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

Dancsó Marcell

JELNYELV SZÖVEGGÉ FORDÍTÁSA SZEKVENCIAFELDOLGOZÁSRA ALKALMAS MESTERSÉGES INTELLIGENCIA MODELLEKKEL

KONZULENS

Dr. Ekler Péter

BUDAPEST, 2023

Tartalomjegyzék

HALLGATÓI NYILATKOZAT

Alulírott **Dancsó Marcell**, szigorló hallgató kijelentem, hogy ezt a szakdolgozatot meg nem engedett segítség nélkül, saját magam készítettem, csak a megadott forrásokat (szakirodalom, eszközök stb.) használtam fel. Minden olyan részt, melyet szó szerint, vagy azonos értelemben, de átfogalmazva más forrásból átvettem, egyértelműen, a forrás megadásával megjelöltem.

Hozzájárulok, hogy a jelen munkám alapadatait (szerző(k), cím, angol és magyar nyelvű tartalmi kivonat, készítés éve, konzulens(ek) neve) a BME VIK nyilvánosan hozzáférhető elektronikus formában, a munka teljes szövegét pedig az egyetem belső hálózatán keresztül (vagy hitelesített felhasználók számára) közzétegye. Kijelentem, hogy a benyújtott munka és annak elektronikus verziója megegyezik. Dékáni engedéllyel titkosított diplomatervek esetén a dolgozat szövege csak 3 év eltelte után válik hozzáférhetővé.

Kelt: Budapest, 2023. 12. 05.

...…….
Dancsó Marcell

Összefoglaló

A nyelvfeldolgozás terén elért sikerek rengetek kaput nyitottak ki előttünk. A hangalapú asszisztensek, alap esetben is hasznos, de legtöbbször forradalmi funkciókat hoznak életünkbe. A mesterséges intelligencián alapuló „voice to text” modellek manapság gyakorlatilag tetszőleges nyelvről képesek felismerni szöveget, valamint a közelmúltban nyelvfeldolgozás területén elért eredményeknek köszönhetően pedig nincsenek határok, a felhasználók, és fejlesztők előtt. Illetve előbbi állítás sajnos csak egy nagy ferdítéssel igaz. Ugyanis világszerte körülbelül 1,5 milliárd [1] ember hallássérült, és több mint 70 millió [2] ember használja a jelnyelvet, mint elsődleges kommunikációs formát. Számukra „saját nyelvükön” ezek a funkciók nem, vagy csak korlátozottan érhetőek el. A jelnyelv egy speciális fajtája az ujjbetűzés. Gyakran használják nevek, címek, telefonszámok, valamint olyan fogalmak közvetítése során, melyekre nincs bevett gesztus. Egy tapasztalt jelelő képes közel kétszer olyan sebesen betűzni, mint egy virtuális billentyűzeten pötyögő egyén, nem beszélve arról, ha minden jelet alkalmazva kommunikál. Így égető egy számukra is kényelmesen használható interfész kialakítása.

Még jelentősebb problémát vet fel a kommunikáció kérdése. Manapság a nyelvek közötti gépi fordítás széles körben és formátumban elérhető az internet kapcsolattal rendelkezőknek. Régen túl vagyunk már az egyszerű szótár alapú fordítókon, a különböző transzformer, és nagy nyelvi modellek térhódításával gyakorlatilag tetszőleges nyelvek között megoldható a kommunikáció. Ez alól kivétel azonban a jelnyelv. Nem létezik olyan megoldás a piacon, ami segíti a jelelő kommunikációját a jelelni nem tudó személy felé. A terület előrehaladásán nem segít, hogy a jelnyelv csakúgy, mint a verbális párjai, nem standardizált. Szinte minden nemzetnek saját jelnyelve van, nem beszélve a helyi sajátosságokról, és dialektusokról. Tovább nehezíti a feladatot, hogy a rendelkezésre álló, nagy méretű, és jó minőségű adatbázisok száma eltörpül a klasszikus nyelvfeldolgozásban megszokottaktól.

Munkámmal a széles tömegek számára elérhető jelnyelv fordító rendszerek fejlesztéséhez járulok hozzá. Megvizsgálom többek között a kesztyűvel történő felismerés, valamint hagyományos képfeldolgozás eredményeit, továbbá részletesen foglalkozom a póz approximáción alapuló algoritmusokkal. A nyílt kérdés utóbbival kapcsolatban, hogy bár rendkívül jó arányban tömöríti a képi adatot, ami lehetővé tenné

nagy mennyiségű, egységes adatbázis konstruálását, de vajon tart-e ott a technológia, hogy megbízható módon kódoljon minden fordításhoz szükséges információt?

Az amerikai jelnyelven belül külön-külön vizsgálom az ujjbetűzés, és általános jelbeszéd lehetőségeit, kitérve a pillanatképből, valamint mozgásszekvenciából dolgozó megoldásokra. A teljesség igénye nélkül összehasonlítom a feladatra adaptált konvolúciós, LSTM, transzformer háló architektúrákat. A jelelés nyelvtani adottságai következtében, hiányoznak a segédigék, valamint gyakran más a szórend a hagyományos angolhoz képest. A primitív fordítás eredményeit ezért kontextus függő módon transzformálom generatív nagy nyelvi modellekkel, valamint az ujjbetűzés esetében a megbízhatóság növelésére is felhasználom őket.

A dolgozatomban bemutatom az adatgyűjtés és a tervezés lépéseit, az alkalmazott mesterséges intelligencia algoritmusok részleteit, valamint elemzem a fejlesztés során felmerült tervezői döntéseket, és alternatíváikat. Ezen felül bemutatásra kerül az általam készített teljes megoldás, mely nagymértékben segíthet a hallássérültekkel való kommunikáció során.

Abstract

The successes achieved in the field of natural language processing have opened numerous doors before us. Voice-based assistants are not only useful, but they often introduce revolutionary functionalities into our lives. Nowadays, artificial intelligence-based voice-to-text models can recognize text from virtually any language, and thanks to recent advancements in natural language processing, there are no boundaries for users and developers. However, the previous statement is a significant distortion of the truth. Globally, approximately 1.5 billion [1] people are hearing impaired, and more than 70 million [2] people use sign language as their primary means of communication. For them, these functionalities are either not available in their „own language", or only in a limited manner. Sign language, particularly finger-spelling, is a special form of communication often used for communicating names, addresses, and phone numbers, or conveying concepts without standard gestures. An experienced signer can spell almost twice as fast as an individual typing on a virtual keyboard, let alone when using all available signs for communication. Hence, there is an urgent need to create an interface that is comfortable for them to use.

An even more significant issue arises in the realm of communication. Machine translation between spoken languages is widely available to those with internet access in various formats. We have long surpassed simple dictionary-based translators, and with the rise of different transformers and large language models, communication between virtually any language has long become a reality. However, sign language even today remains an exception. No solution on the market assists signers in communicating with non-signing individuals. The progress in this area is hindered by the fact that sign language, like its verbal counterparts, is not standardized. Almost every nation has its own sign language, not to mention regional variations and dialects. Furthermore, the task is complicated by the fact that the available large and high-quality datasets are minuscule compared to what is customary in classical natural language processing.

My work contributes to the development of sign language translation systems accessible to the broader masses. I have conducted research encompassing various aspects, including glove-based recognition and traditional image processing methods. However, my primary focus was directed towards the comprehensive exploration of using

pose approximation algorithms. While the latter is highly promising as it compresses visual data effectively, enabling the construction of large, homogeneous databases, the question of whether current technology is capable of reliably encoding all the information needed for translation has so far remained open.

Within American Sign Language, I separately explored finger-spelling and general sign language possibilities, paying close attention to solutions working from both snapshots and motion sequences. Without aiming for completeness, I compared convolutional, recurrent, LSTM, and Transformer network architectures adapted for the task. Due to the grammatical characteristics of sign language, auxiliary verbs are missing, and the word order is often different from traditional English when translated word by word. Therefore, I processed the results contextually using generative large language models and employed them to enhance the quality and reliability of the translation.

In my thesis, I walk through the steps of data collection, provide details about the applied artificial intelligence algorithms, and analyze the design decisions and their alternatives that emerged during development. Furthermore, I present the comprehensive solution I've developed, offering significant support for communication with the hearing impaired.

1 Bevezetés

A nyelvfeldolgozás területén hatalmas áttöréseket értünk el az elmúlt években. A mai „voice to text” modellekkel lehetőségünk van rendkívül sokszínű és természetes módon interakcióba lépni az eszközeinkkel. Kiegészülve a világot alapjaiban megrengető intelligens nagy nyelvi modellekkel, saját virtuális asszisztenst alkalmazhatunk, chat formában érhetjük el a világ tudását és nem utolsó sorban kinyílik a kapu a szabad kommunikáció előtt tetszőleges nyelvek között. A hasonló rendszerek egyik sajnálatos velejárója, hogy csupán hagyományos nyelveken érhetőek el. Bár látszólag a technikai tudás rendelkezésre áll, még sem jelent meg még számottevő jelnyelvet támogató rendszer a piacon. A jelnyelv szöveggé formálásával közel 70 millió (William Woods University, 2016) embernek nyílna lehetősége saját nyelvén interakcióba lépni okos eszközökkel, nem beszélve a jelelni nem tudók felé irányuló kommunikációról. Nagy segítséget nyújthatna továbbá a tanulásban azoknak, akik annak ellenére, hogy hallássérüléssel kénytelenek élni, nincs lehetőségük megtanulni, mind pénzügyi, mind tanulást segítő eszközök híján. Ezen csoport mérete a legtöbb országban meglepően még jelentősebb, mint a jelelni tudók száma, bizonyítva a nyelv komplexitását, és ezzel a feladat nehézségét.

1.1 Jelnyelv alapok

A jelnyelv egy vizuális gesztusokra épülő nyelv, amelyet a hallássérült és a halló közösségek egyaránt használnak a kommunikációhoz. A beszélt nyelvektől eltérően a jelnyelvben a jelentést a kézmozdulatok, az arc kifejezései és a testtartás hordozzák.

Míg sokan tévesen úgy gondolják, hogy a jelnyelv egyetemes, valójában számos különböző változata létezik világszerte. Minden ország rendelkezik sajáttal. Érdekes például, hogy az amerikai jelnyelv (ASL) és a brit jelnyelv (BSL) annyira különbözőek, hogy nem érthetőek egymás számára, annak ellenére, hogy mindkét ország angolul beszél. A két nyelvnek mellesleg teljesen különböző a fejlődése, míg az ASL-nek francia jelnyelvi gyökerei vannak [3], addig a BSL teljesen külön fejlődött [4]. Továbbá hasonlóan a verbális nyelvekhez, itt is léteznek dialektusok, helyi sajátosságok, ami további komplexitáshoz vezet intelligens megoldások kialakításakor.

A legtöbb jelnyelv egy speciális részhalmaza az ujjbetűzés, amely kézzel formázott betűk sorozata. Néhány jelnyelvben az ujjbetűzést gyakran használják nevek, idegen szavak vagy specifikus terminológiák kifejezésére, amelyeknek nincs saját jelük. Más jelnyelvekben az ujjbetűzést csak ritkán használják, és az emberek inkább a teljes kifejezések és mondatok használatát részesítik előnyben. Számos kisebb jelnyelv létezik világszerte, amelyek különleges közösségekben alakultak ki és nincsenek befolyásolva a nagyobb jelnyelvektől. A Kolok Kata [5], más néven balinéz jelnyelv, egy falusi jelnyelv, amely két szomszédos faluban őshonos Bali északi részén, Indonéziában. A Kata Kolokban nincs hivatalos ujjbetűzés, ami rámutat arra, hogy az nem szükséges egy jelnyelv működéséhez. Ezek a kisebb jelnyelvek gyakran egyedülállóak és tükrözik az adott közösség kultúráját és történelmét.

Mint minden nyelvnek, a jelnyelveknek is van saját nyelvtana és szókincse. A szavak és mondatok jelentését a kézformák, helyzetek, mozgások és az arc kifejezései együttesen hordozzák. A nyelvtan komplex és nem feltétlenül követi a hallók által beszélt nyelv szerkezetét.

1.2 Amerikai jelnyelv (ASL)

Az amerikai jelnyelv (ASL) az Egyesült Államokban és Kanadában élő hallássérült közösségek által használt nyelv. Az ASL története a 19. századig nyúlik vissza, amikor Thomas Gallaudet és Laurent Clerc megalapították az Egyesült Államok első hallássérült iskoláját. [3] Az itt használt jelnyelv kombinálta az amerikai helyi jelnyelveket a francia jelnyelvvel, ami az ASL alapját képezte.

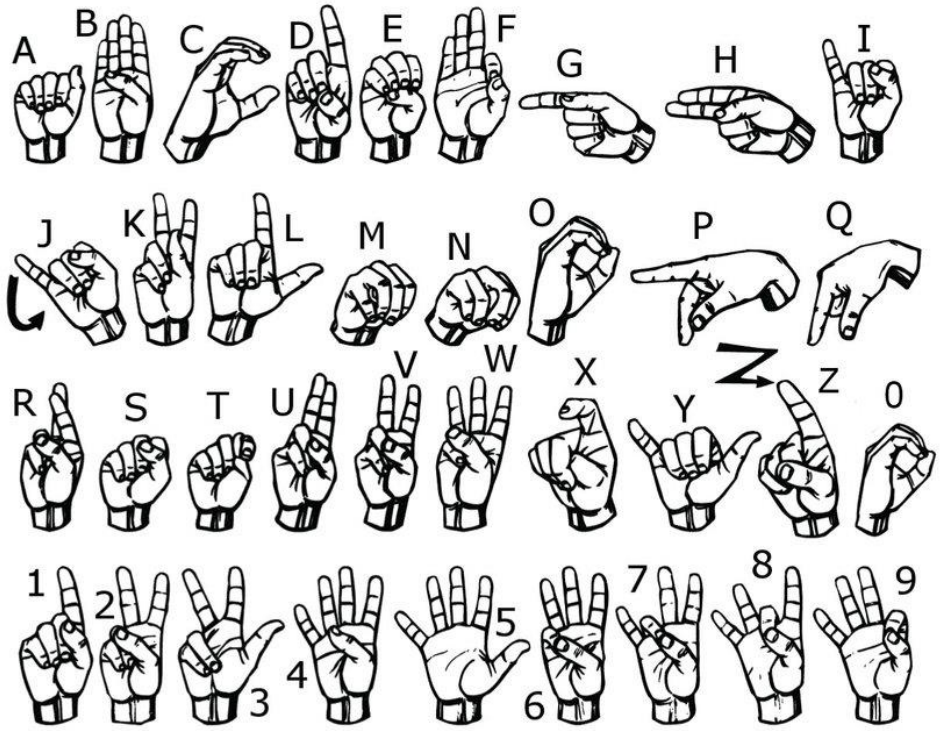
Közel félmillió ember beszéli, ami meglepő, hiszen ez a szám körülbelül 10 százaléka a 40 millió hallássérültnek az országban. Az ASL körülbelül a 7. legtöbbet használt, viszont az interneten szabadon hozzáférhető források által legjobban dokumentált jelnyelv a világon, így a dolgozat is ezzel foglalkozik. Fontos kiemelni azonban, hogy hasonlóan rögzített adathalmazok esetén a modellek, és algoritmusok univerzálisak.

1.2.1 Ujjbetűzés az amerikai jelnyelvben

Mint sok más jelnyelvben, az amerikaiban is található ujjbetűzés. A brit jelnyelvvel ellentétben itt csupán egy kézre van szükség, a számok, illetve angol abc betűinek kommunikációjára. Ez persze nem azt jelenti, hogy az egyéb gesztusoknak nincs

jelentése ebben a kontextusban, viszont a karakterek teljes mértékben beazonosíthatóak csupán a kezek megfigyelésével. pl.: A fejmozgás, vagy előre dőlő testhelyzet ujjbetűzés során hangsúlyozó szereppel bírhat.

Az ujjbetűzés olyan kézformákat használ, amelyek egyes betűket képviselnek, hogy kifejezzenek szavakat. Bár az ujjbetűzés csak az ASL egy része, használják nevek, címek, telefonszámok és egyéb, mobiltelefonon gyakran beírt információk közlésére. Sok siket okostelefon-felhasználó gyorsabban tudja ujjbetűzéssel kifejezni a szavakat mintsem, hogy begépelje őket. Valójában az ASL ujjbetűzése jelentősen gyorsabb lehet, mint a tipikus okostelefonos virtuális billentyűzeten való gépelés (átlagosan 57 szó/perc az amerikai 36 szó/perc átlagához képest [6]).



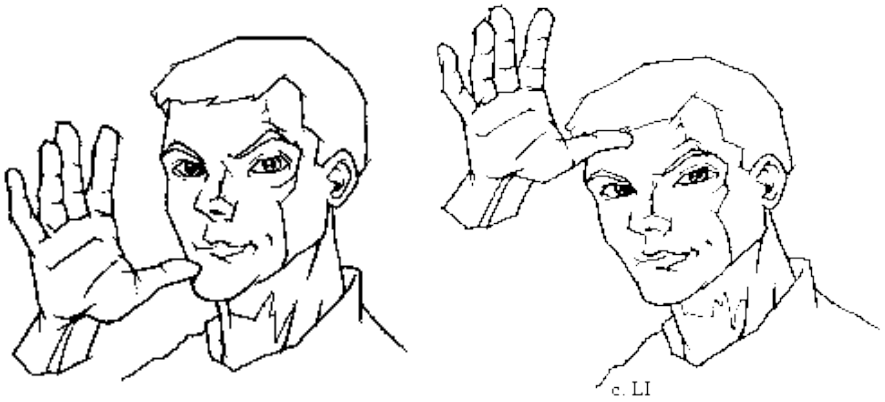
1.1 Ábra: Amerikai jelnyelv ujjbetűzés során használt lehetséges kéz formái.[1]

A legtöbb gesztus statikus, vagyis nem szükséges dinamikus mozdulat a megkülönböztetéshez. Ez alól kivétel a j és z betűk, melyeket ugyanazzal a kéztartással kell jelelni, mint az i és d betűt, csupán az ujjakkal leírt alakzatban térnek el. Ezen felül a számok esetében előfordul, hogy a kéztartás megegyezik egy betűnél használttal. Megkülönböztetni őket a kontextusból lehet.

[1] https://www.lifeprint.com/asl101/fingerspelling/images/abc1280x960.png

1.2.2 Általános ASL

A legfontosabb különbség az ujjbetűzéshez képest, hogy sokkal fontosabb szerepet játszik a fej, és mimika, valamint egyéb gesztusok is. Egy jó példa erre a jelenségre az Anya, illetve Apa gesztus, melyek ugyanúgy széttárt tenyérrel viszont a fej különböző részét érintve fejezhetőek ki. Ez nem is egyedi eset, léteznek adatbázisok, amelyekben kéz alak alapján lehet gesztusokat keresni. [7] Módosító tényező lehet még a másik kéz alakja, és a végrehajtott mozgáskombináció is.



1.2 Ábra: Balra az Anya, jobbra az Apa gesztus az amerikai jelnyelvben.[2]

A hagyományos angolhoz képest általában más a szórend. Tipikusan előre helyezi az idő vagy igeidő információt, majd a mondat témáját, és végül a megjegyzést. Például az "I am going to the store" (Én megyek a boltba) mondatot ASL-ben így jeleznénk: "NOW I STORE GO" (MOST ÉN BOLT MENNI), de az is gyakori, hogy az igét megismétlik a jelelés elején és végén. Ezen felül, mikor írásos formában akarjuk ábrázolni a jelnyelvet, még szokás vizuális annotációkkal is ellátni. Ezt nevezi az ASL „gloss” [8] vagy „glossing” -nak. Ebben az olyan jelek is feltüntetésre kerülnek, melyeknek nincs angol megfelelője, illetve egyéb „non-verbális” gesztusok. Például az ujjbetűzött szavak elé az „fs-’’ karakter sorozat kerül, a szavak felé pedig feltüntetésre kerülhet a szemöldök állása, illetve a birtokos eset is (poss).

poss eyebrows up
ME NAME fs-M-A-R-C-E-L-L

1.3 Ábra: Egyszerű példa az ASL írott változatára.

[2] https://www.lifeprint.com/asl101/pages-signs/m/momdad.htm

1.3 A dolgozat felépítése

A bevezető után megvizsgálom a téma irodalmát, kitérve a segédeszközöket felhasználó, valamint hagyományos képfeldolgozáson alapuló algoritmusokra. Összehasonlítom más kutatások eredményeit, valamint az alkalmazott módszerek hátrányait.

A következő fejezetben felvázolom az elkészített alkalmazás magas szintű architektúráját, vázat adva a további fejezetekben tárgyalt fejlesztési lépéseknek.

Részletesen bemutatom a Környezetből érkező jelek feldolgozására szolgáló algoritmikus apparátust, valamint elemzem a mesterséges intelligencia modellek tanítására gyűjtött és feldolgozott adathalmazokat. A következő fejezetben pedig külön-külön vizsgálom a póz approximáción alapuló ujjbetűzés lehetőségeit mind pillanatképekből, mind szekvenciális képkockákból, egy hagyományos jelelésre kialakított modellt, valamint részletezem a kettő ötvözésének lehetőségét. Ismertetem a kialakított modelleket, nagy hangsúlyt fektetve a tervezői döntések indoklására, alternatívák feltárására, továbbá számos metrika szerint kiértékelem, és összehasonlítom őket. A legjobb modellekhez különböző felhasználási módok szerint csoportosítva kiegészítő algoritmusokat tervezek.

Végül összefoglalom az eredményeket, további javaslatokat teszek a fejlesztésre, és megválaszolom a kérdést: Vajon a póz approximáció alkalmas-e jelen formájában komplex rendszerek kialakítására.

