambda \in 10^{U(-4,-1)} \tag{52}$$

A további paraméterek tekintetében az általunk definiált mindhárom hálózati architektúra megegyezett. Tehát az *ANN1*, *ANN2* és *ANN3* hálók a következő közös paraméterekkel és tulajdonságokkal rendelkeztek:

- Két szint (egy köztes és egy kimeneti)
- Köztes neuronok száma: 40
- Tanító algoritmus: momentummal ellátott *gradient descent*
- Mini tanító halmaz méret: 10 minta
- Momentum: 0.15
- Tanítási ciklusok maximális száma: 1000
- Megállási feltétel: nincs javulás az utolsó 10 ciklusban
- Eltolás kezdeti értéke: 0
- Kezdeti súlyok: véletlenszerűen választott az (53) normális eloszlásból, ahol **W** a súly mátrixot, míg κ a neuronok bemeneteinek a számát adja meg az l. szinten
- Tanulás csökkenés: exponenciálisan az (54) képlet alapján, ahol ε a tanítási ciklus számlálójának értéke

$$\mathbf{W}^l \in \mathcal{N}\left(0, \frac{1}{\sqrt{\kappa}}\right) \tag{53}$$

$$\alpha = \alpha_0 \mathrm{e}^{-\varphi\varepsilon} \tag{54}$$

Szem előtt tartva a végső célt - a valós idejű tevékenység felismerést, ahol az offline kapott eredményeket átültetjük a gyakorlatba - fontos kihangsúlyoznom, hogy mindhárom neurális háló egy saját fejlesztésű gépi-tanuló könyvtár felhasználásával lett létrehozva. A teljes könyvtár Java programozási nyelven van megírva, viszont egy tetemes része már elkészült C++-ban is.

A neurális hálózatok tanítását végző algoritmus kulcsa a *back-propagation* folyamatban rejlik. Szerencsére találtam egy átfogó és részletes leírást adó könyvet a neurális hálózatok tervezéséről és a „tanuló motort" ennek alapján készítettem el [74]. A programban a neurális háló struktúráját, azon belül a köztes és kimeneti szinteket külön-külön osztályokként kezelem. Ez lehetővé teszi, hogy rugalmasan alakíthassam a hálózat struktúráját tetszőleges számú köztes szint létrehozásával.

4.2. Az eltérő neurális háló architektúrák hatékonyság vizsgálata

4.2.1. Hatékonyság vizsgálat négy széles körben használt adatleíróval

Első lépésként a három neurális háló architektúrát kizárólag normalizált nyers adattal akartuk tesztelni. Viszont 100 kísérlet után az *ANN1* háló és a WARD adatbázis használatával a 3.3 fejezetben leírt eredményt kaptuk és világossá vált, hogy a normalizált nyers adat nem biztosít elég információt a hatékony osztályozáshoz. Ezt követően a 4 leggyakoribb idő és frekvenciatartománybeli adatleírók használatával (az adatgyűjtők összes tengelyéről) vizsgáltuk a hálókat, a WARD és a HAPT adatbázisokon egyaránt. Ez a 4 adatleíró a 2. táblázatban is megtalálgató [4], [15] [16], [35]. Név szerint:

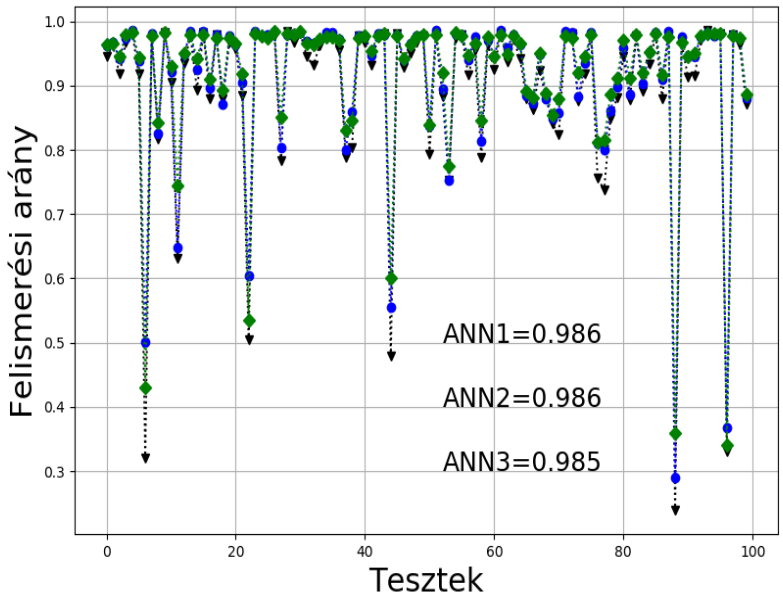
- várható érték
- szórásnégyzet
- alap frekvencia
- spektrum energia

Fontos megemlíteni, hogy az így kapott adatleírók is átestek a (3) képlettel megadott normalizáláson, hogy skálafüggetlenek legyenek.

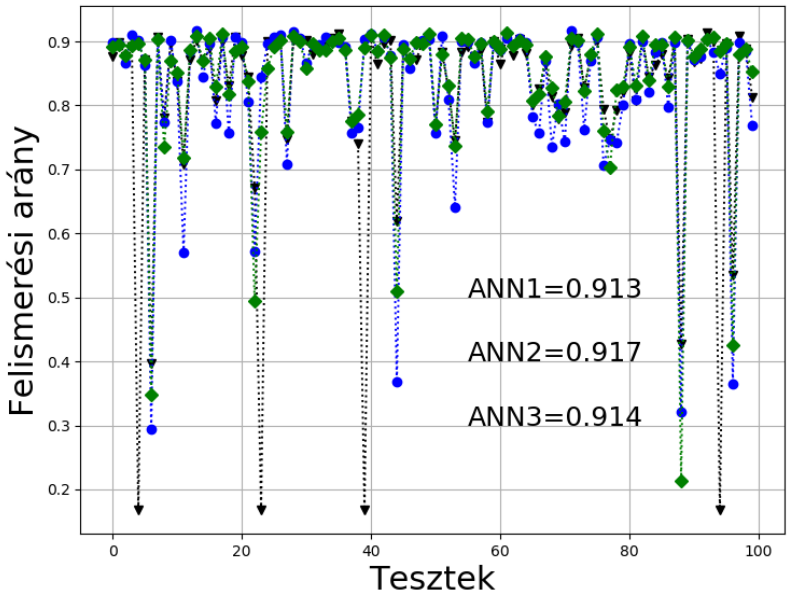
Annak érdekében, hogy a 3 háló azonos paraméterekkel legyen tesztelve, mindhárom esetben az álvéletlenszám generátor magja azonos értékre volt beállítva. Az így kapott felismerési arányok a WARD és HAPT adatbázisokon a 14. és 15. ábrákon látható, ahol a fekete görbe az *ANN1*, a kék az *ANN2* míg a zöld az *ANN3* architektúra felismerési arányait mutatja 100 kísérlet után. Amint láthatjuk, a legmagasabb felismerési aránya a három hálónak 98.6%, 98.6% és 98.5% volt a WARD adatbázison, valamint 91.3%, 91.7%, és 91.4% a HAPT adatbázison. Ha visszapillantunk a 2. táblázatra, akkor láthatjuk, hogy ezek az eredmények a WARD adatbázis esetén már hasonlóan jók a mások által mért korábbi eredményekhez, viszont a HAPT adatbázison nem érik el a korábban mért legjobb eredményt.

4.2.2. Hatékonyság vizsgálat kibővített adatleíró halmazzal

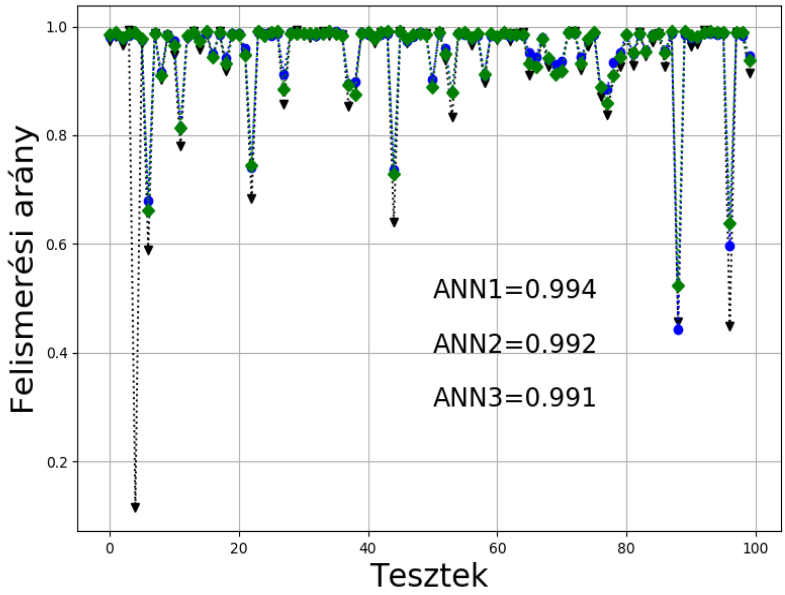
A következő lépésben a 4. táblázatba összegyűjtött összes adatleírót kiszámítottuk a WARD adatbázis szegmentálása során, valamint a HATP adatbázis esetén az adatbázis által nyújtott összes adatleírót felhasználtuk. Fontos megemlítenem, hogy a HAPT adatbázis készítői által biztosított adatleírók és a 4. táblázatba összegyűjtött adatleírók között minimális eltérés van. Mivel néhány adatleíró információt von össze az egyes tengelyekről (például: dőlésszög és a jel nagyságterület) ezért a WARD adatbázis esetén (5 darab 5 tengelyes szenzor) az adatleírók kiszámítását követően egy ablaknyi szélességű időintervallumhoz tartozó adathalmazokat (a nyers adathalmazokat, amelyek az 5 szenzor egyes tengelyein közel azonos időintervallumban volt mérve) egy 385 elemű adatleíró vektor reprezentált és ez adta a hálók bemenetét. Az adatleírók számának kibővítése után, az így kapott felismerési arányok a 16. és 17. ábrákon látható, ahol a görbék színei az előző ábráknál leírt hálókra utalnak. A tesztek során azt is mértük, hogy az aktuális paraméter kombinációval mennyi időt vesz igénybe az egyes neurális hálók tanítása, mivel a felismerési arány mellett a tanuló algoritmus tanítási és döntési ideje is fontos szerepet tölt be. A három háló tanítási ideje a 18. és 19. ábrákon tekinthető meg, ahol az eltelt idő másodpercben értendő. A színek jelentése ugyan az, mint az előző ábrák esetében. Továbbá, mindkét ábrán feltüntettük az egyes architektúrák leghosszabb tanítási idejét is.



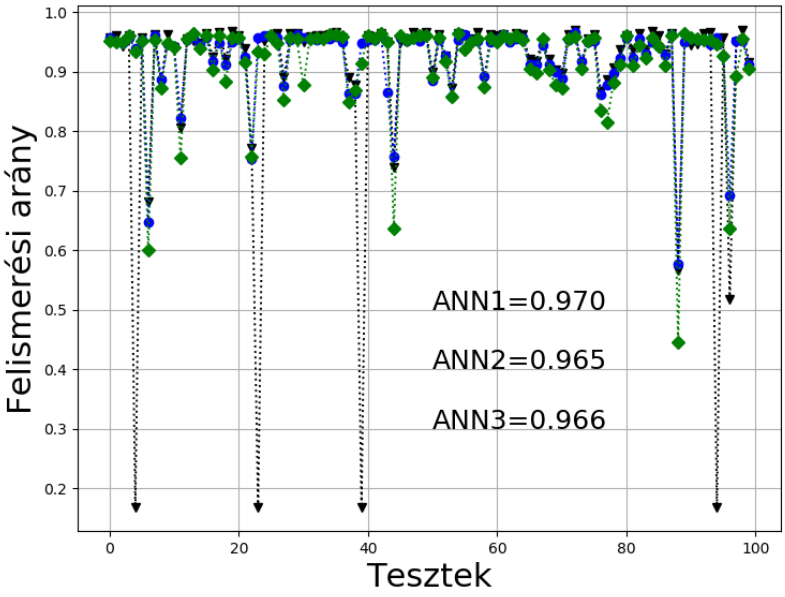
14. ábra. A három különböző neurális háló felismerési arányai 4 adatleíróval a WARD adatbázison



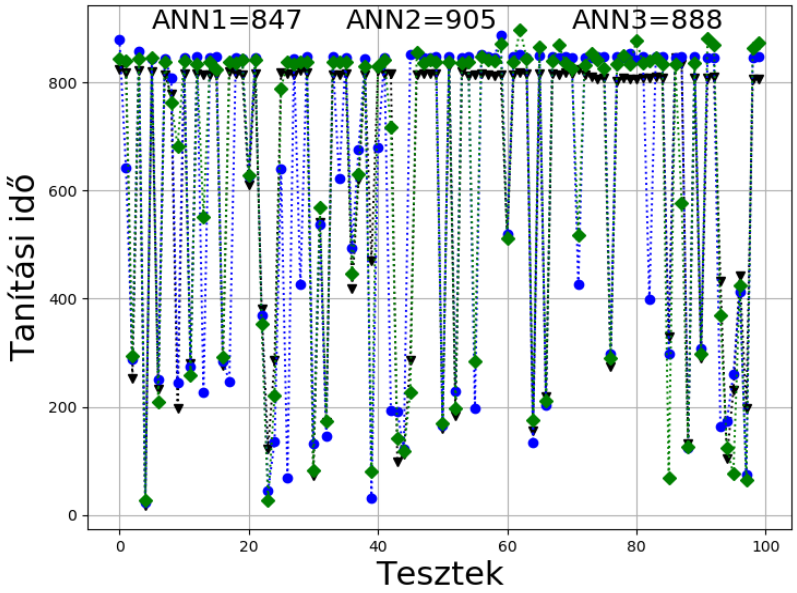
15. ábra. A három különböző neurális háló felismerési arányai 4 adatleíróval a HAPT adatbázison



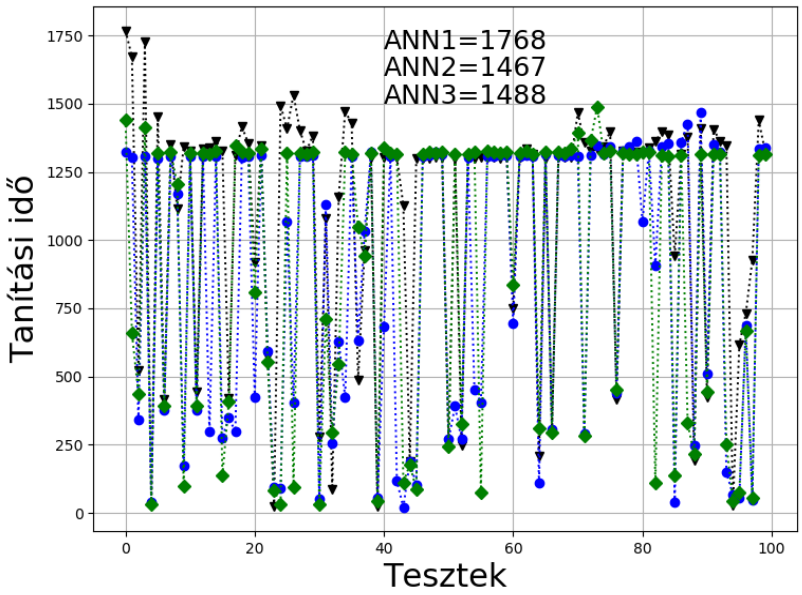
16. ábra. A három különböző neurális háló felismerési arányai a kibővített adatleíró halmazzal a WARD adatbázison



17. ábra. A három különböző neurális háló felismerési arányai a kibővített adatleíró halmazzal a HAPT adatbázison



18. ábra. A három különböző neurális háló tanítási ideje a WARD adatbázis esetén.



