Budapesti Műszaki és Gazdaságtudományi Egyetem
Villamosmérnöki és Informatikai Kar
Méréstechnika és Információs Rendszerek Tanszék

Valósidejű bólintásdetekció gépi tanulási módszerekkel

TDK dolgozat

Készítette:

Tass Mihály

Konzulens:

dr. Hullám Gábor
Révy Gábor

2022

Tartalomjegyzék

Kivonat

Az emberi viselkedésanalízis egyik fontos eleme a nonverbális gesztusok felismerése. A nonverbális kommunikáció fontos elemei a kéz, a kar vagy a felsőtest is, azonban a gesztusok jelentős része az arcról és a fej mozgásából olvasható le. Ilyen többek között a bólintás, melynek detekciója több szempontból is kihívást jelent. Egyfelől különböző kulturális háttérrel bíró emberek eltérő mértékű és gyakoriságú fejmozgást alkalmaznak kommunikációjuk során, másfelől az egyes egyének szintjén is jelentős eltérések lehetnek a bólintás során realizálódó fejdőlésszög mértékében, valamint az egyéb irányú járulékos fejmozgásokban. Mindezek miatt általánosan alkalmazható bólintásdetekciót megvalósítani szakértői módszerekkel nehéz, jellemzően valamilyen korlátozások mellett lehetséges. Emiatt inkább gépi tanulás alapú módszerekre kell hagyatkoznunk, melyekhez azonban megfelelően annotált tanító adathalmaz szükséges. A bólintás kép alapján történő felismeréséhez elérhető több adathalmaz, azonban ezek felhasználása valósidejű detekcióhoz mérsékelt pontossághoz vezet, mivel az apró bólintások azonosítása így nem lesz lehetséges. A videó alapján történő felismerés nagyobb pontossághoz vezethet, ilyen adathalmaz azonban, amely a bólintás szempontjából kellő részletességgel annotált, korlátozott számban elérhető.

Dolgozatomban a bólintás valósidejű detekciójára két különböző módszert vizsgálok meg. Az első módszer egy rejtett Markov-modell alapú klasszikus gépi tanuló megoldás, amely a fej pozíciója alapján következtet a bólintásra. Ezután egy visszacsatolt konvolúciós neurális hálózat alapú módszert mutatok be. Az implementált eljárások teljesítményét a kutatási célokra elérhető, videókat tartalmazó SEWA adathalmazon értékelem ki, majd hasonlítom össze.

Abstract

An important element of human behaviour analysis is the recognition of non-verbal gestures. Hand, arm and the upper body are important elements of nonverbal communication, but a significant part of gestures can be read from facial and head movements. This includes nodding, which is challenging to detect for various reasons. On one hand, people from different cultural backgrounds use different degrees and frequencies of head movements in their communication, and on the other hand, there may be significant differences at the individual level in the degree of head pitch and other types of additional head movements that are realized during nodding. For these reasons, generally applicable nod detection is difficult to achieve using expert methods, typically only possible with some limitations. For this reason, we have to rely instead on machine learning-based methods, which however require a properly annotated training dataset. Several datasets are available for image-based detection of nods, but using them for real-time detection will lead to moderate accuracy, as the identification of small nods will not be possible. Recognition from video can lead to higher accuracy, but such datasets, annotated in sufficient detail for nodding, are available in limited quantities.

In my thesis, I investigate two different methods for the real-time detection of nods. The first method is a classical machine learning solution based on a hidden Markov model that can predict the nod based on the position of the head. Then, I present a method based on a recurrent convolutional neural network. I evaluate and compare the performance of the implemented methods on the SEWA video dataset, which is available for research purposes.

1. fejezet

Bevezetés

Az emberi kommunikáció két fő része a verbális, (hang alapú) és nonverbális (vizuális) kommunikáció. A nonverbális kommunikáció definiálható úgy, mint az arc, a mimika, a test és a hanglejtés segítségével történő közlés, vagy máshogy kifejezve a kommunikáció azon része, ami nem szavak útján történik. A nonverbális (vagy nem verbális) kommunikációnak mára jelentős szakirodalma van, ami érthető, hiszen a nonverbális kommunikáció nem csak az emberek közötti tudatos közlés része, hanem kiolvashatók belőle érzelmek, traumák, akár betegségek is. Adam Mehrabian pszichológiai kutatása szerint[5], az emberi közlés csupán 7%-a történik verbális úton, 38%-a paraverbális jeleken keresztül (pl. hanglejtés), és nagyjából 55%-a nonverbális úton.
A nonverbális jelek képzik tehát a legnagyobb részét ez emberi közlésnek. Ezek között vannak fiziológiai jelzések, amelyek az emberi test működésből adódnak és az evólúció során lettek belénk kódolva (pl. félelem), illetve vannak tanult jelek (pl. bólintás), amelyek különböző társadalmi csoportokban, kultúrákban eltérőek lehetnek.
A nonverbális jeleket leggyakrabban közlési csatornák mentén csoportosítják, ezek a vokális jelek, a tekintet vagy szemkontaktus, a mimika, a gesztusok, a testtartás, a térközszabályozás (proxemika), az emblémák és a kronémika. A viselkedésanalízis egyik fonos eleme a nonverbális gesztusok felismerése. A felismert gesztusokból szakértői rendszerek segítségével lehet detektálni kommunikációs jegyeket, érzelmeket, reakciókat, vagy akár mentális betegségeket. Ezen gesztusok felismerése még egy szakértő számára is nagy kihívás lehet, hiszen 8 különböző kategóriába tartoznak, sokszor rövid idő alatt történnek és különböző társadalmi csoportoknál eltérőek lehetnek. Emiatt a nonverbális kommunikációs jegyek automatizálása kívánatos lehet. A legtöbb közlést az acról és a fej mozgásából lehet leolvasni, így ezeknek a területeknek a monitorozása nagy figyelmet kap a szakirodalomban.
A dolgozat a bólintás detekcióra mutat néhány módszert, ami több szempontból is kihívást jelent. A bólintás egy tanult gesztus, ami nem egy fiziológiai jelzés, tehát az emberi működésből adódó alap jelzés. A világ legtöbb kultúrájában megtalálható és szinte minden esetben az egyetértést, az igenlést fejezi ki. A fejrázást a bólintás párjaként szokták használni, ez rendszerint a bólogatás ellentétét jelenti. Kivételt képeznek a bolgárok, akiknél egyedülálló módon ennek a két gesztusnak a jelentése ellentétes.
A detekciót nehezíti, hogy az emberi kommunikációban a bólogatás nagysága, frekvenciája különböző kultúrákban eltérhet. Szintén változhatnak ezek az értékek, ha két személynek eltérő a habitusa, vagy akár más témáról van szó.
Habár a gesztusok legtöbbször nagyobb mozgással járnak mint a mimikák, mégsem triviális a detektálásuk, mivel a mimikákkal ellentétben egy képről nem dönthető el egyértelműen, hogy történt-e bólintás vagy sem. A detekcióhoz egy videó, avagy egy idősorrendbe rendezett képszekvencia szükséges. Habár több olyan modszerrel találkoztam, ahol többkamerás rendszert, vagy fejre erősíthető eszközt használtak a probléma megoldásához, nekem az volt a célom, hogy egy közepes, vagy alacsony (webkamera) szintű kamera kép alapján

detektáljam a kiválasztott gesztust. Ennek előnye, hogy nagyobb erőforrásigény és felszerelés nélkül használható a megoldás. A dolgozatban megvizsgálok egy rejtett Markov-modell alapú klasszikus gépi tanuló megoldást, ami a fej pozíciója alapján következtet bólintásra, illetve egy visszacsatolt konvolúciós neurális hálózat alapú rendszert. Az implemetált eljárások teljesítményét a BAUM-1 [28] és a SEWA [15] adathalmazokon értékelem ki.

2. fejezet

Feldolgozott szakirodalom

A bólintásdetekció problémája ismert, azonban napjainkban még nem széles körben kutatott. Ennek egyik fő oka lehet, hogy nehéz hozzájutni megfelelően, bólintásra annotált videós adatbázishoz. Éppen ezért, a szakirodalomban főként hagyományos gépi tanuló módszerekkel próbálják megoldani a feladatot. A dolgozat következő részében olyan kutatásokat fogok bemutatni, amelyek relevánsak a témában és olyan módszereket használnak, amik megfelelők lehetnek a sikeres gesztus detekcióban.

2.1. Hagyományos gépi tanulási módszerek

2.1.1. Bólintás, mint gesztus

A bólintás felismerésének automatizálásához szükséges a gesztus minél pontosabb fizikai megfigyelése. A bólintás a fej periodikus, vertikális mozgása a nyakoz és felsőtesthez képest. Legtöbbször egy kérdésre adott "igen" választ, megértést, vagy megerősítést jelent. A gesztus kifejezésében a fejmozgáson kívül ugyanakkor közrejátszhatnak a szemek, illetve a száj is, bár ezek lényegesen kisebb jelentőséggel bírnak, mint a fej mozgása.
A fej mozgásának sebessége, a bólintás nagysága, frekvenciája, illetve az ismétlések száma is eltérő lehet. Egy kis biccentés megtörténhet akár egy másodperc harmada alatt is, viszont egyes esetekben egy lassabb fejmozgás eltarthat akár 2-2.5 másodpercig is. Akármilyen is a gesztus, ha a fej mozgása ismert, egy megfelelő algoritmus segítségével lehet következtetni arra, hogy a bólintás megtörtént-e vagy sem. A bólintás vagy bólogatás detekciója hagyományos gépi tanuló módszerekkel tehát sok esetben két részre osztható: a fej mozgásának monitorozása, illetve a kinyert adatokból a gesztus megtalálása, felismerése. A fejmozgás monitorozása nem kapcsolódik szorosan a bólintás-detekcióhoz, önmagában is gyakran kutatott téma.

2.1.2. Fejmozgás detekció

Ashish Kapoor [13] munkájában a fej mozgását a szem követésével monitorozta. Megoldásában infrared érzékeny kamerát használt, amivel a pupillák pozícióját tudta meghatározni. Ezt a videó összes képkockájára alkalmazva a szem pozíciójából a fej mozgására is következtetni tudott. Kapoor megoldásnak hátránya lehet, hogy egy-egy fényes tárgy, vagy szemüveg pontatlanná teheti a fej mozgásának detekcióját, illetve speciális kamerára van hozzá szükség.
Chen és Odobez [6] Microsoft Kinect camerát használtak, aminek segítségével három dimenzióban tudják monitorozni a fej mozgását. A fej pozíciójának detektálásához a 3D Morphable Model (3DMM) eljárást használták, a teljes háromdimenzós modell felépítéséhez pedig az Iterative Closest Points (ICP) algoritmust. Chen és Odobez robosztusabb és

pontosabb megoldást ad a problémára mint Kapoor, hiszen nem csak két pontot detektál az arcon, hanem egy teljes háromdimenziós modellt épít fel, ugyanakkor ehhez speciális ezközre, a Microsoft Kinect kamerájára van szükség.
A fej mozgásának monitorozása visszavezethető az arcfelismerés problémájára abban az esetben, ha az arcdetekció elég gyors ahhoz, hogy a videó összes képkockájára időben pontos eredényt adjon. A Google kutatói [14] olyan módszert dolgoztak ki, amely képes 468 háromdimenziós nevezetes pontot (landmark pontot) elhelyezni az arcon, akár száz képkockára másodpercenként. A detektált landmark pontok a 2.1 ábrán láthatóak. A megoldáshoz visszacsatolt neurális hálózatot használtak, amit nagyságrendileg harmincezer kép segítségével tanítottak be.
A módszer hagyományos kamerát használ és valós időben képes az arcot pontosan elhelyezni a térben.A fej mozgása monitorozható az egymást követő képkockákon detektált landmark pontok elmozdulását felhasználva. A Microsoft Kinect-es megoldáshoz hasonlóan ennek a megközelítésnek is előnye a robosztusság, a szem detekcióján alapuló módszerhez képest. Emellett a landmark pontok pozíciójából nem csak a fej mozgása detektálható, információt adnak többek között a szemek és a száj mozgásáról is. A dolgozatban bemutatott hagyományos gépi tanuláson alapuló módszerben a fejmozgás monitorozására én is a landmark pontos megoldást alkalmaztam a robosztussága, gyorsasága és megbízhatósága miatt.

2.1. ábra. Az arcra vetített 468 landmark pont

2.1.3. Bólintást detektáló algoritmus

A releváns szakirodalomban a hagyományos gépi tanuló módszereken alaputó bólintásdetekciók különféle megközelítéseket használnak a fej mozgásának monitorozására. Ezzel ellentétben, a fejmozgásból bólintásra következtető algoritmus a legtöbb esetben megegyezik, ez a rejtett Markov-modell (HMM).
Chen és Odobez [6] illetve Kapoor [13] megoldásaikban egyaránt rejtett Markov-modellt használnak. A fej pillanatnyi állapotát a bólintás tekintetében három diszkrét csoportba osztják, ezek a következők: a fej vertikálisan felfelé billen, a fej eredeti pozíciójában van, illetve a fej vertikálisan lefelé billen. Ez a három állapot a modell látható, megfigyelhető állapotai, a rejtett állapotok pedig a tény, hogy történt-e történt bólintás vagy sem. A fej folytonos mozgásából egy olyan szekvenciát lehet létrehozni, ahol minden képkockához

tartozik egy érték, aminek az értékkészlete a három fejpozíció halmaza. Ebből a szekvenciából a Viterbi algoritmus segítségével lehet a legvalószínűbb rejtett állapotszekvenciát kiszámítani, és következtetni a bólintásra. A modell megfelelő működéséhez az állapotátmenetek tanítása szükséges, amihez a Baum-Welch algoritmust szokás használni.
A rejtett Markov-modell előnye, hogy az állapotátmenetek kiszámításához, betanításához nincs szükség nagy mennyiségű pontosan annotált adatra szemben a neurális hálózat alapú megoldásokkal. Nem számít számításigényesnek sem a kiértékelése, sem a tanítása, kis helyen tárolható és gyors. Valósidejű detekcióra tehát minden paraméterében alkalmasnak látszik. A rejtett Markov-modell talán egyetlen hátránya, hogy diszkrét állapotokkal dolgozik, a fej mozgása pedig folytonos. Mivel a bólintás gesztus közben nem egységes a fej kitérésének mértéke, nem triviális feladat a fej pozícióját a fent említett fent-középen-lent diszkrét állapotok egyikébe sorolni.
A dolgozatban bemutatott hagyományos gépi tanulás alapú módszerben a bólintás detekciójához én is rejtett Markov-modellt használtam, alacsony erőforrásigénye és gyorsasága miatt.

2.2. Neurális hálózat alapú módszerek

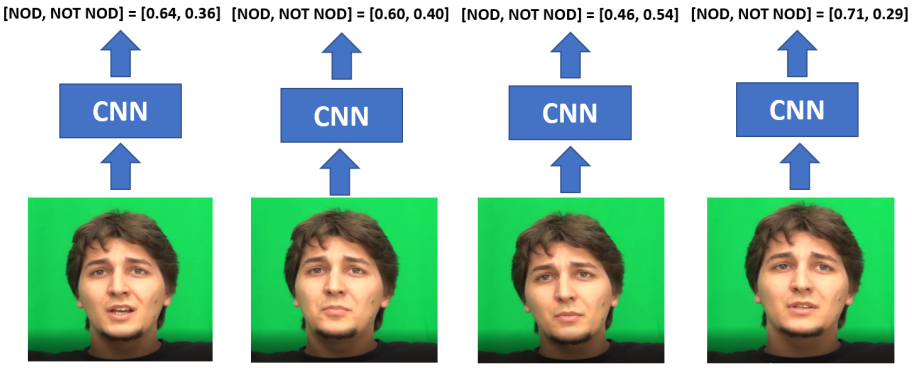
Habár a neurális hálózat alapú megoldások nagyon népszerűek napjainkban, a bólintásdetekció problémájának nem jellemző megközelítése. Ennek oka az lehet, hogy nagyon nehezen lehet hozzájutni megfelelően annotált és elegendő mennyiségű adathoz, a videós adathalamzokra ez hatványozottan igaz. Ha a megfelelő adathalmaz mégis rendelkezésre áll, a bólintás detekció redukálható egy olyan videóklasszifikációs problémára, ahol a videórészleteket két osztályba kell sorolni. Az egyik osztály a "történt bólintás" a másik pedig a "nem történt bólintás". A probléma az osztályok száma miatt egyszerűnek tűnhet, viszont például az aktivitás detekcióval ellentétben, a bólintás detekció esetében a videók tartalmukban nagyon hasonlóak. Ezért fontos, hogy az adathalmaz annotációja pontos legyen, illetve, hogy elegendő adat álljon rendelkezésre a tanításhoz.

2.2.1. Konvolúciós neurális hálózat alapú megoldások

A konvolúciós neurális hálózat (CNN) alapú képfelismerő megoldások nagyon népszerűek napjainkban, mivel kiemelkedő teljesítményt nyújtanak ezen a területen. Ezt a neurális hálózat típust fel lehet használni videó-klasszifikációs problémákhoz is, hiszen a videó nem más, mint időben rendezett képek szekvenciája. A CNN alapú videó osztályozási problémákra számos megoldás született az elmúlt években, leggyakrabban az emberi aktivitás detekció (Human activity recognition, HAR) területén alkalmazzák. A következő pontokban röviden bemutatok néhány konvolúciós neurális hálózat alapú megoldást a videó klasszifikáció problémájára.