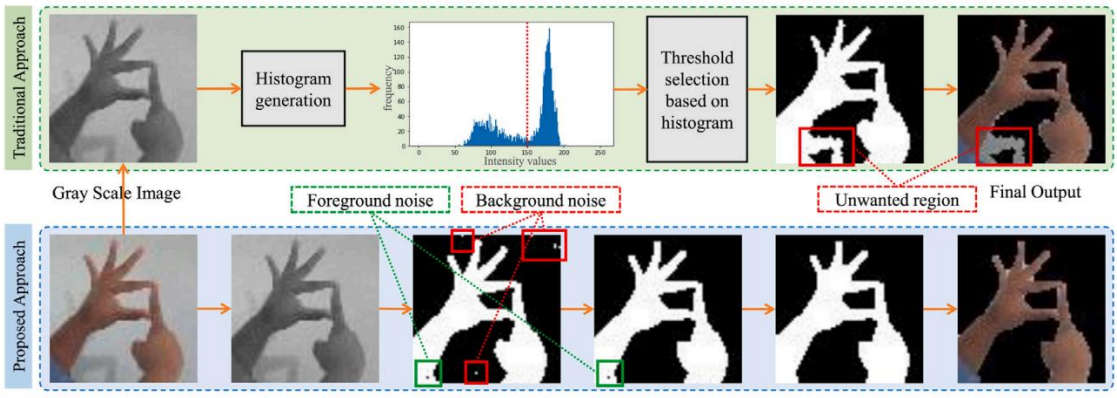
A tanulmány során fejlesztett kódom, valamint mintavideók a kész megoldásról elérhetők online a https://github.com/dancsomarci/sign-language linken.

2 Kapcsolódó kutatások

A második fejezet a jelnyelv-felismerési módszerek széles skáláját vizsgálja meg, a hagyományos képfeldolgozástól kezdve, a speciális segédeszközök használatáig. A cél az, hogy bemutassa a különböző technikák előnyeit és korlátait, valamint azokat a fejlődési irányokat, amelyek jelenleg a legígéretesebbnek tűnnek ezen a területen. A fejezet részletesen elemezi a képfeldolgozáson alapuló módszereket, amelyek a kamera által rögzített képeket dolgozzák fel és alakítják át értelmezhető jelekké. Ezt követően a segédeszközök használatát vizsgálja, mint például a speciálisan kialakított kesztyűk és intelligens gyűrűk, amelyek új perspektívákat nyitnak a gesztusfelismerés területén. A fejezet végül a dolgozat általános célkitűzéseit foglalja össze, amelyek középpontjában a hatékony, segédeszköz nélküli jelnyelv-fordítás áll.

2.1 Hagyományos képfeldolgozáson alapuló módszerek

Az ilyen módszeren alapuló megoldásokra igaz, hogy bár a sikeres projektek átalakítják a kamera képeit valamilyen magasabb szintű reprezentációba, de sosem lépnek ki a képi dimenzióból, nem modellezik a kéz fiziológiáját. Gyakori például a testrészek szegmentálása. [9] Vagy a dinamikus információ kinyerése az egyes képkockák pixeleinek különbségéből. [10]



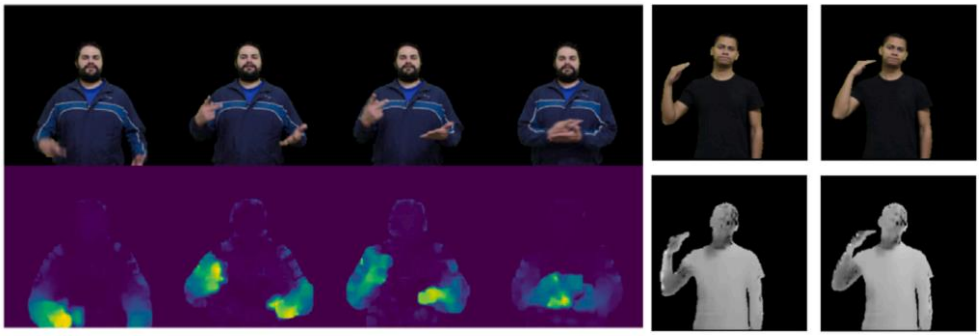
2.1 Ábra: Példa kezek szegmentálására Szín hisztogram segítségével. [9]

2.1.1 Jelnyelv automatikus fordítása többfolyamos 3D CNN-nel és mesterséges mélységtérképek generálásával

Ebben a kutatásban [11] videó képkockáiból azonosítottak be jeleket. Első lépésként a képkockák számát csökkentették le, továbbá egy előtanított VGG16 [12] nevű modellt alkalmaztak csoportosításukra. Ezt követően K-közép klaszterezés és főkomponens-analízis alkalmazásával 10 képet választanak ki a videóból. Erre a modellek és tanítási idő komplexitás csökkentése miatt van szükség. Ezt követően több érdekes algoritmust is kipróbáltak, egy újabb képi dimenzió bevezetésére a videó mellé.

- Optical flow, vagyis a tárgyak látszólagos mozgásának mintázatát közelítő módszert alkalmaztak az egyes képkockák között. Ehhez Gunnar Farneback 2003-ban publikált algoritmusát [13] alkalmazták. Az így keletkező újabb, dinamizmust is ábrázoló képkockákat, az RGB dimenzióhoz csatolták, egy újabb „optical flow” dimenzióként.
- A másik módszer mélység információt tartalmazó képek alkalmazása. Ehhez azonban nem egy második, mélységélességre is érzékeny kamerát alkalmaztak, hanem algoritmikusan generálták. A feladatot egy módosított GAN [14] (Generative adversarial network) modellel végezték, melyben a szokásos zaj helyett, két bemenete van mind a generátornak, mind a diszkriminátornak, és melyen keresztül mindkét modell megkapja az eredeti képet is, így sikeres tanítás esetén lehetősége lesz a generátornak a bemenő képkockára illeszkedő mélység térképet generálni.

Az architektúrát tekintve a továbbiakban rétegekbe szervezett 3D konvolúciós blokkokat használtak, majd az eredményt 2 Dense réteggel osztályozzák.



2.2 Ábra: Balra Gunner Farneback optical flow algoritmus kimenete, jobbra GAN hálózatokkal generált mesterséges mélységtérképek. [11]

A nehézséget, az ilyen jellegű megoldásoknál az okozza általában, hogy mivel videó adathalmazokról van szó, azok mérete és kezelése rendkívül költséges. Ez annak az eredménye, hogy a képek/videók sokkal több információt is hordoznak, mint amire szükség van a fordításhoz, és egyáltalán nem triviális olyan feldolgozó algoritmusokat találni, melyek optimálisak mind a tárolt információ, mind az adatok tömörítése terén. A problémát fokozza, hogy a finom gesztusok, mint az arc mimikája, szemöldök mozgása, mivel minimális mozgással jár, megkívánja a nagy felbontású, jó minőségű képeket. Továbbá az is szembetűnő, hogy a kutatások jelentős része, kevés kategória közül képes gesztusokat azonosítani, ehhez persze a szűkös adathalmazok is hozzájárulnak, de az sem zárható ki, hogy a nagyobb kategória számhoz szükséges megnövekedett adathalmaz, és tanítási idő, korlátozzák a fejlesztőket.

2.2 Segédeszközt használó megoldások

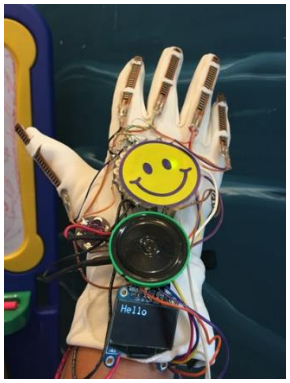
Segédeszköz használata a gesztusok felismerésére egy hatalmas előnnyel jár a hagyományos képfeldolgozást alkalmazó módszerekkel szemben, itt a gesztusok kategorizálásához különböző szenzorok alkalmazásával rendelkezésre áll valós térbeli információ a kezekről, valamint a mozgásokról. Ez hatalmas előnyt jelenthet a pontosság terén hiszen, kamerakép használatakor előfordulhat, hogy olyan szögben kerül rögzítésre a jelelő alany, amiből nem megállapítható a gesztus. Ez persze nem áll fent egy hétköznapi frontális kommunikáció, vagy az eszközeinkkel való interakció során, hisz ekkor egyéb iránt is a kamera felé orientálja magát a jelelő, de a segédeszközbe rejtett szenzorok minden esetben egy biztosabb, és pontosabb képed adnak a gesztusokról.

2.2.1 Kesztyűt használó kutatások

Az egyik jelentős irány a kesztyű használata. Irodalma [15][16] jelentős, néhány évente mindig érkezik egy-egy nagyobb áttörés a területen.

A kesztyű valójában több szenzor összehangolt méréseit [17] használja fel a gesztusok azonosítására. A hüvelykujj kivételével a maradék 4 ujj mozgatásának szabadságfokai limitáltak, elsődleges mozgási tartományuk a tenyér felé hajlítás. A dőlésszögek meghatározásának leggyakrabban használt érzékelője a hajlítás érzékelő szenzor, melynek ellenállása arányosan növekszik az ujjak görbítése során. Gyakran szerelnek továbbá gyorsulásmérőket az ujjak végére, valamint a tenyérre is. Nyomás érzékelőket az ujjbegyekre, valamint az egyik megoldás még a Hall effektust kihasználó

távolság szenzor adatait is feldolgozza. Nyilvánvaló probléma viszont, hogy jeleléskor szükséges az eszközzel rendelkezni, ezáltal a természetességéből könnyen veszíthet a kommunikáció, nem beszélve az anyagi vonzatáról az eszközöknek. További gondot okoz a minden kísérletben változó hardver, melynek köszönhetően legtöbbször egyedi adathalmazra van szükség, ami nem feltétlen robosztus.



2.3 Ábra: Jelnyelv fordító kesztyű prototípus.[3]

2.2.2 SignRing: Amerikai jelnyelv-felismerés IMU szenzorral ellátott gyűrűkkel

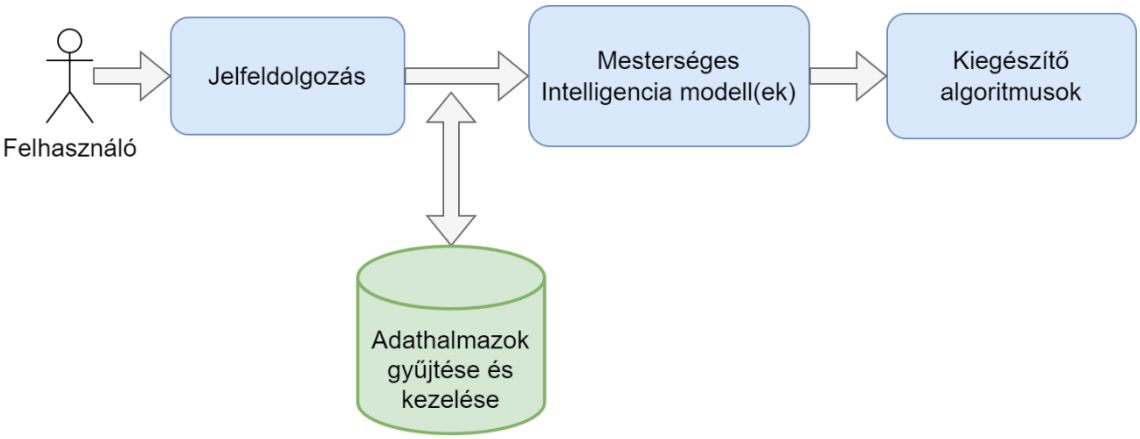
Egy újabb megközelítés a gyűrűbe [18] helyezett inerciális mérőegység (IMU) használata. A hivatkozott kutatásban a két mutatóujjon elhelyezve 6 szabadsági fokos adatokból rekonstruálták a jelelt gesztusokat. Ez egy rendkívül friss, de ígéretesnek tűnő kutatás, mivel egy gyűrű viselete nem akadályozza az embereket a hétköznapokban, és javaslatot tesznek több szögből felvett videó esetén tanító adatok kinyerésére is, mellyel a már meglévő videó alapú adathalmazokat is hasznosítani lehetne. Az általános felhasználhatósága azonban egyelőre nem bizonyított. Ennek oka, hogy mint sok más jelnyelvhez hasonlóan, az amerikai változatban is, nem csak a kezek hordoznak információt. pl.: Azok helyzete az archoz képest jelentősen módosíthatja a gesztus jelentését. Mivel az arc szenzorokkal való monitorozása, végképp egy természet ellenes lehetőség, esetleg előzetes kalibrációra, vagy kontextusfüggő javításra van szükség a fordításhoz. Ezen felül az arc mimikája nélkül elveszítjük a jövőbeli lehetőséget a nyelv teljes értékű fordítására. Így az ilyen módszereket nem tartom teljes értékű megoldásnak az általános jelnyelv fordításra.

[3] https://www.wipo.int/wipo_magazine/en/2019/05/article_0005.html

3 A megközelítés lépései

A kapcsolódó kutatásokból látszik, hogy a megoldásokat két részre lehet osztani: jelfeldolgozásra és mesterséges intelligencia modellekre. Előbbire azért van szükség, mert a modellek nem tudnak bemenetet fogadni közvetlenül a környezetből, szükség van tehát a valóság egy több szempontból is optimális reprezentációjára. Erre volt példa a képeken a kezek szegmentálása kameraképből vagy az ujjak hajlításának érzékelése a kesztyűs kutatásban. Ehhez a lépéshez szorosan kapcsolódik továbbá a modellek tanítása során felhasznált adathalmazok kialakítása is.

A valóságban létezik még egy harmadik lépés is, a kiegészítő algoritmusok formájában, amit nem lehet figyelmen kívül hagyni. Erre éles alkalmazások kialakítása során van szükség. Feladata a modell kimenetének megbízhatóbbá tétele, valamint hasznosítása. Egy jó példa az utóbbira, ha a bemenet az egyes képkockák szegmentált változata, a kimenet pedig egy-egy betű vagy jel, a kiegészítő algoritmus feladata folytonos szöveggé alakítani a kimenetet.



3.1 Ábra: Alkalmazás távlati architektúrája.

4 Környezetből érkező jelek feldolgozása

Érezhető a korábban bemutatott kutatásokból, hogy ez az egyik leghangsúlyosabb része a feladatnak. Ha ugyanis a jelelés reprezentációja nem megfelelő nehéz lesz jó modelleket építeni rá. Továbbá az is megfigyelhető, hogy jelenleg egyáltan nincs széles körben elfogadott megoldás a feladatra. Az előzetes irodalomkutatás, és saját ambícióim alapján négy problémát fogalmaztam meg a jelenleg elérhető megoldásokkal kapcsolatban, amelyekre a dolgozatban javasolt módszer megoldást ad:

1. Mindennemű segédeszköz használata nélkül, a kéz fiziológiai modellje is felhasználásra kerüljön.
2. Univerzális, könnyen kezelhető adatokhalmazokat lehessen létrehozni a tanításra.
3. A már meglévő videó/képi formátumú adatbázisok felhasználhatóak legyenek.
4. Mindezt valós időkorlátok között.

A javaslat egy újszerű képfeldolgozási módszer, melynek lényege, hogy speciális algoritmusokkal előállítható, a képen látható kéz ízületeinek helyzete. A módszer alkalmazható a test többi részén elhelyezkedő jellegzetes pontok beazonosítására is, beleértve a fejet, és az arcot. A megfelelő pontok koordinátáit felhasználva kialakított adat alkalmas a kitűzött célok megvalósítására. A továbbiakban ezt bizonyítom.

4.1 Póz becslő algoritmusok

A póz felismerés a számítógépes látás egyik kulcsfontosságú területe, amely az emberi test alakjának és tartásának felismerésére és nyomon követésére összpontosít. Ezen technológia segítségével a gépek képesek azonosítani és elemezni az emberi test különböző részeit, mint például a kezeket, lábak, fejet és a test főbb ízületeit.

A működése lényegében képfeldolgozási és mély tanulási technikákra épül. A mély neurális hálózatok, különösen a konvolúciós neurális hálózatok (CNN), forradalmasították ezt a területet, lehetővé téve a nagy pontosságú és valós idejű póz detektálást. A hálózatok képesek "tanulni" az emberi anatómia jellemzőiről és azok változatosságáról nagy adathalmazokon keresztül, így képesek azonosítani a testrészeket

és azok relatív elhelyezkedését. Így amikor az adott kamera szemszögéből nem látszanak tökéletesen az ujjak, a kéz és test ezen fiziológiai modelljeit „ismerve” mégis képes értelmes becsléseket adni. Tehát a kamerán keresztüli felismerésen túl egyéb információt is képes belevinni a válaszaiba, innen kapta a nevét: póz becslés. A továbbiakban felváltva hivatkozok az ilyen jellegű algoritmusokra póz felismerés címszó alatt is.

Az elmúlt években jelentős előre lepések történtek ezen a területen. A modellarchitektúrák fejlődésével a póz detektálási rendszerek gyorsabbak és pontosabbak lettek. Ezen kívül az átfogó adathalmazok elérhetősége és nagyobb számítási kapacitás rendelkezésre állása is hozzájárultak a terület fejlődéséhez.

A dolgozatban pózbecslés alapon működő modelleken kísérletezve vizsgálom, hogy az adatok ilyen formában való kódolása alkalmas-e a jelnyelv általános fordítására.

4.1.1 OpenPose

Az OpenPose [19] kiemelkedik a póz detektálás világában, mint az első nyílt forráskódú, valós idejű 2D test-, kéz- és arc-póz detektálási rendszer. A Carnegie Mellon Egyetem által fejlesztett projekt olyan technológiát hozott el a nagyközönségnek, amely a korábbi megoldásoknál jobban skálázható és sokoldalúbb. Míg számos rendszer létezik a póz detektálásra, az OpenPose különösen azzal emelkedik ki, hogy egyetlen neurális hálózattal képes kezelni a test, kéz és arc pózait, így kínálva integráltabb megoldást. Továbbá, a nyílt forráskódú természetének köszönhetően sok kutató és fejlesztő számára elérhetővé vált, ami gyors innovációt és széleskörű alkalmazást tett lehetővé. Ezen funkciók kombinációja teszi az OpenPose-t az iparág egyik vezető eszközévé a póz felismerésben.

4.1 Ábra: Többalakos 2D póz felismerés eredménye az OpenPose algoritmussal. [19]

A bemeneti kép először egy előtanított képfeldolgozó modellen megy keresztül. A következő lépés egy kétfázisú konvolúciós blokkokból álló modell alkalmazása. Az első fázis végén egy úgynevezett végtag „affinitás” mezőt generál. Ez a mező minden képponthoz egy 2 dimenziós vektort tartalmaz, ha adott pont rajta van a végtagon, akkor értéke a 2 legközelebbi felismerni kívánt kulcspont között feszülő vektor normálva, egyébiránt pedig 0. Természetesen a modell ezt csak megközelíti, de tanítás során egy olyan mezőt használnak „elvárt” adatként, ami az összes emberre kiszámolt mezők átlaga. A pontossága ellenőrizhető, két kulcspont között, egy integrállal a két pontot összekötő szakasz mentén.

$$\boldsymbol{E = \int_0^1 L_c\big(p(u)\big) \cdot \frac{d_{j2} - d_{j1}}{|d_{j2} - d_{j1}|_2}\, du}$$

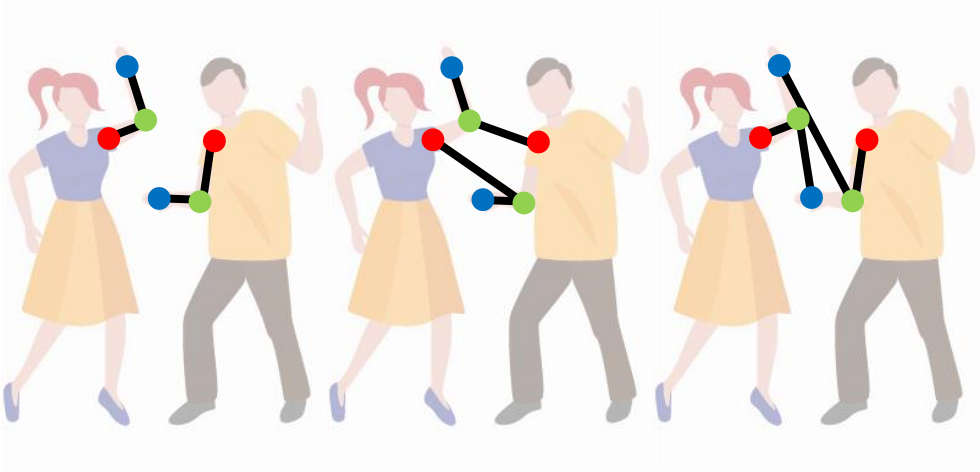
$$\boldsymbol{p(u) = (1-u)d_{j1} + u d_{j2}}$$

4.1 Egyenlet: Végtag affinitás mező pontosságát leíró képletek. [19]

A képletben a kérdéses két kulcspontot d_{j1} és d_{j2} jelöli, $p(u)$ pedig $u \in [0,1]$ tartományon a közöttük való lineáris interpoláció. $L\big(p(u)\big)$ megadja a keresett pontokon az affinitás mező értékeit, majd ez skalárisan szorzódik a két pont között feszülő vektor normáljával. Utóbbi művelet jól megfogja a kapcsolatot a két vektor hasonlósága között, hiszen értéke $[-1,1]$ tartományon annál közelebb van 1-hez minél hasonlóbb irányba néznek. A gyakorlatban az integrált diszkrét tartományokon vett összegzéssel (Σ) közelítik.

