TERMÉSZETTUDOMÁNYI KAR

Burján Csaba

Matematika Bsc, Matematikai elemző szakirány

Mélytanuláson alapuló pózbecslés és alkalmazása

Témavezető: Dr. Lukács András

Számítógéptudományi Tanszék

Budapest, 2023

NYILATKOZAT

Név: Burján Csaba

ELTE Természettudományi Kar, szak: Matematika Bsc

NEPTUN azonosító: A6lKJG

Szakdolgozat címe:

Mélytanuláson alapuló pózbecslés és alkalmazása

A **szakdolgozat** szerzőjeként fegyelmi felelősségem tudatában kijelentem, hogy a dolgozatom önálló szellemi alkotásom, abban a hivatkozások és idézések standard szabályait következetesen alkalmaztam, mások által írt részeket a megfelelő idézés nélkül nem használtam fel.

Budapest, 2023.12.30.

a hallgató

Tartalomjegyzék

Bevezetés

Szakdolgozatom tárgya a neurális háló alapú emberi pózfelismerő rendszerek alkalmazása a sportban és ezen belül a rövidpályás gyorskorcsolyában.

A neurális hálók egyre nagyobb teret hódítanak napjainkban, amit főleg a számítástechnikában bekövetkezett hatalmas fejlődésnek köszönhetnek az elmúlt 10-20 évben. Ugyanis ezek a rendszerek rendkívül nagy számítási kapacitást és adatmennyiséget igényelnek. Napjainkban már szinte bárhol megtalálhatóak, például: tumorokat észlelnek orvosi felvételeken, lakossági energia fogyasztás alapján optimalizálnak erőműveket vagy célzott reklámokat ajánlanak a böngészési előzményeink alapján.

Természetesen a sportban is megtalálhatóak bármilyen statisztikai elemzés részeként vagy sportágspecifikusan akár mozgáselemző rendszerekként. Manapság a sportban még javarészt főleg olyan mozgáselemző rendszereket alkalmaznak, melyek képesek a nagyon precíz 3-dimenziós érzékelésre, hogy ezzel tudják minél pontosabban modellezni a mozgás legapróbb részleteit is. Ezekhez több kamerát, érzékelőt használnak, ahol lehetséges testre rögzített jeladókkal határoznak meg fontos pozíciókat. Ezek előkészítése nem egyszerű és elég költséges is lehet és persze nem mindenhol alkalmazható.

A jövőben ezek egyszerűbb alternatívája lehet egy olyan rendszer mely tisztán videókból vagy egyéb képi anyagokból tud 3-dimenziós modellt építeni. Egy ilyen rendszert fogok bemutatni mely precizitásban még nem képes felvenni a versenyt a fent említett rendszerekkel, de már most használható extra információt szolgáltathat akár olyan nem szokványos mozgáskultúrájú sportban is mint a rövidpályás gyorskorcsolya.

1. fejezet

Neurális hálók

„A madarak inspiráltak minket a repülésre, a bojtorján növények a tépőzárat, és a természet inspirált rengeteg más találmányt is, logikusnak tűnik tehát megvizsgálni az agy architektúráját egy intelligens gép megépítéséhez. Ez a kulcsgondolat, amely inspirálta a mesterséges neurális hálózatokat (angolul ANN-ek). Bár a repülőket madarak ihlették, nem kell csapkodniuk a szárnyukkal. Hasonlóképpen, az ANN-ek fokozatosan meglehetősen eltértek biológiai unokatestvéreiktől. Egyes kutatók még amellett is érvelnek, hogy le kell mondanunk a biológiai analógiával teljes egészében, nehogy kreativitásunkat a biológiailag elfogadható rendszerekre korlátozzuk. Az ANN-ek képezik a mély tanulás lényegét. Sokoldalúak, erősek és nagy teljesítményűek, így ideálisak a nagy és rendkívül összetett gépi tanulási feladatok megoldására, mint például több milliárd kép osztályozása (pl. Google Images), beszédfelismerési szolgáltatások (pl. Apple Siri), több millió felhasználónak ajánljanak videókat (pl. YouTube), vagy megtanulják legyőzni a világbajnokot a Sakkban több millió múltbeli játék megvizsgálásával.” - idézet a [1] könyvből. Az ANN-ek adott bemeneti és kimeneti adatok alapján próbálják „megtanulni” a két adat közti transzformációt, hogy később csak bementi értékekből adhassanak kimenetet. A továbbiakban ezt részletezem, a használt fogalmakat főleg a [2] forrásból idézem.

1.1 Mesterséges Neuron és Perceptron

A Perceptron az egyik legegyszerűbb ANN architektúra, amelyet Frank Rosenblatt talált fel 1957-ben. Ezekből lettek később a „mesterséges neuronok”, amik a mesterséges neuronháló alap építő kövei. A **Mesterséges neuron** egy $a: \mathbb{R} \to \mathbb{R}^n$ leképzés, amit a következőképpen írhatunk fel:

$$a = \sigma\left(\sum_{i=1}^{n} w_i\, x_i + b\right)$$

ahol σ egy $\mathbb{R} \to \mathbb{R}$ aktivációs függvény $x_i \in \mathbb{R}$ ($i \in [1 \dots n]$) bemeneti érték $w_i \in \mathbb{R}$ ($i \in [1 \dots n]$) az ehhez tartozó súly és $b \in \mathbb{R}$ az eltolás (bias). A fent említett **aktivációs függvény** egy olyan függvény, amely kiszámítja a csomópont kimenetét (a bemeneteik, az egyes bemenetek súlyai és az eltolás alapján). Ez azért szükséges mert a nem triviális problémákat

csak nemlineáris aktivációs függvény segítségével lehet megoldani. Ezek közül párat mutatok be, amit napjainkban is használnak.

Sigmoid:

$$\boldsymbol{\sigma = \frac{1}{1+e^{-x}}}$$

Sajnos nem mindig megfelelő, mivel szélsőséges bemeneti értékekre a függvény értékünk 1-hez vagy 0-hoz közelít, ahol a derivált majdnem 0 (ezt majd a későbbiekben látjuk, hogy miért nem jó).

Softmax:

$$\Sigma(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ ahol } i = 1, \dots, K \text{ és } z = (z_1, \dots, z_K) \in \mathbb{R}^K$$

A softmax függvény, más néven softargmaxvagy normalizált exponenciális függvény, K valós számból álló vektort K lehetséges kimeneteli valószínűség eloszlására konvertál. Ez a logisztikus függvény általánosítása több dimenzióra. A softmax függvényt gyakran használják egy neurális hálózat utolsó aktivációs függvényeként, hogy a hálózat kimenetét a várható kimeneti osztályok közötti valószínűségi eloszlásra normalizálják.

Relu:

$$ReLU(x) = max(0, x)$$

A mai neurális halók leggyakrabban használt aktivációs függvénye, mert egyszerűbb vele számolni és jobb tulajdonságokkal bír mint elődjei (mivel deriváltja a pozitív számokra konstans 1 ezért mélyebb hálók esetén sem jellemző rá a „vanishing gradient” probléma)

1.2 Neuron réteg

A fent említett mesterséges neuronokat mesterséges rétegekbe rendezzük, ezek segítségével definiálható $f: \mathbb{R}^n \to \mathbb{R}^m$ leképezés, ahol n a bemeneti vektor hossza, m pedig a neuronok száma. Az

$$\mathrm{f} = \sigma(W(x) + b)$$

függvényt **mesterséges neuronrétegnek** nevezzük, ahol σ egy adott $\mathbb{R} \to \mathbb{R}$ aktivációs függvény $W \in \mathbb{R}^{m \times n}$ mátrix a neuronok súlyait tartalmazza, $x \in \mathbb{R}^n$ a bemeneti vektor, $b \in \mathbb{R}^m$ pedig az eltolásvektor. Itt a leképezés eredménye egy m-hosszúságú vektor. Ezen rétegek egymásutánjából kaphatunk egy neurális hálót melynek legegyszerűbb és leggyakoribb modellje az **előre csatolt neurális hálók**, ahol a neuronrétegeket függvény kompozícióval egymásután kapcsoljuk. Az

$$f_\theta(x) = f_L \circ \cdots f_2 \circ f_1(x)$$

ahol $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ a neuron rétegek paramétereit tartalmazó vektor f_i pedig a hálózat rétegeit reprezentáló függvény.

Egy két extra indexet bevezetve egy mesterséges neuron kimenetét a következőképp kapjuk

$$a_i^j = \sigma\left(w_{i,0}a_0^{j-1} + w_{i,1}a_1^{j-1} + \cdots + w_{i,n}a_n^{j-1} + b_i^{j-1}\right)$$

ahol a az i-edik réteg j-edik neuronja. Ezen neurális hálók végén pedig szeretnénk megnézni, hogy mennyire sikerült jó eredményeket adnunk. A **veszteségfüggvény** az a függvény, amely kiszámítja a modellünk aktuális kimenete és a várt kimenet közötti különbséget. Ez egy módszer annak értékelésére, hogy modellünk mennyire prediktál pontos eredményt. Akkor teljesít jól a modellünk, ha ennek a függvénynek az értéke minél kisebb. Ezekből az egyik ismertebb az L2 loss függvény vagy négyzetes eltérés függvény

$$L = \sum_{j=0}^{n_L - 1} \left(\mathrm{a}_\mathrm{j}^{(\mathrm{L})} - \mathrm{y}_\mathrm{j}\right)^2$$

ahol, y_j az elvárt érték es az $a_j^{(L)}$ az utolsó L -edik réteg megfelelő indexű kimenet értéke. Ezzel meg is van az alap modellünk most már csak a megfelelő értékek kiszámolása következik.

1.3 A modell tanulása gradient descent módszerrel

A gradiens ereszkedés egy nagyon általános optimalizálási algoritmus, amely képes megtalálni az optimális megoldást sokféle problémára. A gradiens ereszkedés általános elképzelése az, hogy a paramétereket iteratív módon módosítjuk és ezzel minimalizáljuk a költségfüggvényt.

A vektorszámításban egy skaláris értékű differenciálható f több változós függvény **gradiense**. Az a ∇f vektormező mely egy p pontra megadja a leggyorsabb növekedés irányát. Ha egy függvény gradiense egy p pontban nem nulla a gradiens nagysága az adott irányú növekedés mértéke.