2.2.1.1. Single-frame CNN

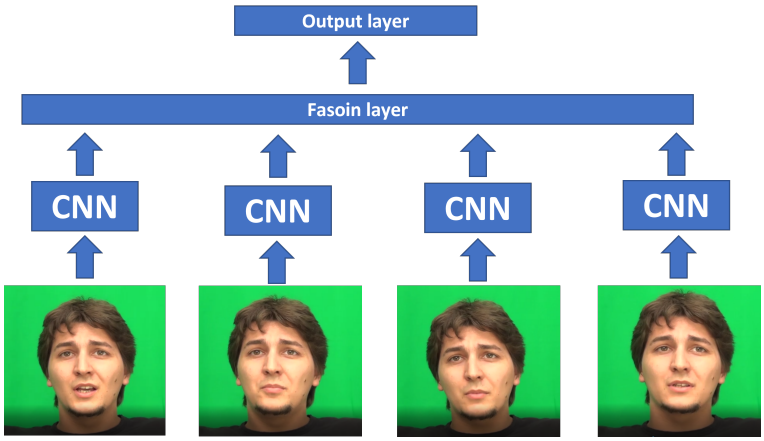
A Single-frame CNN[3] metódus lényege, hogy a videó egyes képkockái teljesen függetlenül kerülnek feldolgozásra (a 2.2 ábrán látható módon). A konvolúciós hálózat a képszekvencia összes elemére becslést ad, hogy melyik osztályba milyen valószínűséggel tartozhat. Az így megkapott valószínűségeket egy előre meghatározott intervallumon kiátlagolva lehet megkapni a végleges valószínűségi vektort. A single-frame eljárás nagy előnye, hogy valós időben jól alkalmazható, hiszen a képszekvencia csak két eleme változik minden új képkocka megjelenésével. Az utolsó elem kivételével a kiszámított valószínűségeket újra fel lehet használni, és csak a friss képkockákhoz tartozó értékeket kell meghatározni. A képkockák külön-külön történő feldolgozásával ugyanakkor elvész a videó időrendisége. A bólintás detekcióban ez problémát jelenthet, ugyanis fontos információ, hogy a fej milyen irányba mozgott. Bólintást jelenthet a fej középről lefelé irányuló mozgása, viszont fordítva ez nem mondható el.



2.2. ábra. Single-frame CNN architektúra

2.2.1.2. Late fusion CNN

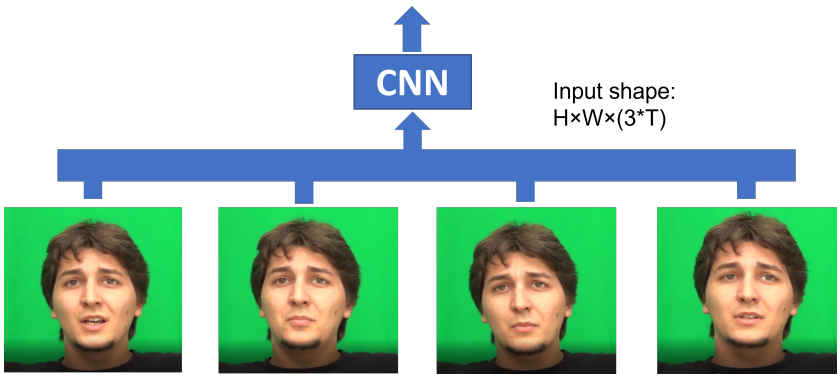
A late fusion metódus nem sokban tér el a Single-frame módszertől. Míg a single-frame módszer az eredményeket a neurális hálózat lefutása után átlagolja ki, a late fusion ezt a hálózatban végzi el egy fúziós szinten (a 2.3 ábrán látható módon). A megvalósításhoz több különálló single-frame hálózatot használnak, melyek különböző képkockákat dolgoznak fel, de a fusion szint közös, itt történik meg a feldolgozott adatok összefuzionálása.



2.3. ábra. Early-fusion CNN architektúra

2.2.1.3. Early fusion CNN

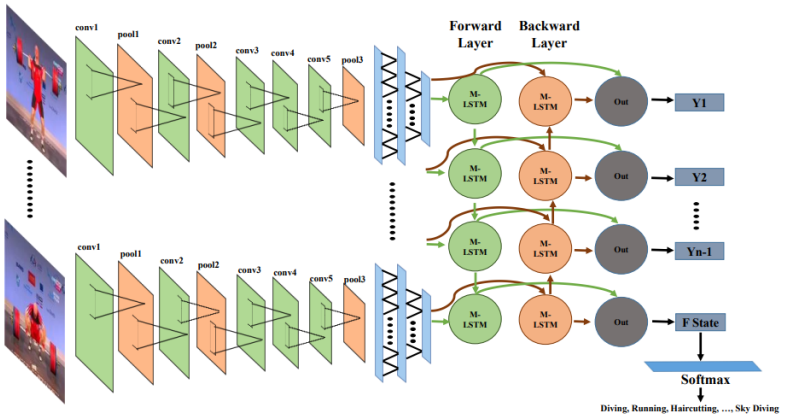
Az early fusion metódus a hasonlít a late fusion-re, viszont itt a videó képkockáinak egyesítése nem a neurális hálózat lefutása után történik, hanem előtte. A videó képeit a feldolgozás előtt egy háromdimenziós tenzorba egyesíti, ami ezután a konvolúciós hálózat bemenete lesz (a 2.4 ábrán látható módon).



2.4. ábra. Early-fusion CNN architektúra

2.3. CNN-RNN architechtúra

A CNN-RNN architechtúra architektúra leggyakoribb felhasználási területe a videó klasszifikáció, általában LSTM visszacsatolt hálózattal valósítják meg. Az architechtúra lényege, hogy a videót képekre bontva egy konvolúciós neurális hálózat (CNN) előre feldolgozza azokat és így továbbítja egy visszacsatolt neurális hálózatnak, ami a klasszifikációt végzi. Tehát konvolúciós neurális hálózat bemenete egy videó összes képkockája, amelyeken jellemző jegyek kinyerését, azaz feature extractiont hajt végre, ezzel kiemelve néhány fontosnak vélt részletet. A videós anyag komplexitása így jelentősen csökken, méghozzá úgy, hogy jelentősen megkönnyíti a visszacsatolt neurálos hálózat (RNN) feladatát, a klasszifikációt.



2.5. ábra. CNN-LSTM architektúra, forrás: [25]

A hálózat visszacsatolt része egy fix hosszúságú képszekvenciából egyenként kinyert jellemzőket egy egységként kapja meg, ezáltal – a single-frame módszerrel ellentétben – figyelembe veszi annak sorrendjét is. A CNN-RNN módszert gyakran használják emberi cselekvés detekcióra [25], arckifejezés detekcióra[18], illetve különböző videóelemzési

problémák megoldására egyaránt. A 2.5 ábrán egy LSTM-mel megvalósított CNN-RNN architektúra látható, amelyet cselekvés detekcióra terveztek. A CNN-RNN architektúra robosztusabb megoldást nyújt a fent említett problémára mint az egyszerűbb, konvolúciós hálózat alapú megközelítések (2.2.1.1 és 2.2.1.2), viszont implementációja összetettebb és tanítása több időt igényel. A feature extraction-t a klasszifikáció mellett minden új képkockára el kell végezni, így a valós idejű alkalmazása valamivel számításigényesebb. A dolgozatomban bemutatott neurális hálózat alapú megoldásban ezt az architektúrát használom, amit a SEWA adatbázis [15] segítségével tanítok és értékelek ki.

3. fejezet

Implementáció elméleti háttere

Ebben a fejezetben a bólintásdetekcióra kidolgozott módszerek elméleti hátterét fogom bemutatni. A bólintás gesztus felismerésére kidolgoztam egy hagyományos gépi tanuló módszert és egy neurális hálózat alapú megoldást is. A hagyományos módszer elméleti hátttereként bemutatom a rejtett Markov-modell (HMM) működését, tanítását és kiértékelését. A neurális hálós megoldás kapcsán a jellemző jegyek kinyerésének szempontjait és a visszacsatolt neurális hálózatok működését mutatom be.

3.1. Rejtett Markov-modell

A rejtett Markov-modell elméleti bemutatásához szükséges a diszkrét Markov-láncok bevezetése. A Markov-lánccal kapcsolatos definíciók és képletek a [19] hivatkozásban megtalálhatók.

3.1.1. Markov-láncok

Legyen adott egy S diszkrét halmaz.
Definíció: S értékű diszkrét valószínűségi változók egy X_0, X_1, X_2,... végtelen sorozatát Markov-láncnak hívjuk, ha a valószínűségi változók között a következő összefüggés fennáll. Minden $n \geq 0$ egész szám és i, j, $i_0, ...i_{n-1} \in$ S állapotok esetén teljesül a

$$P(X_{n+1} = j|X_n = i, X_{n-1} = i_{n-1}, ..., X_1 = i_1, X_0 = i_0) = P(X_{n+1} = j|X_n = i)$$

egyenlőség.
A Markov-lánc időben homogénnek mondható, ha teljesül minden $n \geq 0$ esetre, hogy

$$P(X_{n+1} = j|X_n = i) = P(X_1 = j|X_0 = i)$$

Gyakorlati alkalmazásban az index sokszor az időt jelenti, ezért a Markov-láncra úgy is lehet tekinteni, mint ami valamilyen időben végtelen jelenséget ír le.
A Markov-lánc fontos tulajdonsága, hogy egy jövőbeli állapot nem függ a korábbi megelőző állapotoktól (múlttól), csak a közvetlenül előtte lévőtől (jelentől).
Tehát $X_n = i$ esetén a Markov-lánc jövője $(X_{n+m} = j_m, ..., X_{n+2} = j_2, X_{n+1} = j_1)$ független a múltbéli eseménytől $(X_{n-1} = i_{n-1}, ..., X_1 = i_1, X_0 = i_0)$.
A Markov-láncok fontos leírói a $P(X_{n+1} = j|X_n = i)$ egylépéses átmenetvalószínűségek. Ha ez minden $i, j \in S$ -re adott, akkor az átmenetvalószínűségeket egy $|S| \times |S|$-es A mátrixba gyűjtjük, ahol a mátrix i, j helyén a

$$A_{ij} = P(X_{n+1} = j|X_n = i)$$

valószínűség szerepel. Az A neve átmenetvalószínűségi mátrix. Az A átmenetvalószínűségi mátrix sztochasztikus, azaz minden sorában a valószínűségek összege egy. Azaz

$$\forall i : \sum_{j=1}^{S} A_{ij} = 1$$

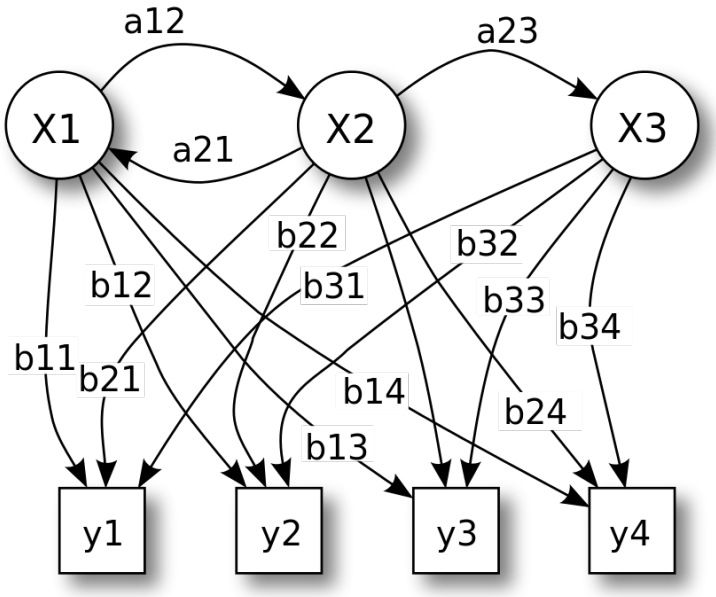
Ezt a tulajdonságot kihasználva a Markov-láncokat ábrázolhatjuk irányított gráfként, ahol a csúcsok az állapotok, és az ij átmenet értéke az A_{ij}. Ekkor az összes csúcs kilépő éleinek értékösszege egy.

3.1.2. Rejtett Markov-modell felépítése

A rejtett Markov-modell egy olyan diszkrét idejű, véges állapotterű, homogén Markov-lánc, ahol vannak ismeretlen állapotok, viszont az átmenetvalószínűség mátrix (A) az ismeretlen állapotok között ebben az esetben is adott. Az ismert állapotok halmazát megfigyelt vagy ismert állapotoknak nevezzük (Y), míg az ismeretlen állapotokra rejtettként hivatkozunk (X). Az állapotátmenet-mátrix mellett definiálható egy eloszlás-mátrix is, amely azt írja le, hogy a megfigyelt állapotok alapján milyen valószínűséggel vagyunk az egyes rejtett állapotokban. A rejtett Markov-modell tehát m megfigyelt és k rejtett állapot esetén az állapotok mellett két mátrixal jellemezhető. Egy $m \times m$-es P állapotátmenet-mátrixszal, ami a megfigyelt állapotok közötti átmenet-valószínűségeket definiálja és egy $k \times m$-es B eloszlás-mátrixszal, ami az ismert változókhoz tartozó rejtett állapotok valószínűségét tárolja. Tehát annak a valószínűsége, hogy bármely k megfigyelt állapothoz j rejtett állapot tartozik:

$$B_{jk} = P(Y_n = k | X_n = j)$$

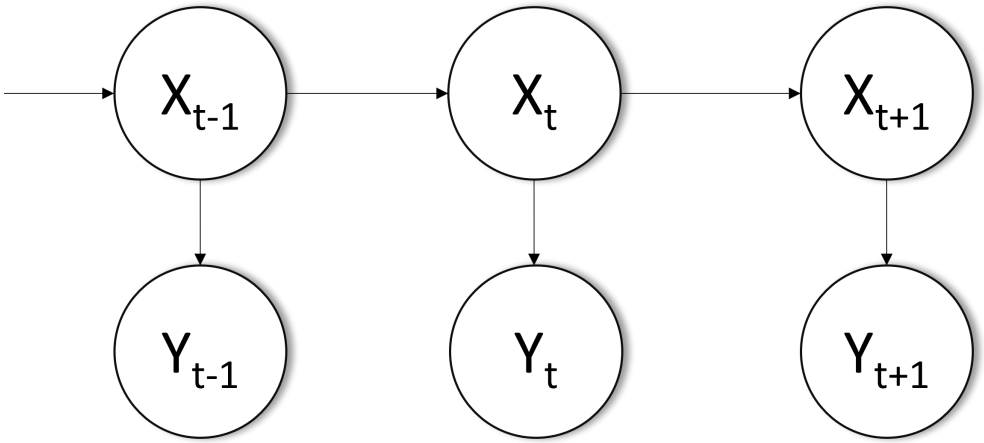
:



3.1. ábra. Rejtett Markov-modell, forrás: [7]

A 3.1-es ábrán az X-szel jelölt csúcsok a rejtett állapotok, a megfigyelt állapotokat pedig Y jelöli. Az átmeneteken az A és B mátrixok értékei láthatók. Szekvenciális adatok feldolgozása esetén, mint például a beszédfelismerés, vagy akár a bólintásdetekció, gyakori a rejtett Markov-modell használata.
A rejtett Markov-modell segítségével egy megfigyeltállapot-szekvenciából következtethetünk a legvalószínűbb rejtettállapot-szekvenciára az eloszlás- és állapotátmenet-mátrixok segítségével. Egy T hosszú S^T rejtettállapot-szekvencia $\{X_1, X_2, ..., X_T\}$ valószínűsége egy adott, szintén T hosszú V^T megfigyeltállapot-szekvenciához $\{Y_1, Y_2, ..., Y_T\}$ a következőképpen számolható:

$$P(V^T, S^T) = P(V^T|S^T)P(S^T) = \prod_{t=1}^{T} P(Y_t|X_t) \prod_{t=1}^{T} P(X_t|X_{t-1})$$



3.2. ábra. Példa a rejtett Markov-modell rejtettállapot-szekvencia predikciójára

A legnagyobb valószínűségű rejtett állapot megtalálásához az összes lehetséges rejtettállapot-szekvencia valószínűségét meg kellene határoznunk. A feladat megoldható így is, viszont ez a megközelítés közel sem nevezhető optimálisnak. Erre a problémára nyújt jobb megoldást a Viterbi algoritmus.

3.1.3. Viterbi algoritmus

A Viterbi algoritmus [9] egy dinamikus programozási algoritmus. Nevét alkotójáról, Andrew Viterbiről kapta, aki 1967-ben hozta létre. A Viterbi algoritmust széles körben használják különböző maximumkeresési problémákra, például a digitális kommunikáció területén vagy a jelfeldolgozásban [27]. A Markov-modell kontextusában a Viterbi algoritmus segítségével meg lehet találni egy megfigyeltállapot-szekvenciához a legnagyobb valószínűségű rejtettállapot-szekvenciát.
Adott a T hosszúságú megfigyeltállapot-szekvencia $\{Y_1, Y_2, ..., Y_T\}$, a keresett, szintén T hosszúságú rejtettállapot-szekvencia $\{X_1, X_2, ..., X_T\}$, és a rejtett Markov-modell pedig $\Theta \rightarrow \{A, B, \pi\}$, ahol A az állapotátmenet-mátrixot, B pedig az eloszlásmátrixot írja le. A következő egyenlet egy adott megfigyeltállapot-szekvenciához adja meg azt a rejtettállapot-szekvenciát, amelynek utolsó eleme t időpillanatban az i állapot volt.

$$\omega_{i,t} = max_{X_1, X_2, ... X_{T-1}} P(X_1, X_2, ..., X_T = i, Y_1, Y_2, ..., Y_T|\Theta)$$

Az egyenlet megoldához teljes rekurziót használhatunk. Ha a szekvencia t pillanatban i állapotban volt, és $t-1$ időpillanatban j állapotba lép, illetve az új megfigyelt állapot k, azaz $Y_{t-1} = k$, akkor

$$\omega_{j,t} = max_i(\omega_{t-1}A_{ij})B_{jk}$$

A módszer rekurzívan alkalmazható, visszafelé i állapotból indulva. Az éppen feldolgozás alatt álló elem összes szomszédja közül mindig az ideálisat kell választani. Az optimális elem a t időpontban tehát

$$\arg\max_i(\omega_{j,t-1})A_{ij}$$

A maximális értékeket elmentve megkapjuk a legnagyobb valószínűségű rejtettállapot-szekvenciát.