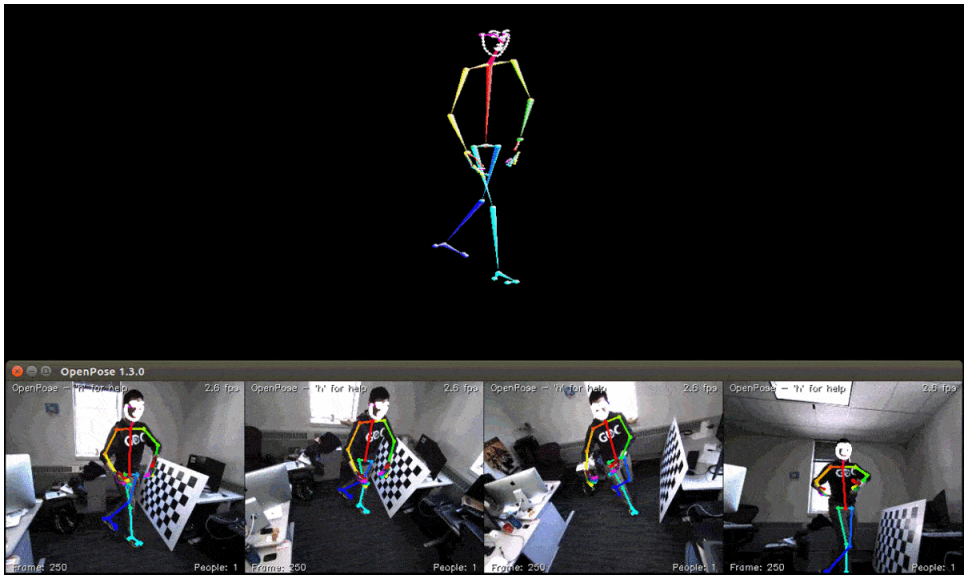
A második iterációban az eredeti bemenet és a kiszámolt mező együttese halad tovább, egy hasonló konvolúciós architektúrába, ami pedig az egyes kulcspontok előfordulási valószínűségét adja. Több azonos típusú, magas valószínűséggel rendelkező pozíció is lehetséges, hiszen a képen lévő emberek száma ismeretlen. Ekkor egy feladat maradt hátra, mégpedig az összetartozó kulcspontok azonosítása. A feladatot lépésekben oldja meg, egy gráfelméleti algoritmussal. Két kulcspont, ami között akarunk összeköttetést, vegyük például a vállat és a könyököt. Képzeljük el úgy a két kategóriába eső kulcspontokat, mint egy párosgráf csúcsait. (Egymással nem akarjuk különböző emberek vállát összekötni, így megvan a két csúcshalmaz.) Két csúcs közötti él súlya legyen a fenti integrállal kiszámolt érték, a csúcsokat jelentő síkbeli pontok között. Ekkor a feladat enyhíthető egy maximális súlyú párosítás keresésével, amire Kőnig Dénes matematikus munkássága nyomán ismerünk polinomiális futásidejű algoritmust.

Általános esetben optimálisan szétválogatni a kulcspontokat változó számú emberre, egy K-dimenziós párosítás keresésére vezethető vissza, ami ismert NP nehéz feladat.



4.2 Ábra: Kulcspontok különböző lehetséges párosításai.

Megjegyzésként említendő, hogy elérhető 3D koordinátákat becslő algoritmus [20] is. Ez azonban több okból sem alkalmas jelenleg a fordításra. Egyrészt több kamera szükséges a működéséhez, másrészt koránt sem felel meg a valós időkorlátoknak. Fontos viszont, hogy a póz adatok univerzalitásából adódóan, ha a jövőben sikerül javítani ezen algoritmuson, összehasonlítható és használható lesz a többi módszerrel. Ezzel szemben egy kesztyűvel történő fordítás esetén ez nem lehetséges, hiszen például egy új kesztyűn lévő eltérő hajlítás érzékelő szenzorok más ellenállás értékeket fognak produkálni fizikai tulajdonságaik miatt, és ezzel a régebbi adathalmazok és arra betanított modellek nem feltétlenül lesznek kompatibilisek az új eszközzel.

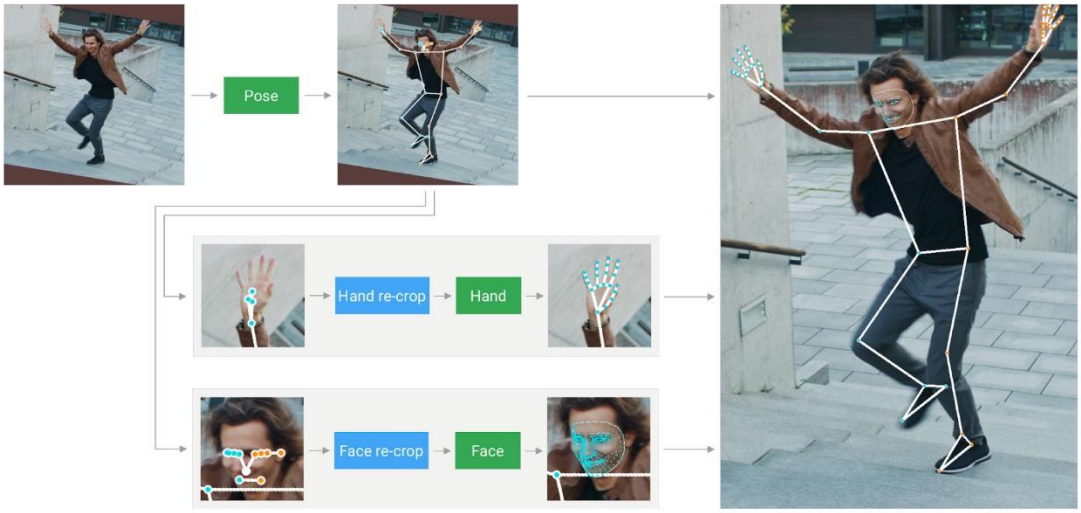


4.3 Ábra: OpenPose 3D képpontok előállítása több kamerával. [20]

4.1.2 MediaPipe Holistic

A MediaPipe Holistic [21] azon kevés modell közé tartozik, amely egyesíti a különböző emberi testrészek detektálását, és még tovább megy: képes a térbeli elhelyezkedésük azonosítására is. A Google által kifejlesztett MediaPipe platformon belül a Holistic modell olyan innovatív technológiákat alkalmaz, amelyek lehetővé teszik az arc, kéz és test pózának egyidejű elemzését.

Ami igazán különlegessé teszi a Holistic modellt, az az integrált megközelítés: több modellt kombinál együtt a lehető legpontosabb eredmények eléréséhez. Például külön modell végzi a testpóz [22], kéz [23], fej/arc [24] kulcspontjainak meghatározását. Ezek az almodellek külön-külön is kiemelkedő teljesítményt nyújtanak területükön, de együtt alkalmazva a Holistic modell egy integrált és átfogó képet ad a felhasználó pózáról és interakciójáról környezetével.



4.4 Ábra: MediaPipe Holistic modell távlati architektúrája. [21]

Sajnos nem elérhető olyan részletes leírás az algoritmikus hátteréről, mint az OpenPosenak, de teljesítménye magáért beszél. Csupán egy embert képes felismerni egy képkockán, de ez egyáltalán nem korlátozó jelenleg, hiszen a feladat egy időpillanatban csak egy jelelő fordítására terjed ki. Előny viszont, hogy nem csak 2D koordinátákat ad vissza, hanem térbeli fogalommal is rendelkezik az ember pozíciójáról. Ez a kezek, jelnyelv céljával való feldolgozása szempontjából hatalmas előnyt jelent. További pozitív tulajdonság, hogy a keretrendszer, és vele együtt a modellek könnyen futtathatóak a különböző platformokon. Elérhető például Androidon, illetve böngészős környezetben is. A valós idejű futás hasonló erőforrás korlátozott környezetekben is támogatott. Ha

folyamatos videóból hajt végre póz felismerést, abban az esetben az egyes képkockák között kihasználja a lokalitásokat, vagyis azt a tényt, hogy a végtagok nem tudnak hirtelen túl nagy távot megtenni fizikai korlátaik miatt, így egy kisebb komplexitású követő algoritmust alkalmaz. Az imént felsorolt előnyök miatt a dolgozat további részében a MediaPipe Holistic, illetve a kézre dedikált verzióját (MediaPipe Hands) fogom használni.

5 Adathalmazok

Két alapvető módszer különíthető el a jelnyelv-felismerés terén: a dinamikus videóból és a statikus képkockákból történő fordítás. Míg a dinamikus videóból kinyert szekvenciális adaton alapuló fordítás lehetővé teszi az egymást követő mozdulatok és azok közti átmenetek azonosítását, ami gyakran kulcsfontosságú a jelentés teljes megértéséhez, a statikus képkockákon alapuló fordítás kihívások elé néz. Ebben az esetben a rendszernek egyetlen pillanatfelvétel alapján kell meghatároznia a mozdulat jelentését, ami korlátozott információt kínál a mozdulat teljes kontextusáról. Ennek ellenére mindkét módszernek megvannak a maga előnyei és alkalmazási területei, amelyekről mélyebben is érdemes beszélni.

5.1 Statikus adathalmazok

Mivel a dolgozat időpontjában nem állt rendelkezésre statikus póz adathalmaz a feladatra, így először képi adatbázisokat kerestem, hogy az adatokat konvertálás során megfelelő formátumúra alakítsam. Mivel az erre a feladatra kiélezett adathalmazok csupán kivágott kezeket tartalmaznak, és a MediaPipe Holistic modell csak az emberi test kontextusában képes részleteket felismerni, így kézen fekvő volt a MediaPipe Hands api használata. A fellelhető adathalmazokban közös volt, hogy a képeket erősen előfeldolgozták, kis felbontásúvá konvertálták és a legtöbb esetben még a szín információt is elhagyták. Pontos specifikáció nem érhető el az api minimális minőségi követelményeiről, így több, egymástól merőben eltérő, nem csak ASL adathalmazt is kipróbáltam:

1. Sign language MNIST [25] : A klasszikus kézzel írt, 28x28 pixeles számjegyeket tartalmazó MNIST adathalmaz variánsa, amerikai jelnyelv feladatra specializálva. A képek hasonlóan névrokonához fekete fehérek, és identikus felbontásúak.
2. University of Exter ASL [26] : Két féle verzióban is elérhető. Tartalmaz alacsony felbontású színes képeket, valamint mélység információt is, bár utóbbi nem került feldolgozásra. A képek minősége elég változatos, és majdnem mindegyik tartalmaz torzítást.

3. Sign language for Alphabets [27] : Szintén fekete fehér adathalmaz, magasabb felbontású, mint az első pontban, viszont nem amerikai jelnyelv.

Első körben az MNIST adathalmazzal kezdtem dolgozni. Nyers formában a képek nem voltak megfelelőek, a MediaPipe Hands api számára. Mivel a minimális minőségi követelményekről leírást nem találtam, így különböző minőség fokozó technikákat alkalmaztam a képeken.

5.1.1 Interpolációs algoritmusok

Az elsődleges gyanúm a képek kis felbontása volt, ezért először ezt próbáltam orvosolni. Az interpolációs algoritmusok kulcsszerepet játszanak a képek felbontásának javításában. Az interpoláció lényegében magasabb felbontású területeken pixelértékek becslését jelenti az alacsony felbontású képről származó információk alapján. A klasszikus interpolációs módszerek, matematikai összefüggések használatával számolják ki ezeket az értékeket, figyelembe véve minden pixel szomszédait. Azonban ezek a hagyományos módszerek néha nem tudják megragadni a bonyolult részleteket. A mélytanulás megjelenésével fejlettebb interpolációs algoritmusokat fejlesztettek ki, amelyek neurális hálózatokat használnak. Ezek a modellek, alacsony és magas felbontású képek párosairól tanulnak, és céljuk, hogy áthidalják az egyszerű matematikai becslések problémáit, így kínálva pontosabb és esztétikailag kellemesebb szuperfelbontási eredményeket.

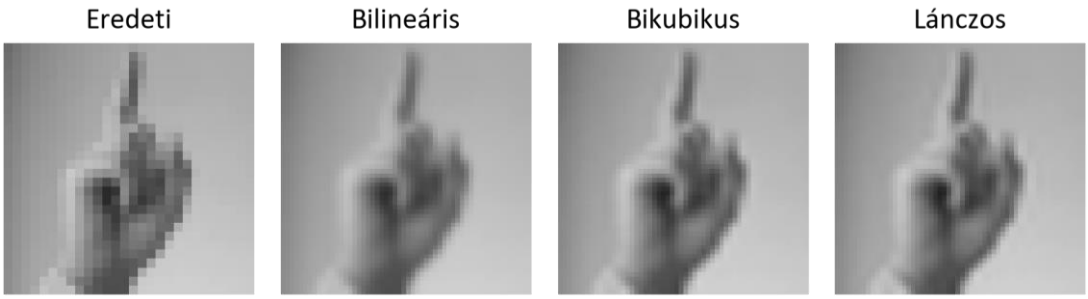
5.1.1.1 Hagyományos matematikai módszerek

Az interpoláció során a matematikai módszerek a pixelértékek közötti összefüggéseket használják fel a becslésekhez, és nem igényelnek külső információ forrást. A kipróbált módszerek:

1. Legközelebbi szomszéd (Nearest Neighbor): Az egyik legegyszerűbb módszer. Itt az új pixel értékét közvetlenül az eredeti képen található legközelebbi pixel értékéből veszi. Nincs súlyozás vagy más számítás, így gyorsan működik, de a végeredmény gyakran "kockás" megjelenésű lehet, különösen nagyobb méretarányú nagyításnál.
2. Lineáris (Bilinear) Interpoláció: Négy közeli pixel értékét veszi figyelembe. A kimeneti pixel értéke a környező pixelértékek súlyozott

átlagaként jön létre, ahol a súlyok az új pixel távolságán alapulnak a négy eredeti pixelhez képest. A bilineáris interpoláció simább képet eredményez, mint a legközelebbi szomszéd módszere.

3. Bikubikus Interpoláció: Két érték interpolációját egy harmadfokú polinóm illesztésével valósítja meg. Hogy a paraméterei egyértelműen kiszámíthatóak legyenek szükséges a végpontokban vett deriváltak értéke, ennek ismeretéhez pedig a környező szomszédok is. Így egy pixel értékének kiszámításakor figyelembe veszi a környező 16 pixel értékeit. Eredménye sokkal természetesebb kinézetet ad, mint a lineáris változat.
4. Lánczos Interpoláció: A módszer egy magyar matematikus, Lánczos Kornél érdeme. Egy ablakozott sinc-függvényt használ a pixelek súlyainak kiszámítására, valamint több pixel értékét veszi figyelembe, mint a bilineáris vagy bikubikus módszerek. A használt (opencv) library a pixel 8x8-as környezetét veszi figyelembe. A sinc-függvény tulajdonságainak köszönhetően a Lánczos kiválóan alkalmas képek élesítésére anélkül, hogy jelentős torzulást okozna.



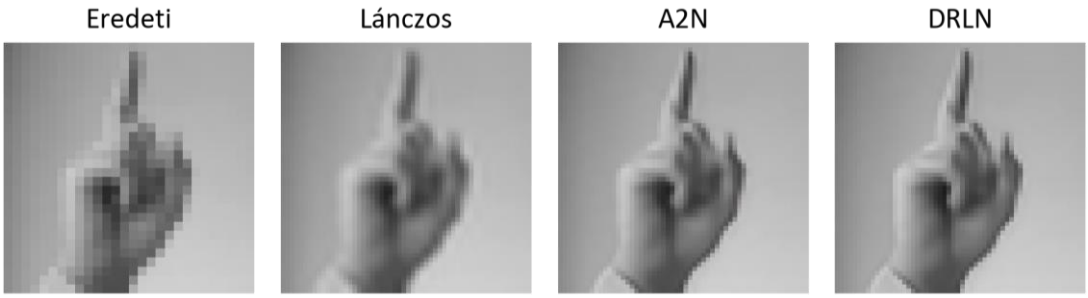
5.1 Ábra: Interpolációs algoritmusok eredménye az ASL MNIST adathalmazon.

5.1.1.2 Mély tanulás alapú módszerek

A mélytanulás-alapú interpolációs technikák számos előnnyel rendelkeznek a hagyományos matematikai módszerekhez képest. Elsődlegesen ezek az algoritmusok képesek észlelni és reprodukálni azokat a bonyolult mintákat és szerkezeteket a képeken, amelyeket a klasszikus módszerek esetleg nem vesznek észre. A hagyományos módszerek rögzített matematikai képletekhez kötöttek, amelyek néha nem képesek alkalmazkodni a valóságos képtartalom széles változatosságához. Ezzel szemben a mély tanulás alapú módszerek hatalmas adatkészleteken tanulnak, lehetővé téve számukra, hogy általánosítsanak és tájékozottabb előrejelzéseket tegyenek a számtalan képminta

alapján, amelyekkel találkoztak. Továbbá, az ilyen modellek képesek hierarchikus jellemzőket megtanulni, ami azt jelenti, hogy észlelni tudják a képek alacsony szintű részleteit, mint a szélek és textúrák, valamint a magas szintű tulajdonságokat, mint az objektum struktúrák és szemantikai kapcsolatok. Ez a holisztikus megértés gyakran olyan interpolált képekhez vezet, amelyek nem csak élesebbek, hanem tartalmukban is koherensebbek, csökkentve az artifaktumokat, olyan vizuális eredményeket előállítva, amelyek közelebb állnak ahhoz, amit az emberi szem elvárna látni. Gyakorlatban sokkal élesebb hatást keltő képeket kapunk, mint a matematikai interpolációval. A kísérletek két modellel is el lettek végezve:

1. A „Densely Residual Laplacian Super-Resolution” tanulmány alapján készített modell, továbbiakban DRLN [28], különlegességét az adja, hogy olyan modulokból épül fel, melyek kombinálják a sűrű reziduális kapcsolatokat, vagyis modulon belül a konvolúciós blokkok minden őket megelőző blokk kimenetét megkapják bemenetükön, valamint egy speciális Laplace figyelmi mechanizmust. Továbbá ezek a modulok is többször szerepelnek egymás után. Paraméterek számában mérve komplexebb, valamint futásidőben jelentősen lassabb, mint a következő tesztalany.
2. Az „Attention in Attention Network for Image Super-Resolution” című írás alapján készült modell, röviden A2N [29], erejét szintén meglévő elemek újszerű felhasználásból nyeri. Alapja egy úgynevezett „Attention in attention block”, mely lényegében dinamikusan tanult súlyokkal kombinálja a bemenet általánosan feldolgozott, valamint figyelmi mechanizmussal ellátott ágát.



5.2 Ábra: Mély tanuláson alapuló felbontás növelő módszerek összehasonlítása Lánczos interpolációval.

5.1.2 Fekete fehér képek színezése mély tanuláson alapuló modellekkel

A képek felbontásának növelése nem járt átütő sikerrel a vizuális felismerhetőség ellenére, így csak egy tulajdonság hiánya okozhatja a MediaPipe könyvtár felismerési problémáját, mégpedig a képek színe, illetve azok hiánya.

Szerencsére túl vagyunk már azon az időszakon, mikor fekete fehér képeket manuálisan kellett megszínezni. Léteznek ugyanis előtanított mesterséges intelligencia modellek, melyek fekete fehér képekből színeket tudnak javasolni, minden pixelnek. Ez úgy lehetséges, hogy sok tanítókép alapján a modellek megtanulják értelmezni a képeken szereplő objektumokat, és hogy a nagy számok törvénye alapján hogyan néznek ki színesben.

A felhasznált modellek a „Colorful Image Colorization” című tanulmány [30] alapján készültek. Alap gondolat, hogy a képeket váltsuk át „LAB” formátumúra. A modellek bemenetként a fénysűrűség (Luminance) csatorna értékeit kapják, és az AB-csatorna értékeit próbálják közelíteni. Egy fekete fehér képen, a tárgyak nagy része több színt is felvehet így, ha regressziót alkalmazunk például euklideszi távolságmetrikával hibafüggvényként, az átlagoló hatás fakó színek felé tolja el a matematikailag optimális eredményeket. A szerzők javaslata, hogy a 10 egységenként felosztott AB-térre alkalmazható klasszifikáció, ami pedig élénkebb színeket eredményez. Adathalmazként felhasználható bármilyen színes kép, színtér váltás után rendelkezésre is áll a bemenet, és elvárt kimenet. A továbbiakban ECCV16 néven hivatkozott modell ezen az elven működik.

Kipróbáltam egy alternatív módon betanított modellt is (SIGGRAPH17 [31]). Az architektúra ugyanaz maradt, de alkalmanként tanulás során adunk a modellnek néhány képpontban segítséget az elvárt színhez. Bár a kutatásaim során ezt nem használtam ki, de így utólag is lehetőség nyílik segíteni a modell színválasztását. A tanítás során megadott segédpontok számát egy geometriai eloszlás függvény határozza meg, és bőven akad olyan bemenet is, amin nincsen „segítség”, így alkalmas a felhasznált feladatra. Tapasztalat szerzés céljából, néhány példán futtatva, bár kevésbé élénk színeket produkál, mégis realisztikusabb képeket eredményez, mint társa.

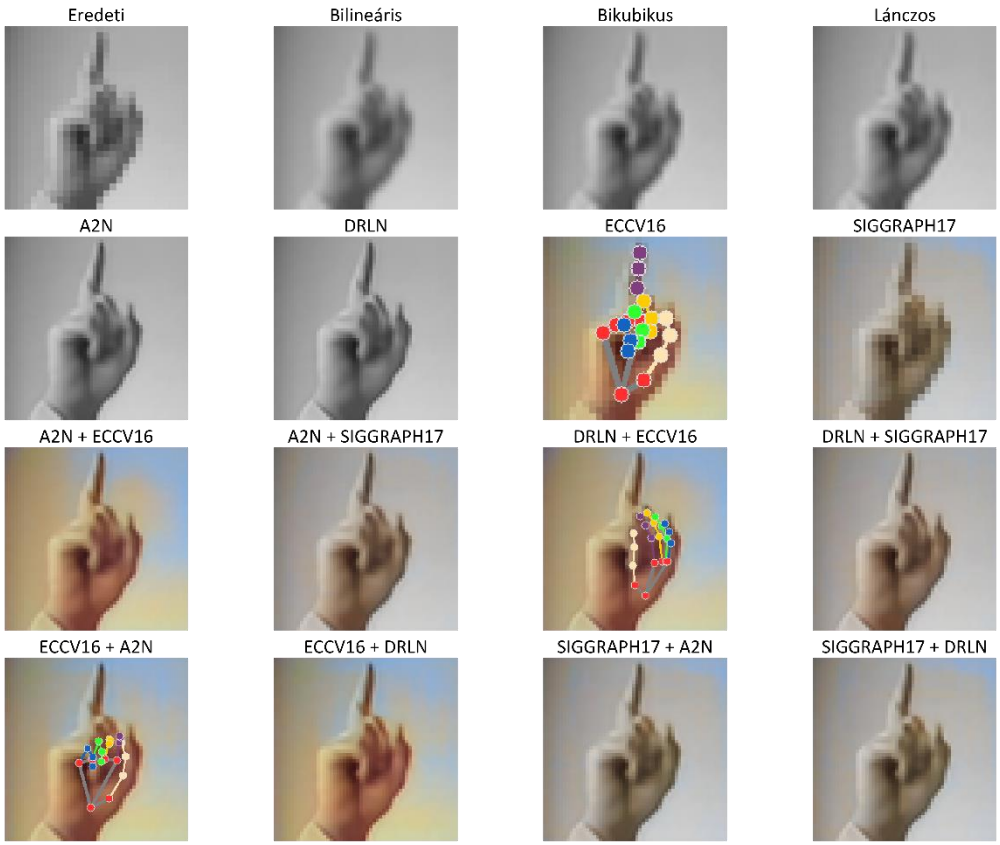
5.3 Ábra: Vizsgált kép színező modellek eredményei Ansel Adams, amerikai fotográfus legendás képén.[4]

5.1.3 Minőség javító algoritmusok összehasonlítása

Bár a felbontás növelő technikák látványos eredményt értek el, amíg a képek fekete fehérek maradtak nem volt képes a MediaPipe api kezeket felismerni. A színezési technikák közül az ECCV16 mutatta a legjobb eredményeket. Ez a modell nem mellesleg a használt neurális háló kialakítása miatt változtatja a bemeneti képek méreteit is. Előfeldolgozásként a Lánczos interpolációval megnöveli a kép méreteit 256x256 pixelre, majd az inferencia után Lineáris interpolációt alkalmaz az eredeti méretre való összenyomáshoz. Végeztem teszteket a felbontás növelés és színezés egymás utáni alkalmazására, ezekben az esetekben eredményeket szintén azok a kombinációk hoztak, melyekben szerepelt az ECCV16 algoritmus, de a felismert póz, közel minden esetben, erősen torzított lett. Az eredményeket validáltam a korábban „Sign language for Alphabets” néven említett adathalmazon is. Mivel az eredmények változatlanok voltak, ezután a teljes ASL-MNIST adatbázist konvertáltam az imént említett színező modellel, ezzel az esetek 30%-ában, közel 8500 esetben, sikerült pózt felismerni a képeken.