Ha $f = f_\theta(x)$ ahol $\theta = (\theta_1, \theta_2, \ldots, \theta_K)$ a paraméterek akkor a gradiens a következő:

$$\nabla f(p) = \begin{bmatrix} \frac{\partial f}{\partial \theta_1(p)} \\ \vdots \\ \frac{\partial f}{\partial \theta_K}(p) \end{bmatrix}$$

A gradiens megmutatja, hogy milyen „irányban” változtassunk a paramétereinken és, hogy arányaiban milyen mértékben, de magát a lépés nagyságát mi választjuk meg és egy konkrét lépés a következőképpen írható le

$$\theta \leftarrow \theta - \lambda \cdot \frac{\partial L(f_\theta(\mathbf{x}), y)}{\partial \theta},$$

ahol θ a hálózat a paramétereit tartalmazó vektor, λ a lépés mérete, L pedig a veszteségfüggvény.

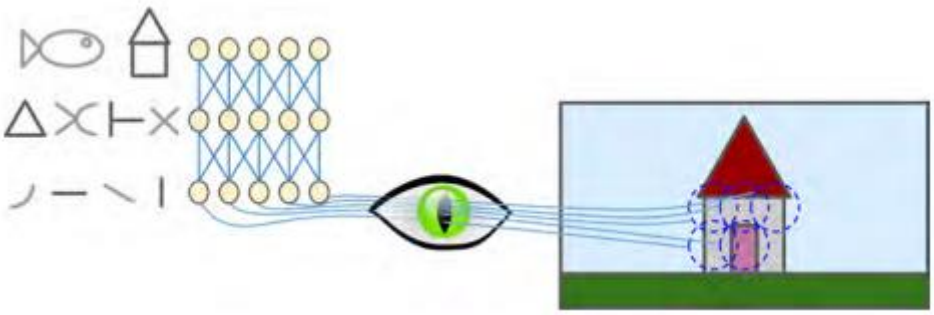
Ezekkel a lépésekkel megtalálhatjuk a függvényünk lokális minimumát, ami persze nem feltétlen jelent jó megoldást. Szerencsére erre a gyakorlatban rengetek megoldás létezik, mint pl.: a sztochasztikus gradiens ereszkedés vagy a „mini-batch” gradiens ereszkedés, ahol random tanuló elem vagy elem-csoport választásával próbálunk kikerülni a lokális minimumokból.

2.fejezet

Konvolúciós Neurális hálók

2.1 Biológiai vonatkozat

„A konvolúciós neurális hálók ötlete egy tanulmányhoz köthető mely az agyunk vizuális kérgét tanulmányozta és már az 1980-as évektől használták képfelismeréshez [3] . Az utóbbi pár évben köszönhetően az exponenciális növekedésnek a számítási kapacitásban és a különböző tanító adatok mennyiségében. Valamint egyéb a szakmában elterjedt megoldásoknak hála mára már rengeteg helyen használják pl.: önvezető autókban, kép-keresésben, automatizált videók kategorizálásban és más területeken is mint hangfelismerés és nyelvi modellek építésében. Mi csak a vizuális felhasználásával fogunk foglalkozni.” [1]



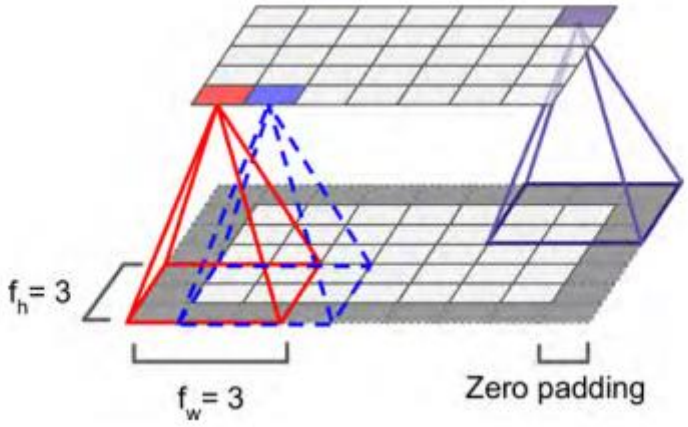
1. ábra A szemünk által érzékelt információból, apróbb formákból hogyan próbál az agyunk egy nagyobb összképet összerakni

2.2 Konvolúciós réteg

A CNN legfontosabb építőeleme a konvolúciós réteg. Az első konvolúciós réteg neuronjai nem kapcsolódnak a bemeneti kép minden egyes pixeléhez (mint az előző fejezetben), csak a pixelekhez a receptív mezőjükben. A második konvolúciós rétegben minden neuron megint csak az előző réteg kis részére koncentrál. Ez az architektúra lehetővé teszi a hálózat számára, hogy az első rejtett réteg alacsony szintű részleteit koncentrálja és a következő réteg ezekből építsen komplexebb információt és így tovább. Ez a hierarchikus struktúra gyakori a valós képeken, ez az egyik oka, hogy miért működnek olyan jól a CNN-ek a képfelismerésben.

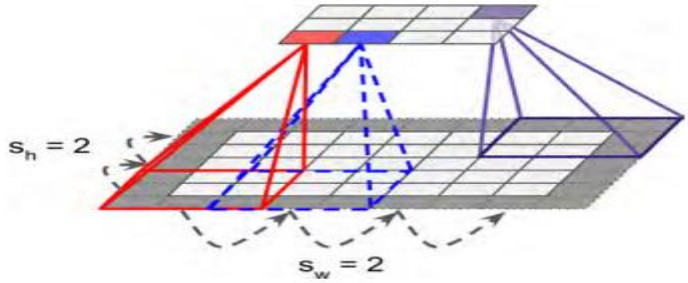
Egy adott réteg i. sorában és j. oszlopában elhelyezkedő neuron az előző réteg $\{i, i + f_h - 1\}$ $és$ $\{j, j + f_w - 1\}$ neuronjainak kimenetével van kapcsolatban, ahol f_h $és$ f_w a befogadó mező hossza és szélessége. Annak érdekében, hogy egy réteg magassága és szélessége

ugyanolyan legyen, mint az előző rétegé, megszokott, hogy nullás értékeket adjunk a bemeneti réteg széléhez.



2. ábra Hogyan adja meg az előző réteg 9 eleme a következő réteg 1 elemét

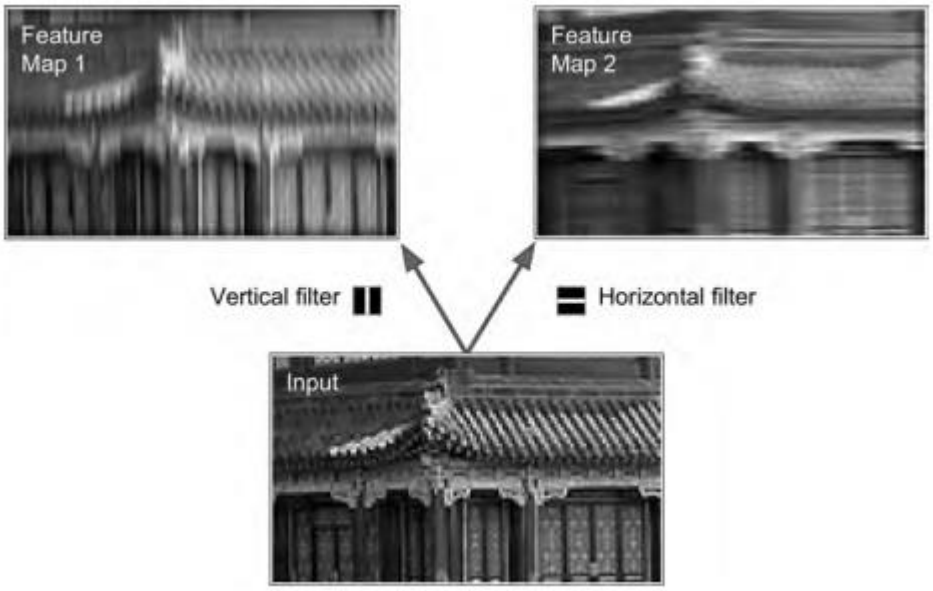
Lehetőség van arra is, hogy egy nagy bemeneti réteget egy sokkal kisebb réteghez kapcsoljunk a befogadó mezők eltolásával. A két egymást követő receptív mező közötti távolságot lépésnek nevezzük. A 3. ábrán egy 5 × 7-es bemeneti réteg (plusz nulla padding) egy 3 × 4-es réteghez van csatlakoztatva, 3 × 3 receptív mezővel és egy 2-es lépéssel (ebben a példában a lépés mindkét irányban azonos, de nem kell, hogy így legyen). A felső réteg i.sorában és j. oszlopában található neuron az előző réteg$\{ i * s_h, i * s_h + f_h - 1\}$ és $\{j * s_w, j * s_w + f_w - 1\}$ oszlopaiban található neuronok kimeneteihez kapcsolódik, ahol s_h és s_w a függőleges és vízszintes lépések.



3. ábra Lépésköz állítása mind a két dimenzióban

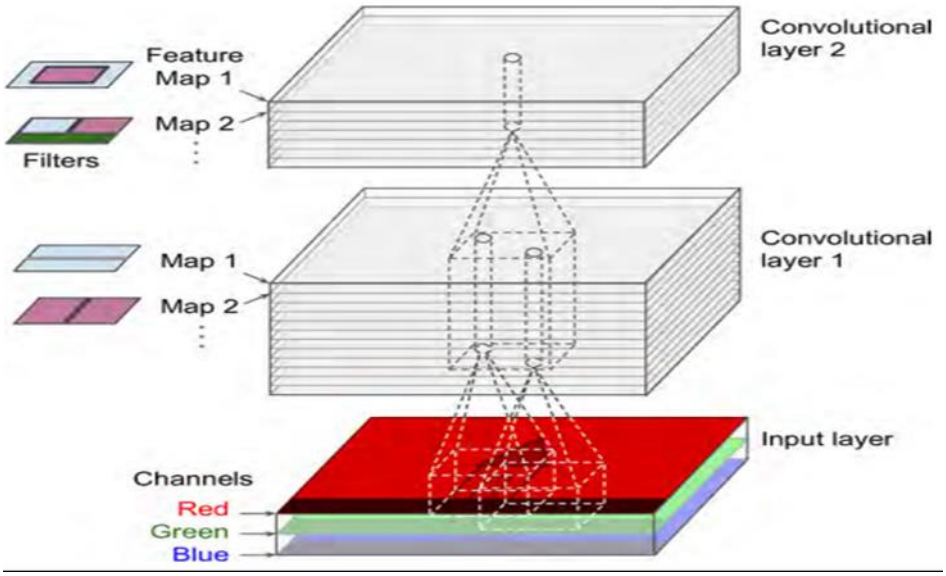
A fentiekben receptív mezőnek nevezett területhez tartozó súlyokat kernelnek vagy filternek nevezik és bár vannak hiperparaméterek, amiket a modellező állít be (kernel mérete, padding

és lépésköz) a modellünk most a kernelben lévő mezők súlyát próbálja optimalizálni. Ezen paraméterek állítgatásával tudunk különböző információkat kinyerni képekből. Egy egyszerűbb példát nézve, ha a kernelünk középső oszlopa csupa egyes míg minden más 0 azzal a vertikális mintákat vagy vonalakat emeljük ki míg ha a középső sor 1 és minden más 0 akkor a horizontális éleket hangsúlyozzuk



4. ábra A kernel súlyainak állításával, hogy nyerhetünk ki különböző információt

Eddig az egyszerűség kedvéért minden konvolúciós réteget 2 dimenzióban ábrázoltunk, de a valóságban egy rétegen belül több csatorna is lehetséges, tehát pontosabb, ha 3D-ben ábrázoljuk őket. Ezenkívül érdemes még megemlíteni, hogy minden csatornához külön kernel tartozik (azonos hiperparaméterekkel de különböző optimalizálható súlyokkal).