3.1.4. Baum-Welch algoritmus

A Baum-Welch algoritmus segítségével a rejtett Markov-modell ismeretlen állapotátmeneteinek valószínűségét, tehát az A és B mátrixok értékeit lehet meghatározni, azaz megfigyelések alapján közelíteni. Nevét két megalkotójáról Leonard E. Baum-ról és Lloyd R. Welch amerikai matematikusokról kapta, akik az 1970-es évek elején dolgozták ki a módszert.
Az algoritmus felvázolásához szükséges néhány változó bevezetése [20]. Adott $Y = \{Y_1, Y_2, Y_3, ...Y_T\}$, T hosszú megfigyeltállapot-szekvencia, illetve $\Theta = (A, B, \pi)$ modell, ahol A a rejtett állapotok átmeneti valószínűségét adja meg, B a rejtett és megfigyelt állapotok összetartozásának valószínűségeit, π pedig a kezdeti állapot valószínűségét. Legyen $\alpha_{i,t}$ az a valószínűség, hogy a Θ modell i rejtett állapotban volt t időpillanatban úgy, hogy adott az összes megfigyelt állapot t előtt.

$$\alpha_{i,t} = P(Y_1, Y_2, ..., Y_t, q_t = X_i|\Theta)$$

Az α értéke a forward algoritmussal számítható. Legyen $\beta_{i,t}$ az a valószínűség, hogy a modell i állapotban volt t időpilanatban úgy, hogy adott a t után következő összes megfigyelt állapot.

$$\beta_{i,t} = P(Y_{t+1}, Y_{t+2}, ..., Y_T|q_t = X_i, \Theta)$$

A β értéke a backward algoritmussal számítható. Az α-át előre paraméternek (forward parameter) szokták nevezni, β-t pedig hátra paraméternek (backward parameter). Míg α csak a t pillanat előtt megfigyelt állapotokat használja (a múltat), β ezzel ellentétben a t pillanat utáni megfigyelt állapotokat veszi figyelembe (a jövőt).
Az α és β paraméterek egy fúziójaként legyen $\gamma_{i,t}$ az a valószínűség, hogy a modell t időpillanatban i rejtett állapotot vesz fel úgy, hogy adott az összes megfigyelt állapot t előtt és t után egyaránt.

$$\gamma_{i,t} = P(q_t = X_i|X, \Theta)$$

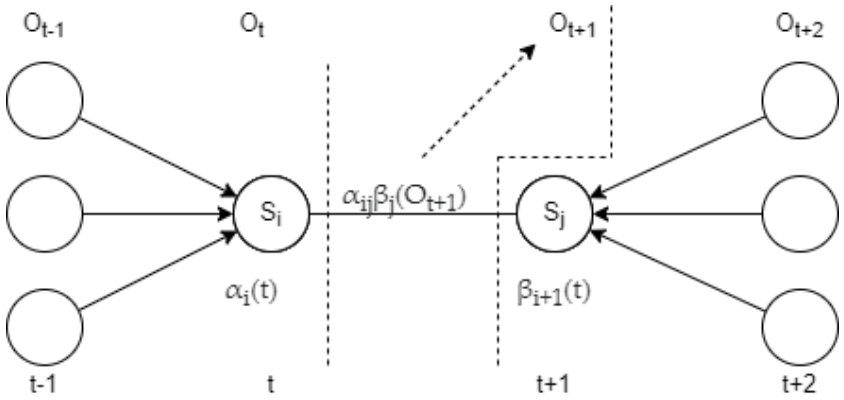
Az α és β felhasználásával bevezethető a $\zeta_{i,j,t}$, ami azt a valószínűséget adja meg, hogy a modell t időpillanatban az i rejtett állapotot veszi fel, $t+1$ időpillanatban pedig j állapotot úgy, hogy adottak a megfigyelt állapotok.

$$\zeta_{i,j,t} = P(q_t = X_i, q_{t+1} = X_j|X, \Theta)$$

A $\zeta_{i,j,t}$ felírható $\alpha_{i,t}$ és $\beta_{j,t}$ felhasználásával

$$\zeta_{i,j,t} = \frac{\alpha_{i,t} A_{ij} B_{jY_{t+1}} \beta_{j,t+1}}{P(X|\Theta)}$$

$$\zeta_{i,j,t} = \frac{\alpha_{i,t} A_{ij} B_{jY_{t+1}} \beta_{j,t+1}}{\sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_{i,j} A_{ij} B_{jY_{t+1}} \beta_{j,t+1}}$$



3.3. ábra. $\zeta_{i,j,t}$ vizuális megjelenítése [11]

A $\zeta_{i,j,t}$ felhasználásával, kifejezhető $\gamma_{i,t}$. Szummázva az összes valószínűséget, amikor i állapotból bármely szomszédos j állapotba lép át a modell, $\gamma_{i,t}$ értékét kapjuk, tehát hogy mennyi valószínűséggel veszi fel a model t időpillanatban az i rejtett állapot értéket.

$$\gamma_{i,t} = \sum_{j=1}^{N} \zeta i, j, t$$

A $\gamma_{i,t}$ változót az összes t-re szummázva megkapjuk, várhatóan hányszor veszi fel a modell az i állapotot. Hasonló módon, szummázva az összes t értékre $\zeta_{i,j,t}$ értékét, megkapjuk, hogy várhatóan hányszor lép a modell i állapotból j állapotba. A bevezetett kifejezésekkel az A mátrix könnyedén frissíthető egy új szkvencia segítségével, hiszen kiszámítható a várható számossága egy állapotátmenetnek i és j között, illetve a várható számossága annak, hogy a modell az i állapotot veszi fel, a kettőnek a hányada pedig megadja, hogy mi az átmenet valószínűsége. A B mátrix j rejtett és k megfigyelt állapota közötti kapcsolatot pedig úgy kapjuk meg, ha j és k együttállásának várható számát elosztjuk a j várható számosságával. A kezdeti értékek a $\gamma_{i,t}$-vel frissíthetők.

$$\bar{A}_{ij} = \frac{\sum_{t=1}^{T-1} \zeta_{i,j,t}}{\sum_{t=1}^{T-1} \gamma_{i,t}} \qquad \bar{B}_{jk} = \frac{\sum_{t=1,Y_t=k}^{T} \gamma_{j,t}}{\sum_{t=1}^{T} \gamma_{i,t}} \qquad \bar{\pi}_i = \gamma_{i,1}$$

Véletlenszerű kezdeti $\Theta = \{A, B, \pi\}$ értékekkel, a tanító-szekvenciát felhasználva lehet frissíteni a modell paramétereit. A frissítést iteratívan végezve a modell értékei konvergálnak egy lokális optimum felé.

3.2. Mesterséges neurális hálózatok

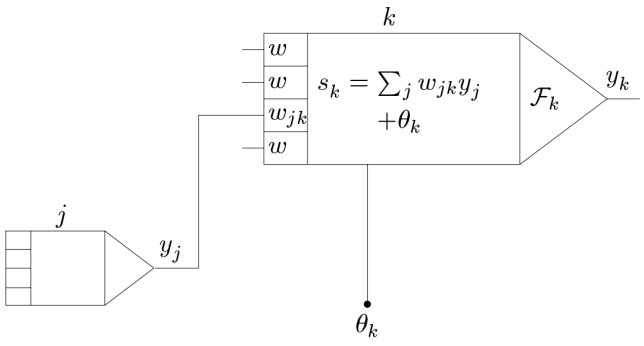
A mesterséges neurális hálózatok (ANN) napjaink egyik legelterjettebb gépi tanulási módszere. Felhasználási területe szinte kimeríthetetlen, használják többek között képfeldolgozásra, adatbányászatra, jelfeldolgozásra, vagy természetes nyelv feldolgozásra. A mesterséges neurális hálózatot a biológiai neurális hálózat analógiájára hozták létre, viszont működésében és tanulásában, taníthatóságában eltér attól. A szakirodalomban a mesterséges jelzőt többnyire akkor használják, ha hangsúlyozni szeretnék, hogy nem a biológiai neurális hálózatról van szó. Az informatika és matematika területén csak neurális hálózatként vagy neuronhálóként hivatkoznak rá.

3.2.1. Neurális hálózatok felépítése

A neurális hálózatok elméletét és topológiáját Ben Krose [17] a következőképpen fogalmazza meg. A neurális hálózatok hasonló típusú, nagy mennyiségű egyszerű műveleti elemet tartalmaznak, melyek összeköttetésben állnak és adatokat közölnek egymással.
A neurális hálózatok fontosabb elemei a következők:

- Műveleti elemek halmaza, avagy neuronok.
- Összeköttetések a neuronok között. Általában w_{jk} értékkel súlyozott, j és k neuronok között.
- Egy propagation szabály, ami kiszámítja a neuron belső értékét (s_k) a különböző bemenetekből.
- Egy transzformációs, vagy aktivációs függvény, ami a neuronok belső értékéből leképzi a kimeneteket.
- Tanuló algoritmus (általában minta alapján való tanulást jelent).

A neuronok emelett rendelkezhetnek állandó értéket szolgáltató bemenettel is. A legtöbb neuron feladata egyszerű, a szomszédoktól és egyéb bemenetektől összegyűjtött adatok alapján kiszámítani a kimenetet, majd továbbítani azt a következő csomópontoknak.



3.4. ábra. A neuron felépítése [17]

A 3.4-es ábrán egy neuron vázlat látható. A Θ_k a neuron állandó bemenete, a többi bemenet felől pedig a szomszédeos neuronok kimenetét kapja, jelen esetben j neuron és annak y_j kimenete van feltüntetve. A szomszédos neuronok kimeneteit a w_{jk} súllyal összegzi, és hozzáadja Θ_k-t. Az így kiszámított belső értéket, s_t-t az y_k függvényen keresztül továbbítja.

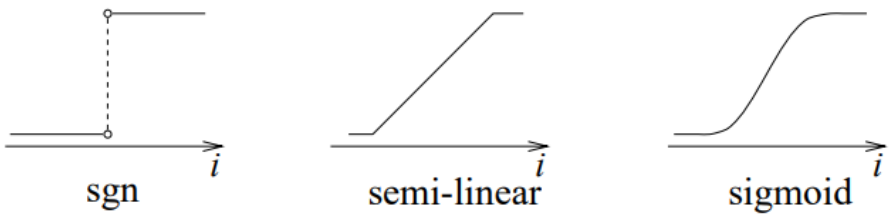
A neuronok működése hasonló, viszont elhelyezkedésük alapján három csoportba lehet őket osztani.

- Bemeneti neuronok: csak a hálózat bemeneteként funkcionálnak.
- Rejtett neuronok: adatfeldolgozó feladatuk van.
- Kimeneti neuronok: csak a hálózat kimeneteként funkcionálnak. (A gyakorlatban nagyon hasonlók a rejtett neuronokhoz.)

A hasonló típusú neuronokat általában rétegekbe (layers) szervezzük. Így beszélhetünk bemeneti, kimeneti és rejtett rétegekről, utóbbiból több is lehet. Egy k neuron belső értékének (s_k) kiszámítása egyszerű, az összes w_{jk}-val súlyozott j szomszédtól származó bemenet összege, tehát a bemenetek lineáris kombinációja, illetve az egyéb bemenetek értékeinek összege.

$$s_k(t) = \sum_j w_{jk}(t) y_j(t) + \Theta_k(t)$$

A bemenetekből a kimenetet tipikusan egy nemlineáris függvény, a transzfer függvény állítja elő a kimenetet. A transzfer függvény leggyakrabban a lépcsőfüggvény (sgn), a ReLU [2], a telítéses lineáris függvény (semi-linear), vagy a sigmoid függvények egyike.



3.5. ábra. Elterjedt transzfer függvények, forrás: [17]

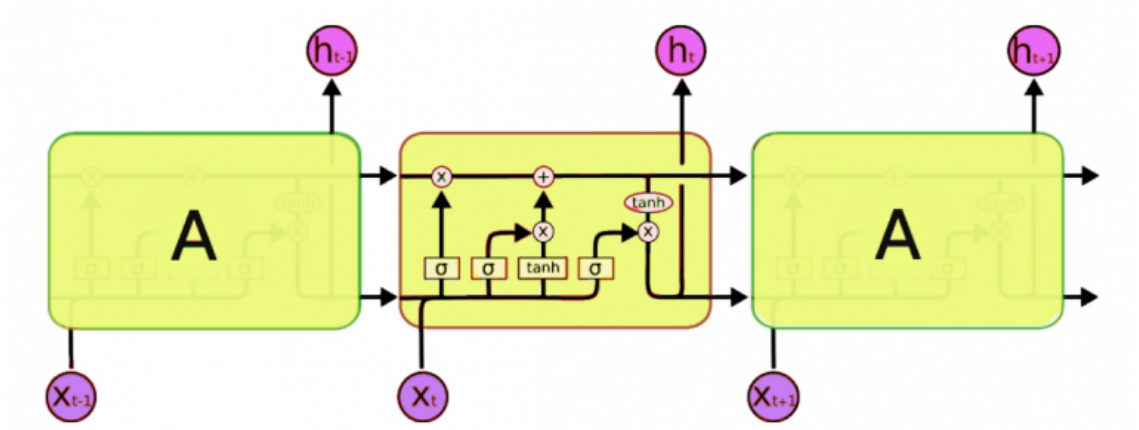
A neurális hálózatokat alapvetően két nagyobb részre lehet osztani a neuronok összeköttetései alapján. Ezek az előrecsatolt és visszacsatolt hálózatok.

- Előrecsatolt hálózatok (feed-forward networks): Az adatok a hálózat bemenetétől a kimenete felé haladnak a rétegeken keresztül, nem haladhatnak visszafelé, illetve egy rétegen belül sem.
- Visszacsatolt hálózatok (recurrent networks): Lokális visszacsatolás esetén megengedett az adat mozgása visszafelé a rétegek között, rétegen belül, sőt, akár neuron szinten is. Globális visszacsatolás esetén a hálózat kimenete visszacsatolható a saját bemenetére.

3.2.2. Elterjedt neurális hálózat architechtúrák

3.2.2.1. LSTM

Az LSTM [12] (Long short-term memory) egy visszacsatolt neurális hálózat architektúra, ami napjainkban igen népszerűnek számít. Az eredeti ötlet Sepp Hochreitertől származott, aki 1997-ben publikálta azt.
Az LSTM a gradiens eltűnés problémájára ad egy megoldást, amely gradiens alapú tanítás közben lép fel. A gradiens adja meg, hogy a hibafüggvény milyen irányba "lejt", ami alapján a hibát csökkentjük minden tanulási iterációban. A visszacsatolás közben a kis gradiens értékek összeszorzódnak és hamar megközelítik a nullát. Ha a gradiens nulla, vagy nulla közeli érték, akkor a hibafüggvényről nehezen eldönthető, hogy milyen irányba érdemes haladni a tanulással. Ennek következtében az első néhány réteg nagyon lassan tanul. Visszacsatolt hálózatok esetén a probléma orvosolható olyan aktivációs függvény alkalmazásával, melynek a deriváltja (meredeksége) nem alacsony, ilyen például a ReLU [2]. Visszacsatolt hálózatok esetében ez nem feltétlenül segít, mert a visszacsatolás közben is eltűnhet ez az érték.
Az LSTM erre a problémára ad megoldást. Az LSTM – hasonlóan az RNN-hez – modulokból épül fel, viszont itt ezek több kapuból álló komplex rendszereket tartalmaznak. Az LSTM újítása a cellastátusz, ami az egész láncon végigmegy és csak kis változtatások történnek rajta a kapuk segítségével. A celláknak három ilyen kapujuk, két bemenetük és két kimenetük van. Bemeneteik az előző cella cellastátusza és az előző cella kimenete, kimeneteik pedig ugyanezek, csak a következő cella számára. A cellák három kapuval rendelkeznek. Ezek a felejtés-kapu (forget gate), input kapu és output kapu. A felejtés-kapu dönti el, hogy a cellastátuszból milyen információnak nem kell belekerülnie a cellastátuszba. Az input kapu dönti el, hogy milyen új adatok kerüljenek bele, az output kapu pedig a kimenetet számítja ki. A 3.6 ábra egy LSTM cella felépítését ábrázolja.