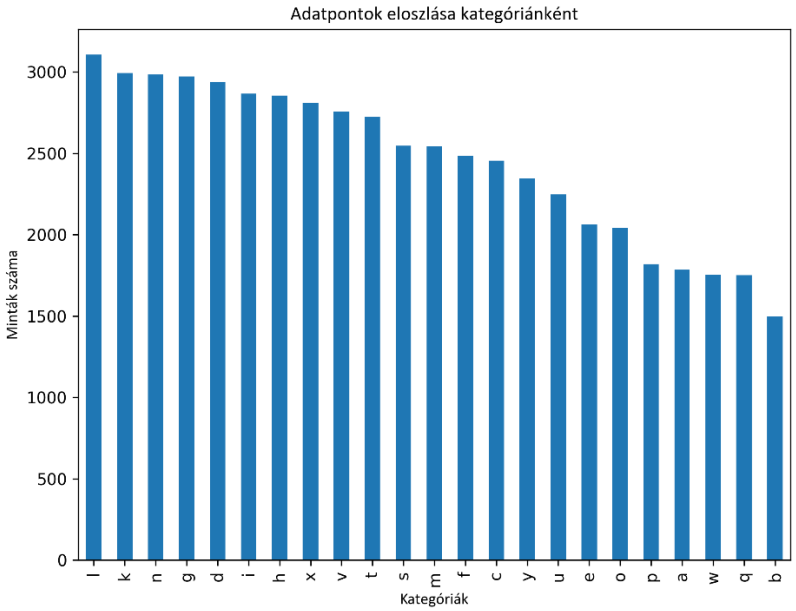
A minőség javító algoritmusokat a „University of Exter ASL” névvel illetett adathalmazon is kipróbáltam. Mivel ez egy színes képeket tartalmazó adathalmaz, a színező algoritmusok nem segítették a találati arányt, de a felbontás növelése Lineáris interpolációval igen. Konvertálás után az eredeti adathalmaz 38%-át sikerült megőrizni.

[4] https://shop.anseladams.com/collections/original-photographs-by-ansel-adams/products/tetons-and-the-snake-river



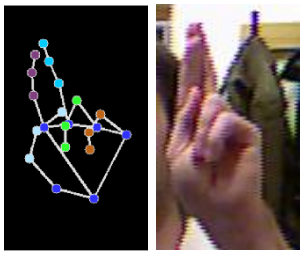
5.4 Ábra: Az összes kipróbált módszer illusztrációja az ASL MNIST adathalmaz egyik képén.

(Két modell név esetén az alkalmazási sorrend balról jobbra történt, valamint, ha sikeres volt a póz felismerés, megjelennek a képen a kulcspontok.)



5.5 Ábra: Az „ASL MNIST” és „University of Exter ASL” adathalmazokból kinyert pózok eloszlása kategóriánként.

A konvertált „R” betűk az esetek nagy százalékában nem megfelelően kerültek átalakításra, ugyanis a „U” és „R” betű között csupán annyi a különbség, hogy a mutató és középső ujj keresztezve van. Az alábbi képen is látható, a modellek hibásan az „U” betűhöz tartozó póz információhoz hasonló adatot ismertek fel az „R” betűként annotált képekről. Ez betudható az adathalmaz kis felbontásának, saját kamerával készült képeken nem tapasztaltam hasonlót.



5.6 Ábra: Helytelenül konvertált „R” betű a University of Exter ASL [26] adathalmazból.

5.1.4 Saját ASL adathalmaz

A más adathalmazokon elért sikerek ellenére, saját ASL ujjbetűzés adathalmazt gyűjtöttem. Az adatok frontális irányba néző webkamerával kerültek rögzítésre. A felvételek három napszakban, két különböző helyszínen készültek két féle kamerával. A felhasznált minták között egyénenként egyenletesen oszlott meg a különböző kategóriákban rögzített felvételek száma, valamint a jobb és bal kéz használata.

A nyers felvételek nem, csak a póz információ került mentésre. A képeken maximum egy kezet kerestem, mivel az ASL-ben minden betű és szám kifejezhető így. Képkockánként 21 db 3 dimenziós kulcspont került rögzítésre. Feltételezzünk 32 bites lebegőpontos számábrázolást, így felvételenként 252 byte adattal kell számolni. Ha el akarnánk tárolni a képeket, erős előfeldolgozás mellett is, például 100x100-as méretű, 8 bites színmélységű, fekete fehér képek esetén 10000 bájtot kellene tárolni, ami így közel 1: 40 tömörítési arányt jelent.

Mivel az adatok átlagosan 15-20 FPS sebességgel kerültek rögzítésre, még a kezek gyors mozgásának hatására is sok hasonló kép keletkezett, ezért 10 egymást követő adatpontból 9 nem került felhasználásra, így biztosítva, hogy legalább 0,5 sec teljen el két minta között.

A külső forrásból származó képekkel ellentétben nem vágtam ki, és izoláltam a kezeket a póz információ kinyerése előtt. Ez két okból is fontos lehet. Egyrészt a póz felismerő API relatív koordinátákat ad vissza. Ezek az értékek 0 és 1 közé esnek, és azt

jelzik, hogy a bemeneti képen a szélesség, és magasság hány százalékánál találhatók az adott pontok. Ha a bemeneti kép más képarányú, a detektált kezek koordinátái elnyúltnak torzítottnak tűnhetnek egymáshoz képest. Hasonló jelenség figyelhető meg a különböző látószögű kamerákon keresztül felismert kezeken is. A másik jelentős különbség, hogy mivel a teljes kép csak egy kis részén helyezkednek el a kéz a koordinátái matematikailag sokkal kisebb mértékben térnek el egymástól, mint a korábban tárgyalt adathalmazok esetében. Ezen probléma kiküszöbölésére nagy hangsúlyt helyeztem a modellek tanítása során használt előfeldolgozásra.

5.2 Szekvenciális adathalmazok

Ebbe a kategóriába azok az adatbázisok kerültek, melyeknek a bemenete nem csupán egy képkockából származtatható, hanem több egymást követőéből, valamint a származtatás után a sorrendi információ is megmarad. Ezen szekvenciák hossza alapján két lehetőség állhat fenn: a kimenet egy jelnek feleltethető meg (pl.: anya) vagy az is egy szekvencia (pl.: egy ujjbetűzött név).

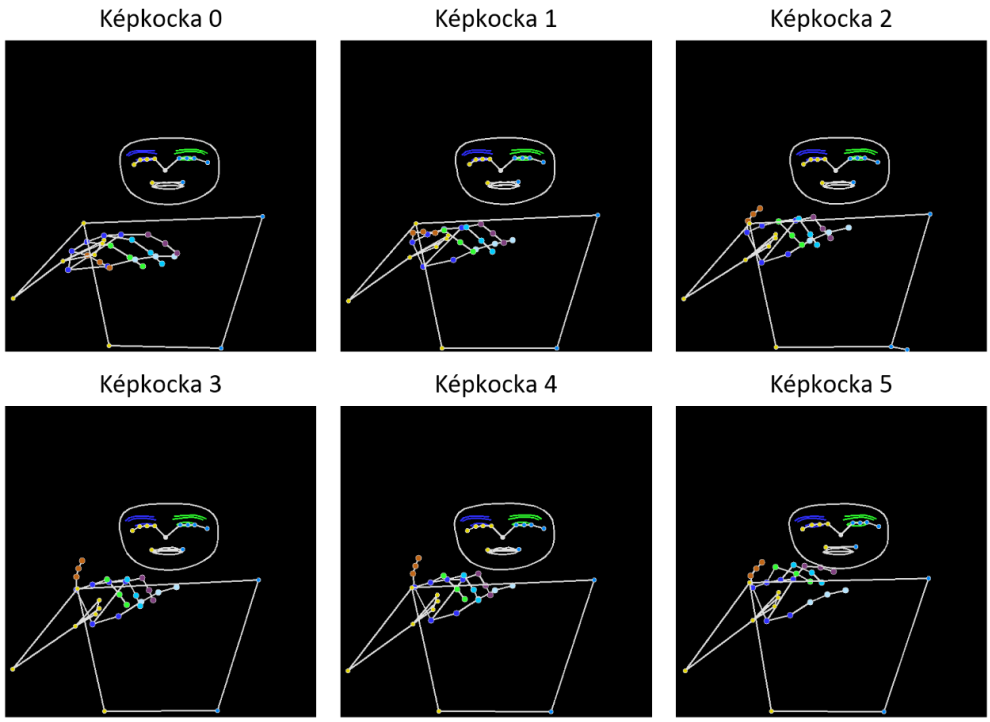
5.2.1 Szevenciális bemenet, szevenciális kimenet

Az adatok egy Google által meghirdetett Kaggle versenyből származnak [32]. A Kaggle egy online platform, amely adattudományi versenyeket szervez, és lehetővé teszi a felhasználók számára, hogy kollaboratív módon osszák meg egymással adathalmazaikat. A verseny célja az Amerikai Jelnyelv (ASL) ujjbetűzésének észlelése és szöveggé történő fordítása volt. Az adatok több mint hárommillió ujjbetűzött karaktert tartalmaznak, amelyeket több mint 100 siket jelnyelv-használó készített okostelefon kamerával, különféle háttér és megvilágítási feltételek mellett. Ez az ujjbetűzés területén elérhető legnagyobb póz adathalmaz.

A jelelőknek szöveget mutattak, amely nagybetűket is tartalmazott. A legtöbben ezt nem közvetítették, de néhány minta azonban tartalmaz ezzel kapcsolatos technikákat:

- Görbe-L kézforma a nagybetűk előtt.
- Fizikailag magasabb elhelyezés térben, mint a kisbetűknél.
- Betű kézformájának megrázása.

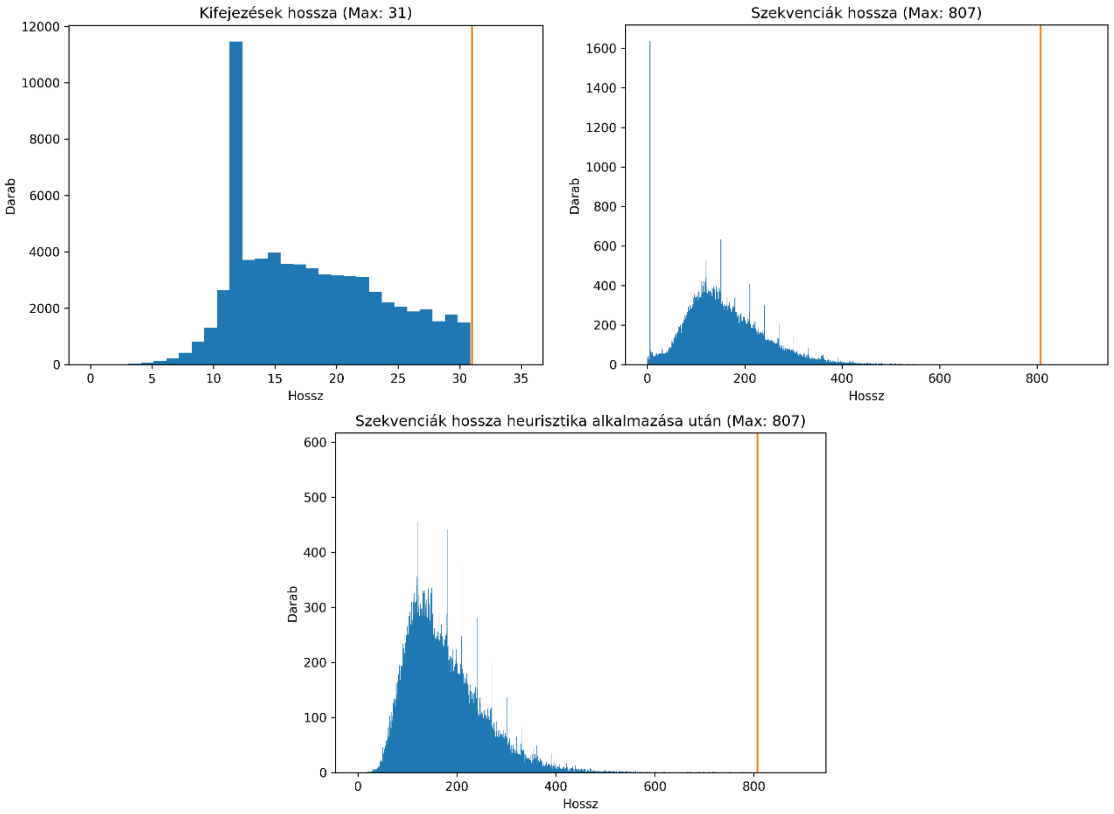
A készítő sajnos a címkéken erre utaló információt nem helyezett el, így a saját megoldásomban erre nem térek ki.



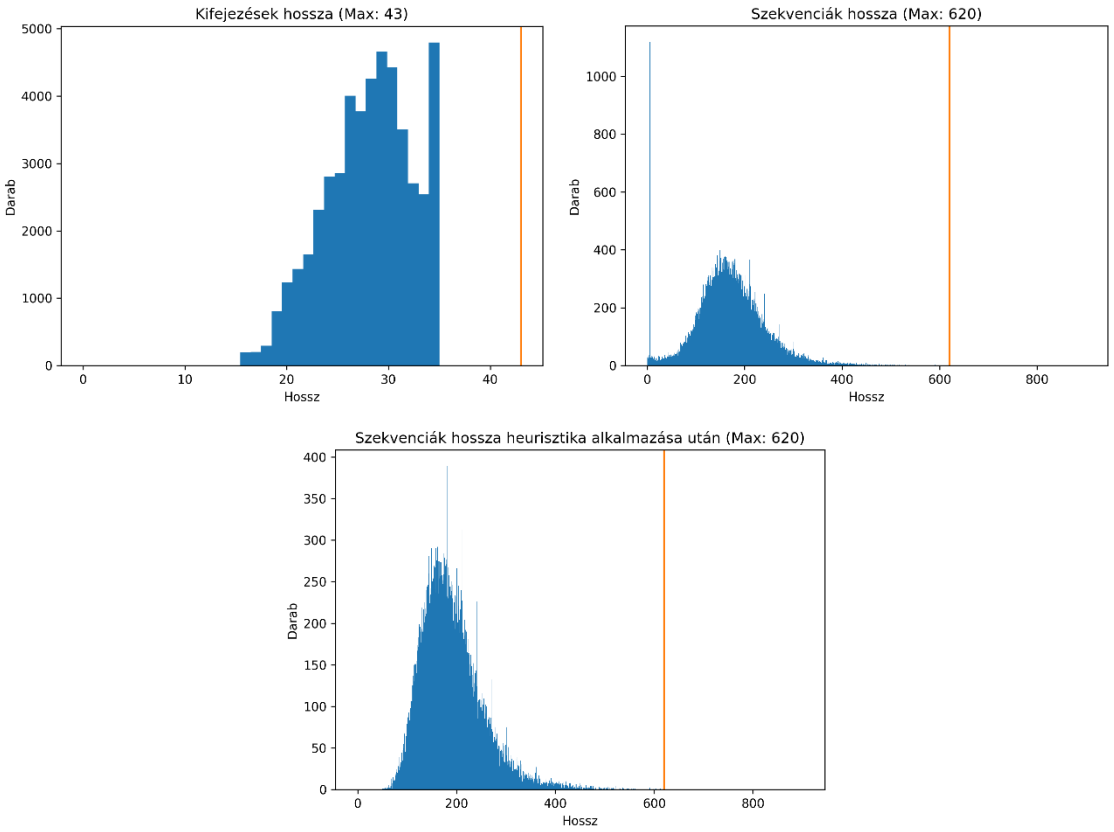
5.7 Ábra: Példa az adathalmazban található szekvenciákra.

Két különböző verzió érhető el az adathalmazon belül. Az egyikben linkek, címek telefonszámok jelelt felvételei találhatóak. A másik összefüggő szöveges mondatokat tartalmaz. Előbbi jobb minőségű, ezért a tanítást ennek 80%-án végeztem, a maradék 10-10%-ot pedig validációs és teszt célokra tettem félre, továbbá az összefüggő szöveges adathalmazon is végeztem teszteket a betanított modellekkel.

Tanítás előtt megvizsgáltam a szekvenciák, illetve a hozzájuk tartozó kifejezések hosszát karakterben. Az eredmények meglepőek: kiugróan sok esetben találtam olyan sorozatokat, melyek különösen rövidek voltak, illetve egyesek pedig rendkívül hosszúak. Az adatok vizuális megjelenítésével arra a megállapításra jutottam, hogy sok felvételen hiányzik a részletes kézmodell eredménye. Ennek oka valószínűleg, hogy a felvételek során használt telefonok nem tudták minden esetben a jobb minőségű póz felismerő modellt futtatni, csak a telefonra optimalizált változatát, illetve, hogy elérték azt a jelelési sebességet, amit az API már nem tudott minden képkockán lekezelni. A valós indok balladai homályban marad, de bizonyára az adatgyűjtés során került a bizonytalanság a rendszerbe. Ha a felvételeken nincs megfelelő mennyiségű kéz adat, akkor nem áll rendelkezésre elég információ a jelek megállapítására. Az ilyen esetek kiszűrésére egy heurisztikát alkalmaztam, melynek lényege, hogy legalább kétszer annyi képkockán kell jelen lennie részletes kéz pozíció adatnak, mint amilyen hosszú a jelelt kifejezés.



5.8 Ábra: ASL Kaggle tanító adathalmaz szekvenciáinak statisztikája.

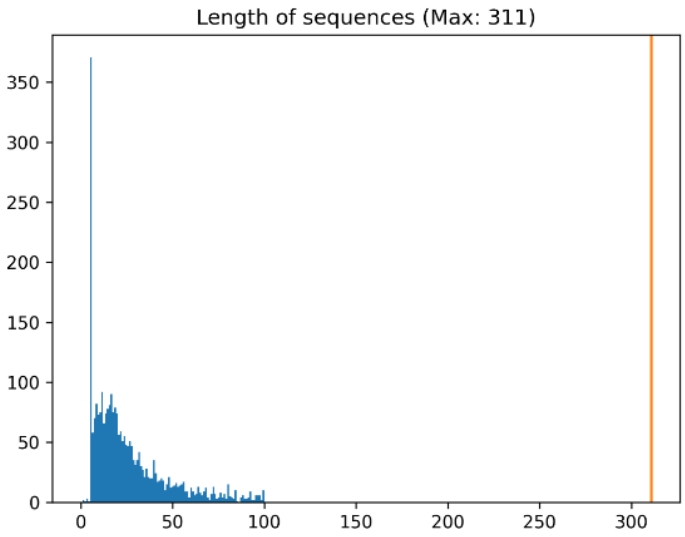


5.9 Ábra: ASL Kaggle kiegészítő adathalmaz szekvenciáinak statisztikája.

További fontos tervezői döntés a sorozatok hosszának megállapítása. Ez a későbbi pontokban leírt modellek technikai kialakítása, valamint az optimális tanítási idő elérése érdekében szükséges. A meghatározottnál rövidebb sorozatokat egy semleges kitöltő (padding) értékkel egészítjük ki, a megvalósításomban nullákkal minden dimenzióban. Ha túl rövid a szekvencia méret, akkor a modellnek nem lesz elég információja sok esetben a kimenet megállapítására, ha viszont túl hosszú, a rövidebb sorozatok nagyrészt csak kitöltő értékeket fognak tartalmazni, és így a modell feleslegesen tanul az adaton, nem beszélve a megnövekedett memória és tanítási idő komplexitásról. A kísérleteim során, valamint az adat statisztikai vizsgálata alapján a választásom a 256-os méretre esett. Ez az érték a legközelebbi kettő hatvány a szekvencia hosszok vonatkozásában számolt 80-90 percentilishez. Kisebb értékek esetén a modellek jelentősen rosszabbul teljesítettek, nagyobb értékek esetében pedig elenyésző volt a nyert pontosság, a jelentősen megnövekedett tanítási időhöz képest.

5.2.2 Szekvenciális bemenet, statikus kimenet

Az adatforrás az előzőhez hasonlóan egy Kaggle platformon meghirdetett versenyből származik. [33] Nagy különbség viszont, hogy a szekvenciák jelentős része rövid, így amikor csoportosítjuk őket (batch), akkor sokkal több kitöltő érték fog szerepelni a tanítási szekvenciákban.



5.10 Ábra: Kaggle ASL Isolated signing adathalmaz statisztikai elemzése.