5. ábra Látható, hogy minden rétegben több csatornában helyezkednek el az információk

Ehhez tartozó matematikai képlet a következő

$$z_{i,j,k} = b_k + \sum_{u=1}^{f_h} \sum_{v=1}^{f_w} \sum_{k'=1}^{f_{n'}} x_{i',j',k'} \cdot w_{u,v,k',k} \quad \text{with} \begin{cases} i' = u \cdot s_h + f_h - 1 \\ j' = v \cdot s_w + f_w - 1 \end{cases}$$

•$z_{i,j,k}$ az i. sorban és j. oszlopban lévő neuron kimenete a k-adik csatornán az l-edik rétegben

• A korábban már említett s_h és s_w a vertikális és horizontális lépésköz f_h és f_w a kernel hossza és szélessége és $f_{n'}$ a csatornák száma az $l-1$-edik szinten.

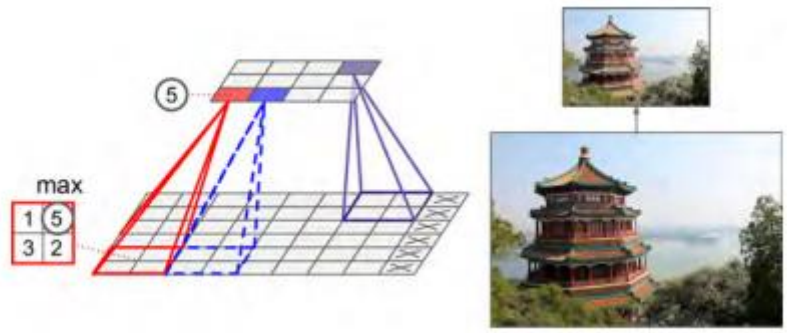
• $x_{i',j',k'}$ a kimenete az l-1 edik szinten lévő neuronnak az i' sorban j' oszlopban a k'

csatornában.

• b_k a bias az l-edik szinten a k-adik rétegben ezzel lehet állitani a csatorna súlyát.

• $w_{u,v,k',k}$ pedig a kapcsolódó súly a k-adik csatornában az l-edik rétegben és a bemenete az u-adik sorban és a v-edik oszlopban van a k' csatornában.

Ezen kívül megemlíthető még a **pooling,** ami igazából csak egy speciális kernel. Itt is meghatározhatjuk a méretet, a lépéshosszt és a padding típusát, csak úgy mint ezelőtt. Az egyesülő neuronoknak azonban nincs súlya, mindössze egy aggregációs függvényt alkalmazunk az elemekre, mint például a max vagy az átlag. A példában egy max függvényt alkalmazunk, a gyakorlatban ez a legelterjedtebb. Ezt legfőképpen azért alkalmazzák, hogy tömörítsék az információt, még egy apró 2 × 2-es kernellel és egy 2-es lépéssel is a kimenet fele akkora lesz mindkét irányban (tehát a területe négyszer kisebb), egyszerűen kiesik a bemeneti értékek 75%-a.



6. ábra Pooling használata, hogy tömörítsük az információt

3. fejezet

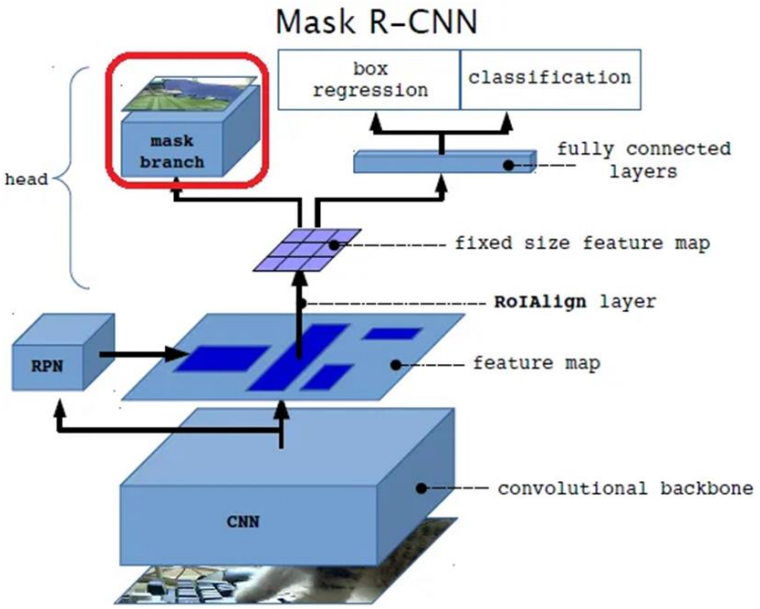
2D pózfelismerés

3.1 Áttekintés

Természetesen a képi információ feldolgozásán kívül másra is használhatóak a konvolúciós neuronhálók, például dokumentumok analizálására a benne használt szavak alapján vagy a földi klíma változásának megértésében. A legtöbb szakterület persze a képi információ feldolgozására használja, legyen szó a telefonba épített arcfelismerőkről, önvezető autók által használt objektum detektálásról és szegmentálásáról. Ezeket az architektúrákat apróbb változtatásokkal könnyen lehet más feladatokra is használni például a pózfelismerésre.

3.2 Mask R-CNN

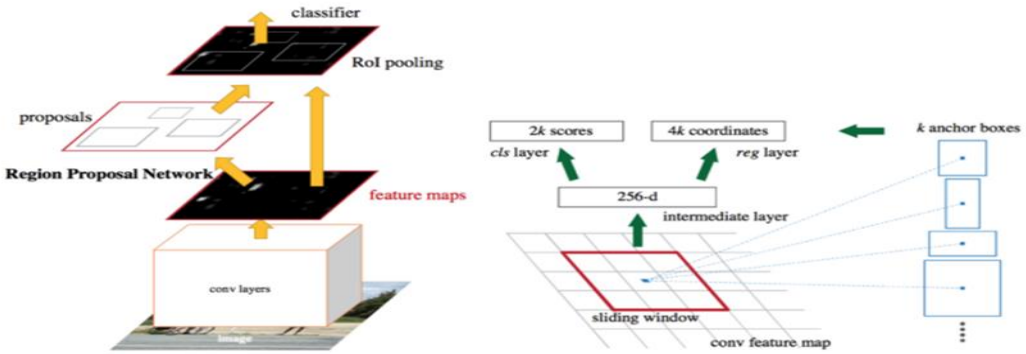
[4] A következő részben bemutatok egy objektum szegmentálásra használt architektúrát, amit kis változtatásokkal könnyen lehet pózfelismerésre használni. Első sorban szeretnénk a képen lévő objektumokat azonosítani (a mi esetünkben csak embereket), de lehetne akár több kategóriában is objektumokat keresni. Ehhez egy alkalmas modell a Mask R-CNN. Ennek megértéséhez elég megérteni a Faster R-CNN működését, ami 2 elődjét (R-CNN, Fast R-CNN) múlta felül számítási gyorsaságban és jóval takarékosabb memóriahasználatban.



7. ábra Mask R-CNN architekrúra

Faster R-CNN

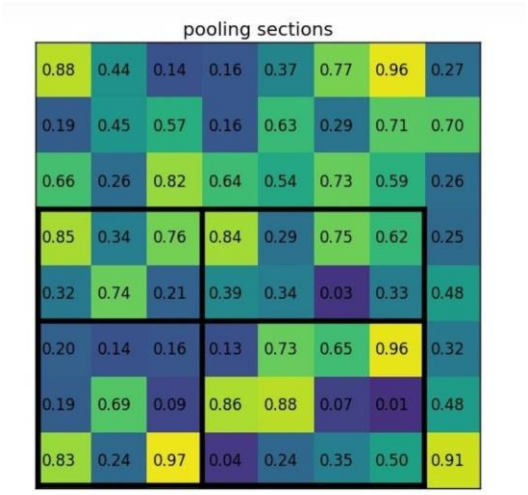
[5] Először egy szabványos főbb jellemzők kivonására alkalmas CNN-t (általában ResNet verziót) alkalmazunk, hogy tömörebb formában kapjuk meg az információt. Ezután ennek a hálózatnak az utolsó még nem teljesen kapcsolódó rétege egy „feature map” kerül felhasználásra a régiójavaslathoz és a végső észleléshez is. A régiójavaslatokhoz egy külön *n x n* -es neuronháló (vagy ablak) „csúszik végig" a feature mapeken. Minden ablakhoz generálunk fix számú ún. horgonyt előre megadott skálázási arányok és méretarányok szerint összesen k darabot (pl 3-3 megadott aránnyal ablakonként 9-et). Ez a hálózat prediktál mindegyik ablakra egy *cls* $\in \mathbb{R}$ 2k („classification” vagy osztályozó pontszám az objektum osztályához kapcsolódóan) és egy *reg* $\in \mathbb{R}$ 4k pontszámot (regressziós pontszám a határoló dobozokhoz). A feature mapen minden *n* × *n-es* ablakhoz tartozik egy megfelelő terület (látómező) az eredeti képen, amely hatással van a feature map adott régiójára. Így a modell javaslatot tud tenni érdekes régiókra különböző méretben és arányban. A tanítás során a kimeneti *cls* és a *reg* pontszámokat veszteségfüggvényen keresztül hasonlítjuk össze a kapott címkékkel a következő módon: Minden „groundtruth” objektum határolódobozhoz rendelünk egy vagy több horgonyt (olyanokat, amelyeknek elég magas a metszés/unió arányuk - nagyobb mint 0.7). A horgonyok, amelyek egy objektumhoz vannak rendelve, 1-es *cls* címkét kapnak, míg mások 0-t. Hasonlóképpen a *reg* címkék a horgonyokhoz az objektum alapján vannak hozzárendelve (ha van ilyen objektum). Ezek a reg címkék lesznek a határoló doboz paraméterei egy horgonyhoz rendelt objektumról. A végső modellben kiválasztjuk a legjobb N (mi esetünkben 100) *cls* pontszámot és a hozzájuk tartozó *reg* értékek lesznek a határolódoboz koordináta javaslatai. A javasolt határolódobozok alapján RoI-k (Regional of Interest vagy magyarul „érdekes területek”) lesznek kivágva a feature mapekből, mint *m* × *m-es* képek. RoI-k és a feature mapünk általában nincsenek tökéletesen összhangban.