19. ábra. A három különböző neurális háló tanítási ideje a HAPT adatbázis esetén.

Ha összehasonlítjuk a kapott eredményeket a 14. 15. 16. és 17. ábrákon, akkor jól látható, hogy a kiterjesztett adatleíró halmaz használatával mindhárom neurális háló architektúra felismerési aránya jelentősen növekedett. Ez a javulás a HAPT adatbázison sokkal szembetűnőbb, mivel itt a felismerési arány megközelítőleg 5.5%-kal növekedett, míg a WARD esetében ez kevesebb, mint 1%. Ez az eredmény is alátámasztja a 3.2.1-es fejezetben ismertetett észrevételt, miszerint az adatleírók használata, valamint azok számának növelése látványosabb javulást eredményez azokban az esetekben, amikor az adatgyűjtő eszközök száma kevesebb. Hiszen a WARD adatbázis esetén 5 adatgyűjtő eszközt használtak, míg a HAPT megalkotásához csak egyet.

A 14-19. ábrákon látható mérési eredmények alapján egy fontos megállapítást is levonhatunk a hálózatok architektúrájával kapcsolatban. Mégpedig azt, hogy mindhárom neurális hálózat közel azonos felismerési arányt biztosított és a tanításuk is közel azonos ideig tartott. Ez egyben azt is jelenti, hogy mindhárom architektúra hasonlóan jó választás lehet, mivel számottevően nem különbözik egymástól a hálók teljesítménye. A három neurális hálózat hatékonyságának vizsgálata a paramétereik kimagaslóan fontos szerepére is rávilágított. Ha megnézzük az előző ábrákat, akkor jól látható, hogy mindhárom hálónál az eltérő paraméter kombinációk hatására a felismerési arány is számottevően megváltozott. Például az *ANN1* struktúra a teljes adatleíró halmazzal, 88%-os (99.4% – 11.6%) felismerési arány eltérést mutatott a legjobb és legrosszabb paraméter kombinációval a WARD adatbázison és 80%-os (97% – 16.8%) felismerés eltérést a HAPT-on.

Végül, azt is megfigyeltük, hogy a leghatékonyabb paraméter kombináció mindkét adatbázison eltérő volt. Például az *ANN1* architektúra leghatékonyabb paraméterei a WARD adatbázison: $\alpha_0 = 0.00061$, $\varphi = 0.00002$, $\lambda = 0.00241$, míg a HAPT adatbázison: $\alpha_0 = 0.00047$, $\varphi = 0.00006$, $\lambda = 0.03716$. Ez jól mutatja a kapcsolatot az adathalmaz és a háló paraméterei között.

Ahogy a 16. és 17. ábrákon látható az *ANN1-es* háló a legjobb paraméter kombinációval 99.4% felismerési arányt nyújtott a WARD adatbázison és 97%-ot a HAPT esetében. Ha ezeket az eredményeket ismét összehasonlítjuk a 2. táblázatban találhatóakkal, akkor azt állapíthatjuk meg, hogy 100 teszt után egy sekély neurális hálóval, ahol a háló három fő paramétere (kezdeti

tanulási arány, tanulást csökkentő faktor, súlyok normalizálásának erőssége) véletlenszerűen lett kiválasztva, jobb eredményt értünk el mindkét adatbázison, mint mások. Az itt kapott eredményeinket összegezve, amelyek a [J5]-ös cikkünkből származnak, jöttek létre a 4. és 5. tézispontok:

4. tézis: *A három neurális háló architektúrával végzett méréseink alapján az egyes hálók legjobb felismerési arányai a WARD adatbázison 99.4%, 99.2% és 99.1% volt, míg a HAPT adatbázison 97%, 96.5% és 96.6%. Ennek alapján azt állapítottuk meg, hogy a három neurális háló architektúra teljesítménye csak minimálisan különbözik egymástól. Ezzel szemben a hálók hiper-paramétereinek kombinációja lényegesen befolyásolja a háló teljesítményét, mivel ugyan az a neurális háló egy helyesen és helytelenül kiválasztott paraméter kombinációval több, mint 80%-os felismerési arány eltérést is produkálhat.*

5. tézis: *A két réteggel rendelkező sekély neurális hálókkal végzett kísérleteink során, amikor az általunk összegyűjtött összes adatleírót felhasználtuk a neurális hálózat bemeneteként, a legmagasabb felismerési arányok a WARD és a HAPT adatbázisokon 99.4% és 97% volt. Ezek az eredmények, a WARD esetében 0.9%-kal, míg a HAPT esetében 0.6%-al jobb, mint a korábban mért legmagasabb felismerési arányok a két adatbázison. Ez azt mutatja, hogy a hiper-paraméter kereséssel és az általunk összegyűjtött, különböző idő és frekvenciatérbeli adatleírók alkalmazásával egy kétszintes, sekély neurális háló használata optimális választás emberi tevékenység felismerésre.*

Annak ellenére, hogy az *ANN1* struktúra konvergenciája néhány esetben megakadt, összességében mégis egy kicsivel magasabb felismerést produkált mindkét adatbázison. Ebből adódóan a későbbi vizsgálataink során csak az *ANN1* architektúrát használtuk.

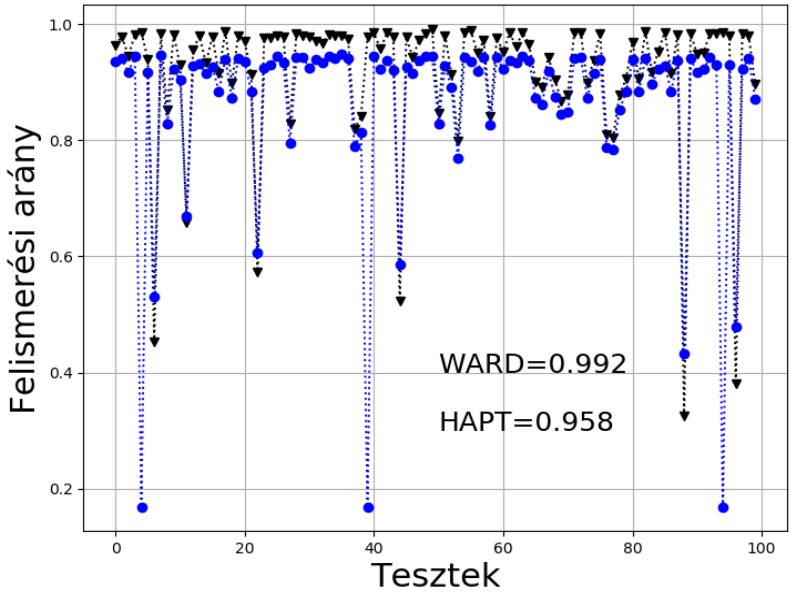
4.2.3. Hatékonyság vizsgálat válogatott adatleírókkal

Annak érdekében, hogy az adatleíró-kiválasztás hatását is megvizsgáljuk, megmértük az *ANN1* háló felismerési arányát a kiválasztás után kapott, csökkentet adatleíró halmazzal. A 3.5 fejezetben bemutatott eredményeink azt mutatták, hogy az általunk létrehozott naiv Bayes keretező típusú módszer egy optimális választás lehet adatleíró kiválasztásra, ezért ebben a

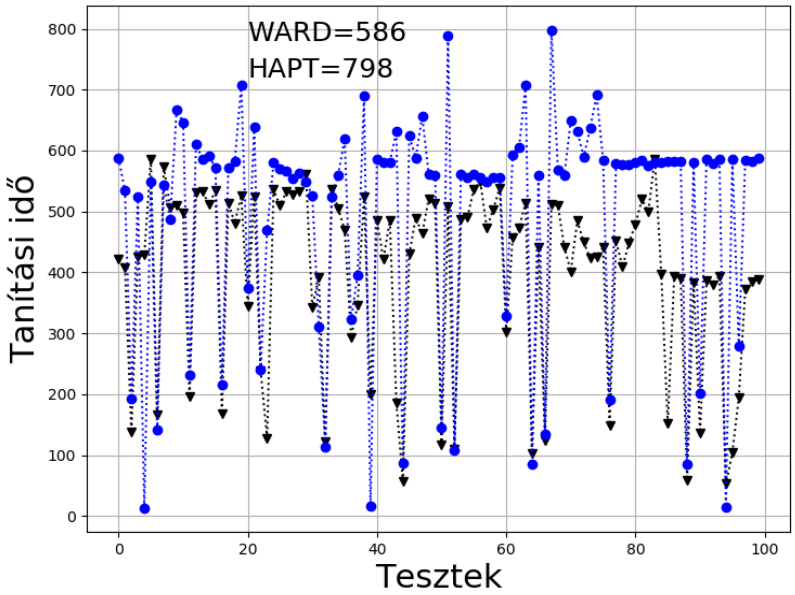
tanulmányban is ezzel a technikával próbáltuk meg a teljes adatleíró halmaz egy, a felismerés szempontjából hatékony részhalmazát meghatározni.

Az átfogó eredmény érdekében mindkét adatbázison lefuttattuk az adatleíró-kiválasztást és az így kapott adathalmaz volt a háló bemenete. A kiválasztást követően a WARD adatbázisból 150 adatleírót, míg a HAPT esetén 200 adatleírót használtunk a teljes adatleíró halmazból. A kiválasztott adatleírók számának meghatározásakor ott húztuk meg a határt, ami után a naiv Bayes módszer hatékonysága már nem növekedett.

A neurális háló felismerési arányai a kiválasztott adatleíró halmazzal a 20. ábrán látható, ahol a fekete görbe a WARD adatbázison mért értékeket mutatja, míg a kék görbe a HAPT adatbázison mért értékekre utal. A háló tanítási ideje változásának a szemléltetése érdekében megmértük a csökkentett adatleíró halmaz okozta, tanulási gyorsulás mértékét is. Ezt a 21. ábrán tekinthetjük meg, ahol az előzőekhez hasonlóan és a dolgozat további ábráin is (ahol releváns) az idő egysége másodpercben értendő.



20. ábra. A neurális háló felismerési aránya a kiválasztott adatleíró halmazokkal.



21. ábra. A neurális háló tanítási ideje a kiválasztott adatleíró halmazokkal.

Ha összehasonlítjuk a 18. 19. és 21. ábrákat, akkor jól látható, hogy a háló tanítási ideje lényegesen csökkent a kiválasztott adatleíró halmazzal. Nyílván az időigénynek fontos szerepe lehet azokban az alkalmazásokban, ahol a hardveres erőforrások meglehetősen korlátozottak, mint például a viselhető szenzoroknál.

Viszont, ha összehasonlítjuk a 20. ábrán látható felismerési arányokat a 16. és 17. ábrákon látható mérési eredményekkel, akkor az is észrevehető, hogy a kiválasztott adatleíró halmazzal a neurális háló teljesítménye kis mértékben csökkent. Például a teljes adatleíró halmazzal a HAPT-on, a legmagasabb felismerési arány 97% volt és ekkor a tanítás 1327 másodpercet vett igénybe, ezzel szemben a kiválogatott adatleíró halmazzal a legjobb eredmény 95.8% volt, viszont ekkor a tanítás csak 619 másodpercig tartott.

Valójában a teljesítmény visszaesés a WARD adatbázis esetén minimális, mivel a teljes és a kiválasztott adatleíró halmazzal elért legmagasabb felismerési arányok között a különbség mindössze 0.2%. Ezzel szemben a HAPT adatbázison ez az eltérés nagyobb, 1.2%-os.

Következésképpen azt állapíthatjuk meg, hogy az adatleíró kiválasztás azokban az alkalmazásokban lehet hasznos, ahol a számítási kapacitás ezt megköveteli.

4.2.4. Kibővített neurális hálózat hatékonyság vizsgálata

A sekély neurális hálózatok vizsgálatának végén, arra kerestük a választ, hogy ha az eddig használt neurális hálót kibővítjük, például egy újabb köztes szinttel, akkor ez hogyan befolyásolja a felismerési arányt és a tanítási időt.

Általában három vagy annál több szint használata nem gyakori, mivel ekkor a súlyok és az eltolások értékeinek a beállítása a hátsó szinteken sokkal lassabb, mint a felső szinteken. Ezt a jelenséget tanulás lelassulásnak vagy más néven „*eltűnő gradiens*” (*vanishing gradient*) problémának is nevezik [75]. Mivel ezekben a hálózatokban a tanulás folyamata lelassul, ezért értelem szerűen a tanítás időigénye is lényegesen növekedni fog.

Ettől a hátránytól eltekintve, néhány gépi-tanulási problémában a sekély neurális háló kiterjesztése számottevő javulást hozott. Egy ide kapcsolódó érdekes tanulmány található az [50] referenciában ahol a szerzők egy hatszintű előrecsatolt neurális hálót hoztak létre, amit a jól ismert MNIST adatbázis kézzel írt számjegyeinek felismerésére használtak. A háló tanítását GPU gyorsítással oldották meg, mivel egy hatalmas méretű hálónál ez igen sok időt venne igénybe egy hagyományos CPU-n. A kibővített háló segítségével a szerzők 0.35% hibaarányt produkáltak, ami az addig elért legjobb eredménynek számított. Ez a hibaarány azért meglepő, mert más kutatók az imént leírt hálónál mélyebb tanuló algoritmusokat is használtak, többnyire konvolúciós neurális hálózatokat, viszont az általuk kapott hibaarány magasabb volt. Ez felveti azt a kérdést is, hogy akkor mi az előnye a konvolúciós hálózatoknak a sekély hálózatok kiterjesztésével szemben?

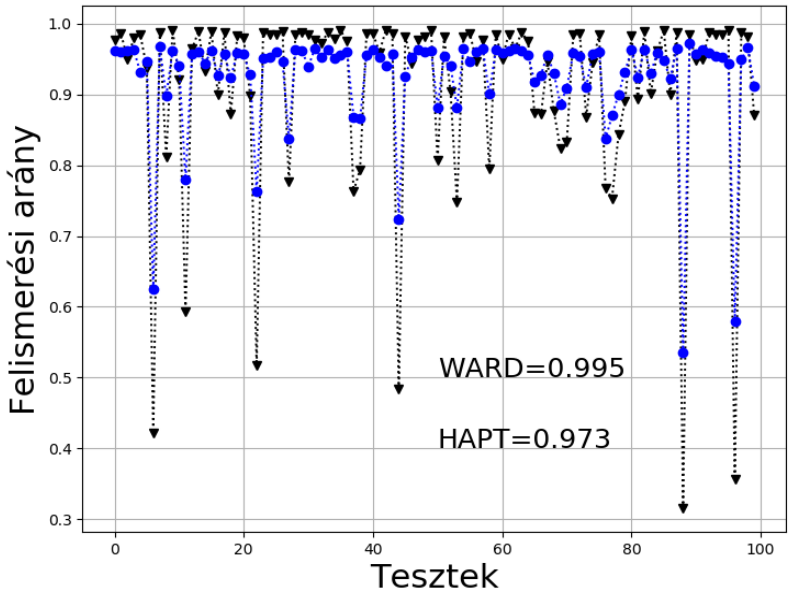
Ezek az eredmények motiváltak minket arra, hogy megvizsgáljuk a háló kibővítésének hatását a felismerési arányon. Az eredeti *ANN1* struktúrát kibővítettük egy újabb köztes szinttel, amely tangens (σ_1) aktivációs függvénnyel rendelkezik, és 20 neuront tartalmaz. A tanulás lassulás miatt a tanítási ciklusok maximális számát 1300-ra növeltük.