6 Elkészült modellek bemutatása

Az előző fejezetben sorra bemutatott adathalmazok más és más típusú modellt kívánnak. Nem lehet például ugyanazt az architektúrát használni egy egyszerű klasszifikációra, mint egy szekvencia előállítására.

Bemenet/Kimenet	statikus	szekvencia
statikus	Statikus ujjbetűzés	x
szekvencia	Hagyományos jelelés	Szekvenciális ujjbetűzés

6.1 Táblázat: Feladat kategorizálása bemenet és kimenet típusa alapján.

A továbbiakban a mátrixban jelölt 3 feladat kapcsán elkészült modelleket mutatom be. Fontos megjegyezni, hogy a statikus bemenet-szekvenciális kimenet feladatot azért nem tárgyalom részletesebben, mert jelen kontextusban egy képkockáról származó póz információból csak egy jelet lehet megállapítani, semmiképpen sem egy sorozatnyi gesztust.

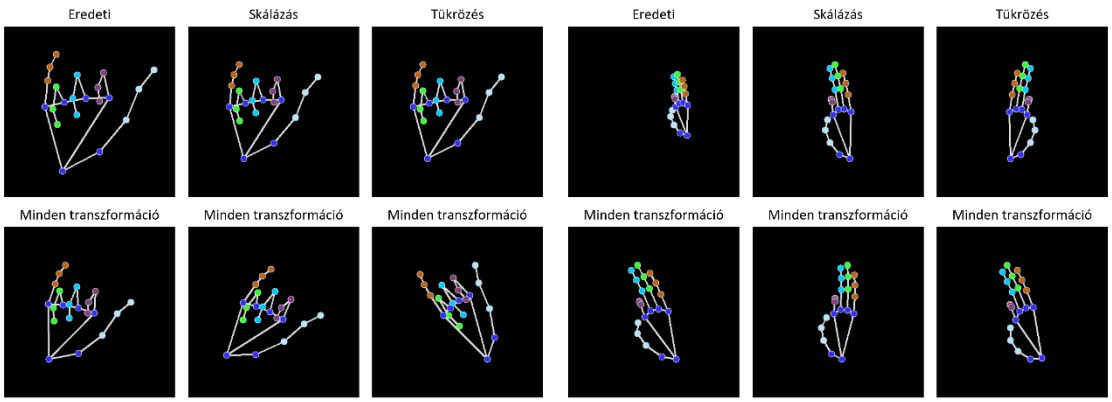
6.1 Statikus ujjbetűzés

A statikus módszer nem véletlen az ujjbetűzés témakörében jön elő, ugyanis a j és z kivételével egyik gesztus sem kíván dinamikus mozdulatot, ez a kettő pedig egy későbbi szakaszban tárgyalt algoritmussal korrigálható. Egy másik probléma is fennáll, ha az 1-9-ig lévő számjegyeket is bevonjuk. Az ilyen „kiterjesztett” ujjbetűzés ugyanis, tartalmaz olyan kéztartásokat, melyek két kategóriába is tartoznak. Ilyen például a 2 és v, melyek gyakorlatilag identikusok kéztartás szempontjából. A gyakorlatban szövegkörnyezetből megkülönböztethetőek, de ha mégis szükség van izolált környezetben a használatukra, olyankor szám esetében a jelelő tenyerét maga felé fordíthatja.

6.1 Ábra: Példa ASL-ben egyforma kézformát használó, mégis különböző jelentésű gesztusok.

6.1.1 Előfeldolgozás

A részfeladat fejlesztése során a statikus-statikus adathalmazok közül a saját frissen rögzített, valamint két meglévő képi adathalmaz konvertált változatát is felhasználtam. Ezek nagyban különböznek egymástól, ezért rendkívül fontos, hogy információ vesztés nélkül hasonló formára hozzuk őket, Első lépésben egyforma méretűre skálázom a kezeket, és eltolom a kulcspontokat a (0.5, 0.5) koordinátába, ami az API relatív koordináta rendszerében a képernyő közepe. Az így keletkezett adatpontokon minden alkalommal amikor a modell bemenetére kerülnek tanítás során egy bizonyos valószínűséggel affin transzformációt hajtok végre. Ez segít a modell általánosító képességének növelésében. A transzformáció csak az xy koordinátákra érvényesül, ugyanis tapasztalati alapon kiderült, hogy a z irányú adatok nagysága nincs korrelációban a kéz fiziológiájával, nem arányos a másik 2 tengely menti kiterjedésével. Csupán annak eldöntésére alkalmas, hogy az egyes ujjak hogyan helyezkednek el térben egymáshoz képest. A bal kézzel jelelők, a jobb kézzel jelelők tükörképei, így bár bal kezes adatok is rendelkezésre állnak, mégis alkalmanként tükrözés történik az y tengelyre. További transzformációk között szerepel a képernyő síkjában történő közelítés, illetve távolítás, xy tengelyek menti nyírás, valamint 15 fokkal való elforgatás. A transzformációk paramétereit empirikus módszerrel hangoltam, nagy számú mintán való teszteléssel, a túlzott torzítás elkerülése érdekében.



6.2 Ábra: Balra a University of Exter ASL, jobbra a saját adathalmazomból származó adatpont előfeldolgozása és adatdúsítása során kialakult minták.

A saját adathalmaz kezei a fenti módszerek alkalmazása után is keskenyebbnek hatnak. A további javítás érdekében a póz detektáló algoritmuson kell változtatni. Erről részletesen a Kiegészítő algoritmusok pontban lesz szó.

6.1.2 Modellek

A bemeneten az értékeket először kulcspontonként, majd x,y,z sorrendben helyeztem el, majd a klasszifikációs feladatként megfogalmazott problémára két féle modellt próbáltam ki:

1. Egyszerű mély neurális háló. (DNN)
2. Konvolúciós rétegekkel kiegészített CNN architektúra.

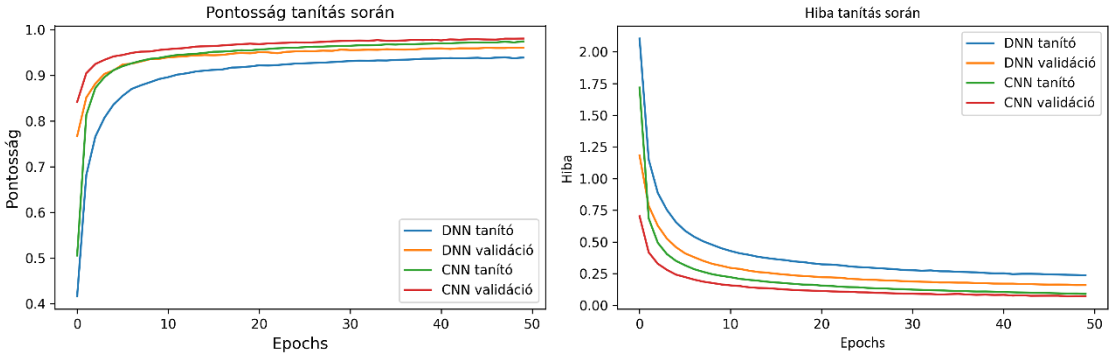
A konvolúciós hálózatban 1D konvolúciót használtam, ami idősoros adatoknál gyakran használt építőelem, de itt szemantikailag máshogy használom. A réteget úgy állítom be, hogy a filterek ne egymás melletti értékeket mintavételezzenek, hanem minden harmadikat. Ez biztosítja, hogy egy-egy filter csak ugyanolyan típusú (pl.: x koordinátákat jelentő) számokat mintavételezzen. Az intuíció, hogy a konvolúciós réteg után minden érték tartalmazzon információt az összes kulcspontról, és persze külön-külön legyen csoportosítva x, y és z szerint. Ehhez a következő modellt készítettem:

```
model = Sequential ([
    Input(shape=(63,)),
    Lambda(lambda x: tf.keras.backend.expand_dims(x, axis=-1)),
    Conv1D(filters=64, kernel_size=21, dilation_rate=3,activation='relu'),
    Flatten(),
    Dense(128, activation='relu'),
    Dropout(0.2),
    Dense(len(categories), activation='softmax')
])
```

Fontos paraméter a „dilation_rate”, hiszen ez biztosítja, hogy hány elemeként kerüljenek feldolgozásra az értékek, illetve fontos még a „kernel_size”, amit 21-re állítok, hogy egy konvolúciós filter minden kulcspont megfelelő koordinátájából mintát vegyen.

További kiegészítésként „Dropout” rétegeket is helyeztem el a hálózatokban, melyek bizonyos valószínűséggel néhány réteg bemenetét 0-ra állítják. Ez egy gyakran alkalmazott technika a modellek általánosító képességének fejlesztésére.

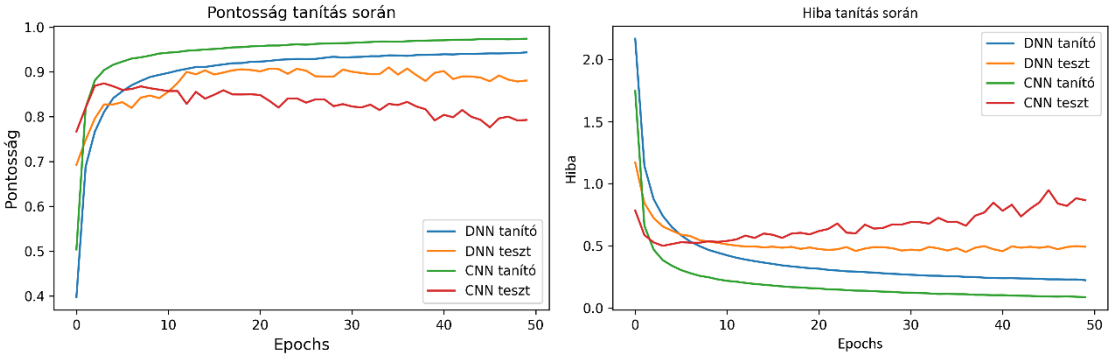
A tanítást az Exter university adathalmaz [26], és az ASL MNIST [25] adathalmazból kinyert póz információból hoztam létre. Az „r” betűk pontossága nem volt megfelelő a konvertált adathalmazoknál, így ezt a kategóriát kihagytam. Tanításra az adatok 80%-át, validációs célokra pedig a maradék 20%-át használtam. Tesztelésre a saját adathalmazomat választottam. Az eredmények így igazolják, hogy alacsony minőségű felvételekből is létrehozható valós környezetben is működő modell.



6.3 Ábra: Hiba, és pontosság alakulása a validációs adathalmazon tanulás során.

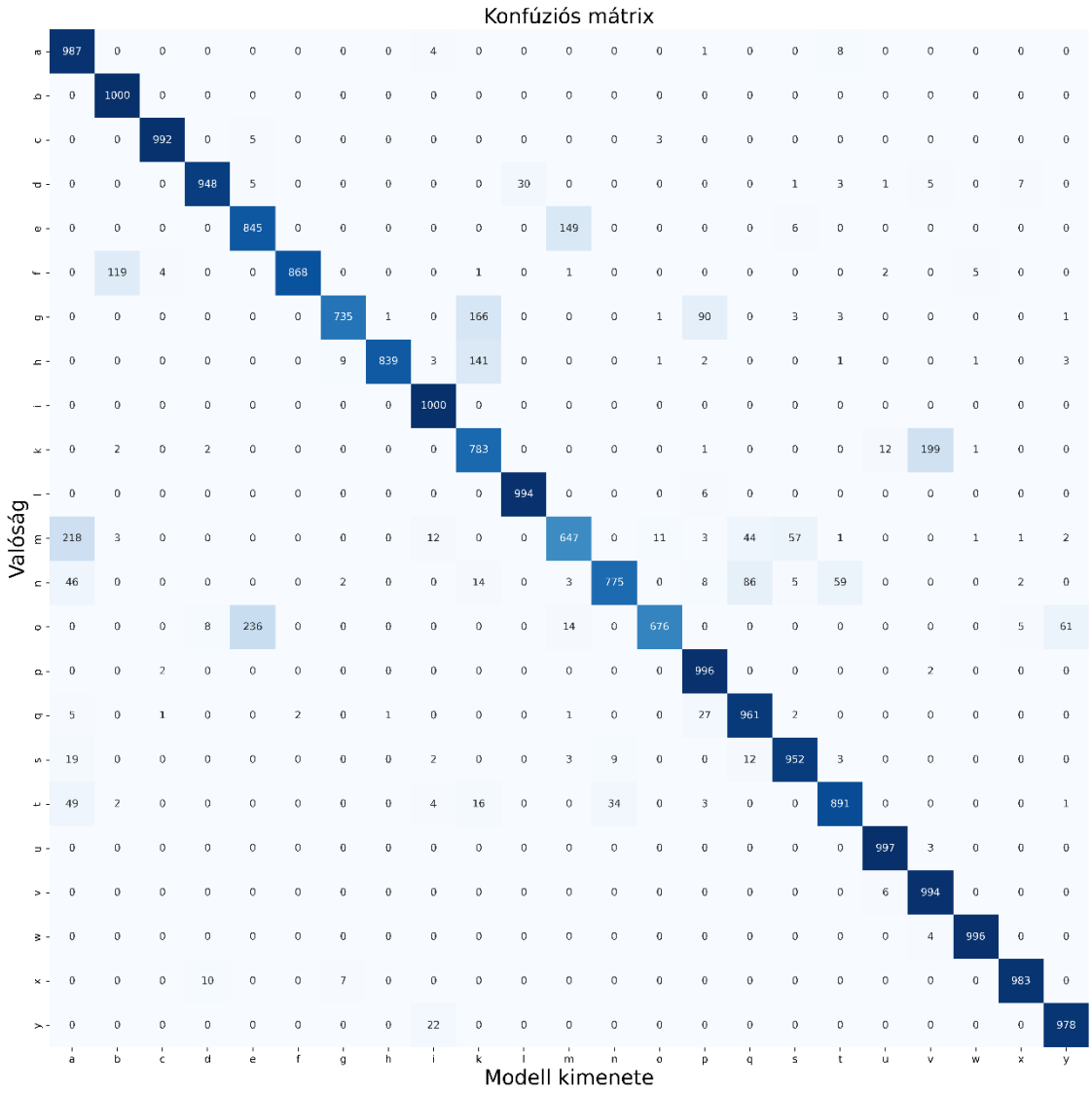
Az ábrákon látszik, hogy a validációs adathalmazon jobb eredményeket ér el a modell, mint a tanítón. Ennek oka, hogy az adatdúsítás dinamikusan működik, tehát minden alkalommal újra és újra random transzformálom a bemeneti képeket, viszont ezt csak a tanító halmazon. Kis túlzással igaz, hogy a modell sosem lát két ugyanolyan képet a tanulás során.

Ezen adatok alapján a CNN architektúra jobban teljesít minden esetben. A tesztelés során viszont a DNN architektúra konzisztensen jobb eredményeket hozott. Ennek okán megvizsgáltam a teszt adathalmazon mért pontosság és hiba alakulását a tanítás során.



6.4 Ábra: Hiba, és pontosság alakulása a teszt adathalmazon tanulás során.

A grafikonok rávilágítanak a megfejtésre, miszerint a modellek 10 epoch körül már túltanulták az adathalmazt, nem képesek jól általánosítani. Az ok, ami miatt ez nem jelentkezett a validációs adathalmazon csak az lehet, hogy túlságosan nagy korreláció áll fenn a tanító adathalmazzal, vagyis a random transzformációk ellenére is hasonlóak. A tanító ciklusokat (epoch) az ábrák alapján módosítva (early stopping), a DNN modell konzisztensen eléri a 90%-ot, a CNN modell pedig a 86%-ot a teszt adathalmazon.



6.5 Ábra: DNN hálózat teszt adathalmazon elért konfúziós mátrixa.

6.1.3 Kiegészítő algoritmusok

A mesterséges intelligencia (AI) modellek fejlesztésekor a legoptimálisabb teljesítmény és pontosság elérése érdekében gyakran szükség van kiegészítő algoritmusokra. Fekete doboz jellegükből fakadóan ugyanis, valószínűségi alapon viselkednek éles adatokon. Előfordulhat például, hogy bizonyos szögben, vagy a környezeti viszonyok hatására, pl.: fény csillanás a kamerában, nem lesz megfelelő a pillanatnyi kimenet. Ez a jelenség egyszerűen kiszűrhető, a többségi döntés algoritmusával.

6.1.3.1 Kimenet korrigálása valószínűségi alapon

A modell eredményei először egy sorba kerülnek, ahonnan többségi döntés alapján kerül ki a modell korrigált válasza. Egyszerűen megvalósítható egy FIFO adatszerkezettel, és így hatékony módon kiszűrhetőek a pillanatnyi anomáliák. Szűrhető továbbá az a jelenség is, amikor ismeretlen pózok eredményéül kis magabiztosságú, csapongó eredményeket ad a modell, ilyenkor nem egyszerű többségi döntést hozunk, hanem csak egy bizonyos százalékú többség hatására adunk eredményt. Ha azonban ilyen jellegű jelentős fölénnyel egyik kategória sem rendelkezik, abban az esetben sikeresen azonosítottuk azt a helyzetet, amikor a modell képtelen a magabiztos azonosításra.

Ennek a megoldásnak egy hátránya van, mégpedig az, hogy nem veszi figyelembe a modell magabiztosságát. A tanítás során ugyanis valószínűségi értékeket is megtanul. Ha például a kimenetéből az látszik, hogy a bemeneten lévő póz 3 karakterhez tartozhat 90%-ban, és mindegyiknek nagyságrendileg egyforma az esélye, pontatlanság az egyiket győztesnek kikiáltani és behelyezni a sorba. Egy ilyen szituációban, ha lemerevítené a kezét a felhasználó, a modell ugyanazt a feltételezhetően rossz megoldást adná vissza, és a többségi döntést kijátszhatná. Egy gyakran alkalmazott technika ilyen esetben, hogy vesszük a modell javaslatait, és valószínűségeik szerint csökkenő sorrendbe rendezzük őket, majd annyit választunk a legnagyobbakból sorban, hogy a valószínűségük összege ne legyen több egy p paraméternél, de maximum k darabot választhatunk. A választásokat átskálázzuk, hogy a kisebb csoportban is 1 legyen az összegük, és valószínűségi alapon választunk közülük elemet a pufferbe. Ha a modell elég magabiztos a választásaiban, a valószínűségi viselkedésből fakadó „melléfogásokat” elfedi a felhasználó elől a többségi döntés, de kevésbé magabiztos esetben pedig nem lesz többsége egyik kategóriának sem. Ezen technika a lehetséges kimenetek számának növelésével egyre jobban működik, valamint a p és k paraméterek a modell úgynevezett „kreativitását” befolyásolják. Előszeretettel szokták ezt az algoritmust alkalmazni nagy nyelvi modellek esetében is. [34]

6.1.3.2 Nem szándékos mozgások elkülönítése

Eddig feltételeztük, hogy a felhasználó minden időpillanatban jelel. Ez a valóságban nem áll fenn, és szükség van ennek megbízható felismerésére.

Első lehetőség egy okos heurisztika alkalmazása. Ha például nem érzékel kezeket a pózfelismerés, biztosan nem jelel a felhasználó. Egy lépéssel tovább is vihető ez az

ötlet, azzal az információval kiegészítve, hogy a jelelés legtöbb esetben a test felső harmada körül, de a fejtől kissé lentebb történik. Ha viszont a rendszert például online konferenciák, megbeszélések spektrumában is el akarjuk képzelni, bizonyára elő fog fordulni, hogy az emberek akaratlanul is könyökölnek az asztalon, vagy az arcukhoz érnek. Az ilyen véletlen mozgások, valamint egyéb különleges testhelyzetek kiszűrésére nem elég tehát egy egyszerű pozíció heurisztika.

Lehetőség van egy újabb, „ismeretlen gesztus" kategória bevezetésére a klasszifikációs feladathoz. Ehhez a saját adathalmazomat kiegészítettem olyan felvételekkel, melyen jeleléshez hasonló, kényszeres cselekvéseket hajtok végre, mint például fej vakarás, könyöklés az asztalon. Az így újra tanított modell képes lesz elkülöníteni néhány esetben a nem jeleléshez tartozó mozgásokat, viszont problémája lesz az olyan esetekkel, mikor a mozgása során, bár nem jelel, mégis felvesz olyan pozíciókat a keze, melyek megfelelnek egy-egy kategóriának. Nem elég tehát a statikus pillanatfelvétel a feladat elvégzésére. A további fejezetek részletesen foglalkoznak olyan módszerek kialakításával, amelyek képesek kezelni ezt az elvi lehetőséget is, adok azonban egy ideiglenes megoldást is:

A javaslat egy másik kiegészítő modell alkalmazása, amely bináris klasszifikációt hajt végre minden időpillanatban, a jelelés tényének megállapítására. Bemenete az elmúlt n db időpillanatból származó információmorzsa is. Adatként ingyenesen elérhető videókat használtam fel, melyeket lerövidítettem, hogy csupán a tényleges jelelést tartalmazzák. A jelelést nem tartalmazó videó kialakítása kissé nehézkesebb, mert nagyon sok lehetséges elfoglaltság, és mozgáskombináció létezik. Választás során olyan szemlélettel kerestem a negatív kategóriába videókat, melyek ülő pozícióban, esetleges konferenciahívás alkalmával fordulhatnak elő. Ebben a feladatban már felhasználtam a testpóz approximációs modelleket is, ugyanis mint azt már korábban említettem a jelelés egy jellegzetes testtartás felvételével szokott történni, és ennek észlelésére hasznos lehet a kezeken kívül más adat is. Az előfeldolgozási lépések megegyeztek a klasszifikációs modell esetében alkalmazottakkal. Az n paramétert, mint a bemeneten szereplő elmúlt képkockák száma, empirikus módon hangolva 15-nek állapítottam meg. A modell architektúrájához LSTM (Long Short term memory) elemeket használtam fel. A tanulmány nem foglalkozik, ezen modell optimalizációjával, de a szekvenciafeldolgozással kapcsolatos mély tanulási apparátus lehetőségeit egy későbbi fejezet külön tárgyalja.