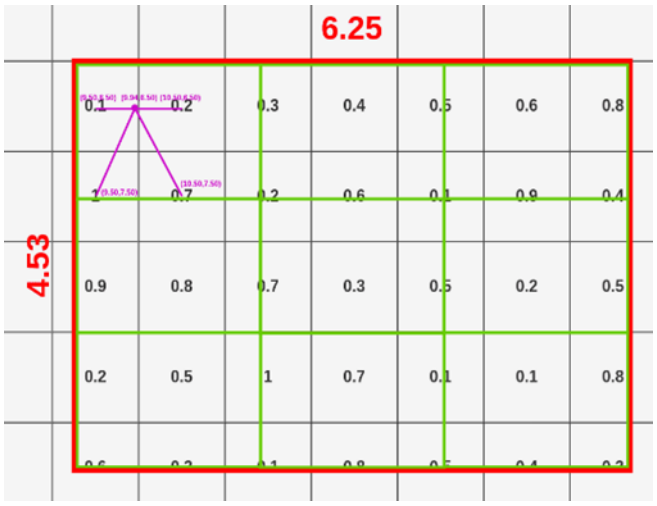
8. ábra Faster R-CNN

3.3 Roi Align

RoI-k kiválasztása RoI Align módszerrel. Nem mindig fontos, hogy a Bboxunk (bounding box) tökéletes fedje az objektumot, hiszen egy autót vagy embert akkor is tudunk észlelni, ha kicsit kilóg valamelyik széle vagy esetleg pont, hogy kicsit nagyobb a határoló doboz, de ha maszkot akarunk illeszteni az objektumunkra akkor már fontos, hogy a Bbox minél pontosabban illeszkedjen, ezért volt szükség egy új pontosabb módszer bevezetésére. A fő különbség az elődök által használt RoI Pooling és a Mask R-CNN által használt RoI Align között a kvantálás. Tudjuk, hogy a Fast R-CNN kétszer alkalmazza a kvantálást. Először a leképezési folyamatban (amikor egy ajánlott Bbox pontos koordinátái helyett a legközelebbi pixeleket adjuk meg), másodszor pedig a pooling folyamatban (amikor egy fix *n x n*-es hálót kell visszaadnunk ezért pár mezőt összevonunk – [9. ábra] ha a Bbox ajánlat egy *5x7*-es terület és nekünk *2x2*-es kell akkor a pixelek miatt nem egyenlő méretű területekkel dolgozunk, ami torzíthat az eredményen). A RoI Align nem használ kvantálást az adatgyűjtéshez. Nem szorítkozunk a természetes számokra. Az objektumunk határait racionális számokkal írjuk le majd az így kapott nem teljesen illeszkedő hálón bilineáris interpolációt hajtunk végre. És így kapunk pontosabb határoló dobozokat (az eredeti Bboxnak sem kell illeszkedni a feature mapünkre és utána ennek felosztásakor sem kell aggódnunk, ha a határok nem illeszkednek).



9. ábra RoI Pooling

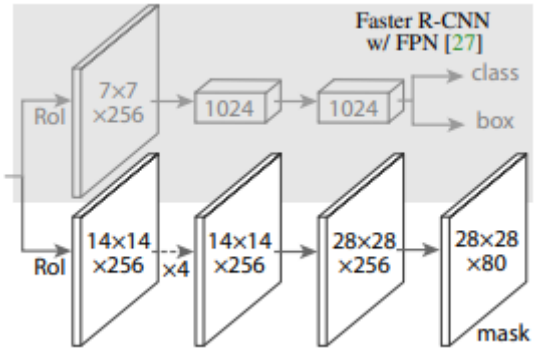


10. ábra RoI Align kvantálás nélkül, bilineáris interpolációval

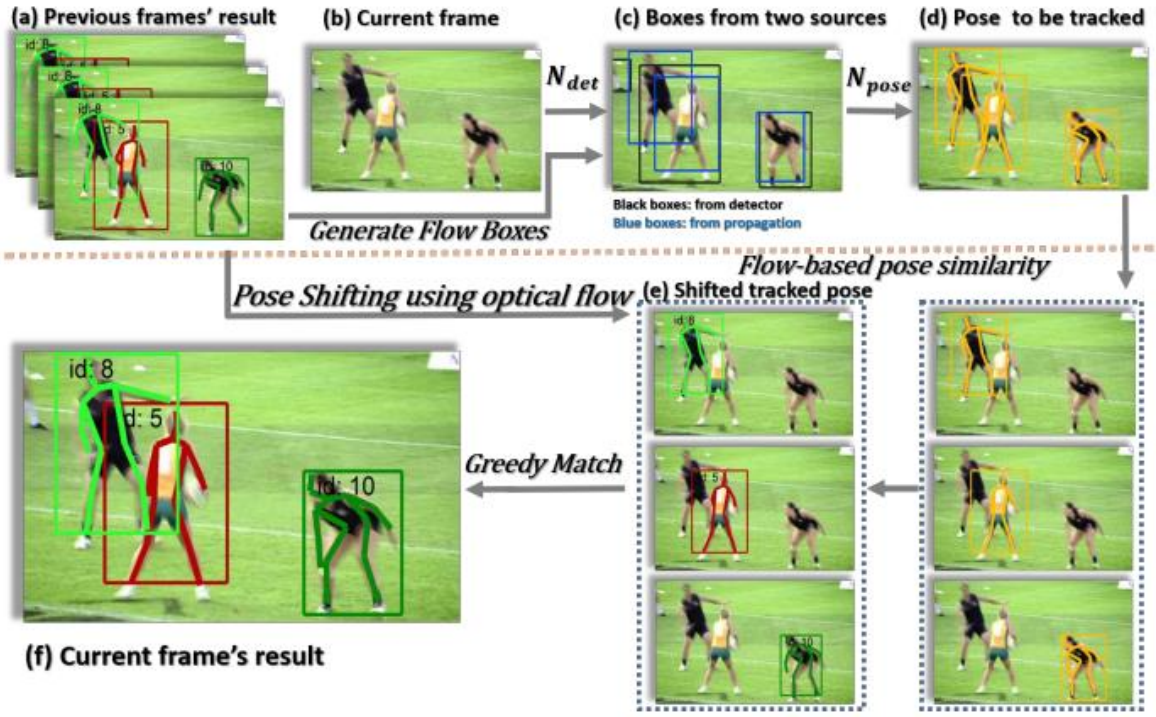
3.4 Fejhálózatok

Más fejhálózatokat alkalmazunk a RoIkon az objektum osztályozásához és a maszk elkészítéséhez (11. ábra) (vagy szemantikus szegmentálásához). Ezt egy újabb kis konvolúciós háló végzi el. Ami minden lehetségeshez osztályhoz jósol maszkokat a RoI-k számára (ez a mi esetünkben csak egy objektumot jelentene), de a veszteségfüggvényben csak a megjósolt osztály maszkját vesszük figyelembe.

Ezt az architektúrát könnyen ki lehet szélesíteni emberi ízületi koordináták detektálására is. Úgy modellezünk egy ízületet, mint egy maskot és K ízülethez K maskot készítünk. Változtatunk egy kicsit a szegmentációs folyamaton. Minden K ízülethez lesz egy $M \times M$ -es tanulási cél, ahol csak 1 pixel van megjelölve „objektumnak”. A tanitás során minden látható tényleges kulcspontnak minimalizáljuk a cross-entropy veszteségét egy softargmax kimeneten (ami arra sarkal, hogy detektáljuk azt az egy pontot), de itt még minden K kulcspont külön van kezelve. Itt egy ResNet-FPN (10. ábra) variánst használunk. A kulcspont detektáláshoz használt fejrendszer egy 3x3-as 512-dimenzios konvolucios réteg egy dekonvolúciós réteggel követve és 2x -es bilineáris felskálázással egy 56x56 -os hálót hagyva (10.ábra - a kép dimenziói nem egyeznek a leírtakkal csak illusztráció), mivel ez jobban használható ízületi detektálásra. Persze ezen kívül vannak más módszerek is pl: más veszteségfüggvény vagy több kulcspont megadása egyszerre.



11. ábra Konvolúciós fejhálózat az ízületetek detektálásához



12. ábra Optikai áramlás

3.5 Pózfelismerés optikai áramlással

[6] A következőkben bemutatok egy technológiát mellyel még jobb észlelést érhetünk el videókon. Ebben a módszerben megpróbálunk nem csak a szegmentálás által észlelt dobozokra hagyatkozni, hanem valahogy lekövetni a detektált objektumok mozgását és ezzel biztosítani, hogy akkor is észleljük az objektumot, ha azt a szegmentáló éppen nem tudja. Nézzük ezt emberekre és pózfelismerő rendszerekre specifikusan. Egy többszemélyes pózkövető rendszer esetében a videókban először megbecsüljük az emberi pózokat a határoló dobozokban, majd követjük ezeket a dobozokat vagy „személyeket" egy egyedi azonosítóval (*id*) a képkockákon

át. Egy P embert úgy jelölünk hogy, $\mathrm{P} = (\mathrm{J}, \mathrm{id})$ ahol $J = \{j_i\}_{\{1:N_J\}}$ az N_j test ízületeinek koordinátái az *id* pedig a fent említett azonosító. Amikor feldolgozzuk a k-adik képet $I^k - t$, már feldolgoztuk az embereket az I^{k-1} képen. Az I^k képen feldolgozott személyekhez pedig *id*-ket rendelünk. Ha egy P_i^k az aktuális kép kockán I^k hozzá van kötve a P_i^{k-1} személyhez az I^{k-1} képen akkor id_i^{k-1} hozzá lesz kötve az id_i^k -hoz (ugyan azt az id-t fogja kapni mind a két képen), ha nem akkor új *id*-t kap P_j^k jelezve egy új embert (hogy mi szerint egyeztetünk meg két embert majd később esik szó). A mi modellünk képkockánkként 2 szett határoló dobozt fog kreálni egyet Mask R-CNN segítségével és egyet az előző képkockák és egy úgynevezett optikai áramlás segítségével majd ezeken alkalmaz egy párosító algoritmust, hogy legjobban észlelje az embereket.