Az így létrehozott neurális háló a 22. ábrán látható felismerési arányokat eredményezte 100 kísérletet követően, a teljes adatleíró halmazzal és a korábban alkalmazott véletlen paraméterekkel. Továbbá, a kibővített háló

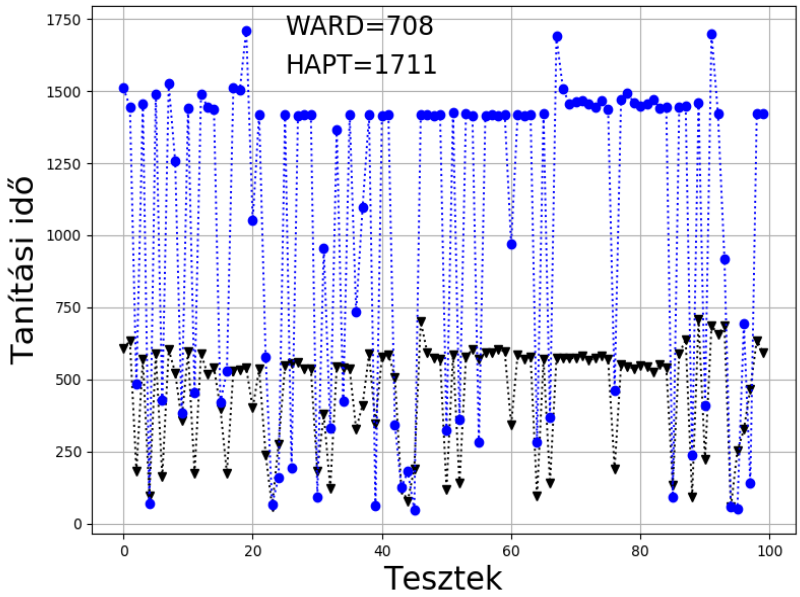
tanítási ideje is megtekinthető a 23. ábrán. A színek jelentése megegyezik az előző ábrákon használttal.

Ahogy az a két ábrán látható, a kibővített háló kis mértékben növelte a felismerést, de egyben a tanítási időt is. Az imént láttuk, hogy az egyetlen köztes réteggel rendelkező neurális háló 99.4%-ot ért el a WARD adatbázison és a tanítása 821 másodpercet vett igénybe, míg a kibővített változattal a legjobb eredmény 99.5% volt, viszont ekkor a tanítási idő is már 974 másodpercre növekedett. Összességében a kibővítés a WARD adatbázison 0.1%-os javulást hozott, míg a HAPT adatbázis esetén 0.3%-ot.

Végül a kibővített háló esetében is jól látszik az egyes paraméter kombinációk hatása a háló teljesítményén. Ebben az esetben a legjobb és legrosszabb felismerési arány közötti különbség 63% (99.5% - 36.8%) a WARD adathalmazán és 44% (97.3% - 53.3%) a HAPT-on. Összességében, mivel a neurális háló kibővítése csak csekély javulást eredményezett, ezért a későbbi munkánk során nem folytattuk a háló bővítés hatékonyság vizsgálatát.



22. ábra. A kibővített neurális háló felismerési arányai.



23. ábra. A kibővített neurális háló tanítási ideje.

5. Összetett és mély neurális hálózatok hatékonyság elemzése

A következő alfejezetek eredményei a [J8]-as cikkünkben találhatóak. Ebben a tanulmányban is a mérési környezetet és az ott használt szoftvereket én készítettem el témavezetőm útmutatása alapján.

5.1. Neurális háló együttesek

A neurális háló együttesek véges számú neurális hálózatból állnak, ahol minden egyes neurális hálózatot ugyan arra az osztályozó feladatra tanítanak be. A neurális hálók a tanításhoz szükséges bemeneti adataikat egy közös adathalmazból kapják. Hasonlóan a korábbiakhoz itt is az a cél, hogy a háló minél jobban közelítse azt az f függvényt $f: \mathbb{R}^m \rightarrow \Gamma$, amely a bemenetén kapott m bemenő adatot leképezi az egyes osztályokat jelölő címkék halmazára (Γ).

Az eredeti elgondolás a tanuló algoritmus együttesek mögött az, hogy az együttest alkotó betanított modellek az adattér eltérő részein fognak általánosítási hibákat ejteni. Ebből kiindulva, ha összességében vizsgáljuk az együttesek döntéseit, akkor a hibás döntések száma csökkenthető az egyedülálló neurális hálóval szemben. Más szavakkal, az együttesek kombinálják az egyes tagjaik hipotéziseit, annak érdekében, hogy egy megbízhatóbb hipotézist származtassanak belőle.

A szakirodalomban már megtalálható számos cikk, ahol gépi-tanuló együtteseket használtak eltérő problémakörökben, mint például a karakter felismerés, pénzügyi előrejelzés, orvosi képfeldolgozás és döntéstámogatás [76-79]. Ezzel szemben az emberi tevékenység felismerésben viszonylag kevés olyan cikk van, ahol együtteseket használnak és ezekben sem neurális hálók alkotják az együttes tagjait. Ennek alapján, mi szerettük volna megvizsgálni olyan együtteseknek a teljesítményét, ahol minden egyes tag egy-egy neurális hálózat.

Általában egy ilyen együttes létrehozása két lépésből áll: a neurális hálók architektúrájának meghatározásából és a döntéseik összekapcsolásából egy adott kritérium alapján.

Valójában az együttesben résztvevő hálózatok betanításának módszere eltérő lehet [80]. Mi a *Bagging* módszert használtuk az egyszerűsége miatt. Ez azt mondja, hogy az eredeti adathalmazból N darab tanítóhalmazt kell létrehozni visszatevéses mintavétellel és az együttes tagjait egy-egy ilyen halmazon kell betanítani. A validációs és tesztfázisokban az egyes tagok döntéseinek ($p_i \in \Gamma$) a kombinációja fogja adni az együttes eredményét.

Munkánk során, mi az egyes döntések alapján az együttes végső döntését a többségi szavazás elve alapján hoztuk meg (*majority voting*), ahol az a döntés lesz a nyerő, amelyik többször szerepel az együttes döntési listájában. Ekkor azonban előfordulhat az is, hogy több osztály is ugyan annyi szavazatot kap. Ebben az esetben a végső döntést az alapján hoztuk meg, hogy melyik neurális háló adott nagyobb aktivációs értéket a kimenetén.

A szakirodalomban megtalálható néhány olyan tanulmány is, ahol az imént bemutatott együttes kialakítás helyett, sokkal összetettebb evolúciós algoritmust használnak erre a célra [80], [81]. Nyilvánvalóan ezek a technikák tovább növelik a tanuló modell bonyolultságát és tanulási idejét, ami a neurális háló tagokkal amúgy is meglehetősen hosszú. Ezért mi a tanulmányunk elkészítéséhez nem használtunk további együttes generáló algoritmusokat.

Összesen négy eltérő tagszámú együttest teszteltünk, amelyeket az együttesben használt neurális hálók száma alapján *E3*, *E4*, *E5* és *E6*-nak neveztünk el. Mivel a korábbi tesztekben az *ANN1* hálózat hatékonynak bizonyult ezért itt is ezt használtuk. Továbbá, itt nem végeztünk egyesével minden neurális hálón paraméter keresést, hanem a kezdeti tanulási arány, a tanulás csökkentő faktor és a súly normalizációnak a legjobb kombinációkat használtuk a korábbi tanulmányból mindkét adatbázishoz. A különböző együttesekkel kapott eredmények, valamint az együttesek tagjainak önálló teljesítménye a 14. táblázatban látható. A táblázatban található értékek három kísérlet átlagértékei.

A táblázatban látható mérési eredmények alapján azt mondhatjuk, hogy az együttes felismerési aránya néhány esetben magasabb, mint az egyes tagjaié, azonban ez nem minden esetben teljesül (erre példa az E5). Tehát azt a következtetést vonhatjuk le, hogy több neurális háló összekapcsolása nem garantálja a teljesítménynövekedést az általunk vizsgált adatbázisokon. Ezt a

jelenséget a [82] cikk szerzői is megfigyelték egy eltérő gépi-tanulási problémában.

14. táblázat. A négy együttes E3, E4, E5 és E6 felismerési arányai

	WARD				**HAPT**			
Tagok	E3	E4	E5	E6	E3	E4	E5	E6
1	0.991	0.993	0.986	0.992	0.961	0.969	0.963	0.967
2	0.990	0.986	0.989	0.993	0.967	0.966	0.967	0.959
3	0.985	0.992	0.992	0.992	0.966	0.966	0.955	0.960
4	-	0.992	0.989	0.990	-	0.965	0.963	0.962
5	-	-	0.987	0.986	-	-	0.964	0.962
6	-	-	-	0.989	-	-	-	0.963
Együttes	*0.993*	*0.996*	*0.991*	*0.995*	*0.968*	*0.969*	*0.966*	*0.965*

Összességében azt mondhatjuk, hogy egy együttes akkor működhet hatékonyan, ha az egyes tagjainak a kimenete között nincs korrelációs kapcsolat. Amennyiben ez nem teljesül, akkor az együttesnek a hatékonysága nem garantált az egyszerű neurális hálóval szemben. Mindezek mellett, az együttes időigénye a tagjai számának függvényében növekszik. Ez azt is jelenti, hogy egy valós idejű alkalmazásban egy együttes használata nem előnyösebb az egyedülálló hálóval szemben.

5.2. Bináris neurális hálózatok

Azok az osztályozási problémák, amelyek számos osztályt foglalnak magukba (ahogy az emberi cselekvés felismerés is) felbonthatóak bináris osztályozási részproblémákra, ahol az osztályozást egy bináris osztályozó algoritmus is elvégezheti. Ekkor az egyes osztályokat jelölő címkék halmaza (Γ) mindössze kételemű.

Ahogy a név is jelzi, ez a módszer az eredeti tanítási halmazt két részre bontja, mintha csak két osztályt kellene elkülöníteni egymástól. Más szavakkal, ez a megközelítés is a jól ismert „oszd meg és uralkodj” stratégiát követi. Elméletileg egy bináris osztályozó döntési határa egyszerűbb, mint a

többosztályos osztályozóké és az osztályozást végző modell megalkotása is egyszerűbb, mivel itt csak két osztályt kell megkülönböztetni egymástól.

A szakirodalomban a leggyakrabban használt bináris osztályozó a *support vector machine*, viszont neurális hálók is alkalmazhatóak erre a célra [83], [84]. Ezen felül, a szakirodalomban különböző módszerek találhatók a bináris osztályok kialakítására, ami egyben a tanítási adathalmaz felosztásának módját is megadja. A két legelterjedtebb módszer a "*one-versus-one*" (*OVO*) és a "*one-versus-all*" (*OVA*). További információ az OVO és az OVA felosztásról a [85] cikkben található.

Rifkin és Klautau munkája alapján [86], az általunk készített tanulmányban kizárólag OVA módszert használtunk, mivel a szerzők megmutatták azt, hogy az OVO-val szemben egyszerűbb OVA módszer is ugyan olyan hatékonysággal alkalmazható, ha az osztályozók megfelelő beállításokkal rendelkeznek.

Valójában az OVA a legegyszerűbb bináris osztály kialakító módszer, ahol az eredeti K osztályú problémát K bináris osztályozási részproblémára tördeljük szét, ahol minden osztályozónak az az egyetlen feladata, hogy a $k.$ osztály elemeit el tudja különíteni a maradék $K - 1$ osztály elemeitől.

Tehát ez a megközelítés azt követeli meg, hogy a bináris osztályozók száma megegyezzen az adott probléma osztályainak a számával, ahol a $k.$ osztályozó tanítási halmazába a $k.$ osztályhoz tartozó elemek pozitív címkével lesznek ellátva (pl.: 1), ezzel szemben negatív (pl.: -1) címkét kap az összes többi minta, amelyik a maradék $K - 1$ osztályba tartozik.

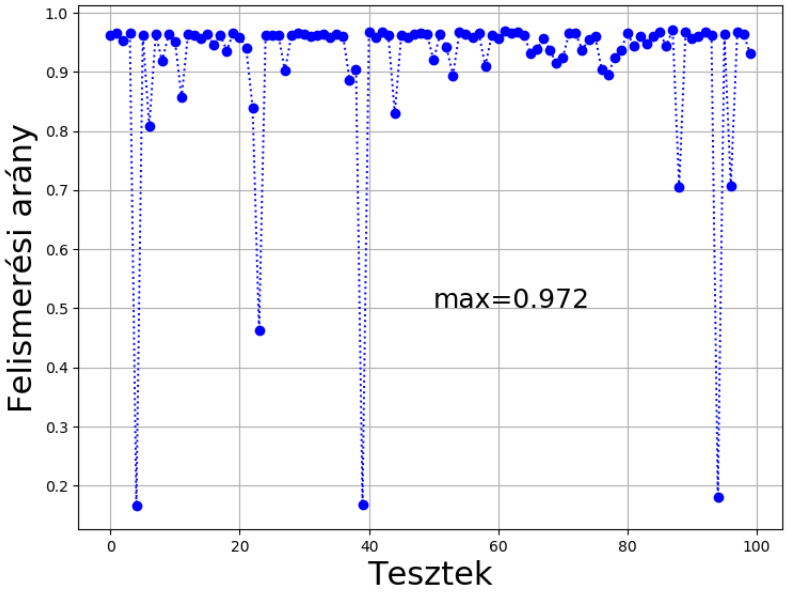
A bináris osztályozók valójában úgy is tekinthetőek, mint egy speciális együttes, ahol minden tag egy osztályt tanul meg a többieket pedig elutasítja. Az együtteseknél bemutatott módon itt is az önálló bináris osztályozók döntései közösen fogják megadni a végleges eredményt. A validációs és teszt fázisokban, hasonlóan a korábbiaknál itt is egy ismeretlen osztályhoz tartozó mintát adunk minden bináris osztályozó bemenetére és az az osztályozó fogja adni a végleges döntést, amelyik pozitív kimenetet produkál. Természetes itt is előfordulhat, hogy több osztályozó is pozitív kimenetet ad. Szerencsére erre több megoldás is létezik, mint például a *maximum konfidencia stratégia (MAX)* vagy a *dinamikusan rendezett OVA* [87].

Mi a *MAX* technikát alkalmaztuk, amely az egyes osztályozók kimenetének a megbízhatóságán alapszik. Ez gyakorlatilag azt jelenti a neurális hálók esetén, hogy a *k.* háló kimenetének aktivációs szintje $c_k \in [-1, 1]$ a *k.* osztály megbízhatóságának tekinthető. Tehát több pozitív aktiváció esetén a végső döntést a legnagyobb aktivációval bíró osztályozó adta, ami a hozzá tartozó osztály címkéje.

A két adatbázisban szereplő, különböző cselekvések alapján, 6 bináris *ANN1* architektúrájú neurális hálót hoztunk létre a HAPT adatbázishoz és 13 hálót a WARD-hoz, azzal a módosítással, hogy most a kimeneti szint csak egy neuront tartalmazott. Az első estben a hálók paraméterei megegyezett a korábban használtakkal. 3 teszt után a felismerési eredmények átlaga a WARD adatbázison 98.9%, míg a HAPT adatbázison 96.54% volt.

Ha összehasonlítjuk ezeket az eredményeket az egyszerű neurális hálóval mért értékekkel, akkor jól látható, hogy ez a felismerési arány nem haladja meg a korábbiakat. Ennek alapján azt feltételeztük, hogy a gyengébb felismerés oka a konstans paraméterek használatából adódik, ezért elvégeztünk néhány további tesztet a HAPT adatbázis használatával, ahol már a bináris hálók paramétereit is véletlen kereséssel határoztuk meg. Az így kapott mérési eredmények a 24. ábrán láthatóak.