6.1.3.3 Összefűzés

Az eddig tárgyalt összetett modell betűzést képes felismerni. Másodpercenként 30 képkocka esetén 30 megfejtett betű keletkezik. Ha ezeket szavakká akarjuk összefűzni elég eltárolni az előző megfejtett betűt, majd csak a következő ettől különbözőt bevenni a megfejtések közé. Ujjbetűzést legtöbbször összefüggő kifejezésekre alkalmazzák, de előfordulhat, hogy külön akarjuk választani a szavakat. Ideiglenes megoldásként a szavak határait a jelelés pillanatnyi szüneteltetése jelentheti, mely akár a korábban tárgyalt jelelés felismerő modellel érzékelünk, vagy amikor a kezek elhagyják a képet.

6.1.3.4 Hibák korrekciója nagy nyelvi modellekkel

Az eddigiekben nem esett szó a j és z betűkről. Ezek ugyanis mozgást igényelnek, és statikus képekből nem különíthetőek el az i és d betűktől. Ezeket a jeleket a modell tervezett módon el fogja rontani, így szükséges egy újabb kiegészítő algoritmus vagy modell, ami korrigálja őket. Első ötletként lehetséges megfigyelni az elmúlt néhány képkockán a kezek mozgását, és ez alapján megkülönböztetni az eseteket. Ez az amerikai jelnyelv esetében működhet, de koránt sem egyszerűen általánosítható más nyelvekre, ahol esetleg jelentősen több a mozgást igénylő jel. További gond, hogy a szavakat is betűző modell jelenleg nem tud dupla betűket adni a kimenetén, az összefűzés természetéből fakadóan.

Szöveges tartalmak javítására alkalmazhatóak a nagy nyelvi modellek képességei. Bemenetként egy empirikus módon kialakított utasítás (prompt) szolgál, mely a modellt arra utasítja, hogy javítsa ki a fent említett hibákat. Az irodalom ezt „zero-shot learning” -nek nevezi, ilyenkor gyakorlatilag egyetlen utasításból „tanul” a modell mindenféle példa ismerete nélkül. Egy fokkal kifinomultabb módszer a „few-shot learning”, amely során példákat is elhelyezünk a prompt-ban. Ez gyakran segít a kimenet megfelelő strukturáltságának kialakításában is.

Négy kategóriában fogalmaztam meg utasításokat:

1. Asszisztens szerepben felkérés a korrekcióra.
2. Asszisztens szerep, explicit jelezve, hogy helyes szavak is lehetségesek.
3. Kontextus felhasználása. (pl.: állatok témaköre)
4. Általános kontextus injektálása. (Előzőhez képest zero-shot, hiszen előre nem ismert a témakör, amiben példákat adhatnánk.)

Az első két eset közül a második jelentősen rosszabb eredményt ér el. Gyakran nem változtat az egyértelműen rossz kifejezéseken, még akkor sem, ha példaként felsorolásra kerül. Az utolsó két esetben jobb teljesítmény érhető el, főleg, ha több lehetséges megoldás is létezik, hiszen a témakört megadva szűkíthető a megoldáshalmaz.

Ez az algoritmus rendkívül költség hatékony lehet egy külső nagy nyelvi modell szolgáltatás igénybevételekor, hiszen sok fejlesztési időt nem igényel, könnyen adaptálható egyéb hibák javítására is, nem beszélve más jelnyelvek integrációjáról. Hátrány viszont, hogy semmi féle minőségi garanciát nem vállal a kimenet helyességére, valamint, ha több nyelvtanilag is helyes megoldás létezik, nem lehet eldönteni melyiket jelelte a felhasználó.

6.1.3.5 MediaPipe póz felismerés kiegészítése

Mint azt már korábban is említettem, ha a bemeneti kép más képarányú, a detektált kezek koordinátái elnyúltnak torzítottnak tűnhetnek egymáshoz képest. Ezt egyszerű módon ki lehet küszöbölni a kezek kivágásával, majd a kis méretű képből póz felismeréssel. Ekkor teljesen közömbös, hogy a felhasználó milyen távol, vagy a képernyő mely szegletén helyezkedik el, minden kivágott képen ugyanolyan arányban fognak látszani a kezek, és így a felismert kulcspont koordináták is hasonlóak lesznek. Negatívum azonban, hogy ezzel elveszítjük a beépített kulcspontkövetés lehetőségét, és így csökken a póz becslés sebessége. Egy potenciális fejlesztési lehetőség lenne, a MediaPipe API felkészítése ilyen jellegű felhasználásra.

6.1.4 Értékelés

A kialakított modell, és algoritmikus apparátus alkalmas különálló szavak betűzésére, és korlátok között összetettebb kifejezésekre is. A különböző adathalmazok kombinációjaként előálló robosztus tanító adatmennyiség alkalmas a modell olyan szintű betanítására, hogy ismeretlen fényviszonyokban, változó kézméretű alanyok, a kamerától eltérő távolságból is kényelmesen használni tudják.

Izolált karakterek felismerésére tökéletes megoldás lenne, ha nem léteznének mozgást igénylő gesztusok, illetve a számokat bevonva identikus kéztartást használó jelek. Szavakká való összefűzés esetén ugyan bizonyos mértékig lehetőség nyílik ezen pontatlanságok korrigálására, de a jelek közötti váltásokból adódó hibák korrekciójára kialakított csúszó ablak méreténél fogva egy körülbelül 1 másodperces késleltetést rak a

rendszerre karakterenként. Ez kezdők számára nem feltétlenül probléma, de haladó jelelők képesek ezzel összemérhető idő alatt jelelni, illetve egyesek akár gyorsabban is, ami kellemetlenségeket okozhat. Ez a probléma feloldható lenne, ha a modellek bemeneteként nem csak egy képkockáról származó adat kerülne, hanem egy bizonyos ablakméret keretein belül a múltbéli képekről is állna rendelkezésre információ. A további fejezetek ilyen modellek kialakításával foglalkoznak.

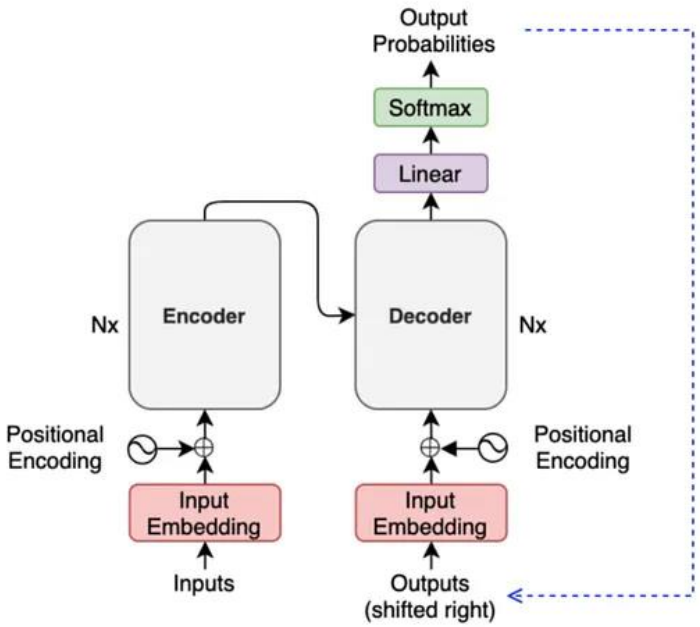
6.2 Szekvenciális ujjbetűzés

A feladatban folyamatos jelelést kell szöveggé alakítani (lásd szekvencia-szekvencia adathalmaz). A feladat sokban hasonlít a hagyományos fordításhoz, azzal a kivétellel, hogy a be és kimeneti sorozat reprezentációja, és felépítése itt merőben különbözik. (A bemenetre póz adat, a kimenetre hagyományos szöveg kerül.) Ennek ellenére egy az általános nyelvfordítás területén gyakran alkalmazott enkóder-dekóder architektúrát választottam a feladat megoldására.

6.2.1 Enkóder-Dekóder architektúra

Az enkóder-dekóder [35] modell egy olyan neurális hálózati struktúra, amely a természetes nyelvfeldolgozás (NLP), a szekvencia-szekvencia, és más feladatok megoldására szolgál. Az ilyen modell két fő részből áll: az enkóderből és a dekóderből. Az enkóder feladata, hogy magas dimenziós bemeneti adatokat alacsony dimenziós reprezentációvá alakítson. Ezt az alacsony dimenziós reprezentációt általában látens térnek nevezik, amely segít az információ tömörítésében és az esszenciális jellemzők megtartásában. Az enkóder hálózat tehát a bemeneti adatokat elemzi, és egy kontextusvektort hoz létre, amely tartalmazza a bemeneti adatok alapvető információit. A dekóder ezt a vektort veszi alapul, és kísérli meg rekonstruálni vagy előállítani a kívánt kimeneti adatokat. A dekóder egy másik neurális hálózat, amely a kontextusvektort használja, hogy a kimeneti adatokat létrehozza vagy előre jelezze. Gyakran előfordul az is, főként fordítási feladatokban, hogy a dekóder több részletben fejti meg a kimenetet, például betűnként, vagy szavanként. Ilyenkor a kontextus vektor mellett a bemenetén megjelennek az előző kimenetei. Erre is gyakran hivatkoznak kontextus néven, ezért a félreértések elkerülése végett jelen írásban az enkóder kimeneteként előállt kontextus vektort (enkóder) kontextusként, míg a dekóder bemenetén megjelenő múltbéli kimenetét dekóder kontextusként hivatkozom. Az egységet amire felosztjuk a dekóder be és kimenetét „token”-nek nevezzük. Egy példával élve, angol-magyar fordítás területén

ahelyett, hogy a „science” szóból egy lépésben „tudomány” jelenjen meg a kimeneten, betűnként futtatjuk a dekóder-t, mely megkapja a tudomány szó részleteit is bemenetként a kontextus vektor mellett, és minden alkalommal csak a következő betűről kell nyilatkoznia. Ez egyrészt azért előnyös, mert kevesebb az összes lehetséges token, ugyanis betűből kevesebb van, mint kombinációikból alkotott szavakból. Így egyszerűbb a feladat, hiszen kevesebb kategória közül egyszerűbben tanul a modell választani. Másrészt a dekóder kontextus sokat segíthet a kimenet formálásában, vegyük például az amerikai jelnyelvben a számok ujjbetűzését: nem minden esetben egyértelmű, hogy a jelelő a gesztushoz tartozó számra, vagy betűre gondol, hiszen ugyanaz a kéztartás szükséges mindkettőhöz. Ha viszont telefonszámot jelel az illető a dekóder kontextusban számok fognak megjelenni, így amikor bizonytalan jelhez érünk a kontextus a szám felé billenti a kimeneti tokenek valószínűségét. Ez az információ mellesleg az enkóder kontextusból is kinyerhető lenne közvetlenül, de a lépésekre bontás, egyrészt egyszerű és jól működő módja a változó hosszúságú kimenet előállításának, valamint ebben az esetben, ha csak az enkóder kontextus lenne a bemenet a dekóder nem tudná, hogy a kimenet melyik tokenjét generálja. Szükség van tehát a dekóder kontextusra is, az enkóder kontextus feldolgozási lépéseinek számontartására.



6.6 Ábra: Enkóder-Dekóder architektúra vázlata. [35]

6.2.2 Saját modell kialakítása

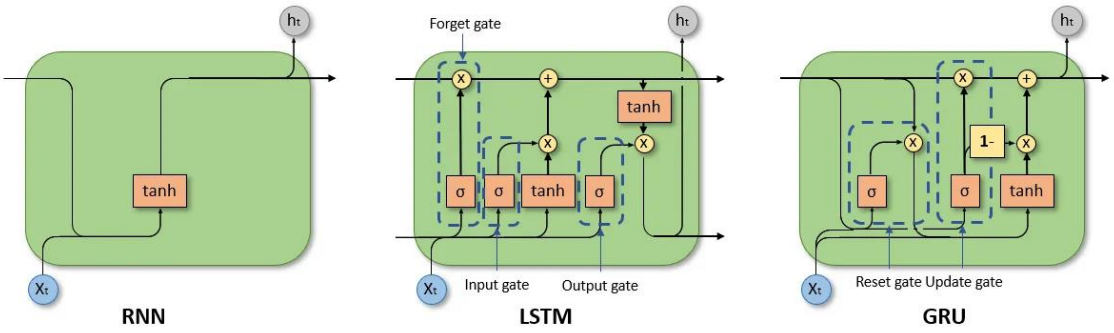
A modell kimenete, és a dekóder kontextus is szöveg, ezt viszont nem lehet közvetlenül a modell bemenetére helyezni. Az előző alpontban felvezetett módon tokenekre kell bontani, és hozzájuk számot rendelni. (Például kategória index.) A feladat jellegéből adódóan érdemes az egyes karaktereket tokennek választani. Egy gyakori lépés még a természetes nyelvfeldolgozás terén, hogy a modell megtanul több dimenziós vektorokkal reprezentálni tokeneket. Ez azért hasznos, mert a modell képes olyan vektor reprezentációt rendelni a tokenekhez, melyek hasonló kategóriák esetén valamilyen távolságmetrika szerint közel lesznek egymáshoz, nagyon különböző kategóriák esetén pedig pont ellenkezőleg távol esnek majd egymástól a vektortérben. Például, ha szavak lennének a tokenek, a „macska” szó vektorreprezentációja tanulás után közel lenne a „cica” szó-hoz tartozó vektorhoz, viszont távol az „egérhez” tartozótól. A konkrét feladatban az „a” karakterhez tartozó reprezentáció távol kerülhet a „b”-hez tartozótól, annak ellenére hogy a kategória indexeléskor sorban kiosztott számok tekintetében egymás mellett lennének, de jeleik nem hasonlítanak.

A póz adatokat nem lehet ilyen formában reprezentálni, de érdemes ennek mintájára olyan tanulható transzformációt végezni rajta, ami esélyt ad hasonló szemantikus reprezentáció kialakítására. Erre 1D konvolúciót alkalmaztam, melyek az idő dimenzió mentén mintavételezik az egyes kulcspont típusokat.

6.2.2.1 Visszacsatolt neurális hálózatok

Egy lépés maradt hátra, mégpedig az enkóder és dekóder belső kialakítása. Első megközelítésben visszacsatolt (rekurrens - RNN) háló architektúrákat próbáltam ki. Ezeket szekvenciális adatok hatékony feldolgozásához fejlesztették ki. A "state" vagy "állapot" az RNN és hasonló visszacsatolt architektúrák kulcsfogalma. Egy RNN-nél az állapot egy belső memória, amely információt tárol az eddig feldolgozott adatszekvenciáról. Minden egyes lépésnél az állapot frissül az aktuális bemenet alapján, és ez az információ halad tovább bemenetként a következő lépéshez. Ezen keresztül az RNN képes "emlékezni" a korábbi lépésekre és figyelembe venni múltbeli információt a döntések során. Előny, hogy ugyanaz az RNN képes feldolgozni különböző hosszúságú szekvenciákat, de a hosszú távú függőségek tanulása nehézkes lehet az egyszerű RNN modellek számára. Három féle verziót próbáltam ki [36]:

1. Az Egyszerű RNN (Simple RNN), minden lépésnél kombinálja az aktuális bemenetét és az előző lépésből származó információt. Nehezen tanulja meg a hosszabb távú függőségeket, ami korlátozza a felhasználhatóságát bizonyos feladatokra. Hosszú szekvenciák esetén gyakran előfordul, hogy tanulás során (hiba-visszaterjesztés algoritmus), a korrekciós értékek (deriváltak) matematikailag túl nagyra, vagy éppen túl kicsire változnak, így nem tanul a hálózat. A többi RNN ezt próbálja gátolni.
2. GRU (Kapuzott Rekurziós Egységek) A GRU egy modernizált változata az RNN-nek, amely két "kaput" használ a döntéshozatalban: az újraindítási és a frissítési kaput. Ezek a kapuk lehetővé teszik a modell számára, hogy hatékonyabban döntsön arról, mely információkat kell megőrizni vagy eldobni az idő során. Ez segít a modellnek abban, hogy jobban kezelje a hosszabb távú információs függőségeket.
3. LSTM (Hosszú Rövid Távú Memória) Az LSTM az RNN egy másik fejlett változata, amely még hatékonyabban kezeli a hosszú távú információs függőségeket. Három kaput használ a döntéshozatalban: a felejtési, a bemeneti és a kimeneti kaput, ezeknek megfelelően képes dönteni arról, mely információkat kell megőrizni, frissíteni vagy eldobni, ami lehetővé teszi számára, hogy még bonyolultabb mintákat tudjon megtanulni.



6.7 Ábra: Visszacsatolt neurális hálózat építőelemeinek vázlatos felépítése. [36]

Az enkóderben elhelyezett visszacsatolt réteg szekvenciálisan feldolgozza a bemenetet. A réteg kialakult belső állapotával inicializálva a dekóder RNN rétegét, annak információja lesz az enkóder „kontextusáról”. A dekóder bemenetére pedig a kimenet egyre hosszabb részletei kerülnek. További lehetőség van arra is, hogy több visszacsatolt

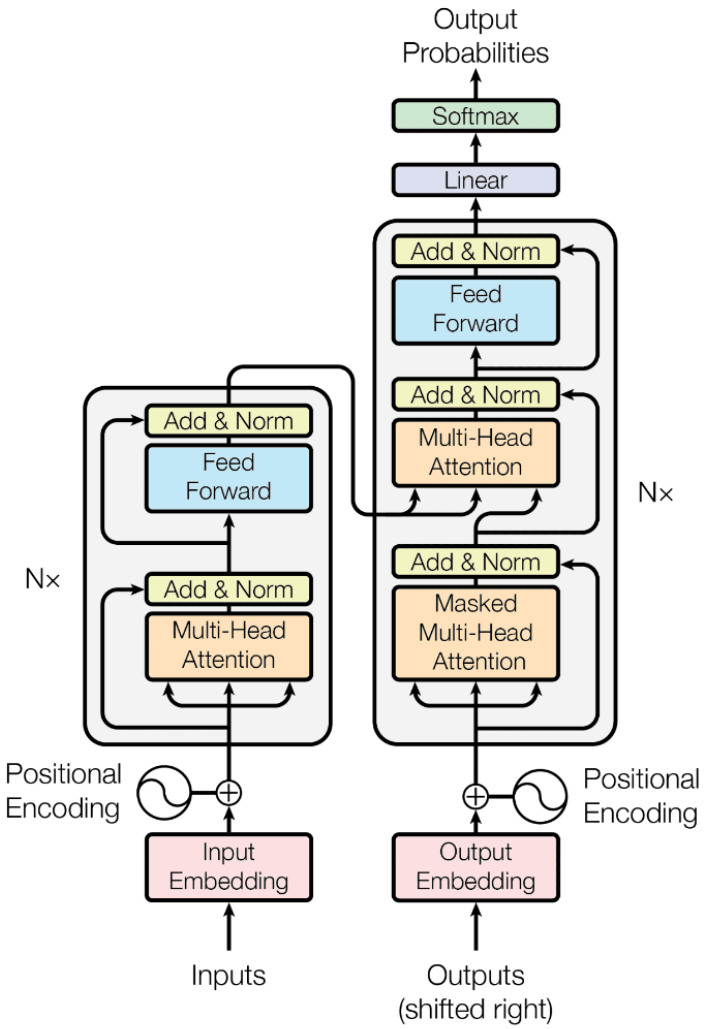
réteget helyezzünk el láncolva egymás után. Mivel minden réteg szekvenciát vár, ezért ilyen esetben szükséges a köztes lépések során számolt kimeneteket is elmenteni, majd ezeket egy szekvenciává összefűzni. Szerencsére ez a felhasznált keretrendszerben egyszerű paraméter állítással megoldható.

6.2.2.2 Transzformer modellek

A transzformer egy forradalmi architektúra típus a mély tanulásban, amely az "Attention is All You Need" című cikkben [37] jelent meg 2017-ben, és melyet eredetileg szövegek fordítására terveztek. Lényege az úgynevezett figyelem mechanizmusban (attention) rejlik, ami lehetővé teszi a modell számára, hogy az egész adatszekvenciára súlyozottan figyeljen, nem csak az aktuális vagy közvetlenül előző lépésekre, továbbá több párhuzamos „fejet” is tartalmazhat a figyelmi mechanizmus (multihead attention), ilyenkor különböző aspektusait tanulja meg megfigyelni az adatnak.

A transzformerek nem rendelkeznek beépített módszerrel a szekvenciális bemenetek sorrendjének figyelembevételére, ellentétben az RNN-ekkel, amelyek szekvenciálisan dolgozzák fel a bemeneteket. Emiatt van szükség a pozícionális kódolásra (positional encoding): hogy információt adjon a modellnek az egyes elemek helyzetéről a szekvenciában. Ezt a kódolást úgy adják hozzá a bemeneti vektorokhoz, hogy az ne torzítsa az eredeti adatokat, de mégis információt adjon a modellnek az elemek sorrendjéről.

A transzformerek nem csak enkóder-dekóder formában élnek, létezik még úgynevezett enkóder only (BERT), és dekóder only (GPT) változata is, ezek arra épülnek, hogy ki lehet úgy alakítani mind az enkóder, mind a dekódert, hogy egymás után kapcsolható legyen. A dolgozatban vizsgált modellt az eredeti 2017-es tanulmány alapján alakítottam ki. Ebben az enkóder és dekóder figyelmi mechanizmust alkalmaz a bemenetére, valamint a dekóder az enkódertől kapott kontextus vektorra. Figyelmi mechanizmusként multihead attention-t használtam 4 fejjel egységesen, valamint 2 enkódert és 4 dekódert találtam optimálisnak a feladatra.



6.8 Ábra: „Attention is all you need" [37] tanulmányban bemutatott transzformer modell architektúrája.