Erre azért lehet szükség mert a videókon nem minden kép éles, sokszor van, hogy egy-egy forma egy képkockán a gyors mozgás miatt elmosódik. A fenti 12-es ábrán a detektor a gyors mozgás miatt kihagyja a bal oldali fekete ruhás személyt. Erre egy megoldás, hogy a közeli képekből állítsunk elő határoló dobozokat az éppen feldolgozó kép számára optikai áramlással. Adott egy ember J_i^{k-1} ízületi koordinátákkal a I^{k-1} képen es az optikai áramlás mező $F_{k-1\to k}$ az I^{k-1} es I^k képek között, megtudnánk becsülni az ízületi koordinátákat $\widehat{J_i^k}$ az I^k képen, ha vesszük J_i^{k-1} es alkalmazzuk rá az áramlási mezőt. Amikor nehéz embert detektálni egy képkockán akkor is tudnánk hozzá rendelni határoló dobozt, ha valamelyik előző képkockán helyesen volt detektálva. Mint az ábrán látható (12/c), a bal oldali fekete ruhás személyt nem detektálta a rendszer, de az optikai áramlással tudunk hozzá határoló dobozt rendelni.

I^k	k^{th} frame
Q	tracked instances queue
L_Q	max capacity of Q
$\mathcal{P}^k$	instances set in k^{th} frame
$\mathcal{J}^k$	instances set of body joints in k^{th} frame
P_i^k	i^{th} instance in k^{th} frame
J_i^k	body joints set of i^{th} instance in k^{th} frame
$F_{k\to l}$	flow field from k^{th} frame to l^{th} frame
M_{sim}	similariy matrix
B_{det}^k	boxes from person detector in k^{th} frame
B_{flow}^k	boxes generated by joint propagating in k^{th} frame
B_{unified}^k	boxes unified by box NMS in k^{th} frame
$\mathcal{N}_{\text{det}}$	person detection network
$\mathcal{N}_{\text{pose}}$	human pose estimation network
$\mathcal{N}_{\text{flow}}$	flow estimation network
$\mathcal{F}_{\text{sim}}$	function for calculating similarity matrix
$\mathcal{F}_{\text{NMS}}$	function for NMS operation
$\mathcal{F}_{\text{FlowBoxGen}}$	function for generating boxes by joint propagating
$\mathcal{F}_{\text{AssignID}}$	function for assigning instance id

13. ábra Pózkövetési algoritmus jelölései

3.6 Pózhasonlóság felismerése

A kérdés már csak az hogyan azonosítjuk ugyan azt az embert több képen át, illetve hogyan döntjük el a 2 generált Bbox közül melyik a jobb? A határoló dobozok IoU (Intersection-over-Union vagy magyarul „metszet/unió") tulajdonságát használni hasonlósági mérőszámként (S_{Bbox}) problémás lehet, ha egy ember túl gyorsan mozog és ezért a dobozok nem feddik egymást eléggé, és a zsúfolt jelenetekben, ahol a dobozok esetleg nem rendelkeznek megfelelő kapcsolattal a tényleges emberrel. Egy finomabb mérőszám lehetne a pózhasonlóság *(S_{Pose}),* amely kiszámítja a test ízületeinek távolságát kettő objektum között, a módszer neve *(OKS – Object Keypoint Similarity „objektum kulcspont hasonlóság").* Mi egy ilyen áramlásalapú pózhasonlósági mérőszámot használunk. Persze a pózhasonlósággal is lehet probléma, ha túl gyorsan változik vagy esetleg a rossz érzékelés miatt.

Adott egy személy J_i^k az I^k -ik képen es egy a J_j^l az I^l képen a kettejük áramlási hasonlósága a következő

$$S_{Flow}(J_i^k, J_j^l) = OKS(\widehat{J_i^l}, J_j^l)$$

ahol OKS reprezentálja két pozíció közti különbséget, ahol $\widehat{J_i^l}$ a jósolt ízületek J_i^k -nak az I^k képről az I^l képre az optikai áramlás mezőt használva $F_{k\to l}$

Más emberekkel vagy tárgyakkal való takarás miatt az emberek gyakran eltűnnek és újra megjelenni. Ezért nem mindig elég két egymást követő képkockát nézni, így megvan az áramlás alapú pózhasonlóság több képkockán át is, amelyet S_{Multi} -flow-nak jelölünk, jelentése a propagált $\widehat{J^k}$ több korábbi képkockából származik.

Az optikai áramlást és az áramlás alapú pózhasonlóságot használva kapjuk pózkövető algoritmust, amely ezt a kettőt kombinálja, ahogy a 14-as ábrán prezentálva van a 13-es ábrai jelölésekkel. Először is megoldjuk a pózbecslési feladatot. Egy képkockán a videóból, egyesítjük a detektor által generált *Bbox*okat és az előző képkockákból az optikai áramlás segítségével generált *Bbox*okat, ezt egy NMS („Non-Maximum Suppression") operáció segítségével tudjuk megtenni. Ezek után becsüljük meg az emberi pózt.

Másodjára megoldjuk a követési problémát. Eltároljuk a követett személyeket egy dupla végű sorba fix hosszall L_Q (a sor hossza) ahol

$$Q = \left[P_{k-1}, P_{k-2}, \dots, P_{k-L_Q}\right]$$

P_{k-1} a követett példányt jelenti, az I^{k-1} képen és L_Q darab előző képet veszünk figyelembe a párosításhoz.

Algorithm 1 The flow-based inference algorithm for video human pose tracking

input: video frames $\{I^k\}$, $Q = []$, Q's max capacity L_Q.
$B^0_{\text{det}} = \mathcal{N}_{\text{det}}(I^0)$
$\mathcal{J}^0 = \mathcal{N}_{\text{pose}}(I^0, B^0_{\text{det}})$
$\mathcal{P}^0 = (\mathcal{J}^0, id)$ ▷ initialize the *id* for the first frame
$Q = [\mathcal{P}_0]$ ▷ append the instance set $\mathcal{P}_0$ to Q
for $k = 1$ **to** ∞ **do**
    $B^k_{\text{det}} = \mathcal{N}_{\text{det}}(I^k)$
    $B^k_{\text{flow}} = \mathcal{F}_{\text{FlowBoxGen}}(\mathcal{J}^{k-1}, F_{k-1\to k})$
    $B^k_{\text{unified}} = \mathcal{F}_{\text{NMS}}(B^k_{\text{det}}, B^k_{\text{flow}})$ ▷ unify detection boxes and flow boxes
    $\mathcal{J}^k = \mathcal{N}_{\text{pose}}(I^k, B^k_{\text{unified}})$
    $M_{\text{sim}} = \mathcal{F}_{\text{sim}}(Q, \mathcal{J}^k)$
    $\mathcal{P}^k = \mathcal{F}_{\text{AssignID}}(M_{\text{sim}}, \mathcal{J}^k)$
    append $\mathcal{P}^k$ to Q ▷ update the Q
end for

14. ábra Pózkövető algoritmus

4.fejezet

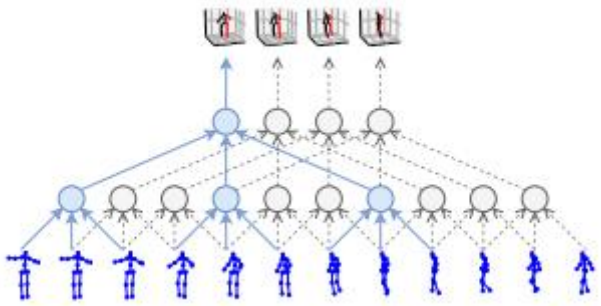
2-dimenzióból 3-dimenzióba emelés

[7] Rengetek problémára már elég is lenne a legtöbb pózfelismerő rendszer 2-dimenziós eredménye. Megmondani épp milyen sporttevékenységet végzünk vagy esetleg egy animációhoz adhat adatokat. Viszont, ha sportolókat akarunk elemezni és versenyezni az idővel akkor muszáj pontos ízületi helyzeteket detektálni és szögeket számolni, hogy kiderítsük mi egy bajnok mozgásának titka.

Ebben a részben egy lehetséges megoldást ismertetünk a fent említett problémára, mely nem feltétlen egyértelmű hiszen egy 2-dimenziós pózhoz többféle 3-dimenziós tényleges pozíció is tartozhat. A korábbi munkák ezt a kétértelműséget az időbeli információk modellezésével kezelték rekurrens neurális hálózatokkal. Másrészt a konvolúciós hálózatok az utóbbi időben nagyon sikeresek lettek az időbeli információ modellezésben olyan feladatokban, amelyeket hagyományosan RNN-kel oldottak meg, mint például a neurális gépi fordítás, nyelvi modellezés, beszédgenerálás és beszéd felismerés. Számunkra azért előnyösebbek a konvolúciós modellek, mert lehetővé teszik a párhuzamosítást mind a batch és idődimenzión át, míg az RNN-ek időben nem párhuzamosíthatók ezért kiszámításuk sokkal lassabb. Ezen kívül a konvolúciós modellekben a gradiens útja a kimenet és a bemenet között fix hosszúságú, függetlenül a sorozat hosszától, ami mérsékli az eltűnő és robbanó gradiens problémákat, amelyek befolyásolják az RNN-eket. A harmadik jó tulajdonság számunkra, hogy a dilatált konvolúciós architektúra az időbeli receptív mező precíz vezérlését is kínálja, ami előnyös az időbeli modellezés szempontjából a 3D-s pózbecslés feladatához.