A 17. és a 24. ábrák összehasonlítása azt mutatja, hogy a bináris neurális hálók a véletlenszerűen kiválasztott paraméterekkel már egy kicsivel jobb eredményt értek el a HAPT adatbázison, mint az egyedülálló háló. Viszont a növekedés csak 0.2% volt. Viszont az időigény itt is számottevően megnövekedett. Egy egyedülálló hálóval a HAPT adatbázison elvégzett korábbi mérések során, a leghosszabb tanítási idő 1768 másodpercig tartott ezzel szemben ez a bináris hálóknál 4562 másodperc volt. Véleményünk szerint egy valós idejű alkalmazásban a bináris hálók időigénye túl nagy ár a minimális felismerési arány növekedésért.



24. ábra. Bináris neurális hálókkal mért felismerési arányok a HAPT adatbázison.

5.3. Konvolúciós neurális hálózatok

A konvolúciós neurális hálózatok a mély tanulás egy kimagasló része. A konvolúciós hálók felépítéséről és működéséről egy általános bevezető a [88] referenciában található. Igazából ez egy többrétegű, előrecsatolt neurális háló, amely abban különbözik a sekély változattól, hogy új típusú rétegekkel van kiegészítve. Az új rétegek a legtöbb esetben konvolúciós és összevonó (*pooling*) szintek.

Ennek a tanuló algoritmusnak az egyik fő előnye a robosztusság, ami azt jelenti, hogy jól tolerálja a különböző adat módosulásokat, ami származhat külső zajból, skálázásból, képfeldolgozásnál forgatásból és megvilágítás változásból, stb. Ezen felül, ez a tanuló módszer figyelembe veszi a tér vagy időbeli kapcsolatokat is a bemenő adathalmazban.

Konvolúciós hálózatot már 1998-ban LeCun és mások kézzel írott számjegyek felismerésére használt a népszerű MNIST adatbázison [89]. Ezt követően számos kutatócsoport és cég használt konvolúciós hálókat képfeldolgozásra, ezen belül is főként objektum felismerésre. Ebből nőtte ki magát az ImageNet verseny is [19], [20], [90] [91]. Ennek keretében az egyes

kutatócsoportok meglehetősen mély konvolúciós hálókat terveztek név szerint GoogleLeNet, AlexNet, ZF Net és VGGNet, amelyek számos konvolúciós, összegző és teljes összeköttetésű (hagyományos köztes szint) szintből állnak.

Gyakorlatilag egy konvolúciós háló ötvözi a digitális szűrést (55) az előrecsatolt neurális hálóval, ahol a háló bemeneti adata az előző konvolúciós szintek mélységeinek megfelelő számú szűrésen esik át. Egy másik megközelítésben, erre a szűrési folyamatra úgy tekinthetünk, mint egy hierarchikus adatleíró-kinyerésre, ezáltal egy adott konvolúciós szinthez tartozó „lapok” (ami a szint mélységét adja) adatleíró tábláknak tekinthető.

$$y[i] = \sum_{j=0}^{N-1} h[j]x[i-j] \tag{55}$$

A konvolúciós szinteken a szűrők értékeit súlyoknak foghatjuk fel, tehát a tanítási folyamat során a szűrők értékei lesznek beállítva, feltételezhetőleg úgy, hogy bizonyos mintákat felerősítsenek, míg másokat elnyomjanak. Igazából, ha jobban átgondoljuk, akkor ez egy mintakeresési folyamat, amit a digitális jelfeldolgozásban, a legtöbb esetben kereszt korrelációval (1) végeznek (szimmetrikus kernel esetén a korreláció és a konvolúció megegyezik). Ha most visszakanyarodunk a [J1] cikkünkhöz, ahol megmutattuk, hogy a mintafelismerésre a korrelációtól az általunk javasolt, négyzetes hibán alapuló módszer jobb eredményt ad, akkor az is felmerülhet, hogy a hagyományos konvolúciós szintek helyett más, hatékonyabb szinteket is ki lehetne dolgozni. Sajnos ez csak egy felvetés, amit még nem tudtunk kellő alapossággal megvizsgálni, ezért a dolgozatban a hagyományos konvolúciós hálózatokat vizsgáltuk.

Általában a konvolúciós réteget vagy rétegeket egy összevonó réteg követ, aminek az a szerepe, hogy az azt megelőző réteg információtartalmát egy kompakt formába összetömörítse. Ezek a szintek tetszőleges számba csatolhatók egymás után és mindkét szint tetszőleges számú, azonos méretű lapokat tartalmazhat, amelyekhez eltérő szűrők tartoznak. A lapok száma adja a szint mélységét (3. dimenzió). A lapokon az egyes neuronokhoz ugyan az a szűrő és eltolás érték tartozik.

Az iménti alapján az *l*. konvolúciós szint *v*. lapján az *i*, *j*. neuron aktivációs értékét formálisan az (56) képlettel írhatjuk le, ahol a szűrő mérete *NxM*, *V* az előző szint lapjainak a száma, *b* az eltolás értéke és ω a laphoz tartozó szűrő

értékei. Általában a konvolúciós szinteken aktivációs függvényként korrigált lineárist (*rectified linear)* (57) használnak [19], [91].

$$a_{i,j}^{l,v} = \sum_{pv}^{V} \sigma\left(\sum_{n}^{N} \sum_{m}^{M} \omega_{n,m}^{l,v} a_{i+n,j+m}^{l-1,pv} + b^{l,v} \right) \tag{56}$$

$$\sigma_5(\eta) = \max(0, \eta) \tag{57}$$

Az összevonó szinten is több összevonási módszer alkalmazható. A szakirodalomban a 2x2 *max-pooling* a legnépszerűbb, ami egy 2x2 régión belül a maximális értéket választja ki és ez kerül át az összevonó szintre [90]. Végül az utolsó konvolúciós vagy összevonó szint összes lapja, összes neuronjának aktivációja fogja táplálni a hagyományos, többrétegű, előrecsatolt neurális hálót.

A neurális hálózatok hatékonyság vizsgálatának lezárásaként mi is megvizsgáltunk két egyszerű architektúrával rendelkező konvolúciós hálót. Hasonlóan a sekély hálózatoknál, a konvolúciós hálózatokhoz is saját programot készítettem Java-ban. Az új szintek leprogramozását és a tanító algoritmus kibővítését Michael Nielson könyve alapján végeztem [72]. A fent leírtak tükrében mindkét hálózatot a következő közös beállításokkal hoztam létre:

- ReLu aktivációs függvény (σ_5) a konvolúciós szinteken
- 2x2 max-pooling az összevonó szinteken
- soft-max aktivációs függvény (σ_4) cross-entropy hibafüggvénnyel a kimeneti rétegen úgy, mint a [19] és [91] referenciákban.

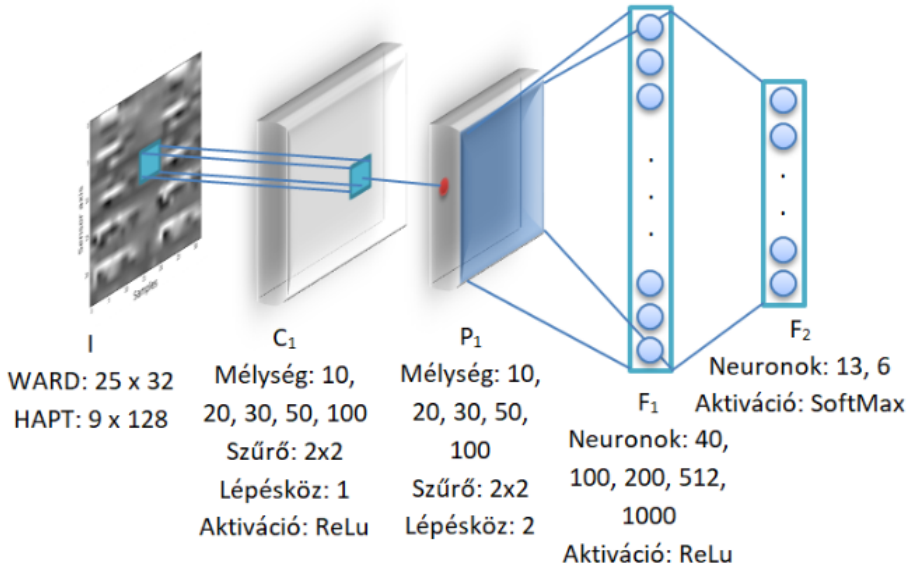
Az általunk használt konvolúciós hálók architektúrájának további beállításait azokra a cikkekre alapoztuk, amelyekben a szerzők konvolúciós hálót használtak cselekvés felismerésre.

Ahogy a 2. fejezetben említettem, a szakirodalomban már fellelhető néhány ilyen cikk (2. táblázat). Például, a [22] cikkben a szerzők két konvolúciós és két összevonó szintet használtak 128 és 256-os mélységgel és két teljes összeköttetésű szintet 512 és 13 neuronnal. A [29] tanulmányban a háló ugyancsak két konvolúciós, két összevonó és két teljes összeköttetésű szintet tartalmazott, ahol a konvolúciós és az összevonó szintek 50 és 40 lapot foglaltak magukba, míg a hagyományos szintek 400 és 18 neuronból álltak. Jiang és Yin [23] különböző struktúrákkal próbálkozott, ahol a leghatékonyabb két konvolúciós és összegző szinttel rendelkezett, amelyek 5

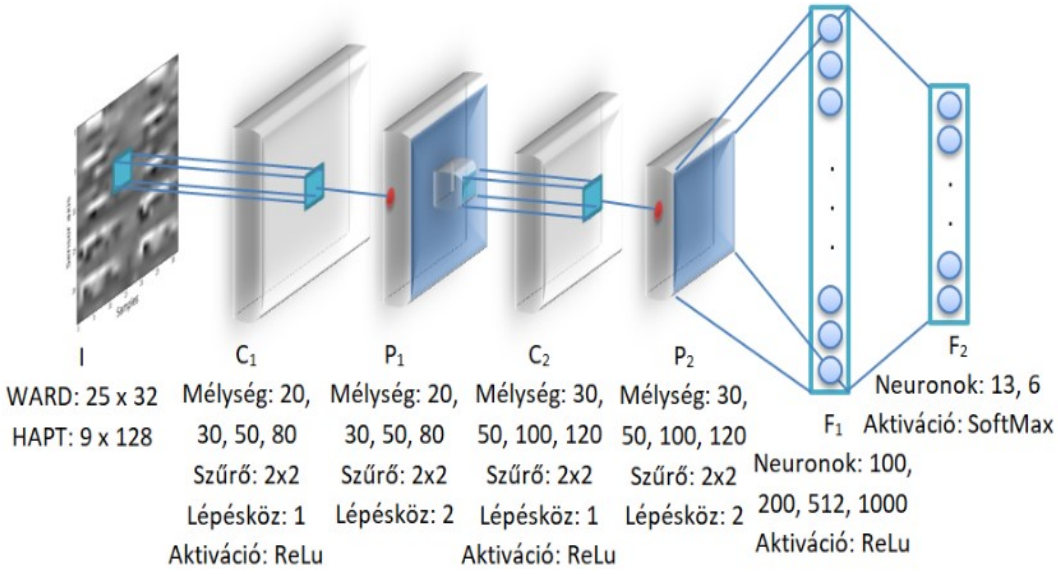
és 10 lap mélységűek voltak és a háló végül két teljes összeköttetésű szinttel zárult. A [26] tanulmány szerzői is különböző konstrukciókkal próbálkozott, ahol a konvolúciós és összegző rétegek száma 1 és 3 között volt 3-tól 9 lap mélységig, viszont a cikkből nem derült ki, hogy melyik összeállítás a leghatékonyabb. A legrészletesebb leírás a konvolúciós háló kialakításáról Ronao és Cho cikkében található [44]. Ellentétben az előző tanulmányokkal, itt a háló bemenete 1 dimenziós volt (szenzorok jelei összefűzve), tehát az első konvolúciós réteg 1 dimenziós konvolúciót végzett. A szerzők eredményei azt mutatták, hogy három konvolúciós szint után a felismerési arány csökken, továbbá arra is rámutattak, hogy egy adott szinten több mint 130 lap felesleges.

A fent bemutatott korábbi eredmények alapján mi két konvolúciós háló struktúrát vizsgáltunk meg (*CNN1* és *CNN2*) különböző mélységekkel és neuronszámmal a hagyományos rétegeken. Az első esetben egy konvolúciós, egy összegző és két teljes összeköttetésű szintet használtunk, míg a második háló két konvolúciós, két összegző és két teljes összeköttetésű szintből állt. A két hálónak a struktúrája a 25. és 26. ábrákon tekinthető meg. Mindkét ábrán feltüntettük, hogy milyen mélységekkel és neuronszámmal végeztünk teszteket. Ahogyan látható, itt a háló bemenete normalizált szenzor adat kétdimenziós formába rendezve, ahol minden sor a szenzorok egyes tengelyeiről származó, egy ablak méretű adathalmaz.

A háló architektúráján felül, a konvolúciós hálók is igénylik a sekély hálóknál bemutatott paramétereket, sőt még azon felül másokat is, mint például a szűrő mérete és az eltolás nagysága. Ebben az esetben, az optimális paraméterek „gyors” megtalálása még egy nyitott kérdés, mivel a korábban használt véletlen keresés túl sok időt vesz igénybe. Ebből kiindulva, mi a korábban is használt paramétereket alkalmaztuk ismét, azzal a módosítással, hogy most a súlynormalizálás értékét megnöveltük ($\lambda = 0.9$). Erre azért volt szükség, mert a konvolúciós hálóknál a normalizációra nagyobb hangsúlyt kell fektetni, mivel a háló növelésével a „túltanítás” (*overfitting*) egy sokkal komolyabb probléma [19]. Végül mind a konvolúciós, mind az összegző rétegeken a szűrő mérete 2x2 volt, 1 és 2 mintányi lépésközzel.



25. ábra. A *CNN1*-el jelölt konvolúciós háló struktúrája, az egyes szintek mélysége és neuronjainak száma, amelyekkel a hálózatot teszteltük



26. ábra. A *CNN2*-vel jelölt konvolúciós háló struktúrája, az egyes szintek mélysége és neuronjainak száma, amelyekkel a hálózatot teszteltük.

Három próbálkozást követően a két konvolúciós háló nyújtotta legjobb eredmények és az átlagos tanítási idejük (másodpercben) a 15. és 16. táblázatokban látható. Mindkét táblázat esetén az architektúra oszlopban található értékek a konvolúciós szintek mélységére és az első teljes összeköttetésű szint neuronjainak a számára utal.