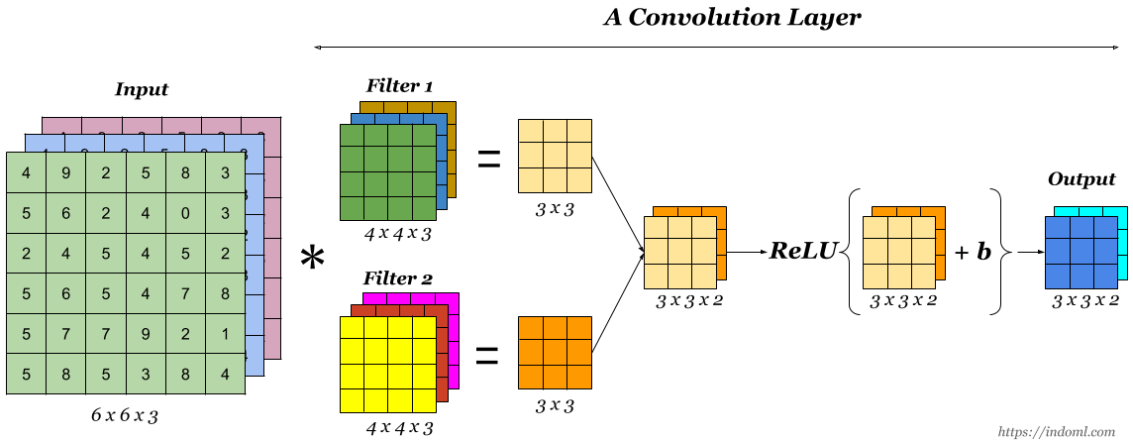


3.6. ábra. Az LSTM cella felépítése, forrás: [23]

A cellastátusz segítségével a hálózat képes régebbi állapotokra emlékezni, így elkerülhető az eltűnő gradiens problémája. Az LSTM nagyon régi konstrukció, de a megjelenése idejében nem kapott nagy figyelmet. Mára az egyik legismertebb visszacsatolt neurális hálózat architektúra. Hasonló, de leegyszerűsített változata a GRU (Gated recurrent units), amit én is használok a CNN-RNN architektúra alapú bólintás-detekció megoldásomban.

3.2.2.2. Jellemzők kinyerése

A jellemzők kinyerése (feature extraction) lényege, hogy a nyers adatot kisebb, kezelhetőbb adattá redukálja úgy, hogy megtartja annak lényeges elemeit.
A képfeldolgozásban nagy teret nyertek a konvolúciós neurális hálózatok, melyek a konvolúció matematikai művelet segítségével hajtanak végre feature extractiont.
A konvolúciós hálózatok, avagy CNN-ek többek között tartalmaznak konvolúciós rétegeket. Ezekben a rétegekben egy konvolúciós ablak segítségével végigiterálnak a kép vagy más bemenetük összes dimenzióján, és konvolválják az ott található értékeket a konvolúciós ablakkal. Az így kapott tenzorok összességét feature mapnek hívjuk. A konvolúció lineáris művelet, ezért a feature mapre (aktivációs térképre) általában alkalmaznak valamilyen nemlinearitást is. A konvolúciós rétegek egymás után helyezkednek el a hálózatban, majd egy összevonó réteg zárja le azokat (pooling layer). Az összevonó réteg valamilyen jellemzőt keres az aktivációs térképeken egy csúszóablakban, és azok alapján csökkenti annak méretét. Az összevonás réteg jellemző metódusai a max pooling (maximális érték keresése az ablakban) és az average pooling (az ablakban található értékek átlaga) [10].
A 3.7 ábrán egy konvoúciós hálózat működése látható.



3.7. ábra. A konvolúciós réteg kimenetének előállításának a bemenete alapján, forrás: [21]

A konvolúciós hálózatokat szinte bármely képfeldolgozási problémára alkalmazzák. Az általam kidolgozott CNN-RNN alapú megoldásban én is ilyet használok.

4. fejezet

Implementáció

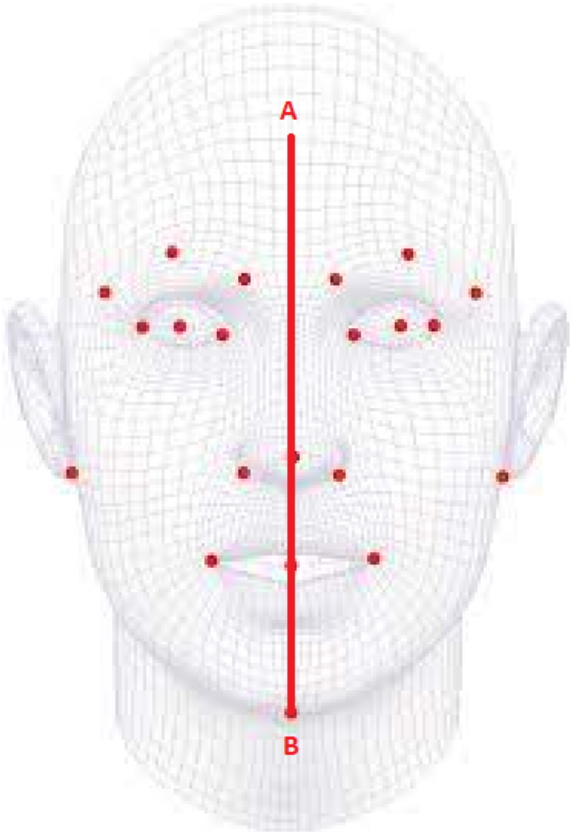
Ebben a fejezetben különböző implementációkat mutatok be a bólintás detektálására. Először egy hagyományos gépi tanulási módszert, majd egy mesterséges neurális hálózat alapú megközelítést. A hagyományos gépi tanulási módszer két részre bontható. A fej mozgásának monitorozása, illetve egy algoritmus, melynek segítségével a fej mozgásából következtetünk a gesztusra.

4.1. Fejmozgás-detekció

A fej mozgásának monitorozására a szakirodalomban leggyakrabban a szem detektálását választják, vagy speciális eszközt alkalmaznak (2.1.2). Az én célkitűzésem az volt, hogy a detekció során csak egy kamera képet fogok használni, ezért a speciális eszközök használatát kizártam. A szem pozíciójának követésével valóban detektálhatóak a nagyobb fejmozdulatok, viszont ez pont a bólintás aspektusában nem szerencsés. Egy bólintás során a fej a horizontális tengely körül forog, így a szem pozíciója nem sokat változik. A kisebb biccentések detekciója így nagyon nehéz feladat. Probléma lehet még a megoldás robosztussága, mivel csak két megfigyelt pont alapján próbáljuk a fej mozgását lekövetni. Én a fej mozgásának detekciójához a mediapipe, Face mesh landmark pontos megoldását használtam [14]. A face mesh egy neurális hálózat alapú rendszer, ami 468 háromdimenziós pontot vetít az emberi arcra, egyszerű kamerakép alapján, akár száz alkalommal másodpercenként. A megoldás tehát robosztus, és gyors is, amivel alkalmas valósidejű gesztus detekcióra. Tapasztalataim szerint a landmark pontok elhelyezése viszonylag pontos amikor nem mozog a fej, viszont gyorsabb mozdulatok esetén kis elcsúszások, hibák észrevehetők. A face mesh képtelen detekcióra, ha az arc egy része takarásban van. Ez általában akkor történik, ha a videó alanya a kezével eltakarja arcát, vagy megtörli azt, bár ezekben az esetekben a bólintás nehezen értelmezhető.
A landmark pontokból a fej mozgására könnyedén lehet következtetni, hiszen két képkocka között a pontok elmozdulása megadja a fej mozgásának irányát, sőt az fps (*képkocka/másodperc*) ismeretében a mozgás sebességét is. A bólintás aspektusában a fej mozgásának azt a komponensét érdemes figyelni, ami horizontális tengely körül történik. Bár nem kizárható, hogy egyéb fejmozgásoknak is van ráhatása a bólintásra, én azt a feltételezést tettem, hogy ezeknek elhanyagolható a jelentősége.

4.1.1. Fejmozgás-detekció 2 pont alapján

A fej mozgásának a horizontális, tehát x tengely körüli forgását legegyszerűbben egy y tengelyen fekvő vektor segítségével lehet mérni. Egy ilyen, vagy ehhez hasonló vektor megkapható valamely állon és valamely homlokon elhelyezkedő pont helyvektorainak különbségeként. A kapott vektort minden képkockán össze lehet hasonlítani a függőleges tengellyel és ki lehet számítani a két vektor által bezárt szöget, amely a fej x tengely körüli elfordulását jelenti.



4.1. ábra. A fej mozgágának detekciója két pont alapján

Legyen A a homlokon, B pedig az állon elhelyezkedő pont, amint az a 4.1 ábrán látható. Az y tengelyen fekvő vektor $(0, 1, 0)$, θ pedig a számított AB vektor és az y tengelyen fekvő vektorok által bezárt szög. A skaláris szorzás tulajdonságából következik a következő összefüggés

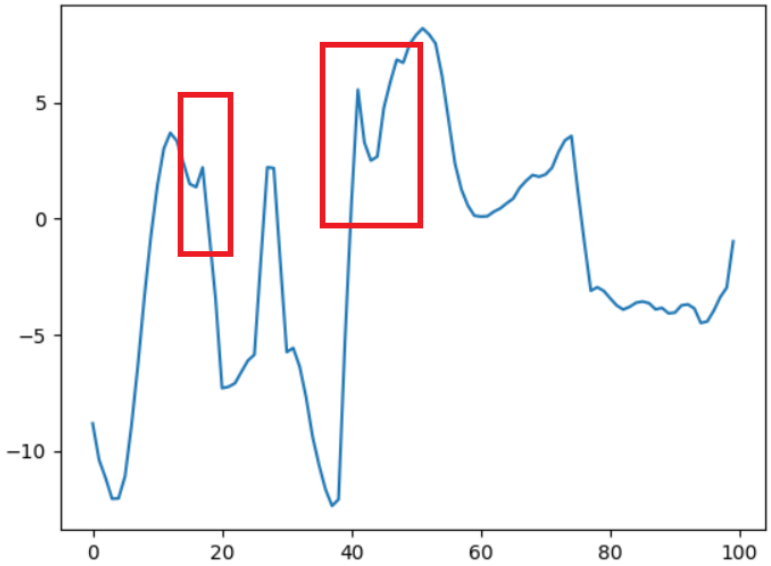
$$(A - B) \cdot (0, 1, 0) = |A - B| \cdot |(0, 1, 0)| \cdot \cos(\theta)$$

Az egyenlet átalakításával a bezárt szög, θ a következőképpen fejezhető ki:

$$\theta = \arccos\left(\frac{(A - B) \cdot (0, 1, 0)}{|A - B| \cdot |(0, 1, 0)|}\right)$$

A detekció ugyanakkor ebben a formában még nem tökéletes. Ha a videón szereplő alany oldalra dönti a fejét, vagyis nem az x, hanem a z tengely körül forgatja, azt ugyan-

olyan elmozdulásnak mérjük, mint a bólogatásra jellemző irányút. Az $(A - B)$ vektor ebben az esetben x irányba mozdul el, ami nem hasznos a bólintásdetekció szempontjából. A probléma megoldására a számított vektor x komponense elhagyható, így az oldalirányú fejmozgást elhagyjuk és csak a feladat szempontjából releváns elmozdulást vesszük figyelembe.
Így adott a fej x tengely körüli forgása, viszont nem megállapítható, hogy a videó alanya felfelé vagy lelfelé fordította a fejét, hiszen a számított vektor y tengellyel való bezárt szöge ezt nem tükrözi. Az $(A - B)$ vektor harmadik, z koordinátája viszont árulkodó ebből a szempontból. Ha a fej előredől, akkor a homlok pozitív z irányba, ellenkező esetben, pedig negatív z irányba mozdul el. A számított szögnek tehát az $(A - B)$ vektor z koordinátája függvényében lehet előjelet adni, aminek segítségével eldönthető, hogy előre vagy hátra billent a fej.
A fejbillenés irányának meghatározása csak akkor működik helyesen, ha az alany fejével egy szintben van a kamera. Ha a kamera feljebb, vagy lejjebb helyezkedik el, akkor az y tengely nem lesz függőleges, és a fej mozgása nélkül is konstans kitérést fogunk detektálni. A probléma megoldható valamelyest azzal, ha a számított vektor bezárt szögét nem az y-tengelyhez, hanem az első képkockán számított vektorhoz képest mérjük. Ez abban az esetben működik, ha az alany nem mozog a kamerához képest a videó közben. Jobb eredményt kapunk, ha nem az első képkockához, hanem az utolsó néhány másodpercben számított átlagos $(A - B)$-hez hasonlítjuk a jelenlegi vektort, így valamilyen szinten kezelhető a probléma.
A detekcióhoz az adatok vizualizálása nem szükséges, az esetleges problémák megtalálásában viszont segíthet.

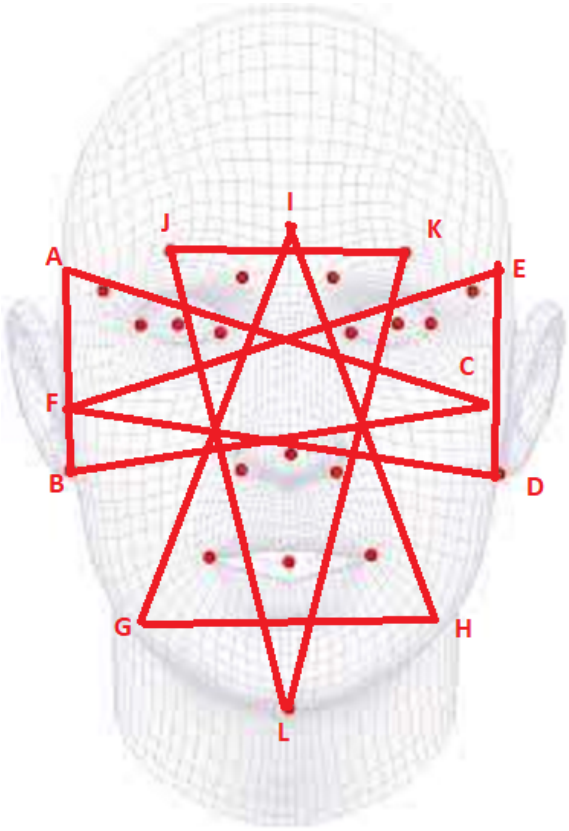


4.2. ábra. A fej mozgásának monitorozása 2 pont alapján

Az ötlet alapvetően jó, viszont a probléma hasonló, mint a szem detekciója esetén lehetne. Két pont ugyanis nem feltétlenül ad elég robosztusságot a pontos detekcióhoz. A 4.2-es ábrán látszik, hogy a 30. és 45. képkocka környékén, ahol a meredekség nagy, tehát a fej mozgása gyors, előfordulhatnak pontatlanságok, melyek a pontok elcsúszásából adódnak. Két pont esetén akár az egyik pont elcsúszása is okozhat olyan pontatlanságokat, amelyek megtéveszthetnék a bólintásdetekciót.
A módszer robosztusabbá tételéhez két pont helyett tizenkét pontot vizsgáltam.

4.1.2. Fejmozgás-detekció 12 pont alapján

A módszer javításánál az alapöletet meghagytam, viszont 2 landmark pont felhasználása helyett 12-őt használtam, így egy-egy pont esetleges elcsúszása, nem okoz akkora pontatlanságot a mért eredményben.



4.3. ábra. A fej mozgásának monitorozása 12 pont alapján

A 12 pont négy háromszögbe rendezhető, ezek az ABC, DEF, GHI és JKL. A 4.3 ábra mutatja ezen háromszögek elhelyezkedését. A háromszögek síkjára állított merőleges, az arcból kifele mutató vektorok eredője az a vektor, ami a 2 pontos megoldásban az AB vektornak feleltethető meg. Az ABC és DEF az x és az y koordinátákra együttesen nagyjából szimmetrikusak. Ez igaz a GHI és JKL háromszögpárra is. Ezen síkokra állított vektorokat felhasználhatjuk arra, hogy az eredőjük alapján meghatározzunk egy, az arcból nagyjából kifelé mutató vektort. Ez a vektor, a vektoriális szorzat segítségével könnyedén kiszámítható. Legyen $k(t)$ ez a vektor a t-edik időpillanatban.

$$k(t) = (A - C) \times (B - C) + (I - H) \times (G - H) + (E - D) \times (F - D) + (K - L) \times (J - L)$$

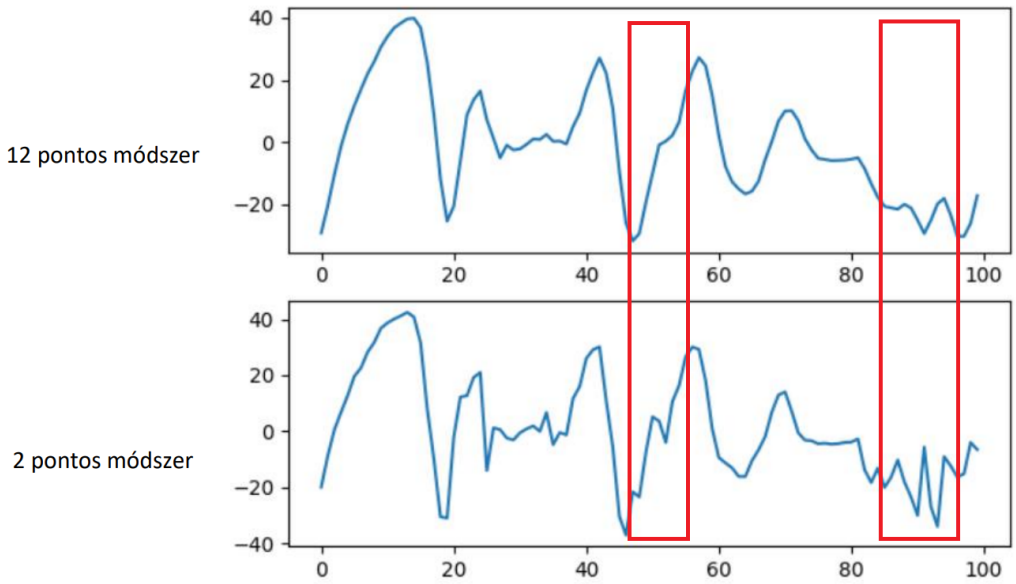
A kamera pozíciójának problémája itt is adott. A z tengelyen fekvő vektorral nem érdemes a számított vektort összehasonlítani, mivel nem garantált, hogy a kamera pont az arc előtt helyezkedik el. Ebben az esetben is egy előre meghatározott n képkocka eredő vektorával (legyen s) mértem össze az aktuális számított vektort, $k(t)$-t.