A modell két részre bontható az elsőben egy (Temporal Convolution) hálózattal állítunk elő 2D ízületi koordinátákból és határoló dobozokból 3D koordinátákat és a tanításhoz egy olyan adathalmazt használunk mely rendelkezik tényleges 3D-s groundtruth koordinátákkal 2D-s képekhez. A modell másik részében felhasználunk előre fel nem címkézett adatokat is és így „félig felügyelt tanítással” is tudjuk segíteni a modell pontosságát. Ez azért előnyös mert nem rendelkezünk még túl sok 3D-s felcímkézett adathalmazzal és ezek előállítása is igen körülményes. Ezt a modellt az a nyelvi fordításban már ismert metodika ihlette miszerint, az enkóderünk jóságát tesztelhetjük, ha ismert egy jó dekóder és ezzel egyesítve oda-vissza fordítva egy szöveget a bemenethez hasonló eredményt kell hogy kapjunk. Ezt a saját problémánkban úgy értelmezzünk, hogy 2D-s koordinátákhoz jósolunk 3D-seket majd

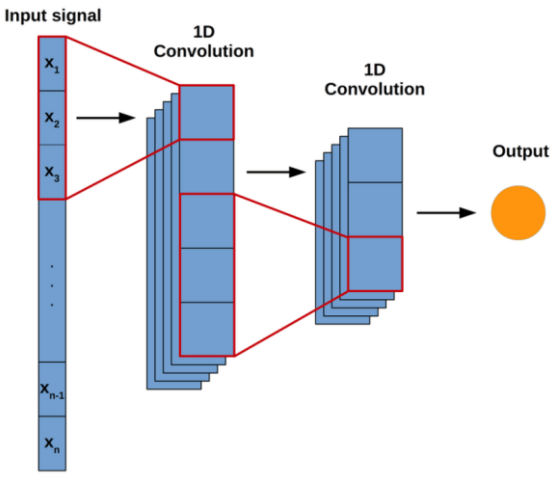
ellenőrzésként visszafordítjuk őket a kamera paraméterek segítségével és az eredetihez közeli eredményt várunk.



15. ábra Temporális konvolúció

4.1 Temporális konvolúció

A temporális konvolúciós hálókról általánosságban elmondható, hogy 1D konvolúciós hálót használnak a bemenet hosszát felfoghatjuk az idő dimenziója ként vagy képkockánkénti információként. Ahhoz, hogy a kimenet vagy több réteg esetén a következő réteg egyik elemét megkapjuk "k-elemre alkalmazzuk az 1D-s konvolúciós kernelt", ahol k a kernel mérete majd a következő elemhez a kernelt eggyel arrébb csúsztatjuk. Több bemeneti csatorna esetén a kernelünk k * C ahol C a bemeneti csatornák száma. Így végső soron 2D-s konvolúciót hajtunk végre, de továbbra is csak 1 dimenzió mentén haladunk és persze több csatornából több csatornába képzés esetén minden kernelnek külön súlyokat kap.



Causal konvolúció

Megkülönböztetünk még ezen kívül causal temporális konvolúciót, ahol minden i-edik elemhez a kimenetben csak a bemenet $\{0, \dots, i-1\}$ elemeit használhatjuk. Ez végső soron azt jelenti, hogy egy képen a koordináták meghatározásához nem vehetjük számba a jövőbeli képeket ez felvett videó esetén egy felesleges korlátozás, de élő videófelvételnél szükséges.

Dilated temporal konvolúció

Az egyik jó tulajdonsága a szekvenciákból jósoló modelleknek, hogy akár az összes bemeneti adat hozzájárul 1 kimeneti adat értékéhez (vagy causal konvolúció esetén az összes kisebb indexű bemeneti adat). Ezt nevezhetjük teljes „történeti fedettségnek". Ahogy ezelőtt láttuk egy konvolúciós réteg ráhatása a következőre vagy a kimenetre a kernel méretétől függ, ha van n rétegünk es a kernel mérete k akkor a befogadó mező

$$r = 1 + n * (k - 1)$$

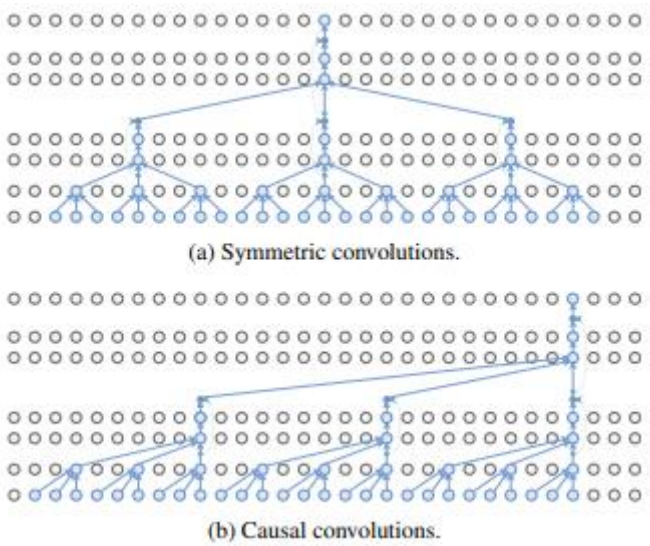
ha meg akarjuk nézni, hogy hány réteg l szükséges a teljes lefedettséghez

$$n = \lceil (l-1)/(k-1) \rceil$$

ez nagyobb bemenet eseten könnyen nagyon mély hálóhoz vezetne rengeteg paraméterrel és hosszú tanulási idővel. Ehelyett bevezethetjük a dilatációt, ami exponenciálisán nagy befogadó mezőt eredményez. A dilatált konvolúció a konvolúció egy sajátos formája ritka szerkezettel, amelynek magpontjai egymástól távol helyezkednek el egyenletesen és közöttük nullákkal töltve. Például, egy diszkrét szűrő h = [1 2 3] (ahol 2 a középpont) d = 2 dilatációs faktorral [1 0 2 0 3] lesz, és [1 0 0 2 0 0 3], ahol d = 3. Ezt a dilatációt rétegenként exponenciálisan növelve kapjuk hogy a látómező

$$r = 1 + \Sigma_{i=0}^{n-1}(k-1)b^i = 1 + (k-1)\frac{b^n - 1}{b-1}$$

ahol b a dilatációs bázis (általában 2 vagy 3)



16. ábra Táguló ideiglenes konvolúció és táguló aktuális konvolúció (b = 3)

17. ábra Bemenetünk 234, képkocka 34 ízületi koordináta (2x17) B = 4 konvolúciós blokk alkotja a modell törzsét zölddel a bemenetek, ahol 2J = 2x17 ízületi koordináta 3d1 = 3-as kernel méret 1 dilatáció és 1024 kimeneti csatorna

4.2 Veszteségfüggvények

A korai munkákban a 3D-s modellek jóságát olyan veszteségfüggvényekkel mérték melyek főleg a „korrektül” detektált (a prediktált és a valós pozíció közti eltérés egy bizonyos határon belül marad - ami általában a végtag hossza * egy arányszám volt) testrészek számát vagy arányát nézték. Manapság inkább az alábbi függvényeket használják

MPJPE – Az átlagos prediktált ízületi koordináták euklideszi távolsága a groundtruth póz koordinátáitól.

$$\boldsymbol{MPJPE(f) = \frac{1}{j} \Sigma_{i=1}^{J} \left|\left| S_p^{(f)}(i) - S_g t^{(f)}(i) \right|\right|_2}$$

ahol f a frame j az ízületek száma S_p a prediktalt és S_{gt} a valós pozíció

ennek egy másik fajtája a

PA-MPJPE először egy Procrustes analízist [8] végez (felskálázással és/vagy forgatással próbálja a két formát a leghasonlóbb alakra hozni) és utána méri a fenti távolságot.

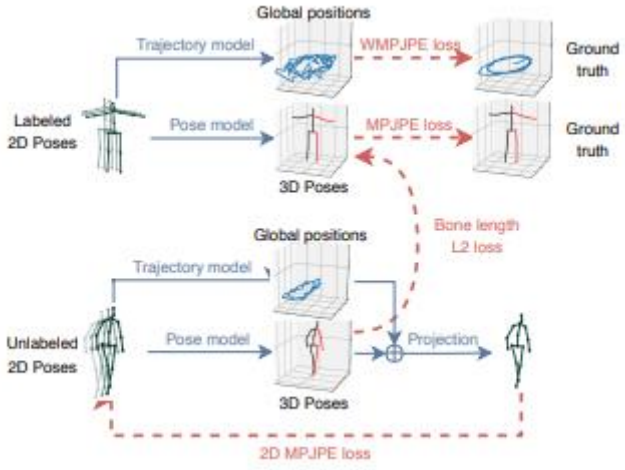
Ezen veszteségfüggvények viszont érzékenyek arra, hogy mekkora az érzékelt ember hiszen minél közelebb van a képen (értsd minél nagyobb) annál nagyobb lenne az eltérés is ezért egy normalizációt hajtanak végre (pl: L2 norm) és az ízületi szögeket megőrizve egy standard csontvázat kapunk fix végtaghosszal és ezen alkalmazzuk a vesztesgfüggvényeket.

4.3 Modell architektúrája

Modellünk 17.ábra egy teljesen konvolúciós architektúra. A bemeneti réteg megkapja az összefűzőt (x, y) koordinátáit a J ember ízületeinek, és időbeli konvolúciót alkalmaz k kernel mérettel és C csatorna számmal. Ezt követik a B ResNet-stílusú blokkok, amelyeket egy „skip connection" [9] vesz körül. Minden blokk először egy 1D konvolúciót hajt végre k kernelmérettel és $D = k^B$ dilatációs tényezővel, majd ezt követi egy konvolúció 1-es kernelmérettel. A konvolúciókat (a legutolsó réteg kivételével) követi egy batchnormalizáció, majd egy ReLu aktiváció és végül egy „drop out" [10] réteg. Minden blokk exponenciálisan növeli a befogadó mezőt k-szorzóval, miközben a paraméterek száma csak lineárisan nő. A szűrő hiperparamétereit, k és D, úgy állítjuk be, hogy a számunkra ideális nagyságú receptív mezőt teljesen lefedje. Végül az utolsó réteg előrejelzést ad a 3D-s pózokról minden képkockára a bemeneti szekvenciában felhasználva múltbeli és jövőbeli adatok egyaránt.