15. táblázat. A *CNN1* háló felismerési arányai és időigénye

Architektúra	**WARD**		**HAPT**	
	Felismerés	**Idő**	**Felismerés**	**Idő**
10-40	0.964	1213	0.846	1615
20-40	0.970	2210	0.853	4071
30-100	0.974	6464	0.864	6972
50-200	0.975	14224	0.883	19985
50-512	0.977	45327	0.898	70671
100-1000	0.977	163026	0.899	286509

16. táblázat. A *CNN2* háló felismerési arányai és időigénye

Architektúra	**WARD**		**HAPT**	
	Felismerés	**Idő**	**Felismerés**	**Idő**
20-30-100	0.956	13010	0.888	16142
20-50-100	0.956	17358	0.895	19394
30-50-200	0.957	24697	0.906	36747
30-50-512	0.972	43183	0.924	62380
50-100-512	0.976	59268	0.928	90921
80-120-1000	0.975	258068	0.942	350677

A fenti táblázatok eredményei azt mutatják, hogy meglepő módon a *CNN1* 50-512 struktúra jobb eredményt ért el, mint a mélyebb *CNN2* a WARD adatbázison. Ezzel szemben a HAPT adatbázison megfigyelhetjük, hogy a struktúra mélyítésével a felismerés is növekedett, ezért itt a *CNN2* 80-120-1000 érte el a legjobb eredményt. Az így kapott legmagasabb felismerési

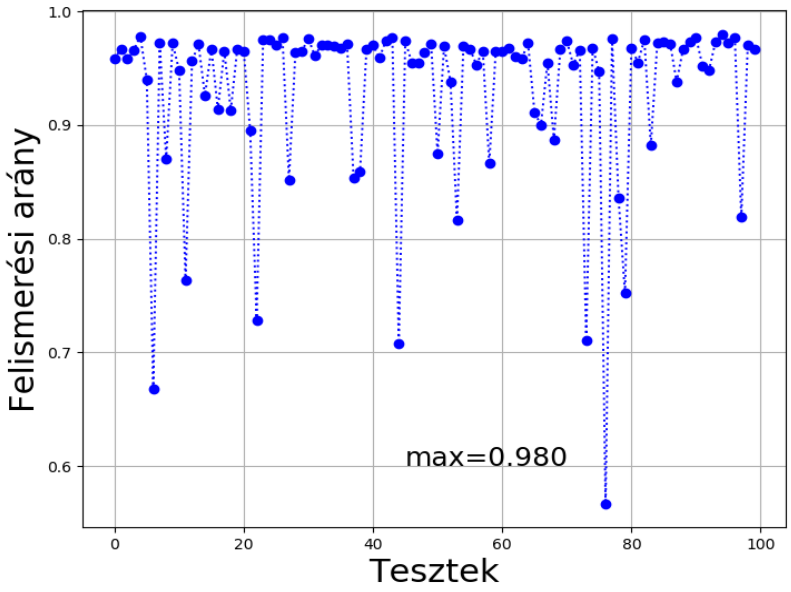
arányok a WARD és a HAPT adatbázisokon 97.7% és 94.2%. Viszont ezek az eredmények nem érik el az egyszerű sekély neurális hálóval mért értékeket, ráadásul a mély konvolúciós hálók tanítási ideje hatalmas. A legmélyebb *CNN2* struktúra betanítása több mint 350000 másodpercig tartott. Ez azt is jelenti, hogy a mély hálók hatalmas időigénye ellehetetleníti a használatukat a kis számítási kapacitással rendelkező eszközön. A tanulmány teljessége érdekében, a korábban használt konstans paraméterek helyett, itt is alkalmaztuk a véletlen paraméterkeresést a *CNN1* 50-512 struktúrán ahol a WARD adatbázist használtuk adatforrásként. Ennek a vizsgálatnak az eredménye a 27. ábrán látható. Annak ellenére, hogy ez a mérési folyamat közel 290 órát vett igénybe, a felismerési arány kevesebb, mint 0.5%-kal növekedett.

Összegezve az itt kapott eredményeket, azt mondhatjuk, hogy az általunk definiált két konvolúciós háló felhasználásával a WARD adatbázison 97.7%, míg a HATP adatbázison 94.2% volt a legjobb felismerési arány. Ezek az értékek azonban elmaradnak a sekély hálóval mért eredményekhez képest, még akkor is, ha itt is alkalmazzuk a részleges paraméter keresést. Amennyiben visszatekintünk a 2. táblázat eredményeire, akkor láthatjuk, hogy más kutatók mélyebb konvolúciós hálókkal is hasonló felismerési százalékokat produkáltak a két adatbázison. Tehát a háló mélyítésével sem garantált a teljesítménynövekedés. Itt felmerülhet a kérdés, hogy mi lehet az oka a konvolúciós hálók „gyenge” teljesítményének. Véleményük szerint erre az egyik ok, a nagy dimenziós adatokban kereshető (*curse of dimesionality*). Ez azt jelenti, hogy minél nagyobb dimenzióban reprezentáljuk a vizsgált adatokat, annál több adatra van szüksége a tanuló algoritmusnak a hatékony működéshez [92]. Ez egy általános probléma, ami minden tanuló algoritmust érint. Viszont, mivel a konvolúciós hálók egy sokkal nagyobb dimenzióba konvertálják a kiindulási nyers adatot a konvolúciós rétegeik által, ezért itt a tanító adathalmaznak is sokkal nagyobbnak kellene lennie, mint egy egyszerű sekély módszer esetében. Viszont az emberi cselekvés felismeréshez készített adatbázisok sokkal kevesebb adatot tartalmaznak, mint például az objektum felismeréshez használt adathalmazok. Sajnos itt az adatbegyűjtés időigényesebb és néha kényelmetlen is lehet, ami kifejezetten igaz az idősebb generáció számára.

Megfigyelhetjük azt is, hogy az objektum felismeréshez használt konvolúciós hálók (például: AlexNet vagy GoogleLeNet) sokkal mélyebbek,

mint amiket a cselekvés felismerésben eddig használtak. Ennek alapján azt feltételezzük, hogy a cselekvés felismerésben használt viselhető szenzorok adatai nem tartalmaznak olyan összetett mintákat, mint a színes képek, ezért nincs is szükség a háló túlzott elmélyítésére. Végül azt is figyelembe kell venni, hogy a konvolúciós hálók nem igényelnek statikus adatleíró-kinyerést, mint a sekély módszerek, mivel (elméletileg) a konvolúciós szintek segítségével végzi el ezt a folyamatot. Viszont a sekély neurális hálók tesztelésekor, mi idő és frekvenciatérbeli adatleírókat is használtunk, míg a konvolúciós hálók csak az időtérbeli jellel dolgoznak. Az aktuális fejezetben ismertetett eredmények a [J8]-as cikkünkből származnak és ennek alapján fogalmazódott meg az 6. tézispont.

6. tézis: *Méréseink azt mutatták, hogy az általunk vizsgált két adatbázison a neurális háló együttesek, bináris neurális hálózatok és a konvolúciós hálók használata nem előnyösebb a sekély, kétszintű neurális hálózattal szemben, mivel ezeknek a komplexebb tanuló algoritmusoknak az alkalmazása vagy nem hozott felismerési arány növekedést, vagy csak 0.2%-os javulást eredményezett. Ezzel szemben a tanítási időigényük a sekély neurális hálózat többszöröse.*



27. ábra. A *CNN1* 50-512 konvolúciós hálóval mért felismerési arányok véletlen paraméterekkel a WARD adatbázison.

6. Offline eredmények tesztelése valós-időben

A kutatásunk lezárásaként, az általunk és mások által kapott offline eredmények megbízhatóságát vizsgáltuk meg valós-környezetben. Erre azért volt szükség, mivel az emberi cselekvés felismerés témakörében született eredmények túlnyomó többsége offline módon, valamilyen nyilvános adatbázison vagy saját készítésű adathalmazokon történt, ahol az adatok begyűjtése speciális környezetben zajlott. Ezek alapján nem rendelkeztünk semmilyen információval arról, hogy a korábbi eredmények mennyire állják meg a helyüket valós környezetben. Az itt elért eredményeink a [J7] és [C2] cikkekben találhatóak. Mindkét cikk méréseit az általunk létrehozott Androidos applikációval végeztük, amelynek az elkészítésében a társszerzők nyújtottak számomra segítséget. Munkánk során arra törekedtünk, hogy a mérést végző eszköz és a tanuló algoritmusok hasonló beállításokkal rendelkezzenek, mint a korábbi tanulmányokban.

Ahogy azt korábban említettem, a cselekvés felismerés egy gépi-tanulási folyamatnak tekinthető, amely adatbegyűjtést, szegmentálást, adatleíró-kinyerést, esetenként adatleíró kiválasztást, tanítást és döntést tartalmaz. A tanuló algoritmuson felül más szempontokat is figyelembe kell venni egy valós-idejű alkalmazás tervezésekor.

Számos kutató foglalkozott már a cselekvés felismeréshez szükséges szenzorok számának és azok elhelyezésének kérdésével. Erre jó példa a WARD és HAPT adatbázisok is. A kutatók nagy része egyetlen adatgyűjtő eszközt használt, amely az emberi test egy adott pontjához volt rögzítve. A [49] cikkben az adatgyűjtő a mellkasra volt erősítve és ennek segítségével, a cikk szerzői a futás, ülés, állás és fekvés hétköznapi cselekvéseket 97.5%, 82.9%, 98.6% és 100% pontossággal ismerték fel. Az [5] cikk szerzői több különböző helyen tesztelték az adatgyűjtő hatékonyságát és azt tapasztalták, hogy ha a zsebben van elhelyezve az eszköz, akkor a futás, ülés és állás cselekvéseket több mint 95%, míg a sétát több mint 90% pontossággal ismerik fel. A [46] tanulmányban az adatgyűjtő egyetlen okostelefon volt, amit ha a zsebben helyeztek el, akkor a kocogás, sétálás, ülés és állás cselekvéseket 90%, 95%, 95% és 100%-os arányban ismertek fel a tanulmány szerzői. A [32] cikk létrehozói szintén egyetlen adatgyűjtő eszközzel dolgoztak, ami az alany derekán volt elhelyezve. Ennek alapján a

futás, sétálás, ülés és állás cselekvésekre 99%, 94.3%, 91.7% és 100% felismerési arányokat kaptak.

Az imént említett tanulmányokhoz hasonlóan a [J7]-es munkánkban mi is egyetlen adatgyűjtőt használtunk, ami egy okostelefon volt. A bevezetőben leírt számos hasznos tulajdonsága miatt (magas szintű programozási környezet, nagy számítási kapacitás, különböző megjelenítési, adattárolási és kommunikációs lehetőségek) esett a választás erre az eszközre. Az adatgyűjtőt a jobb bokán helyeztük el, a [35] tanulmány iránymutatása alapján, ahol a szerzők megmérték, hogy hogyan változik a felismerési arány, ha az adatgyűjtő a derékra, combra vagy a bokára van felhelyezve. A tanulmány szerzői azt a konklúziót vontál le, hogy egyetlen szenzor használata esetén a boka az ideális hely az adatgyűjtő elhelyezésére és ezzel a választással a futás, kocogás és sétálás cselekvéseket 92.6%, 94.8% és 99.7% pontossággal ismerték fel. A mi kísérleteink során, az adatgyűjtő egy tépőzáras tok segítségével volt rögzítve a felmérésben résztvevő önkéntesek bokájára a 28. ábrán látható módon.

28. ábra. Az adatgyűjtő elhelyezése a jobb bokán.

Hasonlóan a fent bemutatott korábbi munkákhoz, mi is csak az elemi, hétköznapi cselekvések felismerésére koncentráltunk munkánk során és ennek alapján 7 cselekvés felismerését tűztük ki célul, név szerint:

- kerékpározás
- futás
- kocogás
- séta
- ülés
- állás
- fekvés

A 3.1 fejezetben bemutatott irányelveket követtük az applikáció mintavételi sebességének és az ablakméret beállításakor. Ebből fakadóan a mintavételi sebességet megközelítőleg 25 Hz-re állítottuk és egy ablak 32 elemből állt (1.28 másodperc), amik között 50% átfedést használtunk a tanítás során. A tesztfázisban nem alkalmaztunk átfedést az ablakok között, hogy ez által is gyorsabb legyen a döntés. Sajnos az operációs rendszer működéséből eredően ez a mintavételi sebesség nem garantált. A valós mintavételi sebesség a beállított vagy annál egy picivel kisebb. Azonban ez a lassú mintavétel egy modern okostelefonnak ez nem okoz problémát, ha az eszköz nincs túlterhelve.

A 2. fejezetben bemutatott szakirodalmi áttekintés és az azt követő fejezetekben kapott mérési eredményeink alapján az applikációba végül két tanuló algoritmust használtunk: a k-legközelebbi szomszédot (1NN) és a sekély mesterséges neurális hálózatot. Az első módszernél, k értékét egyre állítottuk és a jól ismert *Euklideszi* metrikát használtuk az adatleíró vektorok közötti távolság kiszámítására. Annak ellenére, hogy a k-legközelebbi szomszéd $k = 1$ választással a legzajérzékenyebb, több tanulmány is megmutatta, hogy ez a legoptimálisabb választás a tevékenység felismerés esetén [8], [35], [31]. A neurális háló beállításai teljesen megegyezett a 4.1 fejezetben bemutatott *ANN1* hálóval. Mindkét tanuló algoritmus a 4. táblázatban felsorolt adatleírókat kapja meg bemenetként normalizált formában, amit a nyers adatokból számítunk ki. A frekvenciatartománybeli áttéréshez a 3.6 fejezetben bemutatott módosított gyors Fourier transzformációt alkalmaztuk.

6.1. Az applikáció működése

Annak érdekében, hogy megőrizzük a mérésben résztvevő önkéntes adatait, az applikáció használatba vétele előtt létre kell hozni egy fiókot és be kell jelentkezni az általunk üzemeltetett távoli szerverre. A regisztráció során egy felhasználó nevet, az önkéntes korát és nemét kell megadni, ami a szerveren egy SQL adatbázisban lesz megőrizve. A regisztrációs és a bejelentkező képernyők a 29. ábrán láthatóak.

Az applikáció menüjében három fontos menüpont (funkció) érhető el. Az első menüpont alatt lehet tanító adatot gyűjteni minden egyes cselekvéshez. Miután a felhasználó kiválasztott egy cselekvést megadhatja azt az időintervallumot, ameddig adatot szeretne gyűjteni (maximum 4 perc). Ez alatt az idő alatt a szoftver adatot gyűjt a telefon beépített gyorsulás és giroszkóp szenzorjaitól és a mért értékeket fájlokba tárolja. Miután minden cselekvéshez lett adat rögzítve, csak ezt követően lehet a begyűjtött adatokat feltölteni a szerverre (opcionális). Ez a működési elv lehetővé teszi az adatok dinamikus és független begyűjtését.

A második menüpont alatt végezhető el a tanuló algoritmusok betanítása. A felhasználónak választania kellett a két algoritmus közül és ezt követően a háttérben lefutott a tanulási lánc összes eleme. Itt meg kell jegyeznem, hogy a láncban adatleíró kiválasztást nem használunk. A neurális háló tanítását követően a háló súlyai és az eltolás értékei automatikusan mentésre kerülnek, hogy a program következő indításakor ne kellje a tanítást újra elvégezni.

Végül a harmadik menüpont alatt lehet tesztelni a tanuló algoritmus hatékonyságát. A felhasználóknak ismét el kellett végezni a korábbi 7 cselekvést, de most csak 45 másodpercig. Ez alatt, a program tesztadatot gyűjt a szenzoroktól, kiszámítja és normalizálja az adatleírókat és az így kapott adatleíró vektort átadja a tanuló algoritmusnak. A tesztfázis után a felismerési arány egy tortadiagramon lesz látható. Az applikáció egyes funkcióihoz tartozó felhasználói felületek a 30. ábrán láthatóak. A tanulmányhoz tartozó további anyagok (összegyűjtött adathalmazok, részletes mérési eredmények) a projekt hivatalos oldalán érhető el (http://irh.inf.unideb.hu/user/sutoj/har.php).