Az s minden alkalommal újraszámítható az utolsó n darab képkockára számított $k(t)$ segítségével.

$$s = \sum_{i=t-n}^{t} k(i)$$

Az n értéke az fps (*képkocka/másodperc*) függvényében változhat, időbeli értéke a gyakorlatban 3-4 másodpercnek felel meg. A t időpillanatban a fej θ elmozdulási szöge, $k(t)$ és s segítségével a kétpontos módszerhez hasonló módon kiszámítható:

$$\theta = \arccos\left(\frac{k(t) \cdot s}{|k(t)| \cdot |s|}\right)$$



4.4. ábra. A 12 és 2 arc landmark pontot felhasználó módszer összehasonlítása. Pirossal bekeretezve látható a robosztusságbeli különbség a két megközelítés között.

A 4.4-es ábrán látható, hogy a két módszer által mért elmozdulás nagyjából megegyezik. A lokális szélsőértékek helye és értéke lényegében azonos, viszont a meredekebb részeknél, ahol a fej gyorsabban mozog, a 2 pontos módszernek van némi bizonytalansága, pontatlansága. A 12 pontos módszer robosztusabb megoldást nyújt, megtartja a szélsőértékeket, és kisebb pontatlansággal dolgozik.

4.2. Bólintás detektálása a fej mozgásából

A fej mozgásából legtöbb esetben rejtett Markov-modell segítségével detektálják a bólintás gesztusát a szakirodalomban [6, 13]. A bólintást azonban meg lehet próbálni detektálni néhány egyszerű szabály bevezetésével.

4.2.1. Szélsőérték-alapú bólintásdetekció

Egyes bólintások detektálhatók lehetnek néhány, a fej mozgásának nagyságára irányuló szabály bevezetésével. A gesztus megfigyelésével megállapítható néhány általános szabály, melyek igazak a legtöbb nagy vagy egyértelmű bólintásra. A bólintás során a fej lefele és felfele egyaránt mozog. A bólogatás közben a fej mozgásának mértéke nagyobb, mint amikor nem történik bólogatás. Sok esetben a fel és lebillenés nagysága nem tér el szignifikánsan egymástól. A fenti megfigyelések alapján létre lehet hozni egy egyszerű szabályrendszert, aminek segítségével a bólintás detektálható. A szabályok a fej kitérésének nagyságára vonatkoznak, ezért érdemes ezeket egy szekvencián belül a lokális szélsőértékekkel vizsgálni. Legyen Λ egy n hosszú szekvencia, ahol a $\Lambda(t)$ érték a fej elfordulásának szögét tárolja t időpillanatban ($0 \leq t \leq n$). A bólintásra vonatkozó szabályok a következők

- $|max(\Lambda)| - |min(\Lambda)| > const_1$: a szélsőértékek különbsége nagy
- $max(\Lambda) > 0$ és $min(\Lambda) < 0$: a fej előre és hátra is dől
- $\left|\frac{max(\Lambda)}{min(\Lambda)}\right| > const_2$ és $\left|\frac{min(\Lambda)}{max(\Lambda)}\right| > const_2$: A szélsőértékek nagysága nem különbözik szignifikánsan egymástól

A szabályalapú detekció alkalmazása nagyon egyszerű, ha a fej mozgása ismert, viszont az n, a $const_1$ és $const_2$ kostansok meghatározása nem könnyű feladat. A legtöbb bólintás 0.5-1.5 másodperc alatt lezajlik, így érdemes n értékét 1 - 1.5 másodperc körüli értékre állítani. A szélsőérték-alapú bólintásdetekciót a BAUM-1 [28] adatbázis videóin értékeltem ki, az eredményeket a 4.1 táblázatban foglaltam össze.

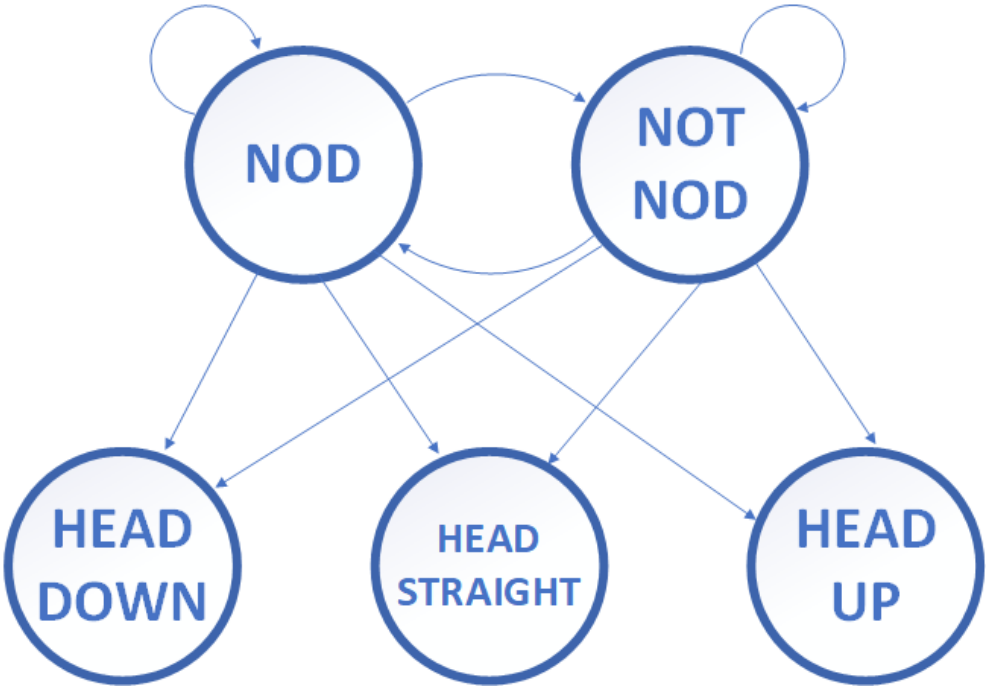
		Predikció	
		bólintást jelzett	nem jelzett bólintást
Valóság	van bólintás	27 db (50.9%)	20 db (37.7%)
	nincs bólintás	6 db (11.3%)	-

4.1. táblázat. A szabályalapú detekció kiértékelésének eredményei

Habár a szélsőérték-alapú módszer a nagyobb, egyértelműbb bólintásokat megtalálta, természetes kommunikációval tesztelve nem teljesített jól. Ennek egyrészt az az oka, hogy a konstansokat nehéz megfelelően beállítani. Másfelől ez a módszer nem képes a kisebb, nagyobb frekvenciájú bólogatásokat megtalálni. A konstansok alacsonyra állításával a detektált gesztusok száma megnő ugyan, de a hamis jelzések ennél jelentősen nagyobb arányban nőnek.
Ez a módszer tehát nem alkalmazható magában bólintás-detekcióra, viszont hibrid bólintás felismerő rendszerben hasznos lehet. A számítási igénye minimális, működéséhez csak néhány konstans tárolása és ezekkel való komparálás szükséges. A konstansokat magasra állítva, nagy bólintás esetén leveheti a terhet a komplexebb, nagyobb számítási kapacitást igénylő rendszerekről.

4.2.2. Rejtett Markov-modell alapú bólintás-detekció

A szélsőérték-alapú bólintás-detekció nem használ fel a fejmozgás nagyságától eltérő jellemzőket. A rejtett Markov-modell képes a bólintás egyéb tulajdonságait is figyelembe venni, mint például annak frekvenciája, vagy hossza. A modellt valós adatokon érdemes tanítani, viszont a megfelelő működéshez nem szükséges akkora mennyiségű adat, mint egy neurális hálózat alapú megoldás betanításához. A hálózat kiértékelése nem számításigényes, tehát gyors, így tökéletesen alkalmazható valós idejű problémák megoldására.
A rejtett Markov modell alkalmazásához az alapvetően folytonos értékekkel rendelkező (idő, elmozdulás) probléma releváns változóit diszkrét csoportokba kell osztani. Én egy nagyon egyszerű modellt alkottam meg a bólintás-detekcióhoz. A modellben három megfigyelt és két rejtett diszkrét állapot van. A megfigyelt állapotok: amikor a fej egyenesen helyezkedik el, amikor lefele van hajtva és amikor felfelé. A rejtett állapotok pedig, hogy történt-e bólintás, vagy sem. A fej pozíciójának egy szekvenciáját a modell megfigyelt állapotaira lehet képezni, és így a Viterbi [9] algoritmus segítségével lehet következtetni egy rejtettállapot-szekvenciára, amiből végül a bólintás gesztus megtörténtére.



4.5. ábra. Rejtett Markov-modell architektúra. Felül a rejtett állapotok, alul a megfigyelt állapotok láthatóak.

Kérdés, hogy mi alapján lehet a görbe folytonos értékeit a modell diszkrét állapotaiba sorolni. Hasonlóan a szélsőérték alapú megoldáshoz, itt is konstansokat vezettem be, melyeket határértékként használtam. Ha egy adott időpillanatban a fej billenésének szöge egy adott konstans felett van, akkor a "HEAD UP" állapotba kerül, ha egy másik adott konstans alatt, akkor a "HEAD DOWN" állapotba, és ha a kettő között, akkor pedig a "HEAD STRAIGHT" megfigyelt állapotba kerül. Az állapotok, és az ezek közötti átmenetek a 4.5 ábrán láthatóak. Az állapotok közötti átmenetek valószínűségeit, tehát az A és B mátrixok értékeit a Baum-Welch algoritmus [4] segítségével lehet meghatározni, betanítani. A modell tanítására a BAUM-1 [28] adatbázis videóit használtam.

4.2.2.1. BAUM-1 adatbázis

A BAUM-1 [28] adatbázis videókat és ahhoz tartozó hanganyagokat tartalmaz. Alkotói azért hozták létre, hogy intelligens rendszereket taníthassanak rajta. Az adatbázis címkézése 8 mentális állapotot különböztet meg. Ezek a boldogság, szomorúság, félelem, düh, undor, bizonytalanság, unottság és kíváncsiság. A videó alanyainak egy meghatározott helyzetbe kellett magukat helyezni, amely kiváltja a fent leírt mentális állapotok egyikét. Az adatbázis alanyai 19 és 65 év közötti török nők és férfiak. Az adatbázis 1184 videót tartalmaz, ezek közül mindegyik annotálva van a 8 mentális állapot egyikével. A felvételeket zöld háttér előtt vették fel, így a környezet nem okoz felesleges zajt a képkockákon. A 4.6 ábra néhány példát mutat az adatbázisban szereplő résztvevőkre.

4.6. ábra. A BAUM-1 adatbázis alanyai

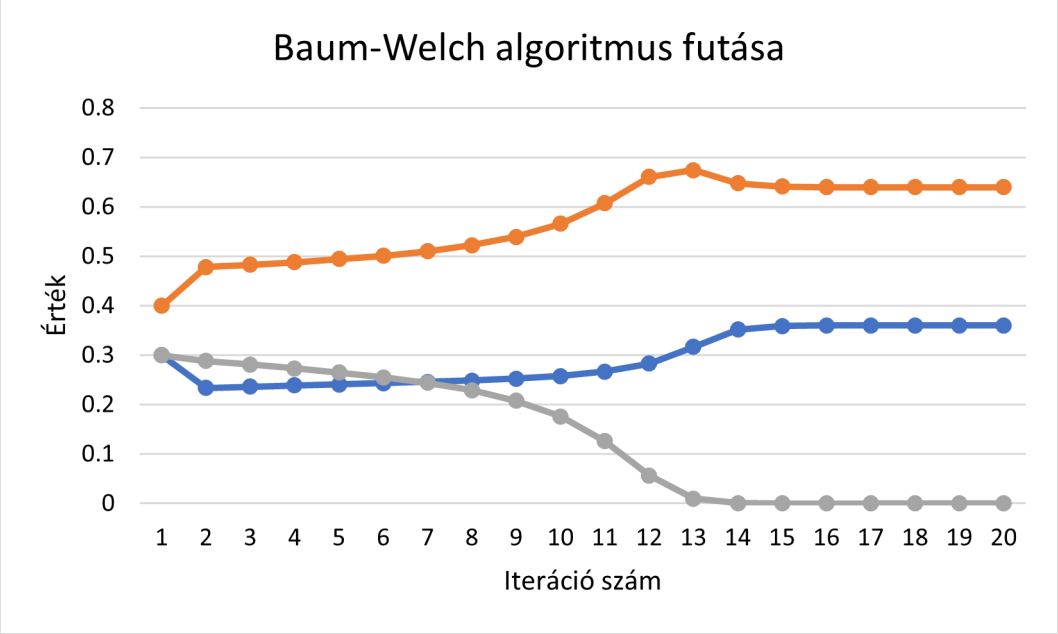
A BAUM-1 adatbázis bólintásra nincs annotálva, így neurális hálózat tanítására nem tudtam használni. Az általam implementált hagyományos gépi tanuló módszerekkel készült bólintásdetekciós rendszereket a BAUM-1 adatbázison teszteltem. A rejtett Markov-modell alapú megoldást ezen az adathalmazon tanítottam, saját címkézéssel.

4.2.2.2. BAUM-1 címkézése

A rejtett Markov-modell tanításához olyan tanító adathalmazra van szükség, amely rejtett állapotok sorozatát tartalmazza. Bólintásra annotált adathalmazhoz egyébként is nehéz hozzájutni, de a rejtett Markov-modell tanításában ez sem segítene, hiszen az állapotátmenetek betanításához nem ilyen típsú annotáció szükséges. A modell állapotátmeneteinek betanítására ezért a BAUM-1 adathalmaz videóinak egy részét címkéztem fel.
A címkézéshez olyan videókat választottam, amelyekben volt bólintás vagy bólogatás. Összesen 27 videót dolgoztam fel, melyekben 17 különböző diák szerepelt. A videókban ezredmásodperc pontossággal megjelöltem, hogy hol vált a fej a "HEAD UP", "HEAD DOWN" és "HEAD STRAIGHT" megfigyelt állapotok között, illetve jelöltem, hogy ha az adott pillanatban észlelhető-e bólintás vagy sem. Az így kapott adatokat egy script segítségével átalakítottam úgy, hogy minden képkockára adott legyen egy rejtett és egy megfigyelt állapot.
A videók átlagosan 6.5 másodperc hosszúak voltak, mindegyikben volt legalább egy bólintás és 30 képkockát tartalmaztak másodpercenként. Az egyes videókból kinyert adatot egymás után fűztem és végeredményként 5286 képkocka hosszú tanító szekvenciát állítottam elő.

4.2.2.3. Rejtett Markov-modell tanítása

A rejtett Markov-modell a Baum-Welch algoritmus [4] segítségével lehet tanítani. Ez az átmeneti valószínűség mátrixok, A és B értékeinek meghatározását jelenti. A BAUM-1 [28] adathalmaz egyes videóinak modellspecifikus felcímkézésével egy 5286 hosszúságú tanító szekvenciát hoztam létre. Érdemes megemlíteni, hogy a Baum-Welch algoritmus használható akár több tanító szekvenciával, ekkor azokat minden iterációban egymás után használja fel. Az így kapott algoritmus lényegében ekvivalens azzal, ha egy hosszú tanító szekvencia lenne. Jelen esetben egy tanító szekvencia mellett döntöttem. Az adatokkal kapcsolatban fontos megjegyezni, hogy a rejtett Markov-modell tanításához nincs szükség akkora tanító adathalmazra, mint például egy neurális hálózat alapú megoldás esetében, illetve az is fontos, hogy az adatok megkeverésének itt nincs jelentősége, ugyanis az algoritmus nagyrészt eloszlásokkal számol, tehát a videók sorrendjének a tanító szekvencián belül nincs jelentősége. A videókon belüli sorrendiséget természetesen tartani kell. A Baum-Welch algoritmus használatához a tanító szekvencia mellett meg kell határozni az A és B átmenetvalószínűségi mátrix kezdeti értékeit, illetve egy kezdeti rejtett állapot valószínűséget és az algoritmus iterációinak számát.
Az A és B márixok kezdeti értéke lehet véletlenszerű. Ha az inicializációkor megadott kezdőértékek nagyon messze esnek a valós értéktől akkor az algoritmusnak több iterációra van szüksége a megfelelő valószínűségek kiszámításához. A kezdeti rejtett valószínűségek esetén is véletlenszerűen adható érték.
Az iterációk számát a tanító szekvencia nagyságának függvényében érdemes megválasztani.

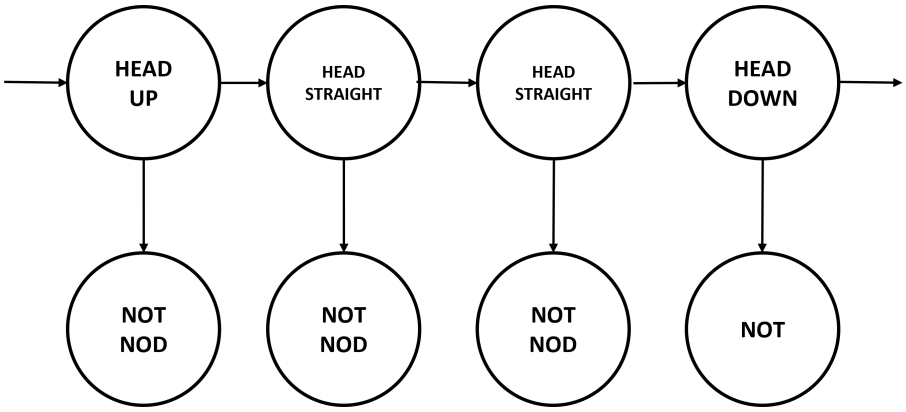


4.7. ábra. A Baum-Welch algoritmus futása.