4.4 Félig-felügyelt megközelítés

Megpróbálunk fel nem címkézett videóképekből is információt kinyerni annak érdekében, hogy javítsuk a modellünk veszteségfüggvényén. Megoldunk egy autoencoding problémát a címkézetlen adatokkal: a kódoló (pozíció becslés) 3D-s pózbecslést végez a 2D-s ízületi koordinátákon és egy dekódoló visszavetíti ezeket a 3D-s pózokat 2D-s koordinátákra. A tanítás során pedig „büntetjük", ha a dekódoló koordinátái nagyon eltérnek az eredetitől. Az alábbi képen látható, ahogy a 2 modell együtt működik és a félig-felügyelt modell egyfajta szabályozóként funkcionál. A két modellt együttesen optimalizáljuk, ahol a batch első felét a felcímkézett adatok teszik ki míg a másodikat a felcímkézetlen adatok. A felcímkézett adatokhoz az groundtruth 3D-s koordinátákat használjuk célként a veszteségfüggvényben míg a felcímkézetlen adatoknál az autoencoder veszteségét használjuk.



18. ábra Félig-felügyelt tanítás

Csonthossz L2 veszteség

Szeretnénk ösztönözni a hihető 3D-s koordináták prediktálását a félig-felügyelt módszer esetén is. Ahelyett, hogy csak „elhisszük" a már betanult paraméterek alapján prediktált kimenetet, bevezetünk egy újabb veszteségfüggvényt, ami összehasonlítja a csonthossz-t a felügyelt és a félig-felügyelt módszer prediktálásánál és büntetjük a nagy eltérést. Ez akkor hoz előnyöket, ha a tanítás során kevés a címkézett adat.

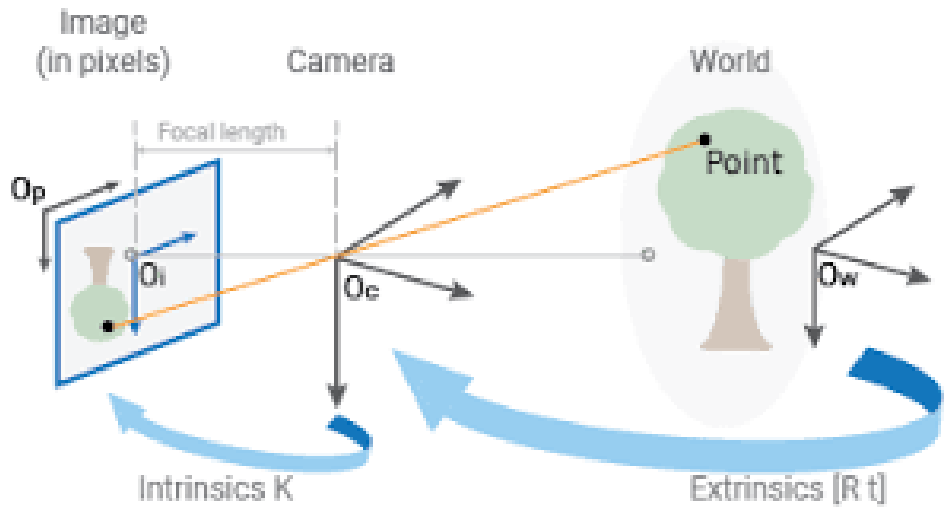
Visszavetítés [11]

A tanitáshoz és értékeléshez szükségesek még a kamera belső paraméterei (ami a legtöbb ma használatos kameránál adott) mivel ezek segítségével tudjuk a félig felügyelt módszernél visszavetíteni a 3D-s modellt 2-dimenzióba. Ebből a témából külön szakdolgozatot lehetne írni ezért csak nagy vonalakban magyaráznám el. A fenti probléma a geometriában felfogható egy egyszerű projekciónak. A valós (X, Y, Z) koordináták felírhatóak homogén koordinátákkal (x_s, y_s) a következő leképzéssel $\vec{q} = M\vec{Q}$

ahol $\vec{q} = \begin{bmatrix} x \\ y \\ w \end{bmatrix}$ $\vec{Q} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}$ és $M = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$

ahol f_x, f_y a fokális távolság (a projekció centrumától milyen messze helyezkedik el a „vászon" amire rögzítünk) és c_x és c_y a principal point offset (a principal point az a pont, ahova a projekció centruma vetül a „vásznunkon" és az offset pedig ennek a lehetséges eltérése a „vászon" közepétől). Így megkapjuk az alakját a pózunknak, de a képen a pontos elhelyezkedését más módszerrel határozzák meg, ezért nincs szükség az extrinsic

paraméterekre (ezek írják le a kamera pozícióját térbeli koordinátákkal és azt, hogy merre néz a kameránk. Ezek nélkül azt feltételezzük, hogy a kameránk az origóban helyezkedik el).



19. ábra Projekció

4.5 Modell tanítása és értékelése

Idő és adat szűkében én nem optimalizáltam a saját feladatunkra a modellt, de szükségesnek tartottam ismertetni hogyan is történik a betanítás és az értékelés mind a tanítás mind pedig egy inferencia során.

A tanítás és az értékelés is kamera térben történik (a kameránk szemszögéből és minden egyes prediktált objektumon az $x = 0$ $y = 0$ pontba transzformáljuk a „root jointot”, ami esetünkben a csípőcsontok felezőpontja). Egyszerre mindig csak 1 embert tudunk prediktálni ezért a kimenet egy olyan 3-dimenziós csontváz lesz, ami mindig a kép közepén helyezkedik el fix végtaghosszokkal. Ezért lesz a veszteségfüggvényünk következetes. A tanítás során próbálunk egymástól független videórészleteken tanítani úgy, hogy a videóklip hossza egyezen a befogadó mező méretével (243 képkocka) és mindig csak egy kimeneti pozíciót prediktálunk. A batch normalizáción paraméterei is eltérnek a szokásostól a 4 blokkon (8 normalizáción át) a béta hyperparamétert 0.1-ről 0.001-re csökkentjük. Ami azt jelenti, hogy a mozgó átlag szórása és várható értéke szinte egyáltalán nem függ az előző rétegektől

A tanitásnál egy batchet oly módon töltünk fel, hogy az első felében csak felcímkézett adatokon értékelünk a másik felében pedig ugyanezen adatokon a félig-felügyelt modellt tanítjuk. Az leső részben az adathalmazt elkészítő csapat 3 protokollt is leír a modell jóságának mérésére (veszteség függvények) ezekből főleg az első kettőt használják (MPJPE és PA-MPJPE). Szükséges még megjegyezni a félig felügyelt modell visszavetítését. Itt mivel mindig egy

középre eltolt embert kapunk a visszavetített alak sosem egyezne az eredeti képen láthatóval. Ezért erre egy külön modellt tanítanak, ami ezt a lokalizációs hibát minimalizálja egy regresszióval. Tehát főleg videóknál az ember mozgása alapján próbálnak jósolni egy pályát, amivel a megfelelő helyre vetítik vissza a képre, mivel az eredményünknek fix csonthossza van és a távolságot vagy mélységet nem igazán tükrözi. A két architektúra megegyezik, de nem osztoznak a súlyokon mert ez negatívan hat a tanulás során. Mivel egyre nehezebb a szabályozni a pályát, ha egy ember messzebb van a kamerától, ezért optimalizáljuk a súlyozott ízületi pozíció hiba átlagot (W-MPJPE) veszteségfüggvényt

$$E = \frac{1}{y_z} \left|\left| f(x) - y \right|\right|$$

vagyis minden minta súlyozva van az igazi mélység inverzével a kameratérben. Szabályozni egy precíz pályát a messzi embereknél relatíve felesleges hísz az ízületek amúgy is egy kis pont köré tömörülnek.