6.2.3 Teacher forcing algoritmus

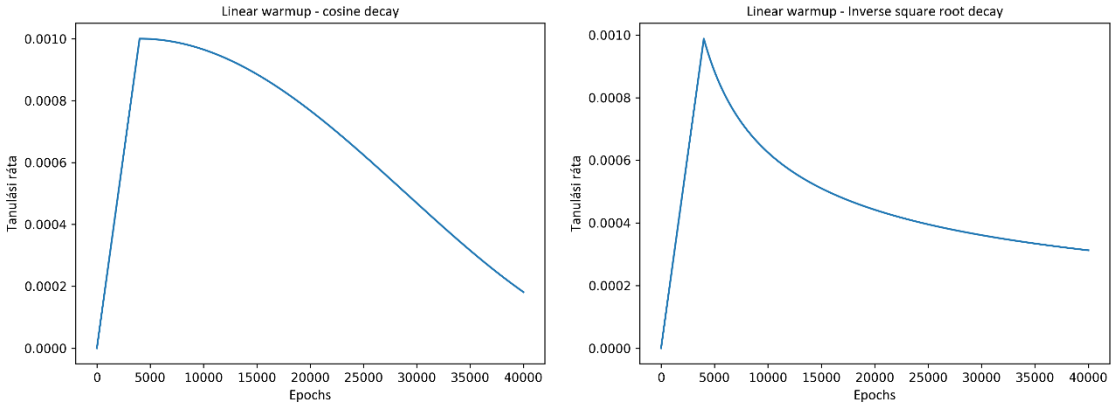
Amikor soros adatokon alapuló modelleket tanítunk (például szövegfelismerés vagy fordítás esetén), az eredeti megközelítés az, hogy a modell által az előző lépésben előállított kimenetet használjuk fel bemenetként a következő lépésekben. A gond ezzel a megközelítéssel az, hogy a tanítás során a modell hibája összeadódik, mivel minden rosszul megállapított token hatással van az összes következő lépésre. A "teacher forcing" [38] technika itt jön képbe. Az ötlet az, hogy a tanítás során a valódi célbemenetet használjuk fel a következő lépésekben, nem pedig a modell által az előállított kimenetet. Ez általában gyorsabb és hatékonyabb tanuláshoz vezet. Példaként vegyünk egy egyszerű fordítási feladatot, ahol a bemeneti mondat: "I am learning" és a célmondat (fordítás): "Én Tanulok", az egységnyi kimenet pedig a szavak. Ha nem használunk teacher forcingot, és

a modell az első szó után nem megfelelő eredményt ad (például "Te"), akkor ez a hiba befolyásolhatja a következő kimeneteket is. Teacher forcing használatával azonban a második szó „kitalálásához” közvetlenül az "Én Tanulok" első szavát, azaz "Én"-t használjuk bemenetként, függetlenül attól, hogy a modell mit adott első szóként.

6.2.4 Metrikák és egyéb hiperparaméterek

A bemeneti adatok, mint póz szekvenciák, illetve hozzá tartozó szövegek nem egyforma hosszúságúak, viszont tanítás során nem egyesével kerülnek feldolgozásra, hanem csoportosítva (batch). A szekvenciák mindkét esetben ki vannak egészítve egy maximális méretre, hogy a hossza mindnek egyforma legyen. Ezért a tanítás során használt hibafüggvénynek, illetve találati arány (accuracy) függvénynek is egy maszkolt verzióját használtam, amik ignorálják a kitöltő (padding) karaktereket.

Optimalizálóként az „adam” algoritmust alkalmaztam testreszabott tanulási rátával (learning rate). Az alkalmazott módszer a „warmup and decay”. A tanulás elején, amikor a modell súlyai teljesen véletlenszerűek, túl nagy lépések a súlyok frissítésében instabilitást okozhatnak. Az első (warmup) periódus alacsony tanulási rátával kezdődik, így a modell lassabban alkalmazkodik a kezdeti időszakban, megelőzve a divergenciát. A maximum elérése után értékét fokozatosan csökkentve (decay), a tanulás későbbi szakaszaiban megakadályozza, hogy a modell túlságosan alkalmazkodjon az adathalmazhoz. (overfitting) Az RNN alapú hálózatoknál lineárisan növelem a tanulási rátát, majd koszinuszos a lecsengés. A transzformereknél az „Attention is all you need” [37] tanulmányban ajánlott „linear warmup inverse squareroot” formulát használtam.



6.9 Ábra: Tanítás során felhasznált tanulási ráta ábrázolása a tanítási lépések függvényében.

6.2.5 Tanítás

Az 5.2.1 fejezetben bemutatott szekvencia-szekvencia adathalmazt használtam fel. A tanító adatok 80%-át használtam a modellek paramétereinek hangolására, 10%-át validációs célokra, és a maradék 10%-án teszteltem. Ezen felül szükségét éreztem a tanító halmaztól függetlenebb mintákon is összehasonlítani a modelleket, de sajnos a verseny során használt hivatalos teszt adathalmaz nyilvánosan nem elérhető, ezért a korábban már említett kiegészítő (supplemental_metadata.csv) sorozatokat is felhasználtam tesztelési célokra. A vizsgált metrikák: Levenshtein távolság (LEV), Szavak hiba aránya (Word error rate - WER), Karakter színtű hiba arány (Character error rate - CER), továbbá a kimenet generálását Teacher forcing (TF) módszerrel, és a modell saját kimenetét kumulálva is elvégeztem, így mindegyik metrikából 2 verzió is elérhető. Az egyszerű enkóder-dekóder modelleket, a visszacsatolt hálózati egység elnevezése azonosítja, míg a transzformer modellek esetén a PE rövidítés Pozicionálás kódolást jelent. A modellek egységesen 30 Epoch hosszan voltak tanítva, ahol 1 Epoch az adathalmaz egyszeri végig futtatása az egyes modelleken.

Modell/Metrika	LEV	LEV TF	WER	WER TF	CER	CER TF
Egyszerű RNN	18.690	12.030	1.000	1.022	1.077	0.681
LSTM	8.655	4.280	0.871	0.796	0.457	0.220
GRU	7.370	3.605	0.808	0.738	0.376	0.185
Transformer PE	3.150	1.940	0.663	0.615	0.175	0.112
Transformer	**2.220**	**1.170**	**0.453**	**0.4222**	**0.124**	**0.068**

6.2 Táblázat: Modellek teljesítménye a tanító adathalmaz tesztelési célokra elkülönített 10%-án.

Modell/Metrika	LEV	LEV TF	WER	WER TF	CER	CER TF
Egyszerű RNN	27.900	22.495	1.000	1.000	1.000	0.807
LSTM	20.500	19.430	0.999	1.000	0.733	0.693
GRU	19.665	18.570	1.000	1.000	0.704	0.664
Transformer PE	**11.780**	**9.725**	**0.990**	**0.967**	**0.418**	**0.345**
Transformer	12.585	10.040	0.999	0.975	0.446	0.356

6.3 Táblázat: Modellek teljesítménye a kiegészítő adathalmazon.

A legjobb eredményt a transzformer architektúrák érték el. Hipotézis arra, hogy miért teljesített jobban a Pozíció kódolást használó verzió a kiegészítő adathalmazon, a következő: a pozíció kódolás feltehetően sokkal jobb általánosító képességgel ruházza fel a modellt. Ez a dolgozat keretein belül nem került bizonyításra, de jó lehetőséget ad további kutatások indítására.

A kiegészítő adathalmazon azért jelentősen rosszabbak az eredmények, mert egyrészt jellegét tekintve nem linkek, címek találhatóak benne jelelt formában, mint a tanító adathalmaz esetén, hanem összefüggő szöveg. Másrészt a kiegészítő adathalmazon átlagosan 6.7, míg a másik esetében 10.41 képkocka jut egy jelelt karakterre, így minőségben jelentősen rosszabb.

6.2.6 Kiegészítő algoritmusok

Egy korábbi fejezetben említésre került, hogy a modellek bemeneti kontextusa (sorozatok hossza) limitált. Szükséges valamilyen technika, annak kezelésére mikor a fordított jelelés túllóg a kontextuson.

6.2.6.1 Rövid szekvenciák fordítása.

Ennek lényege, hogy az egyes betűket, próbáljuk meg fordítani. Ehhez a modell bemenetére csak az elmúlt néhány jelelt karaktert kapja meg, valamint a dekóder az utolsó néhány megfejtett tokent. Az intuíció az, hogy ha a modell megtanulta a karakterek jelelését, akkor izolált módon is képes őket megfejteni. A modell ilyen használata során nagyon érzékeny a bemeneti ablak méretére. Ennek oka egyszerű, különböző tempóban jelelnek az emberek, így nem egyforma mennyiségű póz adat kerül a bemenetre. Továbbfejlesztési lehetőségként említem meg, hogy feltételezhetően lehetséges egymást követő felvételekről a kéz sebességének, és egyben a jelelő gyakorlatiságának megállapítása, valamint ezen adatok összefüggésbe hozása a paraméterek értékével.

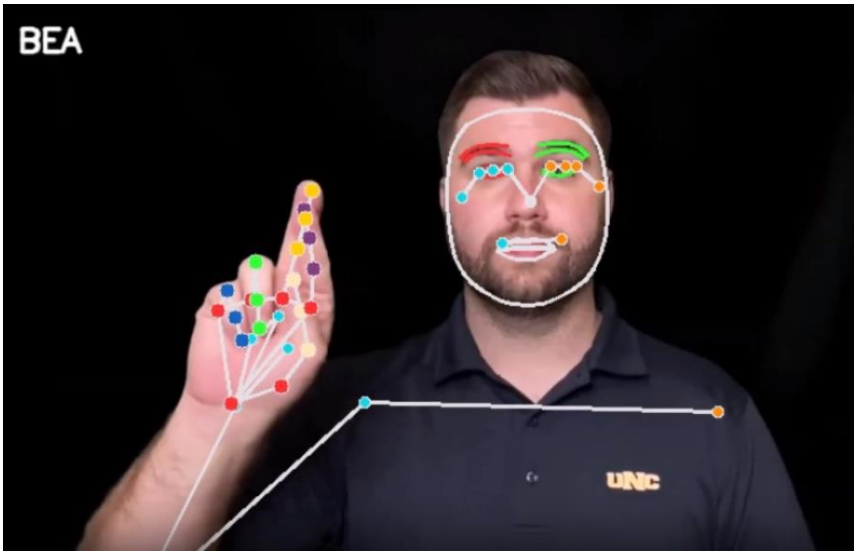
6.2.6.2 Hosszú egybefüggő szekvenciák fordítása

Hasonlóan a tanító adatokhoz, bevár egy hosszabb egybefüggő sorozatot, majd egyben fordítja le. Figyelembe veszi a modell magabiztosságát, és paraméterben szabályozható, milyen értéktől adható a modell kimenete a kontextushoz. Sokkal pontosabb, és jobb minőségű fordításokat ad ez a módszer, azonban van egy hátulütője: a modell architektúrája miatt a maximális szekvencia méret tanítási időben eldől. Ha hosszabb szekvenciákat szeretnénk fordítani valamilyen módon itt is darabolni kell azokat, ami kis mértékben visszavetheti a teljesítményt. Jelelést detektáló modellel való kombinációt is kipróbáltam, ilyenkor csak azon adatokat gyűjtjük egy pufferbe majd fordítjuk, melyeken jelelés történik. Ez a módszer rosszabb eredményeket hozott.

6.2.7 Értékelés

Összességében jelentős előrelépést értem el az Amerikai Jelnyelv ujjbetűzésének felismerésében és fordításában a transzformerek és kiegészítő algoritmusok segítségével, a statikus ujjbetűzéshez képest.

A modell felépítésében az enkóder-dekóder architektúra és saját fejlesztésű megoldásaim együttes alkalmazása volt a kulcs. A domináns kéz kiválasztása és a kezek normalizálása a pózadatokhoz jelentősen hozzájárult az eredmények javulásához. A visszacsatolt neurális hálózatok, valamint a transzformerek alkalmazása lehetővé tette a modell számára, hogy hatékonyabban dolgozzon az időben változó jelekkel. A feladatra legjobbnak a transzformer architektúrát találtam. A tanítási folyamat során különböző hiperparaméterek és tanulási stratégiák, mint például a "teacher forcing" és a "warmup and decay" algoritmusok, valamint a maszkolt hibafüggvény alkalmazása szintén elősegítették a magasabb teljesítményt. Végezetül a valós videókhoz való alkalmazkodást segítő kiegészítő algoritmusok bevezetése azt mutatja, hogy a modell nemcsak a teszt adatkészleteken, hanem valós körülmények között is jól teljesít. Összességében ezek az innovációk és optimalizálások javították a teljesítményt az ujjbetűzés felismerésében és fordításában, a statikus bemenetű modellekhez képest.



6.10 Ábra: Rövid szekvenciákat fordító modell kipróbálása videón rögzített jeleléssel.[5]
(A jelelt szó a „bear” vagy medve.)

[5] https://www.youtube.com/watch?v=sbHJYaZ8UWE&ab_channel=DEAFiningASL

6.3 Hagyományos jelelés

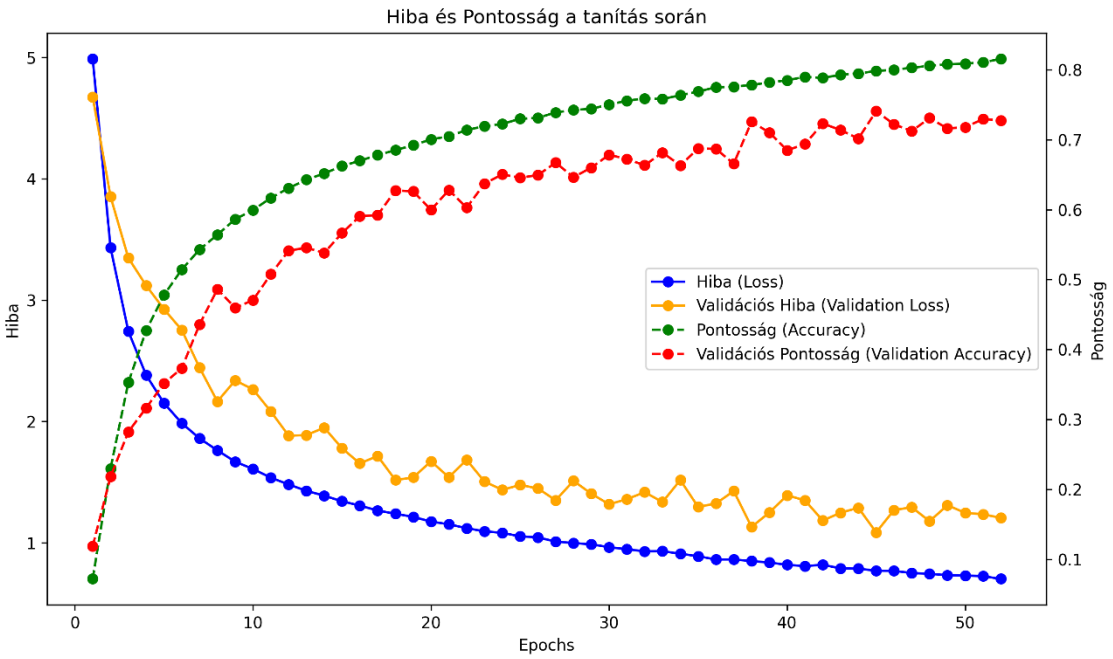
Az ujjbetűzés és a hagyományos jelelés közötti különbségek megértése elengedhetetlen azok számára, akik a siket és nagyothalló közösségekkel szeretnének kommunikálni. Az ujjbetűzés, amelyet gyakran kézábécének is neveznek, az egyes betűk kézmozdulatokkal való ábrázolására épül. Ez a módszer különösen hasznos lehet egyes szavak, nevek és a speciális szakkifejezések pontos közvetítésében, amelyeknek nincs meghatározott gesztusa a jelnyelvben. Ezzel szemben a hagyományos jelelés abban különbözik, hogy a lehetséges gesztusok száma jelentősen több.

Ebben a részfeladatban az 5.2.2-ben bemutatott adathalmaz segítségével azt vizsgálom, hogy szekvenciális póz adatokból megkülönböztethető-e nagy számú kategória, a betűk és számok szűk halmazával ellentétben. Számszerűen 250 jel közül a modellnek választ kell adni a klasszifikációs feladatra.

6.3.1 Architektúra

A vizsgált architektúra Hoyeol Sohn Kaggle versenyen alkotott, győztes modelljének leírása alapján készült. [39] Az alap ötlet, hogy a bemeneti szekvenciából állítsunk elő egy kontextus vektort. Az 6.2.1 fejezetben tárgyalt enkóder-dekóder modellek enkóder részei is pontosan ezt a feladatot hajtják végre. Itt azonban a kontextus vektort a dekóder helyett egy más felépítésű, klasszifikációra kialakított hálózatba vezetjük.

Amiben újat mutat Hoyeol Sohn megoldása, hogy egymás után pakolt enkóder rétegek helyett, konvolúció blokkokat használ, hatékony csatorna figyelemmel (Efficient channel attention - ECA) ötvözve. Az ECA egy olyan figyelem alapú mechanizmus, melynek célja a konvolúciós neurális hálózatok (CNN-ek) hatékonyságának és teljesítményének javítása azáltal, hogy kiemeli vagy elnyomja a különböző csatornák jellemzőit. Ez az adathalmazban jelen lévő emelkedett számú padding érték miatt rendkívül előnyösnek bizonyult. További előnye, hogy hatékonyabb és gyorsabb, mint a klasszikus figyelmi mechanizmus alapú megoldások. A modell ezen része tekinthető gyakorlatilag egy különleges beágyazó rétegként a póz szekvenciák feldolgozására.



6.11 Ábra: Hiba és pontosság alakulása tanítás során.

6.3.2 Értékelés

A saját megvalósításban a modell képes Hoyeul Sohn modelljéhez hasonló teljesítményt nyújtani pontosság terén. Sajnos a teszt adathalmaz nem publikus, így mélyrehatóbb összehasonlítás nem lehetséges. Mivel a szekvencia-szekvencia modelleket jobbnak ítéltem meg a feladatra, ehhez a megoldáshoz nem készítettem kiegészítő algoritmusokat, de ettől függetlenül ezen kutatás folytatásaként érdemes lenne megvizsgálni, hogy a modell tanult kontextus vektora, valóban szemantikailag helyes-e, vagyis hasonló jelekhez valamilyen távolság metrika alapján hasonló kontextus vektor értékek tartoznak.

6.4 Technikai megvalósítás

A megvalósítás során a Tensorflow keretrendszert használtam. Ez az eszköz a mélytanulásban és gépi tanulásban elérhető modellek kialakítása során nyújt kiemelkedő támogatást. Ezek közül három fontos megközelítés emelkedik ki:

1. A **Sequential** API az egyik leginkább egyszerűsített és struktúrált módja a neurális hálózatok létrehozásának a TensorFlow-ban. Ez az API lehetővé teszi a rétegek egymás utáni hozzáadását lineáris módon, ami ideális az egyszerű előre csatolt neurális hálózatok modellezéséhez. A statikus ujjbetűzés rész DNN és CNN modellje is ezzel az API-val készült.

2. A **Functional** API erőteljes eszköztár a modellépítéshez, amely lehetővé teszi bonyolultabb, elágazó struktúrák definiálását és tetszőleges kapcsolatok kialakítását a rétegek között. Ezáltal rugalmasabb modelleket hozhatunk létre, például többszintű és többszörös bemenetű, vagy akár többszintű kimenettel rendelkező modelleket.
3. A **Subclassing** API a legnagyobb szabadságot biztosítja a modell kialakításában, lehetővé téve az egyedi rétegek és műveletek saját osztályainak létrehozását. Ez az eljárás ideális, ha teljes kreatív kontrollra van szükségünk a modellünk struktúrájában, és olyan egyedi összetettségű modellek építését tűzzük ki célul, amelyek a standard API-k által kínáltakon túlmutatnak. Lehetőség van az előző kettővel való kombinálásra is. Ezt kihasználva készült az összes enkóder-dekóder architektúra.

6.4.1 Sorosítás

A tanítás végeztével a modelleket perzisztens módon tárolni kell. Tensorflow-ban ehhez három kulcsfogalmat kell érteni:

1. **Modell architektúra** a neurális hálózat rétegeinek konfigurációjára utal, meghatározza azok szerkezetét beleértve:
    a. Típusait (pl.: Dense, Konvolúciós, LSTM stb.)
    b. Beállított paramétereit (pl.: neuronok száma egy Dense rétegben)
    c. Modell topológia, avagy milyen sorrendben kapcsolódnak egymáshoz a rétegek.
2. **Súlyok**, melyek eredetileg random paraméterek, de tanítás során optimális értékek fele konvergálnak.
3. **Végrehajtási gráf**, amely specifikus a TensorFlow működési módjához, és a neurális hálózatban végrehajtott műveleteket és adatáramlást modellezi. A gráfban a csomópontok a műveletek (mint matematikai vagy réteg-számítások), míg az élek a csomópontok között áramló tenzorokat (többdimenziós adattömböket) képviselik.

Ha a fejlesztés során csupán a beépített alapelemeket használtuk fel. Például Dense, Konvolúciós, Dropout réteg, akkor lehetőség van csupán az első kettőt elmenteni, ez egy beépített függvénnyel meg is tehető, majd szintén egy egyszerű hívással visszatölthető. Ha saját rétegeket is fejlesztünk, már nincs ilyen egyszerű dolgunk. Ilyenkor, ha menteni szeretnénk, szükséges a réteg szerializációs logikáját testre szabni, és betöltéskor szükséges a réteget reprezentáló objektumból egy példány. Ez nehézkes lehet hisz, ha a rétegnek vannak függőségei azokat is hordozni kell. Az egyes verziók között nem biztos, hogy garantált a kompatibilitás. Ezen felül komplex esetekben nem egyszerű megvalósítani a mentési logikát sem. A harmadik módszer a gráf konverzió. Amikor egy modellt „SavedModel" formátumban mentünk, a TensorFlow az egész modellt, beleértve az egyéni rétegeket és függvényeket is, számítási gráffá alakítja át. Ez a gráf teljes reprezentációja a végrehajtott számításoknak, és nem igényli az eredeti Python kódot a futtatáshoz. Egy hátránya van, hogy komplexebb testreszabott modelleknél könnyű belefutni olyan hibaüzenetekbe, amiket nem egyszerű feloldani, tehát a módszer nem annyira felhasználó barát.

6.4.2 Paraméterek kezelése

A kialakított modelleknek rengetek külső paramétere van. Legyen szó az adatok reprezentációjáról, előfeldolgozásáról, bemeneti és kimeneti tenzorok formájáról, illetve jelentéséről. Ahhoz, hogy a mentett modellt használni tudjuk sok inicializáló kód szükséges, melyet manuálisan kell karbantartani és projektről projektre hordozni.