A 4.7 ábrán a B mátrix egyes értékeinek változása látható. Látszik, hogy a 17-edik iteráció után az valószínűségek felvették végleges értéküket a jelenlegi tanító szekvencia alapján. Ellentétben a neurális hálózatokkal, a rejtett Markov-modell tanítása ekkora adathalmaz esetén maximum néhány másodpercet vesz igénybe.

4.2.2.4. Rejtett Markov-modell kiértékelése

A rejtett Markov-modell kiértékelése során egy megfigyelt állapotszekvencia alapján következtetünk az ehhez tartozó legnagyobb valószínűségű állapotszekvenciára. A megalkotott modell alapján a fej mozgását a három diszkrét állapotra bontjuk. Ezen állapotok egy előre meghatározott hosszúságú szekvenciáját átadjuk a modellnek, ami a sorozat összes eleméhez hozzárendel egy rejtett állapotot, jelen esetben ezeknek értékei a "NOD" (történt bólintás) és a "NOT NOD" (nem történt bólintás). Ezt a szekvenciát a Viterbi algoritmus [9] segítségével lehet kiszámítani. Ezt a metódust pedig a rejtett Markov-modell kiértékelésének (decoding) nevezzük. A 4.8 ábrán látható egy példa a rejtett Markov-modell kiértékelésére.



4.8. ábra. Rejtett Markov-modell kiértékelése. Felül a megfigyelt állapotszekvencia, alul a Viterbi-algoritmus által meghatározott rejtett állapotok láthatóak.

Fontos megállapítani, hogy a kapott szekvenciában egy "NOD" állapot nem feltétlenül jelenti azt, hogy tényleg történt bólintás, mivel az egymást követő tanítómintákban sem egyszer szerepelt a bólintás állapot, amikor a gesztus megtörtént. A bólintás tényleges megtalálásához a teljes szekvenciát érdemes vizsgálni, és bevezetni néhány olyan szabályt, ami a kiszámított n hosszúságú szekvencia alapján eldönti, hogy ténylegesen megtörtént-e a gesztus az előző n képkocka alatt. A szabály bevezetéséhez fontos megállapítani a vizsgált szekvencia hosszát (n), ami szintén egy tervezői döntés. Mivel a bólintás 0.5-1.5 másodperc alatt szinte minden esetben megtörténik, ezért érdemes az n hosszú szekvencia időben mért értékének 1 másodperc körül lennie. Én 0.8 másodperc hosszúságú szekvenciákat vizsgáltam. Ennyi idő alatt a bólintás gesztusok jelentős része megtörténik, viszont hosszabb szekvencia vizsgálata esetén, akár két különböző bólintás részei is bekerülhetnek a feldolgozott sorozatba, ami megnehezíti azok különálló detekcióját. A módszer többféle paraméterbeállítással való kiértékelése és tesztelése alapján a rejtett Markov-modell alapú megoldás akkor teljesített a legjobban, ha legalább három bólintás volt egymás után a számított szekvenciában. Tehát ha az összefüggő bólintások száma meghaladja a k küszöböt ($k = 3$), akkor nagy valószínűséggel bólintás történt. A méréseket 30 fps (*képkocka/másodperc*) mellett készítettem, viszont a képfrissítés mértéke változhat. Az előre meghatározott konstansok (k és n) viszont az fps függvényében is kifejezhetők, így általánosabban lehet a módszert alkalmazni.

- $n := fps \cdot 0.8$
- $k := \frac{n}{8}$

A beállított konstans értékekkel a BAUM-1 adathalmaz [28] 17 videóján végeztem teszteket. Ezeken 10 különböző alany szerepelt. A videókban összesen 36 bólintás történt, melyek közül 30-at jelzett helyesen a módszer. Az eredményeket a 4.2 táblázatban foglaltam össze.

		Predikció	
		bólintást jelzett	nem jelzett bólintást
Valóság	van bólintás	30 db (65.2%)	10 db (21.7%)
	nincs bólintás	6 db (13%)	-

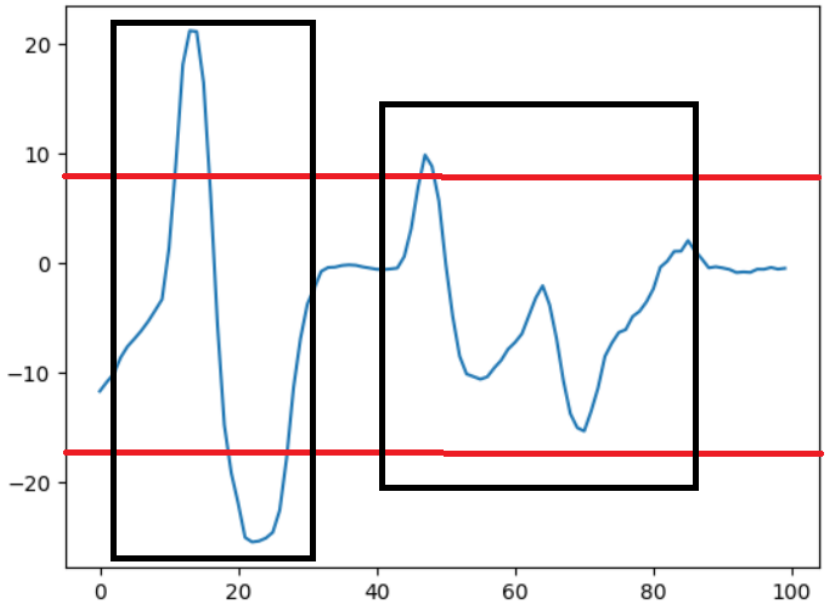
4.2. táblázat. A rejtett Markov-modell alapú detekció kiértékelésének eredményei

A helyesen detektált gesztusok mennyisége alapvetően jó, viszont a téves jelzések száma is igen magas. Ennek oka az, hogy a videóalanyok teljesen más intervallumban mozgathatják a fejüket. Amíg egyes alanyok alig lépnek túl egy 6 fokos intervallumon a bólintást is beleértve, addig mások esetében ez az intervallum elérheti a 20 fokot, bólintással pedig akár 40-45 fokot is. Ez adódhat habitusból, korból vagy kulturális háttérből. Emellett akár ugyanaz a személy különböző témák hallatán, vagy eltérő érzelmi állapotokban is teljesen más nagyságban használhatja ezt a nonverbális gesztust.
Ennek a módszernek a gyengesége tehát hasonló, mint a szélsőérték alapú megoldásnak. Míg ott a fej mozgásának szélsőértékeit hasonlítottuk össze konstansokkal, itt a rejtett Markov-modell diszkrét állapotaiba való beosztás során komparáltunk néhány előre fixált értékkel. Az emberi gesztusokat pedig nehéz leírni néhány konstans segítségével. A konstansokat alacsonyra állítva a detektált gesztusok száma megnő ugyan (TP), viszont a hamis jelzések (FP) mennyisége is megugrik. A konstansok magas értékre állításával csak a legnagyobb bólintások szűrhetők ki, ilyenkor a megtalált gesztusok száma csökken.

A 4.9-es ábrán, feketével vannak bekeretezve a bólintások. Míg az első esetben (0-20) a vártnak megfelelően a bólintás egyes részei külön állapotba kerülnek, a második esetben egy tiszta bólintás nem kerül detektálásra, mivel a hozzá tartozó megfigyelt állapotszekvencia szinte csak "HEAD STRAIGHT" elemekből áll.
A probléma megoldására a rejtett Markov-modell állapothatárait dinamikusan állítottam be.

4.2.3. Rejtett Markov-modell dinamikus állapothatárokkal

A rejtett Markov-modell alapú megoldás legnagyobb problémája az, hogy a konstans állapothatárok segítségével nehéz a fej mozgását megfelelően diszkrét állapotokra bontani. Megfigyelhető ugyanakkor, hogy az egyes videóalanyok egy bizonyos témán belül vagy nagyobb gesztusokat használnak, vagy kisebbeket. Tehát ha egy ember a természetes kommunikáció közben nagy gesztusokat használ, akkor általában a bólintás mértéke is nagynak tekinthető. Ha a kommunikáció közben a fej keveset mozog, akkor valószínűleg a bólintás közbeni fejmozgás is kisebb lesz.
Erre a megfigyelésre építve dolgoztam ki a dinamikus állapothatárokkal rendelkező rejtett Markov-modell alapú megoldást. Ez a módszer lényegében a szélsőérték alapú és a rejtett Markov-modell alapú megoldás egyesítésével jött létre. A bólintás megtalálásához a rejtett Markov-modellt használom, a megfigyelt állapotszekvencia előállításához pedig a fej mozgásának szélsőértékeit.
A diszkrét állapotok határait itt is konstansok határozzák meg, viszont nem egy állapothatárpár van, hanem három. Az alany fejmozgásának utolsó f képkockájában mért minimális és maximális kitérések különbsége határozza meg, hogy mely állapothatárok



4.9. ábra. A rejtett Markov-modell állapothatárai. Feketével vannak bekeretezve a bólintásokhoz tartozó bólintás-szög értékek, pirossal pedig a Markov-modell "HEAD UP", "HEAD STRAINGHT" és "HEAD DOWN" rejtett állapotait határoló küszöbök.

érvényesek rá. A 4.10 ábrán a választott dinamikus határok láthatóak.

Az állapothatárok ebben az esetben különböző gesztusokat használó emberekre eltérőek lehetnek, illetve akár detekció közben változhatnak is, ha esetleg más téma kerülne szóba. Az f értékét 3 másodpercnek megfelelő képkockaszámot adtam.
A módszert ugyanazon a 17 videón végeztem, mint az előző módszernél, az eredmények a 4.3 táblázatban láthatóak.

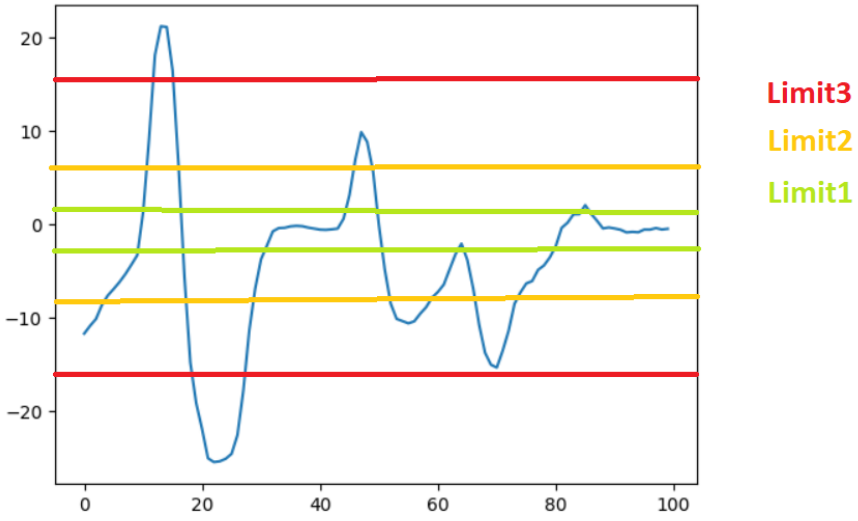
		Predikció	
		bólintást jelzett	nem jelzett bólintást
Valóság	van bólintás	31 db (73.8%)	6 db (14.2%)
	nincs bólintás	5 db (11.9%)	-

4.3. táblázat. A dinamikus határokkal kiegészített rejtett Markov-modell alapú detekció kiértékelésének eredményei

A dinamikus állapothatárok bevezetésével minden téren javult az alkalmazás teljesítménye, viszont a hamis jelzések számán sikerült igazán csökkenteni.

4.2.4. Rejtett Markov-modell alapú megoldások értékelése

A rejtett Markov-modell alapú megoldások valós idejű alkalmazásra teljesen megfelelnek. A felhasználáshoz elég egy webkamera szintű eszköz, a landmarkpontok detekciója másod-



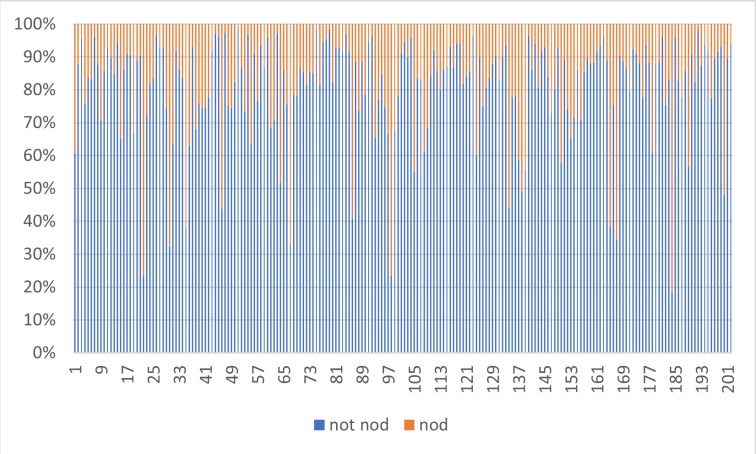
4.10. ábra. A rejtett Markov-modell dinamikus állapothatárai. Zölddel a legkisebb, pirossal a legnagyobb bólintásszög-intervallumhoz tartozó küszöbök intervallumai vannak jelölve.

percenként akár 100-szor is végrehajtható. A fej pozíciójának diszkrét állapotokba sorolása ugyancsak nem jelent problémát, illetve a Markov-modell kiértékelése is a másodperc töredéke alatt elvégezhető. A fejmozgás szekvenciát érdemes FIFO [26] elven tárolni, így képkockánként elég csak egy értéket módosítani. Ezzel a megoldással tehát egy kis számításigényű, könnyedén alkalmazható valósidejű megoldást kapunk, amelyet szinte bármilyen környezetben alkalmazni lehet.

4.3. SEWA adathalmaz

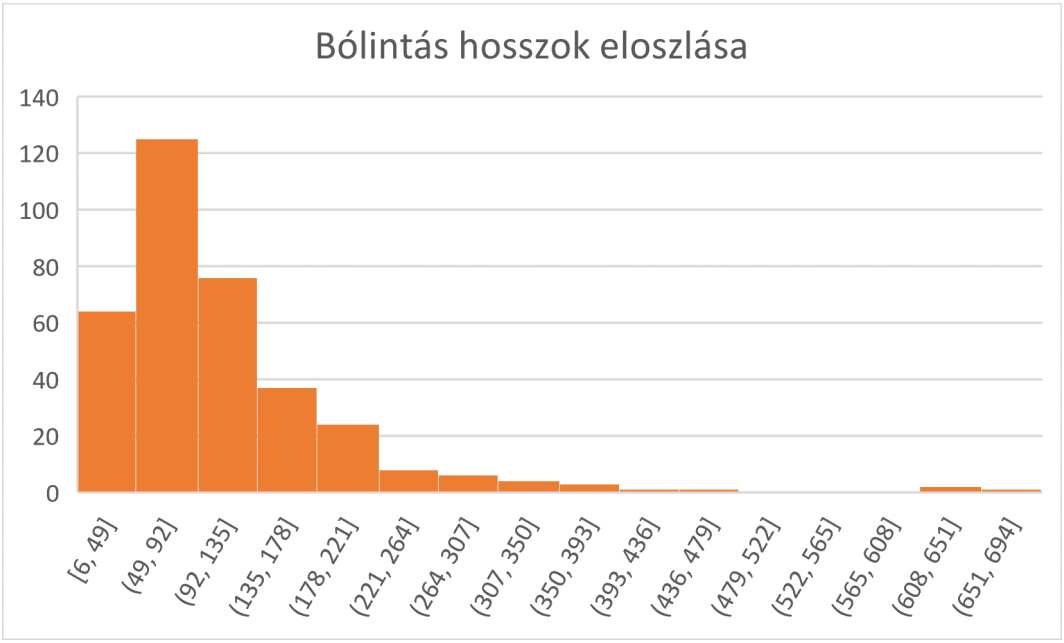
A gépi tanulásról általánosan elmondható, hogy fontos a megfelelő méretű és minőségű adathalmaz kiválasztása, amin a kidolgozott módszer tanítani és/vagy tesztelni lehet. Nincs ez másképp az emberi gesztus-detekció területén sem. A mesterséges neurális hálózat alapú megoldások esetén kritikus kérdés a megfelelő méretű és címkézésű adathalmaz. A SEWA [15], egy címkézett videó és hanganyagokat tartalmazó adatbázis, ami emberi viselkedés analízis problémák megoldásában nyújt segítséget. Az adatbázis alanyai hat különböző kulturális közegbe tartoznak. Angol, német, görög, szerb, kínai és magyar személy egyaránt megtalálható. A videókban szerepelnek nők és férfiak, korukat tekintve 20 és 60 év közöttiek.
Az adatbázisban 1525 percnyi videó- és hanganyag található, melyek különböző érzelmekre és gesztusokra annotálva vannak.