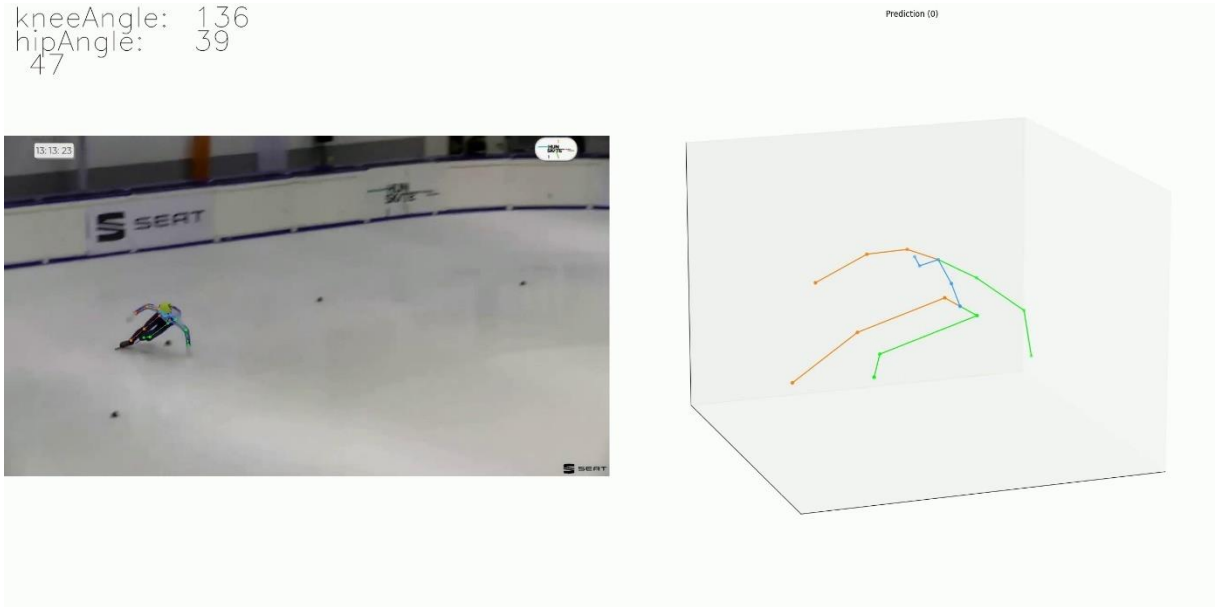
5. fejezet

Alkalmazás

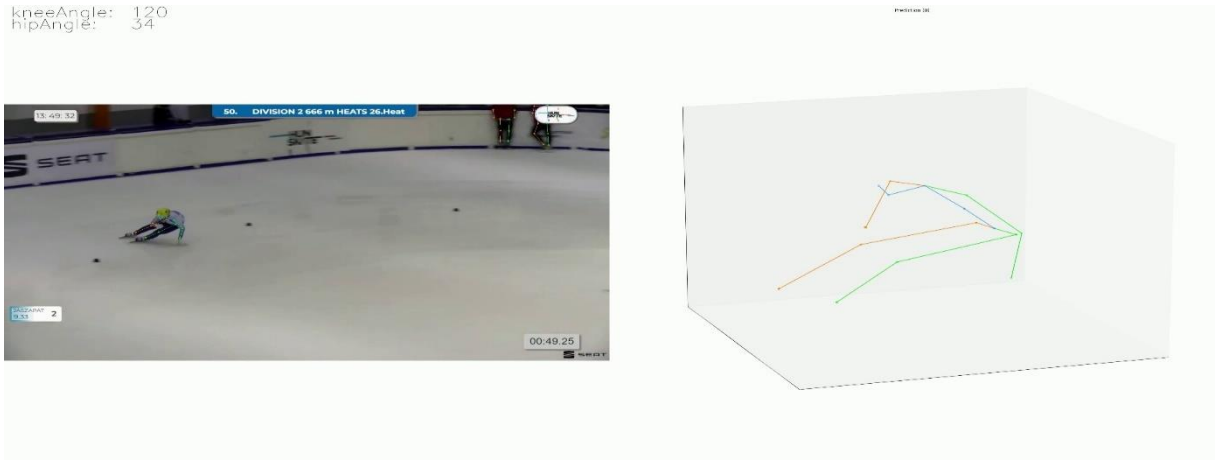
Végül bemutatom, hogy az esetemben ezt a technológiát hogyan alkalmaztam rövidpályás gyorskorcsolya videók elemzésére. A modellben szerencsére van implementált funkció mind a kapott koordináták fájlba mentésére és ezek vizualizálására is. A koordinátákat egy .json fájlba mentve könnyen lehet olvasni és képkockánként értékelni az eredményeket. A sportágban a gyors sebesség miatt kulcsfontosságot kap a videóelemzés, és ebben a különböző ízületi szögek mérése. Ezeket pedig a térbeli koordinátákkal egyszerűen ki lehet számolni:

$$\theta = arccos\left(\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}\right)$$

A következő képeken látható, hogy lehet pozíciók mélységét (térdszöget) kanyarban a centrifugális erő elleni bedőlés szögét (dőlés szöget) mérni. Utóbbinál kihasználjuk, hogy a nagy sebesség és gyorsulás miatt fellépő ellenerők stabilizálására a kanyar közepe környékén lerakjuk a kezünket a jégre. Ezzel a boka és a kéz vonala nagyjából azonosítja a képen a vízszintet és erre tudunk mérni egy dőlésszöget. Persze amelyik képen nincs jégen a kéz ott ez az érték figyelmen kívül hagyandó. Mérhető még egyes pozíciókban a lökés teljessége is (eléri-e a lábnyújtás a 180º-ot).



20. ábra Jobb térdszög 136º dőlés szög 39º

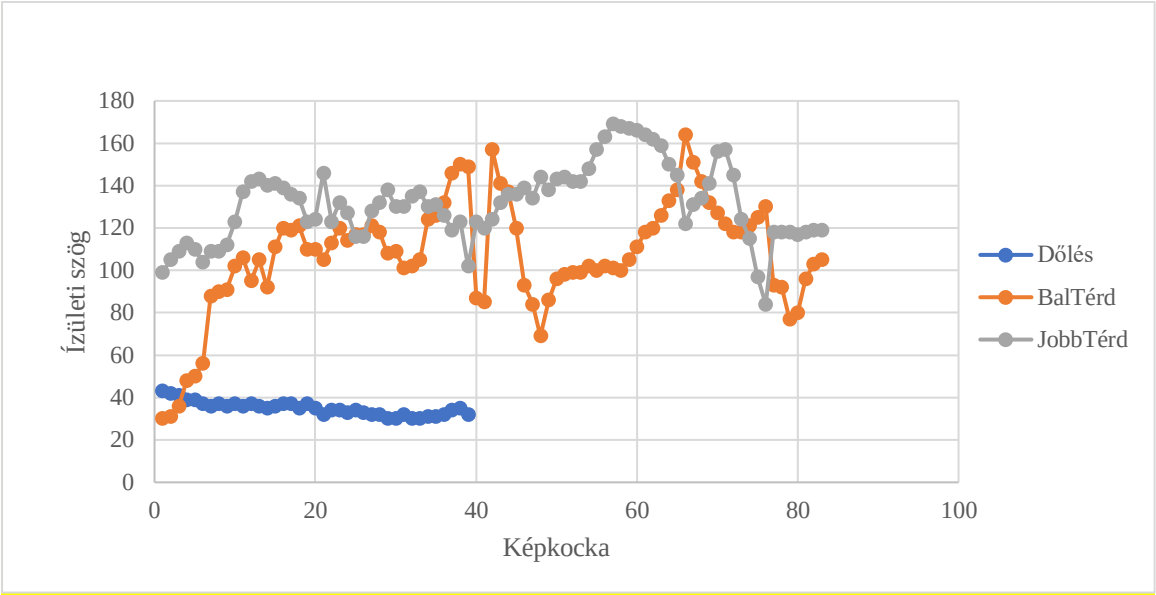


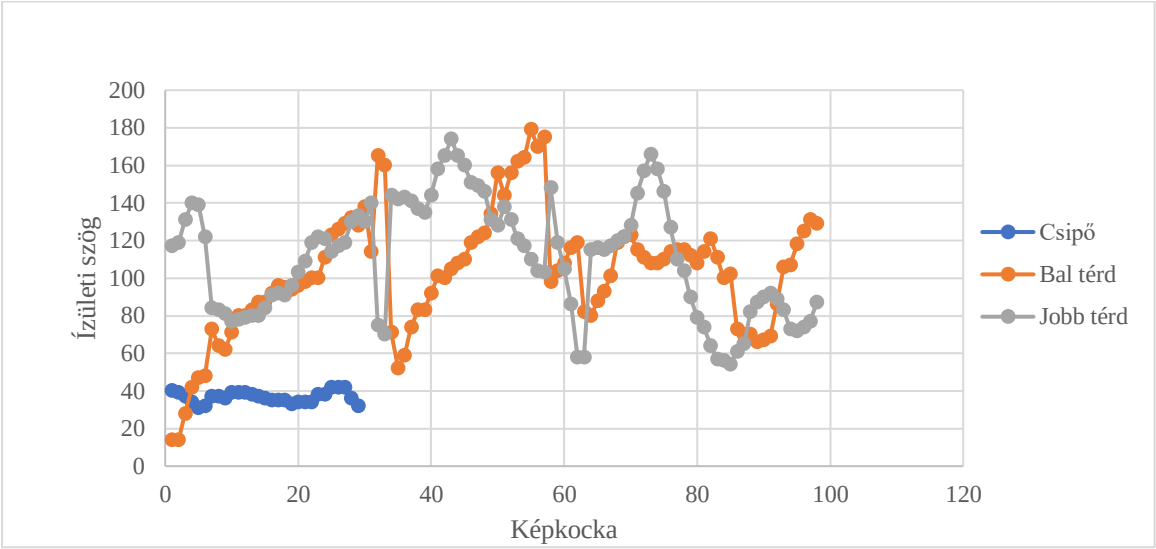
21. ábra Térdszög 120º dőlés szög 34º

22. ábra Térdszög 175º

23. ábra Lábak összekeverése

A képekből kinyert ízületi szögekre tekinthetünk akár idősorként is. Így ezen eredmények elemzésére más módszereket is használhatunk. Megpróbálhatjuk az esetleges észlelési hibákból adódó zajt csökkenteni. A kapott eredményekre függvényt illeszteni. Időben ismétlődő trendeket keresni. Esetleg két korcsolyázó mozgását is vizuálisan könnyebben tudjuk összehasonlítani.





Természetesen a gyorskorcsolya rendkívül speciális mozgású sport és lenyűgöző, hogy a modell relatíve mennyire jól kiigazodott ezen a mozgáskultúrán, de néha összekevert pár lábat és természetesen a 3D-s koordináták és a belőlük számolt szögek sem voltak mindig közel a valósághoz. Ennek ellenére elképzelhető, hogy a jövőben nem kellenek majd ruhába vart

szenzorok és egyéb trükkök arra, hogy hatékonyan elemezhessük ezen extrém mozgású sportokat is csupán videófelvételekből.

Irodalomjegyzék

[1] A. Géron, Hands-On Machine Learning with Scikit-Learn & TensorFlow, O'Reilly Media, Inc, 2019.

[2] I. Godfellow, Y. Bengio és A. Courville, Deep Learning, http://www.deeplearningbook.org: MIT Press, 2016.

[3] K. Fukushima, „https://www.cs.princeton.edu," [Online]. Available: https://www.cs.princeton.edu/courses/archive/spr08/cos598B/Readings/Fukushima1980.pdf.

[4] G. Gkioxari, P. Dollár and R. Girshick, "Mask R-CNN," 2017. [Online]. Available: https://openaccess.thecvf.com/content_ICCV_2017/papers/He_Mask_R-CNN_ICCV_2017_paper.pdf.

[5] R. Shaoqing, H. Kaiming, R. Girshick és J. Sun, „Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks," 2016. [Online]. Available: https://arxiv.org/pdf/1506.01497.pdf.

[6] X. Bin, W. Haiping és W. Yichen, „Simple Baselines for Human Pose Estimation and Tracking," 2018. [Online]. Available: https://arxiv.org/pdf/1804.06208.pdf.

[7] D. Pavllo, C. Feichtenhofer, D. Grangier és M. Auli, „3D human pose estimation in video with temporal convolutions and semi-supervised training," 2019. [Online]. Available: https://arxiv.org/pdf/1811.11742.pdf.

[8] A. Ross, „https://www.cse.sc.edu," [Online]. Available: https://www.cse.sc.edu/~songwang/CourseProj/proj2004/ross/ross.pdf.

[9] D. Wu, W. Yisen , X. Shu-Tao, J. Bailey és M. Xingjun, „SKIP CONNECTIONS MATTER: ON THE TRANSFERABILITY OF ADVERSARIAL EXAMPLES GENERATED," [Online]. Available: https://arxiv.org/pdf/2002.05990.pdf.

[10] N. Srivastava, G. Hinton, . K. Alex, S. Ilya és . S. Ruslan, „Dropout: A Simple Way to Prevent Neural Networks from,” [Online]. Available: https://www.cs.toronto.edu/~rsalakhu/papers/srivastava14a.pdf.

[11] Unknown, „https://www.baeldung.com,” [Online]. Available: https://www.baeldung.com/cs/focal-length-intrinsic-camera-parameters.