6.2. Valós idejű tesztek

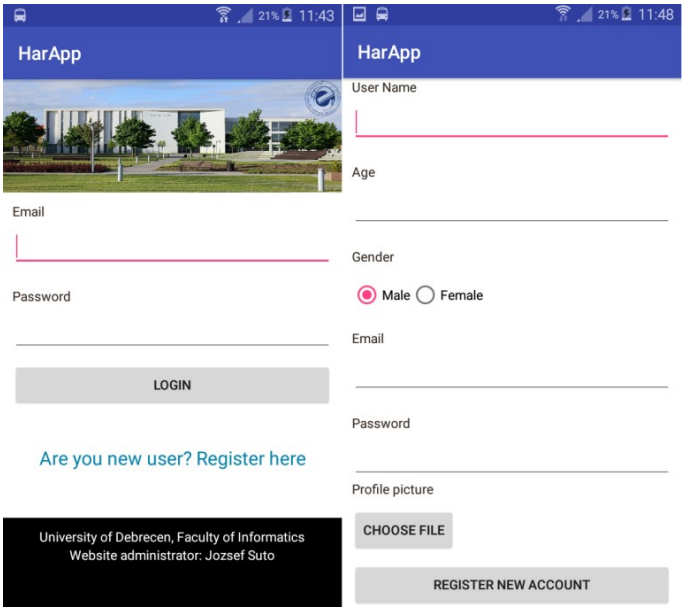
A valós idejű mérések során betartottuk a 2000-ben módosított Helsinki Nyilatkozat pontjait, amelyek az emberi kísérletekre vonatkoznak. A kísérletben három eltérő korú önkéntes vett részt. Az első és a harmadik önkéntes 27 és 28 éves férfiak voltak, míg a második egy 17 éves nő.

A tesztek előtt mindannyian tájékoztatást kaptak a kísérletről és az applikáció működéséről. Arra kértük őket, hogy képességeikhez mérten, a maximális számú adatot gyűjtsék össze minden cselekvéshez. Habár a leghosszabb adatgyűjtési időtartam minden cselekvés esetén mindössze 4 perc, ami nem okoz problémát a statikus cselekvéseknél (például: ülés, állás) viszont egy 4 perces futás igen megterhelő lehet, főként az idős korosztály számára. A 4 perces felső korlátot az önkéntesek visszajelzései alapján határoztuk meg, mivel ennél hosszabb adatbegyűjtési ciklus már kényelmetlen lenne.

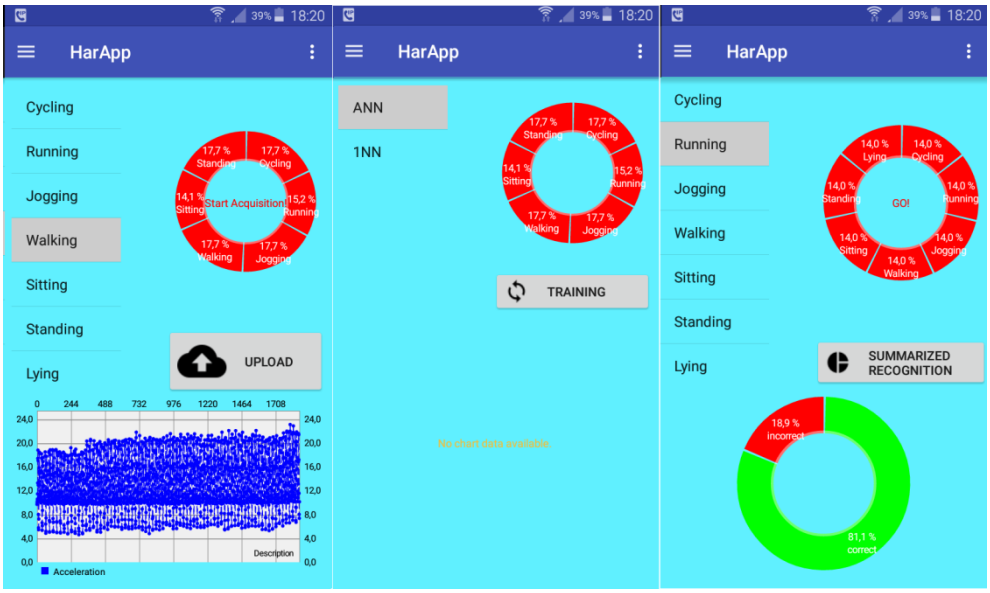
Az adatok begyűjtése után első körben megvizsgáltuk az applikációba implementált tanuló algoritmusok időigényét. A maximális tanító adathalmazzal (4 perces adatgyűjtés minden cselekvéshez) és a 45 másodperces tesztfázissal, a következő adat-előfeldolgozási, tanítási és döntési időket mértünk egy Google Nexus 4 telefonon:

- neurális háló tanítási ideje: 1471 másodperc
- 1NN tanítási ideje: 0 másodperc
  (nincs valós tanítás)
- neurális háló döntési ideje: 0.001 másodperc
- 1NN döntési ideje: 0.088 másodperc
- adat elő-feldolgozási idő tanításkor: 8.0 másodperc
- adat elő-feldolgozási idő döntéskor: 0.003 másodperc

Minden önkéntes az adatgyűjtést és a teszteket egymástól függetlenül végezte a mi felügyeletünk nélkül. Az így kapott felismerési arányok a két tanuló algoritmussal a 17. és 18. táblázatokban látható, ahol az utolsó sor az átlagos felismerési arány.



29. ábra. A bejelentkező és a regisztrációs képernyő.



30. ábra. A három menüpont felhasználói felületei.

Annak ellenére, hogy csak kisszámú önkéntessel végeztük el a felmérést, már így is jól megfigyelhető, hogy az általunk kapott valós idejű eredmények nem olyan magasak, mint a fejezet bevezetésében bemutatott tanulmányoknál, vagy mint a 2. táblázatban láthatóak eredmények. Részletes

elemzés után találtunk néhány lehetséges okot, amivel megmagyarázható ez a teljesítmény eltérés.

A vizsgálataink azt mutatták, hogy az egyik lehetséges ok a valós idejű adat nagy szórása. Ahogy korábban már többször említettem, a legtöbb kutató nyilvános adatbázisokon dolgozott, ahol az adathalmaz szórása számottevően kisebb. Például mind a WARD, mind a HAPT adatbázisok elkészítése során az adatgyűjtés zárthelyen történt a készítők felügyelete alatt. Ez a speciális környezet homogénebb adathalmazt eredményezett, amely nem fedi le a hétköznapi élet különböző szituációt. Annak érdekében, hogy ezt megmutassuk, ismét a WARD adatbázist vettük alapul, ahol az első három alany adathalmazából azt a részhalmazt használtuk fel, amelyet a jobb bokán elhelyezett adatgyűjtő rögzített a fekvés, ülés, állás, sétálás és kocogás cselekvések elvégzése közben. Ezt az adathalmazt adtuk át az applikációban is felhasznált tanuló szoftvernek (ahol a tanuló algoritmus neurális hálózat volt), ami a három személy adatai alapján átlagosan 99%, 92.2% és 97.2%-osan ismerte fel az imént említett 5 cselekvést. Ahhoz, hogy az általunk rögzített adatok és a WARD adatbázis adatai közötti eltérést szemléltessük, kételemű adatleíró vektorokat generáltunk 3 különböző időben rögzített séta cselekvés adathalmazaiból (séta A, séta B, séta C), amelyek a mi kísérletünk és a WARD adatbázis 1. alanyaitól származnak, és ezek eloszlását ábrázoltuk a 31. és 32. ábrákon. Az ábrák alapján megfigyelhető, hogy az adatbázisban az adatleíró vektorok kisebb szórással rendelkeznek a valós környezetben mérttel szemben. Ebben az esetben az átlagtól sokkal eltérő adatleíró vektorok a valós osztályuk döntési határán kívülre eshetnek.

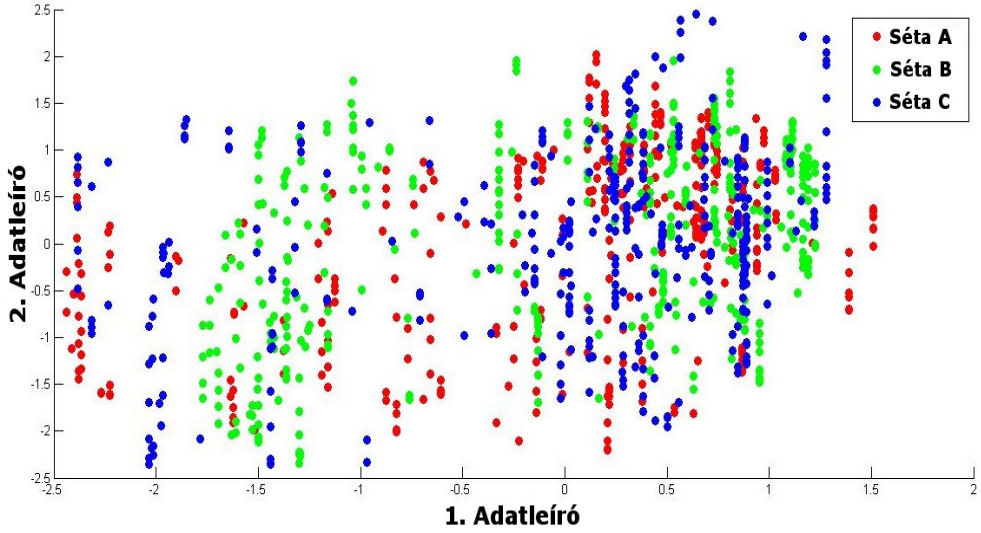
Az is megfigyelhető a 17. és 18. táblázatokból, hogy az ülés felismerése sok esetben nem volt hatékony. Ezt az adathalmazok közötti átfedéssel indokoljuk. A 33. ábra erre mutat példát, amelyen a kísérleteink során, az ülés és állás cselekvésekhez rögzített adathalmazokból számított adatleírók (kételemű) közötti átfedés látható. Mindkét cselekvésnél a láb ugyan abban a pozícióban is lehet, tehát mindkét cselekvéshez megegyező vagy hasonló adatok generálódhatnak. Nyilván ha egy tesztvektor erre a területre esik, akkor a tanuló algoritmus nem tud megbízható döntést hozni. Ez a probléma megoldható legalább egy további adatgyűjtő eszköz bevonásával, amit a test egy megfelelő pontjához kell rögzíteni (például: comb).

17. táblázat. Felismerési arányok az 1NN-el

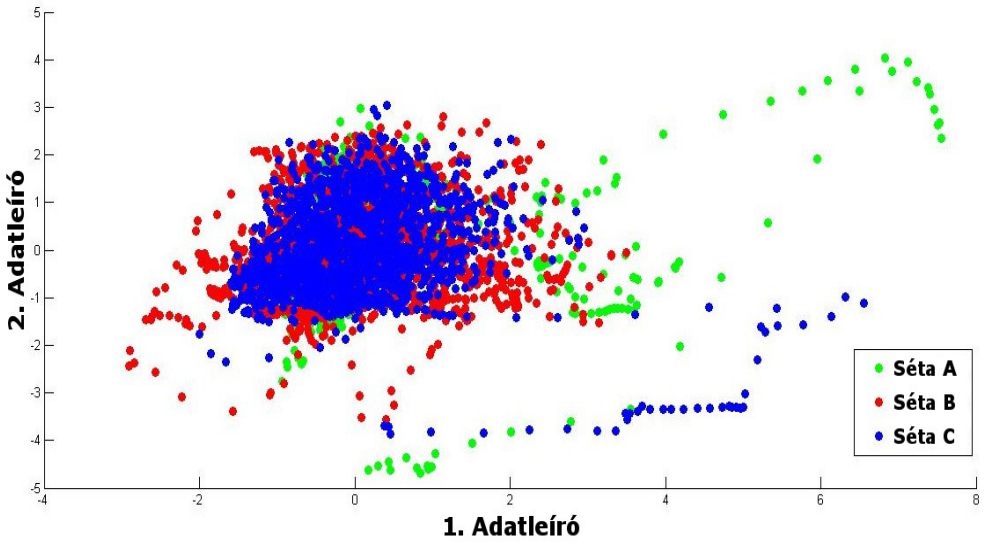
Cselekvés	Önkéntesek		
	1.	2.	3.
Fekvés	51.4%	100%	68.6%
Állás	85.7%	88.6%	74.3%
Ülés	37.1%	5.7%	74.3%
Séta	91.4%	74.3%	57.1%
Kocogás	57.1%	37.1%	57.1%
Futás	97.3%	65.7%	85.7%
Kerékpározás	88.6%	77.1%	71.4%
Átlag	72.7%	64.1%	69.5%

18. Felismerési arányok neurális hálóval

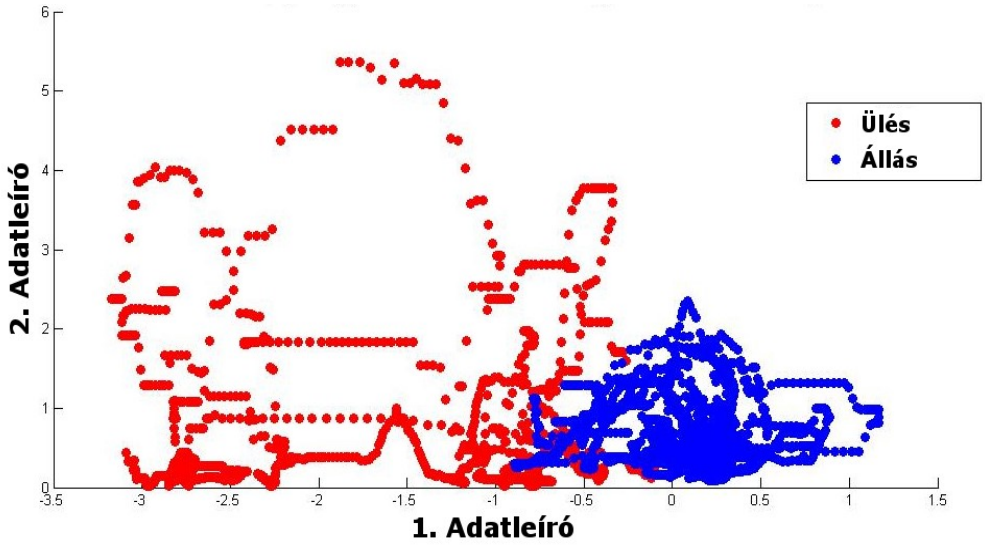
Cselekvés	Önkéntesek		
	1.	2.	3.
Fekvés	94.6%	89.2%	86.5%
Állás	91.9%	89.2%	94.6%
Ülés	16.2%	10.8%	83.8%
Séta	78.4%	91.9%	89.2%
Kocogás	8.1%	10.8%	83.8%
Futás	97.3%	89.2%	91.9%
Kerékpározás	13.5%	97.3%	91.9%
Átlag	57.1%	68.3%	88.8%



31. ábra. Adatleíró vektorok eloszlása a WARD adatbázisban.



32. ábra. Adatleíró vektorok eloszlása a mi kísérletünkben



33. ábra. Átfedés az ülés és állás cselekvések adatleírói között

Végül a két tanuló algoritmus közötti teljesítmény eltérés nem volt olyan számottevő, mint amit mi vártunk. Annak ellenére, hogy a neurális háló kevésbé zajérzékeny és sokkal komplexebb döntési határ generálására képes, mint az 1NN, mégsem produkált jobb eredményt. Ez az észrevétel egyrészt azt mutatja, hogy az adatleíró kiválasztásnak zajelnyomó hatása van, tehát azt feltételeztük, hogy nem zaj a teljesítmény csökkenés oka. Valójában a neurális háló gyengébb teljesítménye a hiányos tanító adathalmazban keresendő. Ha visszatekintünk a 3.5 fejezetben kapott eredményekre, ahol szintén 1NN-el és neurális hálóval dolgoztunk, akkor ott is megfigyelhető, hogy a neurális háló teljesítményben nem tudta számottevően felülmúlni az 1NN-t. Ha ehhez a megfigyeléshez hozzávesszük az 5.3 fejezetben (konvolúciós hálók) szintén jelentkező adathiányt, akkor ezt a problémát általánosan úgy fogalmazhatjuk meg, hogy minél összetettebb egy tanuló algoritmus, annál nagyobb tanító adathalmazra van szükség a hatékony működéséhez.