Ebben segít a tf.Module, ami egy alapvető osztály a TensorFlow-ban, és mely különösen hasznos az alacsony szintű keretrendszer funkciók használatakor vagy modellek nulláról történő építésekor. Segít a változók és almodulok szervezésében, egyszerűsíti a mentési és betöltési folyamatokat, lehetővé teszi a modellkomponensek újra felhasználását, és zökkenőmentesen integrálódik a TensorFlow ökoszisztémájába. A tf.Module különösen akkor hasznos, amikor a modellek nem illeszkednek a standard paradigmákhoz vagy amikor alacsony szintű funkciókra van szükség, mint például komplex tenzorműveletek megvalósításában, vagy kutatási és kísérleti környezetben, ahol nagyobb rugalmasságra vagy alacsonyabb szintű irányításra van szükség a modell építése és működése során.

Vegyük például a kifejlesztett enkóder-dekóder modellt. Ismerni kell, hogy milyen póz adatokból dolgozik, hordozni kell a pontos előfeldolgozáshoz szükséges

kódot, valamint emlékezni kell, hogy a modell mekkora bemeneti szekvenciát, illetve kontextust tud kezelni, valamint a kimenetek szemantikáját is tudni kell értelmezni. (Például melyik kimenet milyen karakterhez tartozik.) Jó lenne egy egységes keretbe foglalni ezeket. Erre tökéletes megoldás volt a tf.Module.

A megvalósított architektúrában minden modell egy tf.Module-on belül működik. Utóbbin két metódus érhető el: egy „info” nevű, amivel lekérdezhető pontosan melyik kulcspontokra van szüksége a fordításhoz. Ez alapján írható olyan logika, mely automatikusan a modell számára befogadható formátumra hozza a MediaPipe api kimenetét. Ez a logika is megvalósítható lenne a tf.Module-ban, de ekkor a modell nem lesz használható más póz felismerő könyvtárral a jövőben, ezért ezt egy külön osztályként valósítottam meg. A másik metódus a „predict” névre hallgat, és egymagában elvégzi a bemenet előfeldolgozását, modell futtatását (inferencia), valamint kimenetének értelmezését. A visszaadott érték egy python string a megfelelő karakterrel, és egy hozzá tartozó magabiztossági százalék. Ez az egész objektum egyszerűen sorosítható, majd bármi féle inicializáló kód nélkül betölthető.

7 Általános jelnyelv fordítás

A jelnyelvek teljes értékű nyelvek, melyek saját szintaktikai és morfológiai szabályokkal rendelkeznek, és nem csupán a beszélt nyelv szó szerinti átültetései. A jelnyelvben a jelek, arckifejezések, testtartás és a térbeli elhelyezkedés összjátéka adja át az üzenet teljes tartalmát és érzelmi mélységét, ami lehetővé teszi a bonyolult és árnyalt kommunikációt, ami messze túlmutat az egyszerű szó-szóra történő fordításon.

Az ASL-ben nagyságrendileg 10000 jel létezik, melyek nagy része dinamikát is igényel, így a statikus módszerek biztosan nem alkalmasak a feladatra. Az egyetlen megoldás tehát, hogy szekvenciális bemenetből dolgozzunk. Ha pedig folyékony jelelést akarunk fordítani, akkor a feladat, hasonlóan az előző fejezetben tárgyalt dinamikus ujjbetűzéshez, egy szekvencia-szekvencia algoritmust igényel. Léteznek olyan módszerek, amivel egy-egy jeleket felismerő modell kimenetét lehet szekvenciává összefűzni, de ez egyrészt nehézkes, másrészt a kutatásom tapasztalatai alapján egy szekvencia be és kimenetű modellt javaslok potenciális megoldásként az általános fordításra. Azon belül is a jövőt, a szekvenciális ujjbetűzés részben tárgyalt modellekhez hasonlóan, egy módosított transzformer architektúrában látom. Persze míg ott tokenként a betűk, és számok szolgáltak, addig általános jelnyelvben célszerű lehet a külön gesztussal rendelkező szavakat, kifejezéseket is felvenni közéjük. Ehhez jóval nagyobb méretű és komplexitású modellekre van szükség, valamint a feladat nehezedése miatt még több adatra, mint az előző fejezetekben vizsgált tanítások során. Párhuzam vonható a nagy nyelvi modellek (LLM – Large Language Model), és a „nagy jelnyelvi modellek” (LSLM – Large Sign Language Model) között. Előbbi esetben a tanító tokenek száma sok esetben a trilliós nagyságrenddel vetekszik, míg csak viszonyítási alapként a szekvenciális ujjbetűzés során felhasznált adathalmaz csupán 3 millió tokent tartalmazott. Véleményem szerint ez a hatalmas szakadék szab gátat jelenleg a terület fejlődésének, és nem a megfelelő algoritmusok hiánya.

8 Összefoglalás és továbbfejlesztési irányok

A dolgozatomban arra a kérdésre kerestem a választ, hogy alkalmasak-e a jelenleg elérhető póz felismerő algoritmusok jelnyelv teljes értékű fordítására. Ehhez először megvizsgáltam néhány konkurens tanulmányt mind a segédeszközökkel dolgozó megoldások, mind a hagyományos képfeldolgozás területéről. Részletesen foglalkoztam a póz felismerő modellek algoritmikus hátterével, valamint három részfeladaton keresztül jártam körbe az amerikai jelnyelv fordítását.

1. Statikus ujjbetűzés témakörén belül sikerekkel övezve foglalkoztam képek felbontásának növelésével mind hagyományos matematikai, mind mély tanuláson alapuló módszerekkel, fekete fehér képek megszínezésével, ezáltal képi médiumon alapuló adathalmazokat sikeresen konvertáltam póz adatbázissá. Ezen felül saját adatbázist hoztam létre. Az együttes adaton sikerült olyan modellt fejlesztenem, mely változó környezeti viszonyok közepette is képes megállni a helyét.
2. Dinamikus ujjbetűzés terén több szekvencia feldolgozásra alkalmas modell eredményeit hasonlítottam össze. A teljesség igénye nélkül megvizsgáltam a visszacsatolt neurális háló elemekből képzett enkóder-dekóder hálózatokat, valamint egy általános nyelv fordítására szánt transzformer modellt is adaptáltam a problémára. Az eredményeket professzionális jelelést tartalmazó videófelvételeken is ellenőriztem.
3. Kitekintést tettem az általános jelnyelv fordítására, valamint megvizsgáltam egy modellt, amely képes 250 jel között nagy százalékban különbséget tenni.

A mesterséges intelligencia modellek tanításán kívül részletesen foglalkoztam azok éles környezetben történő alkalmazhatóságának fejlesztésével, valamint komplex alkalmazásba történő integrációjával.

8.1 Összegzés

A póz felismerést teljes mértékben alkalmasnak tartom a jelnyelv átfogó fordítására, és ebben látom a terület jövőjét. A dolgozat elején megfogalmazott, ezzel a módszerrel kapcsolatos célokat teljesítettem, hisz a megoldásaim, a 3D pózfelismerésen

keresztül a kéz valós fiziológiai modelljével dolgoznak. Saját könnyen kezelhető, univerzális és információt jól tömörítő adatbázist hoztam létre. A képi adathalmaz javítására szolgáló, színező és felbontás növelő módszerek bizonyítják a meglévő adatbázisok felhasználhatóságát. Az elkészült alkalmazások pedig valós időben végzik a fordítási feladatokat.

Úgy gondolom, hogy a területben rengetek potenciál rejlik még. Ezt bizonyítják az aktívan szervezett Kaggle versenyek, ahonnan két adathalmaz is származik, és melyek csupán pár hónapja értek véget. Remélhetőleg a múltbéli friss aktivitás a nagy cégek részéről, mint a Google, aki a versenyeket szervezte, vagy az Nvidia mérnökei, akik megnyerték, nagyban előre fogja lendíteni a póz adathalmazok fejlődését, és hamarosan láthatunk majd a piacon is „nagy jelnyelvi modelleket”.

A tanulmány során fejlesztett kódom, valamint mintavideók a kész megoldásról elérhetők online a https://github.com/dancsomarci/sign-language linken.

8.2 Továbbfejlesztési lehetőségek

Bár a kutatást sikeresnek könyvelem el, a valóság az, hogy amivel foglalkoztam az csupán a jéghegy csúcsa. Számos további lehetőség vár még kipróbálásra, amik közül néhányat szeretnék kiemelni.

8.2.1 Szekvenciális adathalmazok bővítése

Mint azt már korábban említettem a területet jelenleg a robosztus, nagyméretű adathalmazok hiánya fogja vissza. Ennek tudatában pozitívan állítom, hogy csak idő kérdése lenne a jelenleg nagy népszerűségnek örvendő nyelvi modellek mintájára „nagy jelnyelvi modellek” megjelenése. A probléma abban a tényben gyökerezik, hogy míg szöveges adatforrások az emberek interneten folytatott napi tevékenységének köszönhetően temérdek mennyiségben rendelkezésre állnak, a jelnyelv viszont természeténél fogva kevésbé elterjedt az online térben. A javaslat egy platform létrehozása, ahol a résztvevők egységes keretek között tudnak adatot szolgáltatni kutatási célokra, mind kamerával való interakció során, mind meglévő képi, illetve videó formátumú adathalmazok feltöltésével és automatikus konvertálásával. A lényeg az lenne, hogy egy szabványosított póz formátumot, és adatbázist hozzunk létre, mely ezen túl a terület alapjául szolgálna.

8.2.2 Póz felismerő algoritmusok fejlesztése

Bár a jelenleg elérhető megoldások jól működnek, néhány területen van még lehetőség a fejlődésre:

- Erős háttérvilágítás, a bőrhöz hasonló színű, textúrájú tárgyak jelenlétében gyakran bizonytalan a felismerés. Szükséges egy kontrasztos háttér a jó eredmények érdekében.
- Kamerától távol lévő testrészek esetén a felismerés bizonytalansága nagyobb. Valós tesztek során arra is fény derült, hogy a fejlesztett modellek konfidenciája a kamerától való távolodás során csökken.
- Kezek egymással, valamint saját magával történő interakciója (pl.: keresztbe tett ujjak) gyakran bizonytalan.
- A jelenlegi modellek nagy része a valós idejű működést célozza meg. Célszerű lenne olyan megoldásokat fejleszteni, ahol nem kritérium a csökkentett hardver igény, és gyors inferencia, cserébe akár rosszabb minőségű videókból is képes jó minőségű adatot kibányászni. A jövőben a hardver fejlődésével ezek is elérhetőek lesznek majd akár mobilos környezetben is, de addig is jobb minőségű adattal lehetne modelleket kialakítani.

8.2.3 Hagyományos jelelés

A dolgozatban csak korlátozott mértékben foglalkoztam az ezen a területen fellelhető adathalmazok és modellek vizsgálatával. Ennek oka, hogy a kutatás a póz adat felhasználhatóságának vizsgálatára irányult, valamint a szekvencia-szekvencia feladatra kialakított modellek sikeresebbek voltak. Azonban ezen a területen számos megválaszolatlan kérdés létezik még. Ilyen például, hogy lehetséges-e olyan rendszert tervezni, amely újra tanítás nélkül képes új gesztusok azonosítására. Egy lehetséges ötlet, a klasszifikációs, illetve szekvencia fordítás feladat tanítása során megtanult belső kontextusvektor vizsgálata. Vajon a modell tényleg szemantikailag helyes reprezentációt tanul meg, ahol az egymáshoz kapcsolódó hasonló jelek közeli vektorokba képződnek le? Hogyan lehetne ezt a tulajdonságot előidézni, segíteni? Megmarad-e ez a tulajdonság ismeretlen gesztusok estén is?

8.2.4 Jelelés detektálás

A 6.1.3.2 fejezetben bemutatott kezdetleges jelelés detektáló modell fejlesztése kulcsfontosságú, ha a fordító modelleket éles környezetben, például egy konferencia hívásban akarjuk alkalmazni. Ekkor ugyanis az önkénytelen mozgások felismerése elengedhetetlen a megfelelő működéshez. Ebben a témakörben léteznek már póz felismerésre épülő megoldások. Például a „Real-Time Sign Language Detection using Human Pose Estimation” [40] című tanulmányban a szerzők 91%-os pontossággal ismerték fel a jelelést. Ez remek eredmény, de érdemes lenne megvizsgálni, hogy a dolgozatomban fejlesztett modellek működőképesek-e ilyen pontosság mellett is, hiszen, ha kommunikáció közben véletlenül rosszul azonosítjuk a jelelés tényét, nem biztos, hogy a hiányos információ is elég a megfelelő minőségű fordításhoz. Túl nagy késés esetén például levágódhat a mondat eleje, és ezzel jelentősen romolhat a pontosság.

További érdekes kísérlet lenne kombinálni a saját modelljeimet az imént említett tanulmány [40] fő ötletével, miszerint a nyers póz információ helyett az egyes képkockák pózai között számolt különbségeket a bemenetre helyezve, a fordító modell feltehetően jobban meg tudná különböztetni a szüneteket, illetve véletlen mozdulatokat.

9 Köszönetnyilvánítás

A kutatás az Európai Unió támogatásával valósult meg, az RRF-2.3.1-21-2022-00004 azonosítójú, Mesterséges Intelligencia Nemzeti Laboratórium projekt keretében.

10 Irodalomjegyzék

[1] World Health Organization, *"Deafness and hearing loss,"* 2023. https://www.who.int/health-topics/hearing-loss#tab=tab_2 (accessed Nov. 30, 2023).

[2] William Woods University, *"Sign language around the world,"* 2016. https://asl-blog.williamwoods.edu/2016/01/sign-language-around-the-world/ (accessed Dec. 01, 2023).

[3] Gallaudet University, *"American Sign Language & French Sign Language."* https://gallaudet.edu/museum/history/american-sign-language-and-french-sign-language/ (accessed Dec. 01, 2023).

[4] H. D. W. Stiles, *"A Brief History of BSL,"* 2012. https://blogs.ucl.ac.uk/library-rnid/2012/07/06/a-brief-history-of-bsl/ (accessed Dec. 01, 2023).

[5] H. Lutzenberger, *"Manual and Nonmanual Features of Name Signs in Kata Kolok and Sign Language of the Netherlands,"* Sign Language Studies, vol. 18, no. 4, pp. 546–569, 2018, Accessed: Dec. 01, 2023. [Online]. Available: https://www.jstor.org/stable/26637448

[6] Google, Google - *American Sign Language Fingerspelling Recognition*. 2023. Accessed: Dec. 01, 2023. [Online]. Available: https://www.kaggle.com/c/asl-fingerspelling

[7] N. K. Caselli, Z. S. Sehyr, A. M. Cohen-Goldberg, and K. Emmorey, *"ASL-LEX: A lexical database of American Sign Language,"* Behavior Research Methods, vol. 49, no. 2, pp. 784–801, Apr. 2017, doi: 10.3758/s13428-016-0742-0.

[8] S. J. Supalla, J. H. Cripps, and A. P. J. Byrne, *"Why American Sign Language Gloss Must Matter,"* American Annals of the Deaf, vol. 161, no. 5, pp. 540–551, 2017, Accessed: Dec. 01, 2023. [Online]. Available: https://www.jstor.org/stable/26235305

[9] S. Das, Md. S. Imtiaz, N. H. Neom, N. Siddique, and H. Wang, *"A hybrid approach for Bangla sign language recognition using deep transfer learning model with random forest classifier,"* Expert Systems with Applications, vol. 213, p. 118914, 2023, doi: https://doi.org/10.1016/j.eswa.2022.118914.

[10] M. M. Balaha et al., *"A vision-based deep learning approach for independent-users Arabic sign language interpretation,"* Multimedia Tools and Applications, vol. 82, no. 5, pp. 6807–6826, Feb. 2023, doi: 10.1007/s11042-022-13423-9.

[11] G. Z. de Castro, R. R. Guerra, and F. G. Guimarães, *"Automatic translation of sign language with multi-stream 3D CNN and generation of artificial depth maps,"* Expert Systems with Applications, vol. 215, p. 119394, 2023, doi: https://doi.org/10.1016/j.eswa.2022.119394.

[12] K. Simonyan and A. Zisserman, *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015.

[13] G. Farnebäck, *"Two-Frame Motion Estimation Based on Polynomial Expansion,"* in In: Image analysis, Jun. 2003, vol. 2749, pp. 363–370. doi: 10.1007/3-540-45103-X_50.

[14] I. Goodfellow et al., *"Generative Adversarial Networks,"* Commun. ACM, vol. 63, no. 11, pp. 139–144, Oct. 2020, doi: 10.1145/3422622.

[15] K. S. Abhishek, L. C. F. Qubeley, and D. Ho, *"Glove-based hand gesture recognition sign language translator using capacitive touch sensor,"* in 2016 IEEE International Conference on Electron Devices and Solid-State Circuits (EDSSC), 2016, pp. 334–337. doi: 10.1109/EDSSC.2016.7785276.

[16] R. Wu, S. Seo, L. Ma, J. Bae, and T. Kim, *"Full-Fiber Auxetic-Interlaced Yarn Sensor for Sign-Language Translation Glove Assisted by Artificial Neural Network,"* Nano-Micro Letters, vol. 14, no. 1, p. 139, Jul. 2022, doi: 10.1007/s40820-022-00887-5.

[17] M. A. Ahmed, B. B. Zaidan, A. A. Zaidan, M. M. Salih, and M. M. B. Lakulu, *"A Review on Systems-Based Sensory Gloves for Sign Language Recognition State of the Art between 2007 and 2017,"* Sensors (Basel), vol. 18, no. 7, p. 2208, Jul. 2018, doi: 10.3390/s18072208.

[18] J. Li et al., *"SignRing: Continuous American Sign Language Recognition Using IMU Rings and Virtual IMU Data,"* Proc. ACM Interact. Mob. Wearable Ubiquitous Technol., vol. 7, no. 3, Sep. 2023, doi: 10.1145/3610881.

[19] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, *"Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields,"* 2017.

[20] T. Zhao, *"OpenPose Advanced Doc - 3-D Reconstruction Module and Demo,"* 2022. https://github.com/CMU-Perceptual-Computing-Lab/openpose/blob/master/doc/advanced/3d_reconstruction_module.md (accessed Nov. 30, 2023).

[21] Google, *"MediaPipe Holistic,"* 2023. https://github.com/google/mediapipe/blob/master/docs/solutions/holistic.md (accessed Nov. 30, 2023).

[22] V. Bazarevsky, I. Grishchenko, K. Raveendran, T. Zhu, F. Zhang, and M. Grundmann, *"BlazePose: On-device Real-time Body Pose tracking,"* CoRR, vol. abs/2006.10204, 2020, [Online]. Available: https://arxiv.org/abs/2006.10204

[23] F. Zhang et al., *"MediaPipe Hands: On-device Real-time Hand Tracking,"* CoRR, vol. abs/2006.10214, 2020, [Online]. Available: https://arxiv.org/abs/2006.10214

[24] V. Bazarevsky, Y. Kartynnik, A. Vakunov, K. Raveendran, and M. Grundmann, *"BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs,"* CoRR, vol. abs/1907.05047, 2019, [Online]. Available: http://arxiv.org/abs/1907.05047

[25] Kaggle user: tecperson, *"Sign Language MNIST,"* 2017. https://www.kaggle.com/datasets/datamunge/sign-language-mnist (accessed Dec. 01, 2023).

[26] N. Pugeault, *"ASL Finger Spelling Dataset."* https://empslocal.ex.ac.uk/people/staff/np331/index.php?section=FingerSpellingDataset (accessed Dec. 01, 2023).

[27] Kaggle user: Muhammad Khalid, *"Sign Language for Alphabets,"* 2019. https://www.kaggle.com/datasets/muhammadkhalid/sign-language-for-alphabets (accessed Dec. 01, 2023).

[28] S. Anwar and N. Barnes, *Densely Residual Laplacian Super-Resolution*. 2019.

[29] H. Chen, J. Gu, and Z. Zhang, *"Attention in Attention Network for Image Super-Resolution,"* CoRR, vol. abs/2104.09497, 2021, [Online]. Available: https://arxiv.org/abs/2104.09497

[30] R. Zhang, P. Isola, and A. A. Efros, *"Colorful Image Colorization,"* CoRR, vol. abs/1603.08511, 2016, [Online]. Available: http://arxiv.org/abs/1603.08511

[31] R. Zhang et al., *"Real-Time User-Guided Image Colorization with Learned Deep Priors,"* CoRR, vol. abs/1705.02999, 2017, [Online]. Available: http://arxiv.org/abs/1705.02999

[32] Ashley Chow, Google - *American Sign Language Fingerspelling Recognition. Kaggle*, 2023. [Online]. Available: https://kaggle.com/competitions/asl-fingerspelling

[33] Ashley Chow, Google - *Isolated Sign Language Recognition.* Kaggle, 2023. [Online]. Available: https://kaggle.com/competitions/asl-signs

[34] V. Singh, "A Guide to Controlling LLM Model Output: Exploring Top-k, Top-p, and Temperature Parameters," 2023. https://ivibudh.medium.com/a-guide-to-controlling-llm-model-output-exploring-top-k-top-p-and-temperature-parameters-ed6a31313910

[35] D. Karunakaran, *"Fine-tuning Large Language Models series: Internal mechanism of LLMs,"* 2023. https://medium.com/intro-to-artificial-intelligence/fine-tuning-large-language-models-series-part1-internal-mechanism-of-llms-1cc062536b08

[36] J. Dancker, *"A Brief Introduction to Recurrent Neural Networks,"* 2022. https://towardsdatascience.com/a-brief-introduction-to-recurrent-neural-networks-638f64a61ff4

[37] A. Vaswani et al., *"Attention Is All You Need,"* CoRR, vol. abs/1706.03762, 2017, [Online]. Available: http://arxiv.org/abs/1706.03762

[38] W. Wong, *"What is Teacher Forcing?,"* 2019. https://towardsdatascience.com/what-is-teacher-forcing-3da6217fed1c

[39] H. Sohn, *"1st place solution - 1DCNN combined with Transformer,"* 2023. https://www.kaggle.com/competitions/asl-signs/discussion/406684

[40] Moryossef, A., Tsochantaridis, I., Aharoni, R., Ebling, S., & Narayanan, S. (2020). Real-Time Sign Language Detection using Human Pose Estimation. CoRR, abs/2008.04637. https://arxiv.org/abs/2008.04637