Az adatbázis egyes videói bólintásra is tartalmaznak címkézést, így én is fel tudtam használni a munkámhoz. Összesen 538 videóban annotálták a bólintás jelenlétét, ezekben összesen 352 bólintás annotáció található. A bólintások annotációja képkocka szinten történt. Egy bólintáshoz három adat tartozik: melyik videóban volt a bólintás, melyik volt az első képkocka, ahol a bólogatás kezdődött és melyik volt az utolsó képkockája a bólogatásnak. Érdekesség, hogy az 538 bólintásra annotált videó közül csupán 202 videóban van tényleges bólintás. Ez nem azt jelenti, hogy a maradék 336 videóban nincs hasznos információ, hiszen a bólintás-detekcióban fontos szerepe van annak, hogy mikor biztosan nem történt meg a gesztus.



4.11. ábra. A bólintással annotált szakaszok hosszának aránya a tartalmazó videó hosszához képest.

A videók összesen 500153 képkockából állnak, ebből 39044 képkockán történik bólogatás. A SEWA adatbázis 50 fps (*képkocka/másodperc*) képfrissítéssel rendelkező videókat tartalmaz, így összesen 10003 másodpercnyi videó van a gesztusra címkézve, amelyből 780 másodperc bólogatás. A teljes bólintásra annotált részben a bólintások hányada valamivel 8% alatt van. Azokban a videókban, ahol történt bólintás, ott az idő majdnem 20%-át teszi ki a gesztus. A 4.11 ábrán videóra lebontva látható a bólintással annotált szakaszok hosszának aránya a videó hosszához képest. A bólintások átlagosan 110.92 frame hosszúságúak, ami 2.2 másodpercnek felel meg.



4.12. ábra. A bólintással annotált szakaszok hosszának (képkocka) eloszlása

A bólintás hosszak mediánja 89, ami 1.78 másodpercnek felel meg, viszont az adathalmazban található 6 hosszúságú bólogatás és 683 képkocka hosszú bólogatás is. A 6

képkocka hosszúságú bólintás 0.12, a 683 hosszúságú pedig 13,66 másodpercnek felel meg. A 6 hosszúságú bólintás esetén valószínűleg hibás adatról beszélünk, a 683 hosszúságú bólintás pedig valószínűleg nem egy gesztust, hanem egy bólogatás sorozatot takar, tekintve, hogy egy bólintásról nagyjából 1, de maximum 1.5 másodperc után megállapítható, hogy megtörtént a gesztus.
A 4.12 ábrán látszik, hogy nem csak az egy hosszú bólogatás sorozat került be az adatok közé, hanem több, egyértelműen bólogatás sorozatot tartalmazó címkézés készült. Ez a típusú annotáció a bólintás-detekció szempontjából nem szerencsés, hiszen ilyen hosszú, akár 12 másodpercen felüli bólogatás nem történik sokszor a gyakorlatban. Az adathalmaz felhasználásánál azzal a feltételezéssel éltem, hogy az ilyen hosszú bólogatás sorozatok bármely 1-1.5 másodperces részéből kiderül, hogy megtörtént a gesztus, és így akár különálló bólogatásokként is tudom kezelni az ilyen sorozatokat.

4.4. CNN-RNN architektúra alapú megoldás

A CNN-RNN architektúra két részből épül fel: egy konvolúciós neurális hálózatból, illetve egy visszacsatolt neurális hálózatból. A két neurális hálózat együtt alkot komplex rendszert, amit leggyakrabban videó klasszifikációra használnak. A CNN-RNN architektúra mellett CRNN-ként (convolutional recurrent neural network) is szoktak rá hivatkozni. Az implementációkban gyakran LSTM vagy GRU visszacsatolt hálózatokat használnak, így a CNN-LSTM és a CNN-GRU megnevezésekkel is lehet találkozni, viszont az alap ötlet lényegében azonos. A videó képkockáin a konvolúciós hálózat egyenként jellemző kinyerést (feature extractiont) hajt végre, tehát kiemeli belőle a fontos jellemzőket. A képkockákat egy háromdimenziós tenzorként veszi át, és a kiemelt részleteket egy egydimenziós feature vektorban adja vissza. A videó képkockából kiemelt feature vektorokat egyben adjuk át a visszacsatolt hálózatnak, így a videó képkockáinak sorrendje is megmarad. A jellemzők kinyerésével a képek dimenziója csökkenthető és kevesebb, de hasznos adat kerül továbbításra a klasszifikációra.
A bólintás gesztus felismerése is egy fajta videó klasszifikációs probléma, így alkalmazható rá a CNN-RNN architektúra alapú megoldás. Bólintásdetekció esetében a videókat két osztály egyikébe kell sorolni, tehát bináris osztályozásról van szó. Az egyik osztály az amikor történt bólintás, a másik pedig amikor nem történt. A megoldásomban a konvolúciós neurális hálózatnak, a Keras [1] előre tanított CNN-jei közül a MobileNetV2 [22], a NASNetMobile [29] és az InceptionV3 [24] hálókat használtam. Visszacsatolt hálózatként egy 7 rétegű modellt hoztam létre.

4.4.1. Adathalmaz előkészítése

A neurális hálózatok tanításának egyik legfontosabb eleme az adatok hálózatba való betáplálása. A videó klasszifikáció tanításához az összes osztályból nagyjából hasonló mennyiségű videó szükséges. A bólintásdetekció aspektusában ez azt jelenti, hogy kellenek azonos hosszúságú bólintást biztosan tartalmazó és bólintást biztosan nem tartalmazó videók.
Kérdés, hogy mekkora legyen a videók hossza. A legtöbb bólintás 1-1.5 másodperc alatt történik, így nem érdemes sokkal hosszabb videókat vizsgálni. Az adathalmazzal kapcsolatban feltételeztem azt, hogy egy hosszabb bólintássorozat végig bólogatásokat tartalmaz, tehát bármely kivágott része külön is értelmezhető bólintásnak. Ezt kihasználva a hosszabb annotált részeket kisebbekre tudom vágni és így fel tudom őket többször is használni a tanításhoz illetve teszteléshez.
A vizsgált videó hosszának meghatározása nem egyszerű feladat, mivel mind a rövidebb, mind a hosszabb videóknak megvan a maga előnye, illetve hátránya. A rövidebb videók mellett szól, hogy ebben az esetben a rendelkezésre álló adatbázisból több minta állítható elő. A hosszabb videók mellett az szól, hogy a több ideig tartó gesztusok is belefférnek, így mindig detektálni lehet azokat. Úgy döntöttem, hogy három különböző videóhosszal próbálom ki a CNN-RNN architektúrát.

- 60 képkocka: 1.2 másodpercnek felel meg, összesen 447 darab minta
- 50 képkocka: 1 másodpercnek felel meg, összesen 583 darab minta
- 40 képkocka: 0.8 másodpercnek felel meg, összesen 651 darab minta

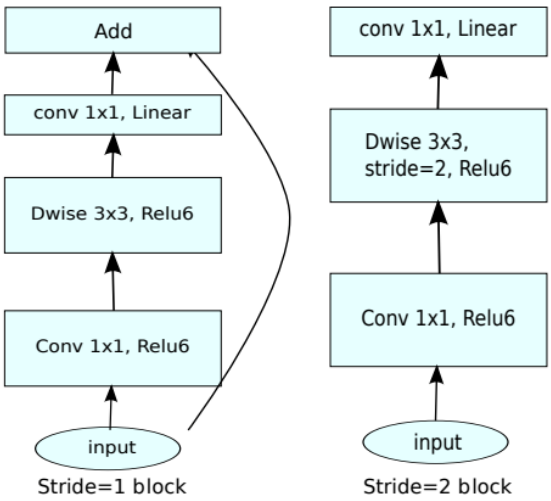
A bólintást tartalmazó videók mellé nagyjából ugyanennyi videót vágtam, amely azonos hosszúságú és garantáltan nem található benne bólintás. A neurális hálózat tanításakor az adatok 20%-át használtam tesztelésre, 80%-át pedig tanításra.

4.4.2. Konvolúciós hálózatok jellemzők kinyerésére

A CNN-RNN architechtúrában nagy szerepe van a konvolúciós hálózat kiválasztásának, hiszen ez dolgozza fel az adatot a visszacsatolt hálózat számára. Ha a kiszűrt részletek nem elég jók, az RNN rész sem tud megfelelően teljesíteni.
Valósidejű CNN-RNN alkalmazás esetén különösen fontos a konvolúciós hálózat sebessége, mivel működés közben minden képkockára ki kell értékelni. A konvolúciós hálózat kiválasztása egyfajta kompromisszum a sebesség és a teljesítmény között. Nagy számítási kapacitás birtokában lehet összetett, pontos hálózatot választani, viszont valósidejű alkalmazások esetében fontos a sebesség, ami viszont az eredmény romlásával járhat.
Az implementációban három különböző előre tanított konvolúciós hálózatot használtam a jellemzők kinyerésére, majd ezek eredményét hasonlítottam össze. Az általam kipróbált konvolúciós hálózatok kis számítási kapacitást igényelnek, ezért potenciálisan beépíthetők egy valósidejű alkalmazásba.

4.4.2.1. MobileNetV2

A MobileNetV2 [22] egy konvolúciós neurális hálózat architektúra, amit a Google kutatói fejlesztettek ki 2017-ben. Az architektúra célja, hogy alacsonyabb teljesítményű eszközökön, mobilokon, egyes beágyazott rendszereken is tudjon teljesíteni. CPU-n nagyjából 25.9 miliszekundum alatt értékel ki egy képet. Gyorsaságához képest igen jól teljesít, így valós idejű alkalmazások felhasználásához alkalmas lehet. A hálózat architektúrája a 4.13 ábrán látható.

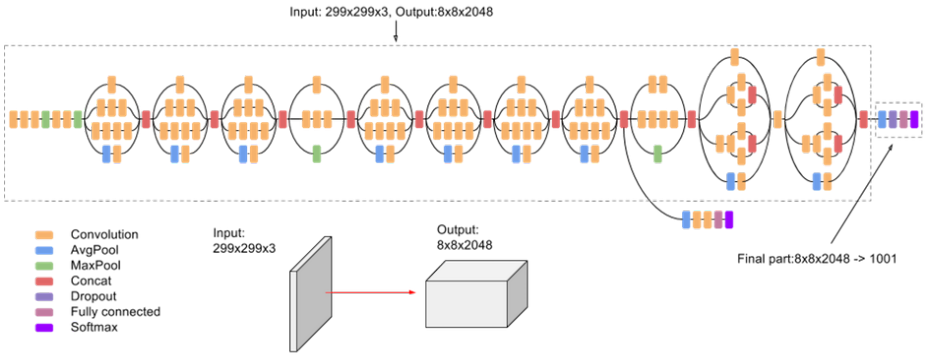


4.13. ábra. MobileNetV2 architektúra, forrás: [22]

4.4.2.2. InceptionV3

Az InceptionV3 [24] egy konvolúciós neurális hálózat architektúra, ami az ImageNet [8] adatbázison tesztelve 78.1% pontosságot ért el. Szimmetrikus és aszimmetrikus blokkokból épül fel, melyek között vannak konvolúciós, average pooling, és max pooling rétegek. Sebességét tekintve nem olyan gyors, mint a MobileNetV2: egy kiértékelési lépés nagyjából 42.2 miliszekundumot vesz igénybe CPU-n futtatva. Teljesítmény szempontjából viszont

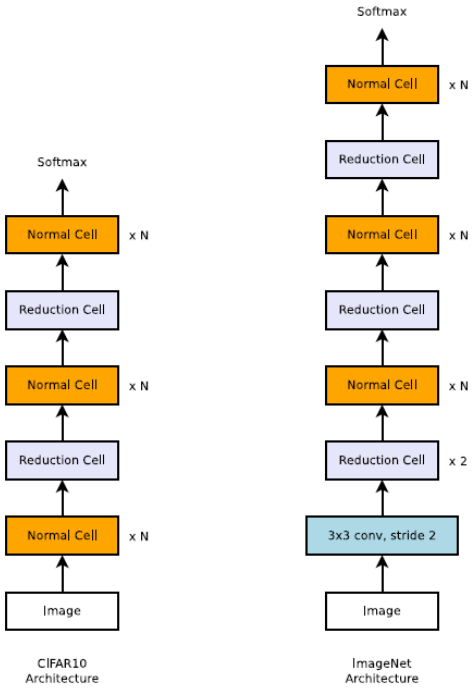
jobb annál. Az InceptionV3 a képből 2048 feature-t emel ki. A hálózat achitektúrája a 4.14 ábrán látható.



4.14. ábra. InceptionV3 architektúra, forrás: [24]

4.4.2.3. NASNetMobile

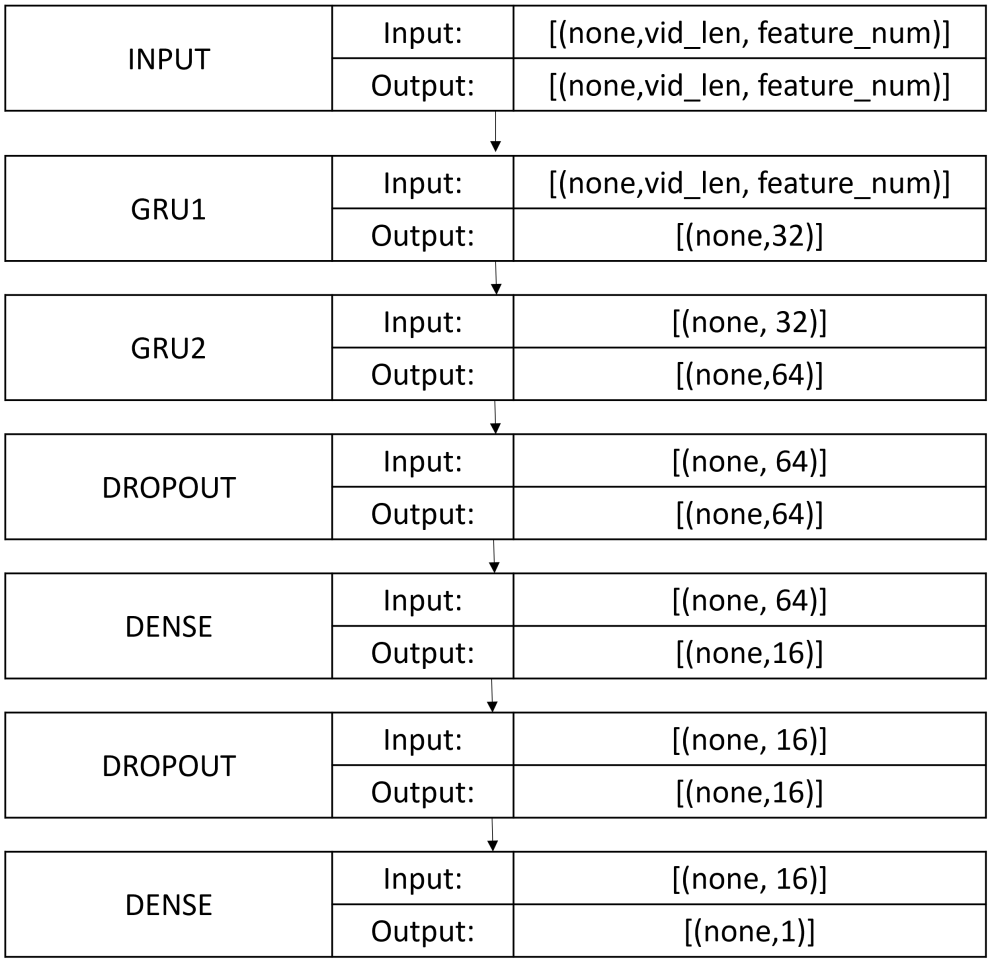
A NASNetMobile [29] egy konvolúciós neurális hálózat architektúra, amit a Google fejlesztői dolgoztak ki. A NASNetMobile a NASNet egy könnyített változata, célja – a MobileNetV2-höz hasonlóan – a gyors működés, akár telefonkészülékeken is; CPU-n futtatva nagyjából 27.2 milliszekundum alatt hajt végre egy lépést. A NASNet (Neural Architecture Search Network) megalkotásánál a konvolúciós blokkok architektúráját egy kis adathalmazra (CIFAR-10 [16]) optimalizálták, majd ezt általánosították egy nagy adathalmazra (ImageNet [8]) a blokkok ismétlésével. A hálózat architektúrája a 4.15 ábrán látható.



4.15. ábra. NASNetMobile architektúra, forrás: [29]

4.4.3. Visszacsatolt neurális hálózat modell

A CNN-RNN architektúra RNN, vagyis visszacsatolt neurális hálózat részeként egy egyszerű szekvenciális modellt alkottam meg. Bemenetként az elemzett videó összes képkockájához tartozó jellemzőket kapja meg, kimenetként pedig a bólintás megtörténtét adja vissza. A modell legfontosabb eleme a két GRU (Gated Recurrent Unit) réteg, melyek 32 és 64 egységgel rendelkeznek, aktivációs függvényként pedig ReLU-t használnak. A hálózat mélyítése érdekében egy 16 egységgel rendelkező teljesen összekötött réteget adtam a modellhez. A túltanulás érdekében két dropout réteg került be még a modellbe, amelyek relatív gyakoriságának értékét 0.2-re állítottam, a kimeneti réteg aktivációs függvényének pedig softmax-ot használtam. A megtervezett visszacsatolt neurális hálózat architektúrája a 4.16 ábrán látható.