Ahhoz, hogy a fenti észrevételt alátámasszuk, az applikációnkban módosítottuk az ablakok közötti átfedést a tanítási fázisban 50%-ról megközelítőleg 90%-ra és az 1. számú önkéntessel ismét elvégeztettük a kísérletet. A mesterségesen dúsított tanító adathalmazzal a neurális háló átlagos felismerési aránya megközelítőleg 18%-kal növekedett a korábbi méréshez képest, míg az 1NN esetén ez kevesebb, mint 10%-os javulást

hozott. Nyilván a dúsított adathalmazzal a tanulási vagy a döntési ideje is növekedett a tanuló algoritmusoknak:

- neurális háló tanítási ideje: 8434 másodperc
- 1NN döntési ideje: 0.64 másodperc
- adat-előfeldolgozási idő tanításkor: 16 másodperc

A fent leírt eredmények a [J7] cikkben kerültek publikálásra. Ennek alapján jött létre a 7. tézispont:

7. tézis: *Miután összehasonlítottuk az általunk kapott valós-idejű mérési eredményeket a korábbi offline mérési eredményekkel, azt tapasztaltuk, hogy a valós-idejű mérések során kisebb felismerési arányokat kaptunk. Erre két indokot találtunk. Egyrészt a valós-idejű adatoknak nagyobb a szórása, mint a speciális környezetben összegyűjtött adatoké. A korábban használt adathalmazok speciális környezetben lettek begyűjtve, ami homogénebb adathalmazt eredményezett, és ez nem fedi le a hétköznapi élet lehetséges élethelyzeteit. Másrészt függetlenül attól, hogy hol helyezzük el az adatgyűjtőnket, egyetlen adatgyűjtő használata esetén, az egyes elemi cselekvésekhez rögzített adathalmazok között átfedések keletkezhetnek, mivel az adatgyűjtőben lévő inerciális szenzorok, az eltérő cselekvéseknél, ugyan abban vagy hasonló pozícióban is lehetnek.*

7. Összefoglalás

Az értekezés elején bemutattam az emberi cselekvés felismerés szakirodalmi hátterét, annak kérdéseit és bizonytalanságait, amelyek a dolgozatomat alkotó tanulmányok elkészítését motiválták.

A kutatásunk elején megvizsgáltuk különböző szenzor kombinációk hatékonyságát a WARD adatbázis 13 cselekvésének felismerésére. Ezen felül az adatleírók nélküli és az adatleírókkal kibővített normalizált nyers adatok hatékonyságát is elemeztük egy kétszintű neurális hálózaton. A kapott eredmények alapján azt állapítottuk meg, hogy a nyers adatból számított adatleírók felhasználásával a tanuló algoritmus hatékonysága növelhető és minél kevesebb az adatgyűjtő eszközök száma, annál fontosabb az adatleírók használata.

Ezt követően megvizsgáltuk és megmutattuk, hogy miért jobb az adatleírók használata a normalizált nyers adatok helyett mesterséges neurális háló használatakor. Erre azt az indokot találtuk, hogy míg a nyers értékek eloszlása (és egyben a cselekvést jellemző minta) a csúszó ablakon belül nagymértékben eltérő, addig az adatleírók értékei függetlenek az adatelemek eloszlásától.

A következő lépésben összegyűjtöttük a szakirodalomból a legismertebb adatleíró-kinyerő módszereket és ezzel teszteltük a tanuló algoritmusok hatékonyságát. Az adatleírók összegyűjtését követően 8 szűrő és egy általunk létrehozott, naiv Bayes alapú keretező, adatleíró-kiválasztó módszerrel vizsgáltuk meg, hogy vajon létezik az adatleíróknak egy olyan szűk halmaza, amely a tanuló algoritmustól és az adott személytől függetlenül, hatékonyan alkalmazható. A kapott eredmények azt mutatták, hogy az adatleíró kombinációk hatékonysága személyfüggő és így nem lehet egy általánosan alkalmazható adatleíró halmazt definiálni.

A frekvenciatérbeli adatleírók kiszámításához kapcsolódóan, megvizsgáltuk a gyors Fourier transzformáció implementációs lehetőségeit. Azt vettük észre, hogy a hagyományos radix-2 algoritmus nem használja ki a szorzófaktorok közötti mindhárom összefüggést, ami által kevesebb szorzófaktort kell kiszámítani és tárolni. Ebből adódóan elkészítettünk a radix-2 algoritmushoz, egy függvényt, aminek felhasználásával a módosított FFT algoritmus mindössze N/8+1 szorzófaktor kiszámítását és tárolását

igényli. Ez azt jelenti, hogy még az FFT futtatása előtt, elegendő az első N/8+1 szorzófaktort letárolni és ez által kevesebb memóriát kell felhasználnunk.

Ezek után a neurális hálók eltérő hatékonyságára kerestük az okot. Első lépésként három, azonos szinttel és neuronszámmal, viszont eltérő aktivációs és hibafüggvénnyel rendelkező háló architektúrát teszteltünk véletlenszerűen kiválasztott hiper-paraméterekkel és eltérő bemeneti adatokkal. Munkánk eredményei rámutattak arra, hogy a hálók paramétereinek nagyobb hatása van a teljesítményükön, mint az architektúra kiválasztásának. Továbbá, azt is megmutattuk, hogy a véletlen hiper-paraméter kereséssel és az adatleírók számának növelésével jobb eredményt értünk el, mint mások két nyilvános adatbázison.

Ezt követően megvizsgáltuk a hatékonyságát egy kibővített, sekély neurális hálózatnak, sekély neurális hálók együttesének, bináris neurális hálózatoknak és két konvolúciós hálónak. Az itt kapott eredményeket összehasonlítva a sekély neurális hálóval kapottal, azt a következtetést vontuk le, hogy az általunk vizsgált adatbázisokon, az összetettebb vagy mélyebb módszerek használata nem célszerű, mivel a tanuló algoritmus komplexitása vagy nem, vagy csak csekély javulást eredményez, viszont a döntési és legfőképp a tanulási időigénye a sekély háló sokszorosa lehet. Tehát, egy valós idejű cselekvés felismerő rendszerben, jelenleg célszerűbb egy kétszintű sekély neurális háló használata az összetettebb változatokkal szemben.

Végül az általunk és mások által kapott offline eredmények megbízhatóságát teszteltük valós környezetben. Ehhez a tanulmányhoz egy okostelefont használtunk adatgyűjtő és adat kiértékelő eszközként, amit egy általunk készített Android alapú applikáció végzett. Az applikációt és a tesztkörnyezet kialakítását (adatgyűjtő elhelyezése, mintavételi sebesség, ablakméret, stb.) a szakirodalom útmutatásai alapján végeztük. Az applikációban a felismerési arányokat két tanuló algoritmussal vizsgáltuk – k-legközelebbi szomszéd és sekély neurális hálózat. A kapott eredményeket összehasonlítva a korábban mért offline eredményekkel eltérést tapasztaltunk. Ezt az eltérést egyrészt azzal magyarázzuk, hogy a valós környezetből származó adatokból számított adatleírók szórása nagyobb, mint a felügyelt környezetben gyűjtött adatoknál. Így lehetnek olyan adatleíró vektorok, amelyek a valós osztályuk döntési határán kívülre fognak esni.

Másrészt egyetlen szenzor használata esetén, az egyes cselekvésekhez tartozó adathalmazok átfedhetnek egymáson, függetlenül az adatgyűjtő elhelyezésének helyétől, ami hibás osztályozáshoz vezet. A tanulmány végén, az applikációban használt két tanuló algoritmus eredményeit és a korábbi eredményeinket is felhasználva megfigyeltük, hogy minél összetettebb egy tanuló algoritmus, annál nagyobb tanító adathalmazra van szükség a hatékony működéséhez.

Irodalomjegyzék

[1] Frank LD, Engelke PO (2001) The built environment and human activity patterns: exploring the impacts of urban form on public health, J. Plan. Lit., 16:202-218.

[2] Godfrey A, Conway R, Meagher D, Olaighin G (2008) Direct measurement of human movement by accelerometry, Med. Eng. Phys., 30:1364-1386.

[3] Sebestyen G, Tirea A, Albert R (2012) Monitoring human activity through portable devices, Carpathian Journal of Electronic and Computer Engineering, 5:101-106.

[4] Gao L, Bourke AK., Nelson J (2014) Evaluation of accelerometer based multi-sensor versus single sensor activity recognition systems, Med. Eng. Phys., 36:779-785.

[5] Maurer U, Smailagic A, Siewiorek DP, Deisher M (2006) Activity recognition and monitoring using multiple sensors on different body positions. In: Proc. of the International Workshop on Wearable and Implementable Body Sensor Networks, Cambridge, pp. 112-116.

[6] Orha I, Oniga S (2015) Wearable sensor network for activity recognition using inertial sensors, Carpathian Journal of Electronics and Computer Engineering, 8:3-6.

[7] Bayat A, Pomplun M, Tran DA (2014) A study on human activity recognition using accelerometer data from smartphones, Procedia Computer Science, 34:450-457.

[8] Duarte F, Lourenco A, Abrantes A (2014) Classification of physical activities using a smart phone: evaluation study using multiple users, Procedia Technology, 17:239-247.

[9] Botta A, Donato W, Persico V, Pecape A (2015) Integration of cloud computing and Internet of Things: a survey, Future. Gener. Comp. Sy., 56:684-700.

[10] De Maio C, Fenza G, Loia V, Orciuoli F (2017) Distributed online Temporal Fuzzy Concept Analysis for stream processing in smart cities. J. Parallel. Distr. Com. doi: http://doi.org/10.1016/j.jpdc.2017.02.002.

[11] Shoaib M, Bosch S, Incel OD, Scholten H, Havinga PJM (2015) A survey of online activity recognition using mobile phones, Sensors, 15:2059-2085.

[12] Godfrey A, Bourke AK, Olaighin GM, Van de Ven P, Nelson J (2011) Activity classification using a single chest mounted tri-axial accelerometer, Med. Eng. Phys., 33:1127-1135.

[13] Lugade V, Fortune E, Morrow M, Kaufman K (2014) Validity of using tri-axial accelerometers to measure human movement – part I: posture and movement detection, Med. Eng. Phys., 36:169-176.

[14] Chern-Sheng L, Huang CH, Yun-Long L, Chuang-Chien C, Chi-Shih C (2007) Wearable device for real-time monitoring of human falls, Measurement, 40:831-840.

[15] Lara OD, Labrador MA (2013) A survey on human activity recognition using wearable sensors, IEEE Commun. Surv. Tut., 15:1192-1209.

[16] Chernbumroong S, Cang S, Atkins A, Yu H (2013) Elderly activities recognition and classification for applications in assisted living, Expert Syst. Appl., 40:1662-1674.

[17] Ponce H, Villasenor MLM, Pechuan LM (2016) A novel wearable sensor-based human activity recognition approach using artificial hydrocarbon networks, Sensors, 16:1-28.

[18] Collobert R, Weston J, Bottou L, Karlen M, Kavukcuoglu K, Kuksa P (2011) Natural language processing (almost) from scratch, J. Mach. Learn. Res., 12:2493-2537.

[19] Krizevsky A, Sutskerev I, Hinton GE (2012) ImageNet classification with deep convolutional neural networks. In: Proc. of the Neural Information Processing Systems, Nevada, pp. 1-9.

[20] Simonyan K, Zisserman A (2015) Very deep convolutional networks for large-scale image recognition. In: Proc. of the 5th International Conference on Learning Representations, San Diego, pp. 1-14.

[21] Simard PY, Steinkraus D, Platt JC (2003) Best practice for convolutional neural networks applied to visual document analysis. In Proc. of the 7th International Conference on Document Analysis and Recognition, Washington, pp. 958-962.

[22] Sheng M, Jiang J, Su B, Tang Q, Yahya AA, Wang G (2016) Short-time activity recognition with wearable sensors using convolutional neural networks. In: Proc. of the 15th ACM SIGGRAPH Conference on Virtual-Reality Continuum and Its Applications in Industry, Zhuhai, pp. 413-416.

[23] Jiang W, Yin Z (2015) Human activity recognition using wearable sensors by deep convolutional neural networks. In: Proc. of the 23th ACM International Conference on Multimedia, Brisbane, pp. 1307-1310.

[24] Chen Y, Xue Y (2015) Deep learning approach to human activity recognition based on single accelerometer. In: Proc. of the 2015 IEEE International Conference on Systems, Man, and Cybernetics, Hong Kong, pp. 1488-1492.

[25] Bhattacharya S, Lane ND (2016) From smart to deep: robust activity recognition on smartwatches using deep learning. In: Proc. of the 2016 IEEE International Conference on Pervasive Computing and Communication Workshops, Sydney, pp. 1-6.

[26] Hammerla NY, Halloran S, Plots T (2016) Deep, convolutional, and recurrent models for human activity recognition using wearables. In: Proc. of the 25th International Joint Conference on Artificial Intelligence, New York, pp. 1533-1540.

[27] Zheng M, Nguyen LT, Yu B, Mengshoel OJ, Zhu J, Wu P, Zhang J (2014) Convolutional neural networks for human activity recognition using mobile sensors. In: Proc. of the 6th International Conference on Mobile Computing, Applications and Services, Austin, pp. 197-205.

[28] Gjoreski H, Bizjak J, Gjoreski M, Gams M. (2016) Comparing deep a classical machine learning methods for human activity recognition using wrist accelerometer. In: Proc. of the 25th International Joint Conference on Artificial Intelligence, New York, pp. 1-7.

[29] Yang JB, Nguyen MN, San PP, Li XL, Krishnaswamy S (2015) Deep convolutional neural networks on multichannel time series for human activity recognition. In: Proc. of the 24th International Joint Conference on Artificial Intelligence, Buenos Aires, pp. 3995-4001.

[30] Saez Y, Baldominos A, Isasi P (2016) A comparison study of classifier algorithms for cross-person physical activity recognition, Sensors, 17:66-92.

[31] Rahman HA, Ge D, Faucheur AL, Prioux J, Carrault G (2017) Advanced classification of ambulatory activities using spectral density distances and heart rates. Biomed. Signal Poces., 34:9-15.

[32] Yang JY, Wang JS, Chen YP (2008) Using acceleration measurements for activity recognition: an effective learning algorithm for constructing neural classifiers, Pattern Recogn. Lett., 29:2213-2220.

[33] Khan AM, Lee YK, Lee SY, Kim TS (2010) A triaxial accelerometer-based physical-activity recognition via augmented-signal features and a hierarchical recognizer, IEEE T. Inf. Technol. B., 14:1166-1172.

[34] Kilinc O, Dalzell A, Uluturk I, Uysal I (2015) Inertial based recognition of daily activities with ANNs and spectrotemporal features. In: Proc. of

the IEEE 14th International Conference on Machine Learning and Applications, Miami, pp. 733-738.

[35] Preece JS, Goulermas JY, Kenney LPJ, Howard D (2009) A comparison of feature extraction methods for classification of dynamic activities from accelerometer data, IEEE T. Bio-Med. Eng., 56:871-879.

[36] Ordonez FJ, Roggen D (2016) Deep convolutional and LSTM recurrent neural networks for multimodal wearable activity recognition, Sensors, 16:115-130.

[37] Yang AY, Jafari R, Sastry SS, Bajcsy R (2009) Distributed recognition of human actions using wearable motion sensor networks, J. Amb. Intel. Smart En., 1:103-115.

[38] Anguita D, Ghio A, Oneto L, Parra X, Reyes-Ortiz L (2013) A public domain dataset for human activity recognition using smartphones. In: Proc. of the 21th European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, pp. 437-442.

[39] Pinardi S, Bisiani R (2010) Movement recognition with intelligent multisensory analysis, a lexical approach. In: Proc. of the 6th international Conference on Intelligent Environment, Kuala Lumpur, pp. 170-177.