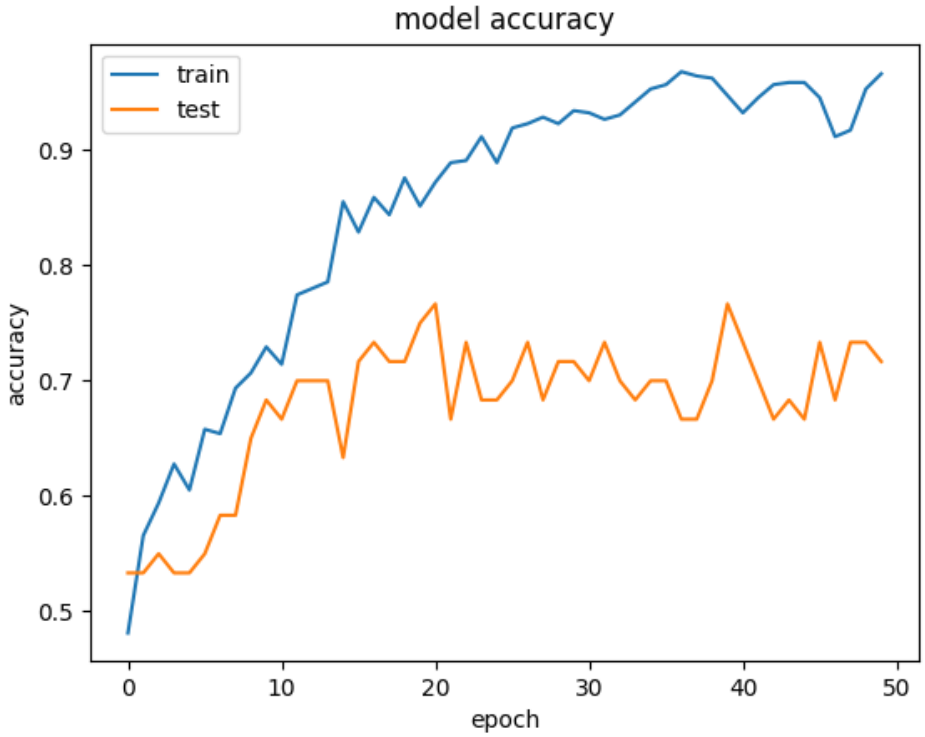


4.16. ábra. A megtervezett visszacsatolt neurális hálózat architektúrája.

4.4.4. A CNN-RNN modell tanítása

Az előző alfejezetben bemutatott modellt három különböző konvolúciós hálózattal tanítottam be. Az InceptionV3, MobilNetV2 és NASNetMobile konvolúciós hálózatok mind relatíve kis teljesítményigénnyel rendelkeznek, ezért alkalmasak lehetnek akár valósidőben működni egy bólintás-detektáló rendszer keretein belül. A vizsgált videó hosszánál is

különböző értékeket próbáltam ki: megvizsgáltam, hogyan teljesít a modell 40, 50 illetve 60 képkocka hosszú videók esetén. A tanításhoz használt adatokat előre feldolgoztam úgy, hogy adott hosszúságú bólintást tartalmazó és nem tartalmazó videókat kivágtam a teljes adathalmazból. Figyeltem arra, hogy nagyjából ugyanannyi bólintást tartalmazó videó legyen, mint amin nem történik meg a gesztus. A fájlokat megkeverve, 4-1 arányban használtam tanító, illetve tesztelő mintaként.
A videók képkockáit egyesével a konvolúciós hálózatba tápláltam, a kimenetekből pedig összeállítottam a tenzort, ami a visszacsatolt hálózat bemeneteként szolgál.
A tanítást 50 iteráción keresztül végeztem, a batch méretet pedig 16 egységre állítottam. A tanításhoz ADAM optimalizációt használtam.
A visszacsatolt hálózat tanítása nem volt sok idő, viszont az összes képkockára futtatott konvolúciós hálózat lefutása jelentős időbe telt. A fejmozgás detekció alatt megfigyeltem, hogy a fej 30 fps esetén sem mozog sokat két képkocka között, ezért a tanítási idő lecsökkentése miatt, a másodpercenkénti 50 képkocka helyett inkább csak 25-öt vizsgáltam. Ez jelentősen felgyorsította a konvolúciós hálózat futási idejét, és az eredményben nem jelentkezett szignifikáns különbség.



4.17. ábra. NasNetMobile pontossága 40 képkocka hosszúságú szekvenciákon való tanulás közben

A 4.17 ábra a NASNetMobile alkalmazásával tanított CNN-RNN rendszer, tanítás során elért pontossága látszik 40 képkocka hosszúságú videók vizsgálata közben. Az ábrán a teszt és a tanítási adathalmazokon elért pontosság van feltüntetve a tanítási iterációk függvényében.

4.4.5. A CNN-RNN architektúra eredményei

A CNN-RNN architektúra betanítása után a következő eredmények születtek.

Cnn/Frame num	40	50	60
InceptionV3	77.8%	74.12%	72.11%
MobileNetV2	74.8%	72.79%	68.23%
NASNetMobile	69.11%	67.23%	67.12%

4.18. ábra. A CNN-RNN architektúra eredményei

Az eredményekből jól látszik, hogy a várakozásoknak megfelelően a InceptionV3 konvolúciós hálózat szerepelt a legjobban (ahogy az a 4.18 táblázatban látható). Az is észrevehető, hogy 40 frame vizsgálata közben születtek a legjobb eredmények. Valószínűleg jobb eredmények születtek volna, ha a bólintások pontosan lettek volna annotálva, ugyanis egy 12 másodperc hosszúságú bólogatás sorozatban előfordulnak olyan intervallumok, ahol igazából nem történik meg a gesztus. Ezek a hibák mind rombolják a hatékonyságot, ahhoz pedig nincs elég adat, hogy a jelentőségük minimális legyen.

4.4.6. A CNN-RNN architektúra értékelése

A CNN-RNN architektúra alapú megoldást folyamatos megfigyelésre is lehet alkalmazni, nem csak egy-egy videó klasszifikálására. Folyamatos detekció esetén, a kiválasztott hosszúságú képkocka szekvenciát FIFO [26] elv-szerűen érdemes kezelni. Minden képkockára el kell végezni a jellemzők kinyerését, majd frissíteni a várakozási sor értékeit. Minden új képkockával tehát csak egyszer kell a konvolúciós hálózatnak dolgoznia, az elmentett adatokból pedig előállítható az a tenzor, amire a visszacsatolt neurális hálózat prediktálni tud. A CNN-RNN valósidejű alkalmazása esetében nem olyan egyszerű a helyzet, mint a Markov-modell alapú megoldásoknál. A konvolúciós hálózatok megfelelő GPU segítségével gyorsan lefutnak, de tagadhatatlan, hogy kisebb teljesítményű eszközökön, esetleg CPU-n futtatva nem olyan gyorsak. A helyzet valamelyest javítható, ha csak minden második képkockát vesszük fel a várakozási sorba. Ezzel a valósidejű kiértékelés valamivel gyengül, mivel előfordulhat, hogy lesz késleltetés a rendszerben. Emellett fontos, hogy kisebb számítási kapacitással rendelkező eszközön a valamivel pontatlanabb, de jelentősen gyorsabb MobileNetV2 konvolúciós hálózatot érdemes alkalmazni.

4.5. Bólintás detekciós módszerek összehasonlítása

A dolgozatban két módszert mutattam be a bólintás gesztus felismerésére. Az egyik hagyományos gépi tanuló módszer, ami rejtett Markov-modellt használ a bólintás detektálására. A másik egy hibrid neurális hálózat alapú módszer, ami a CNN-RNN architektúra alkalmazásával készült.
A rejtett Markov-modell alapú megoldás előnye, hogy valós időben alkalmazható szinte bármilyen eszközön. A fej mozgásának monitorozására használt landmark pont detektor akár 100 alkalommal is ki tud értékelődni másodpercenként. A rejtett Markov-modell kiértékelése szintén nagyon gyors, nem igényel nagy számítási kapacitást. A BAUM-1 [28] adatbázison kiértékelve 73.8% pontossággal dolgozott. A rejtett Markov-modell alapú megoldás hátránya, hogy kis adathalmazon lett tanítva, a videókon szereplő alanyok korban és kulturálisan nem tértek el egymástól. Nagyobb adathalmazon tanítva, konstans értékeit helyesen beállítva valószínűleg még lehetne javítani a módszeren.
A CNN-RNN módszer előnye a rejtett Markov-modell alapú megoldáshoz képest, hogy a tanító adathalmazban kulturálisan, korban, nem alapján eltérő alanyok voltak, így a megoldás széleskörben alkalmazható lehet. Hátránya ugyanakkor, hogy a konvolúciós hálózat kiértékeléséhez valós időben számításigényes művelet. Habár a SEWA [15] adatbázison kiértékelve a legjobb eredmény 77.8% pontosság lett, ennek valósidejű felhasználásához megfelelő hardverelem szükséges. MobileNetV2 konvolúciós hálózat használatával ugyanakkor 22-25 *képkocka/másodperc* frissítéssel lehet alkalmazni, ennek a legjobb eredménye 74.8% volt.

5. fejezet

Összegzés

Munkám során valósidejű bólintás detekcióra dolgoztam ki egy hagyományos gépi tanulási módszert és egy neurális hálózat alapú megoldást. A valósidejű megoldásnál a MediaPipe landmark pont detekcióját használtam a fej mozgásának monitorozására. A fej mozgásából a bólintás kiszűrésére rejtett Markov-modellt használtam, amit a Baum-Welch algoritmus segítségével tanítottam be. A tanításra a BAUM-1 adatbázisból használtam fel videókat, de mivel a rejtett Markov-modell speciális címkézést igényel, ezt én állítottam elő. A rejtett Markov-modell valósidejű kiértékeléséhez a Viterbi algoritmus használtam. A módszer javítása érdekében a modell konstans állapothatárait dinamikusra változtattam, amivel 73%-os pontosságot értem el a BAUM-1 adathalmaz a tanítástól eltérő videóin. A mesterséges neurális hálózat alapú megoldáshoz a CNN-RNN architektúrát alkalmaztam. Tanító adathalmaznak a SEWA adatbázis videóit használtam. A hibrid rendszerben három előre tanított konvolúciós hálózatot próbáltam ki különböző hosszúságú videókat vizsgálva. A CNN-RNN architektúra legjobb eredménye 77%-os pontosság volt.
A rejtett Markov-modell alapú megoldás javítható lenne a konstansok pontos beállításával, illetve nagyobb adathalmazon való tanítással. A CNN-RNN architektúra felhasználásával is jobb eredményt lehetne elérni, ha az adathalmaz pontosabban lenne annotálva, és/vagy több adat állna rendelkezésre. A detekció javítása érdekében jellemzők kinyerése helyett fel lehetne használni a landmark pontokat. Ezzel a CNN-RNN módszerhez hasonló megoldást lehetne kidolgozni úgy, hogy a számításigényes konvolúciós hálózat nem akadályozná a valósidejű detekciót. Érdemes lehet megvizsgálni azt is, hogy egy visszacsatolt hálózat képes-e megtanulni a bólintást csupán a fej horizontális tengely körüli elforgásából.

Köszönetnyilvánítás

A munkám létrejöttében nagy szerepet vállaltak konzulenseim, Révy Gábor és dr. Hullám Gábor. Köszönöm nekik a folyamatos konzultálási lehetőséget és rendelkezésreállást. Köszönet a SEWA[15] és a BAUM-1[28] adatbázis kezelőinek, akik hozzáférést adtak az adathalmazokhoz.

Irodalomjegyzék

[1] Martín Abadi – Ashish Agarwal – Paul Barham – Eugene Brevdo – Zhifeng Chen – Craig Citro – Greg S. Corrado – Andy Davis – Jeffrey Dean – Matthieu Devin – Sanjay Ghemawat – Ian Goodfellow – Andrew Harp – Geoffrey Irving – Michael Isard – Yangqing Jia – Rafal Jozefowicz – Lukasz Kaiser – Manjunath Kudlur – Josh Levenberg – Dandelion Mané – Rajat Monga – Sherry Moore – Derek Murray – Chris Olah – Mike Schuster – Jonathon Shlens – Benoit Steiner – Ilya Sutskever – Kunal Talwar – Paul Tucker – Vincent Vanhoucke – Vijay Vasudevan – Fernanda Viégas – Oriol Vinyals – Pete Warden – Martin Wattenberg – Martin Wicke – Yuan Yu – Xiaoqiang Zheng: TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL `https://www.tensorflow.org/`. Software available from tensorflow.org.

[2] Abien Fred Agarap: Deep learning using rectified linear units (relu). *arXiv preprint arXiv:1803.08375*, 2018.

[3] V Aruna – S Aruna Deepthi – R Leelavathi: Human activity recognition using single frame cnn. In *Applications of Artificial Intelligence and Machine Learning*. 2022, Springer, 205–214. p.

[4] Leonard E Baum – Ted Petrie – George Soules – Norman Weiss: A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains. *The annals of mathematical statistics*, 41. évf. (1970) 1. sz., 164–171. p.

[5] Thomas V Bonoma – Leonard C Felder: Nonverbal communication in marketing: Toward a communicational analysis. *Journal of Marketing Research*, 14. évf. (1977) 2. sz., 169–180. p.

[6] Yiqiang Chen – Yu Yu – Jean-Marc Odobez: Head nod detection from a full 3d model. In *Proceedings of the IEEE International Conference on Computer Vision Workshops* (konferenciaanyag). 2015, 136–144. p.

[7] Wikimedia Commons: Hidden markov model with output, 2012.
URL `https://commons.wikimedia.org/wiki/File:HiddenMarkovModel.svg`.

[8] Jia Deng – Wei Dong – Richard Socher – Li-Jia Li – Kai Li – Li Fei-Fei: Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition* (konferenciaanyag). 2009, Ieee, 248–255. p.

[9] G David Forney: The viterbi algorithm. *Proceedings of the IEEE*, 61. évf. (1973) 3. sz., 268–278. p.

[10] Hossein Gholamalinezhad – Hossein Khosravi: Pooling methods in deep neural networks, a review. *arXiv preprint arXiv:2009.07485*, 2020.

[11] Risto Hinno: *Baum-Welch*. 2021. 03. `https://medium.com/mlearning-ai/baum-welch-algorithm-4d4514cf9dbe`.

[12] Sepp Hochreiter – Jürgen Schmidhuber: Long short-term memory. *Neural computation*, 9. évf. (1997) 8. sz., 1735–1780. p.

[13] Ashish Kapoor – Rosalind W Picard: A real-time head nod and shake detector. In *Proceedings of the 2001 workshop on Perceptive user interfaces* (konferenciaanyag). 2001, 1–5. p.

[14] Yury Kartynnik – Artsiom Ablavatski – Ivan Grishchenko – Matthias Grundmann: Real-time facial surface geometry from monocular video on mobile gpus. *arXiv preprint arXiv:1907.06724*, 2019.

[15] Jean Kossaifi – Robert Walecki – Yannis Panagakis – Jie Shen – Maximilian Schmitt – Fabien Ringeval – Jing Han – Vedhas Pandit – Antoine Toisoul – Björn Schuller és mások: Sewa db: A rich database for audio-visual emotion and sentiment research in the wild. *IEEE transactions on pattern analysis and machine intelligence*, 43. évf. (2019) 3. sz., 1022–1040. p.

[16] Alex Krizhevsky – Geoffrey Hinton és mások: Learning multiple layers of features from tiny images. 2009.

[17] Ben Krose – Patrick van der Smagt: *An introduction to neural networks*. 2011.

[18] Tzuu-Hseng S Li – Ping-Huan Kuo – Ting-Nan Tsai – Po-Chien Luan: Cnn and lstm based facial expression analysis model for a humanoid robot. *IEEE Access*, 7. évf. (2019), 93998–94011. p.

[19] BME math: *Markov láncok*. BME, 2000. 01. `http://math.bme.hu/~mogy/oktatas/VillamosMSc_Sztoch/het_4_Markov.pdf`.

[20] Andrew McCallum: *Baum-Welch alg.* 2004. 03. `https://people.cs.umass.edu/~mccallum/courses/inlp2004a/lect10-hmm2.pdf`.

[21] Sabina Pokhrel: *cnn picture*. 2019. 09. `https://towardsdatascience.com/beginners-guide-to-understanding-convolutional-neural-networks-ae9ed58bb17d`.

[22] Mark Sandler – Andrew Howard – Menglong Zhu – Andrey Zhmoginov – Liang-Chieh Chen: Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (konferenciaanyag). 2018, 4510–4520. p.

[23] Pranj52 Srivastava: *lstm*. 12. `https://www.analyticsvidhya.com/blog/2017/12/fundamentals-of-deep-learning-introduction-to-lstm/`.

[24] Christian Szegedy – Vincent Vanhoucke – Sergey Ioffe – Jon Shlens – Zbigniew Wojna: Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (konferenciaanyag). 2016, 2818–2826. p.

[25] Amin Ullah – Jamil Ahmad – Khan Muhammad – Muhammad Sajjad – Sung Wook Baik: Action recognition in video sequences using deep bi-directional lstm with cnn features. *IEEE access*, 6. évf. (2017), 1155–1166. p.

[26] Wikipedia: *LIFO*. 01. `https://en.wikipedia.org/wiki/FIFO`.

[27] Guilin Yao – Hongxun Yao – Xin Liu – Feng Jiang: Real time large vocabulary continuous sign language recognition based on op/viterbi algorithm. In *18th International Conference on Pattern Recognition (ICPR'06)* (konferenciaanyag), 3. köt. 2006, IEEE, 312–315. p.

[28] Sara Zhalehpour – Onur Onder – Zahid Akhtar – Cigdem Eroglu Erdem: Baum-1: A spontaneous audio-visual face database of affective and mental states. *IEEE Transactions on Affective Computing*, 8. évf. (2016) 3. sz., 300–313. p.

[29] Barret Zoph – Vijay Vasudevan – Jonathon Shlens – Quoc V Le: Learning transferable architectures for scalable image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (konferenciaanyag). 2018, 8697–8710. p.