[40] Su B, Tang Q, Wang G, Sheng M (2016) The recognition of human daily actins with wearable motion sensor systems, Lecture Notes in Computer Science: Transaction on Edutainment XII, 9292:68-77.

[41] Reiss A, Hendeby G, Sticker D (2013) A competitive approach for human activity recognition on smartphones. In: European Symposium of Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, pp. 455-460.

[42] Kastner M, Strickert M, Villmann T (2013) A sparse kernelized matrix learning vector quantization model for human activity recognition. In: European Symposium of Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, pp. 449-454.

[43] Parades BR, Aung H, Berthouze NB (2013) One-vs-one classifier ensemble with majority voting for activity recognition. In: European Symposium of Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, pp. 443-448.

[44] Ronao CA, Cho SB (2016) Human activity recognition with smartphone sensors using deep learning neural networks, Expert Syst. Appl., 59:235-244.

[45] Kouris I, Koutsouis D (2013) Application of data mining techniques to efficiently monitor chronic diseases using wireless body area networks and smartphones. Universal Journal of Biomedical Engineering, 1:23-31.

[46] Ayu MA, Ismail SA, Matin AFA, Montoro T (2012) A comparison study of classifier algorithms for mobile-phone's accelerometer based activity recognition. Engineering Procedia, 41:224-229.

[47] Ayachi FS, Nguyen HP, Pelletier CL, Goubault E, Boissy P, Duval C (2016) Wavelet-based algorithm for auto detection of daily living activities of older adults captured by multiple inertial measurement units (IMUs), Physiol. Meas., 37:442-461.

[48] Karantonis DM, Narayanan MR, Mathie M, Lovell NH, Celler BG (2006) Implementation of real-time human movement classifier using a triaxial accelerometer for ambulatory monitoring, IEEE T. Inf. Technol. B., 10:156-167.

[49] Wang J, Chen R, Sun X, She MFH, Wu Y (2011) Recognizing Human Daily Activities From Accelerometer Signal, Procedia Engineering, 15:1780-1786.

[50] Ciresan, DC, Meier U, Gambardella LM, Schmidhuber J (2010) Big deep simple neural nets excel on hand-written digit recognition, Neural Comput., 22:1-14.

[51] Kavanagh JJ, Menz BH (2008) Accelerometry: a technique for quantifying movement patterns during walking, Gait Posture, 28:1-15.

[52] Polikar R (1999) The story of wavelets, Physics and Modern Topics in Mechanical and Electrical Engineering, pp. 192-197.

[53] Cheng CH, Wang PSP (2005) Handbook of Pattern Recognition and Computer Vision, 3th ed., World Scientific.

[54] Hall MA, Smith LA (1999) Feature selection for machine learning: comparing a correlation based filter approach to the wrapper, In: Proc. of Florida Artificial Intelligence Symposium, Florida, pp. 235-239.

[55] Saeys Y, Inza I, Larranaga P (2007) A review of feature selection techniques in bioinformatics, Bioinformatics, 23:2507-2517.

[56] Gou Q, Liu B, Chen CW (2016) A two-layer and multi-strategy framework for human activity recognition using smartphone. In: Proc. of the IEEE International Conference on Communications, Kuala Lumpur, pp. 120-126.

[57] Jatoba CL, Grobmann U, Kunze U, Ottenbacher J, Stork W (2008) Context-aware mobile health monitoring: evaluation of different pattern

recognition methods for classification of physical activity. In: Proc. of the 30th Annual International IEEE EMBS Conference, Vancouver, pp. 5250-5253.

[58] Zhao Z, Morstatte F, Sharma S, Alelyani S, Anand A, Liu H (2011) Advancing feature selection research- ASU feature selection repository. Technical Report, Arizona State University, http: //featureselection.asu.edu/old/featureselectiontechreport.pdf

[59] Ertugrul OF, Kaya Y (2016) Determining the optimal number of body-worn sensors for human activity recognition. Soft Comput., 20:1-8.

[60] Marsland S (2015) Machine Learning – An Algorithmic Perspective, 2th ed., CRC Press, USA, pp. 27-31.

[61] Guyon I, Elisseeff A (2003) An introduction to variable and feature selection, J. Mach. Learn. Res., 3:1157-1182.

[62] Parag T, Elgammal A, Mittal A (2006) A framework for feature selection for background subtraction, In: Proc. of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, New York, pp. 1916-1923.

[63] He X, Cai D, Niyogi P (2005) Laplacian score for feature selection, In: Proc. of the 18th Conference on Advances in Neural Information Processing Systems, Montreal, pp. 507-514.

[64] Bashir K, Xiang T, Gong S (2008) Feature selection on gait energy image for human identification, In: Proc. of the IEEE International Conference on Acoustics, Speech and Signal Processing, Las Vegas, pp. 985-988.

[65] Karegowda AG, Majunath AS, Jayaram MA (2010) Comparative study of attribute selection using gain ration and correlation based feature selection, International Journal of Information Technology and Knowledge Management, 2:271-277.

[66] Hulse JV, Khoshgoftaar TM, Napolitano A, Wald R (2009) Feature selection with high-dimensional imbalanced data, In: Proc. of the IEEE International Conference on Data Mining Workshops, Miami, pp. 507-514.

[67] Kohavi R, John GH (1997) Wrappers for feature subset selection, Artif. Intell., 97:273-324.

[68] Hall MA, Smith MA (1999) Feature selection for machine learning: comparing a correlation-based filter approach to the wrapper, In: Proc. of the Florida Artificial Intelligence Symposium, Florida, pp. 235-239.

[69] Jayaram MA, Karegowda AG, Manjunath AS (2010) Feature subset selection problem using wrapper approach in supervised learning, International Journal of Computer Applications, 1:13-17.

[70] Tan L, Jiang J (2013) Digital Signal Processing Fundamentals and Applications, 2nd ed., Elsevier, pp. 161-288.

[71] Jia L, Li B, Gao Y, Tenhunen H (1998) Implementation of a low power 128-point FFT. In: Proc. of the 5th Int. Conf. on Solid-State and Integrated Circuit Technology, Beijing, pp. 369-372.

[72] Nielsen MA (2015) Neural Networks and Deep Learning. Determination Press. Ebook. http://neuralnetworksanddeeplearning.com.

[73] Bergstra J, Bengio Y (2012) Random search for hyper-parameter optimization, J. Mach. Learn. Res., 12:281-305.

[74] Hagan MT, Demuth HB, Beale MH, Jesus OD (2014) Neural network design. 2th ed. eBook, http://hagan/okstate.edu/NNDesign.pdf.

[75] Glorot X, Bengio Y (2010) Understanding the difficulty of training deep feedforward neural networks. In: Proc. of the 13th International Conference on Artificial Intelligence and Statistics, Sardinia, pp. 249-256.

[76] Hansen LK, Liisberg C, Salamon P (1992) Ensemble methods for handwritten digit recognition. In: Proc. of the IEEE Workshop on Neural Networks for Signal Processing II, Helsingoer, pp. 333-342.

[77] Giacinto G, Roli F (2001) Design of effective neural network ensembles for image classification purposes. Image Vision. Comput., 19:699-707.

[78] Das R, Turkoglu I, Sengur A (2009) Effective diagnosis of heath disease through neural networks ensembles. Expert Syst. Appl., 36:7675-7680.

[79] Tsai CF, Wu JW (2008) Using neural network ensembles for bankruptcy prediction and credit scoring. Expert Syst. Appl., 34:2639-2649.

[80] Zhao Y, Gao J, Yang X (2005) A survey of neural network ensembles. In: Proc. of the International Conference on Neural Networks and Brain, Beijing, pp. 438-442.

[81] Yao X, Islam MN (2008) Evolving artificial neural network ensembles. IEEE Comput. Intell. M., 3:31-42.

[82] Zhou ZH, Wu J, Tang W (2002) Ensembling neural networks: many could be better than all. Artif. Intell., 137:239-263.

[83] Aly M (2005) Survey on multiclass classification methods. http://www.vision.caltech.edu/malaa/publications/aly05multiclass.pdf.

[84] Ou G, Murphey YM (2007) Multi-class pattern classification using neural networks. Pattern Recogn., 40:4-18.

[85] Galar M, Fernandez A, Barrenechea E, Bustince H, Herrera F (2011) An overview of ensemble methods for binary classifiers in multi-class problems: experimental study on one-vs-one and one-vs-all schemes. Pattern Recogn., 44:1761-1776.

[86] Rifkin R, Klautau A (2004) In defense of one-vs-all classification. J. Mach. Learn. Res., 5:101-141.

[87] Lorena AC, Carvalho A, Gama J (2008) A review on the combination of binary classifiers in multiclass problems. Artif. Intell. Rev., 30:19-37.

[88] Arel I, Rose DC, Karnowski TP (2010) Deep machine learning – a new frontier in artificial intelligence research. IEEE Comput. Intell. M., 5:13-18.

[89] LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient-based learning applied to document recognition. *P. IEEE* 86:2278-2324.

[90] Zeiler MD, Fergus M (2014) Visualizing and understanding convolutional networks. In: Proc. of the European Conference on Computer Vision, Zurich, pp. 818-833.

[91] Szegedy C, Liu W, Jia Y, Sermanet P, Reed S, Anguelov D, Erhan D, Vanhoucker V, Rabinovich R (2015) Going deeper with convolutions. In: Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, Boston, pp. 1-9.

[92] Marsland S (2015) Machine Learning an Algorithmic Perspective, 2th ed., CRC Press, pp. 17-19.

A. Függelék

1. táblázat. Felismerési arányok az 1. alany adathalmazán

Kiválasztó	ANN		1NN	
	N = 5	N = 6	N = 5	N = 6
CFS	93.1	94.7	98.4	99.7
CHI	84.9	92.6	100	100
FCBF	86.2	90.7	99.7	99.9
FIS	87.2	93.4	91.9	94.6
IG	18.0	24.7	12.5	12.5
KW	83.9	89.0	97.4	99.6
MRMR	87.9	90.8	95.9	98.7
T-teszt	12.7	51.7	43.5	77.4
Keretező	92.2	93.0	96.3	98.3

2. táblázat. Felismerési arányok az 2. alany adathalmazán

Kiválasztó	ANN		1NN	
	N = 5	N = 6	N = 5	N = 6
CFS	96.5	97.8	100	100
CHI	96.6	97.9	99.9	100
FCBF	82.5	85.5	99.7	99.9
FIS	71.3	84.7	81.7	83.4
IG	63.2	69.0	95.9	88.0
KW	89.9	92.2	99.8	99.9
MRMR	94.4	94.3	100	100
T-teszt	43.1	72.8	98.5	99.8
Keretező	96.4	98.6	98.7	99.4

3. táblázat. Felismerési arányok az 3. alany adathalmazán

Kiválasztó	ANN		1NN	
	N = 5	N = 6	N = 5	N = 6
CFS	95.0	96.0	88.5	99.8
CHI	93.7	94.6	100	100
FCBF	96.2	96.1	99.0	99.9
FIS	94.0	89.4	76.2	93.9
IG	62.3	65.4	62.2	62.9
KW	88.1	89.7	99.4	99.4
MRMR	82.6	83.9	79.5	79.7
T-teszt	62.3	65.5	62.2	66.4
Keretező	95.3	96.3	99.9	99.9

4. táblázat. Felismerési arányok az 4. alany adathalmazán

Kiválasztó	ANN		1NN	
	N = 5	N = 6	N = 5	N = 6
CFS	92.1	94.6	99.9	100
CHI	92.2	93.4	100	100
FCBF	96.2	96.1	99.0	99.9
FIS	75.0	84.7	71.0	77.1
IG	57.9	66.7	51.5	62.5
KW	82.3	83.0	68.1	68.9
MRMR	82.4	86.9	85.2	89.7
T-teszt	62.7	66.8	62.5	66.5
Keretező	93.7	95.0	100	100

5. táblázat. Felismerési arányok az 5. alany adathalmazán

Kiválasztó	ANN		1NN	
	N = 5	N = 6	N = 5	N = 6
CFS	90.7	95.5	99.6	99.5
CHI	92.3	94.5	100	100
FCBF	93.1	94.2	70.0	74.1
FIS	85.4	92.6	93.5	97.5
IG	38.4	47.9	26.2	59.3
KW	82.8	87.8	88.2	95.6
MRMR	85.8	88.2	62.7	74.7
T-teszt	59.5	61.1	82.0	92.3
Keretező	94.3	96.5	70.6	82.9

6. táblázat. Felismerési arányok az 6. alany adathalmazán

Kiválasztó	ANN		1NN	
	N = 5	N = 6	N = 5	N = 6
CFS	93.0	97.6	100	100
CHI	93.8	96.9	100	100
FCBF	92.2	95.3	100	100
FIS	58.7	62.3	83.1	91.8
IG	46.9	67.3	58.1	70.7
KW	81.4	91.9	85.5	95.7
MRMR	82.6	85.3	87.4	90.7
T-teszt	66.0	68.1	93.6	99.5
Keretező	94.5	96.5	100	100

7. táblázat. Felismerési arányok az 8. alany adathalmazán

Kiválasztó	**ANN**		**1NN**	
	N = 5	N = 6	N = 5	N = 6
CFS	92.5	93.8	99.2	99.7
CHI	92.4	89.8	100	100
FCBF	91.8	95.7	94.7	98.8
FIS	88.0	70.5	98.3	99.9
IG	47.9	57.3	42.5	48.6
KW	87.8	91.0	83.3	87.7
MRMR	84.4	87.0	96.7	99.0
T-teszt	47.9	57.9	42.5	46.9
Keretező	95.8	96.8	100	100

B. Függelék: Publikációs Lista

Idegen nyelvű folyóiratcikkek

[J1] Suto J, Oniga S, Buchman A (2015) Real time human activity monitoring, Annales Mathematicae et Informaticae, 44:187-196.

[J2] Oniga S, Suto J (2015) Optimal recognition method of human activities using artificial neural networks, Meas. Sci. Rev., 15:323-327. **IF.: 0.969**.

[J3] Suto J, Oniga S (2015) A new relation between "twiddle factors" in the fast Fourier transformation. Elektron. Elektrotech., 21:56-59. **IF.: 0.389**.

[J4] Oniga S, Suto J (2016) Activity recognition in adaptive assistive systems using artificial neural networks, Elektron. Elektrotech., 22:68-72. **IF.: 0.859**.

[J5] Suto J, Oniga S (2017) Efficiency investigation of artificial neural networks in human activity recognition. J. Amb. Intel. Hum. Comp. https://doi.org/10.1007/s12652-017-0513-5. **IF.: 1.588** (2016).

[J6] Suto J, Oniga S, Pop-Sitar P (2017) Feature analysis to human activity recognition, Int. J. Comp. Commun., 12:116-130. **IF.: 1.374** (2016).

[J7] Suto J, Oniga S, Lung C, Ioan O (2018) Comparison of offline and real-time human activity recognition results using machine learning techniques, Neural. Comput. Appl. https://doi.org/10.1007/s00521-018-3437-x. **IF.: 2.505** (2016).

[J8] Suto J, Oniga S (2018) Efficiency investigation from shallow to deep neural network techniques in human activity recognition. Elbírálás alatt.

Idegen nyelvű konferenciacikkek

[C1] Suto J, Oniga S, Pop-Sitar P (2016) Comparison of wrapper and filter feature selection algorithms on human activity recognition. In: Proc. of the 6th International Conference on Computers Communications and Control, Oradea, Baile Felix, pp. 124-129.

[C2] Suto J, Oniga S, Lung C, Ioan O (2017) Recognition rate difference between real-time and offline human activity recognition, In: Proc of the International Conference on Internet of Things for the Global Community, Funchal, pp. 1-6.