VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
PROGRAMŲ SISTEMŲ KATEDRA

Kursinis darbas

Gestų kalbos atpažinimas naudojant internetinę kamerą

(Sign language recognition using web camera)

Atliko: 3 kurso I grupės studentas
Pranciškus Ambrazas (parašas)

Darbo vadovas:
asist. Linas Petkevičius (parašas)

Vilnius, 2017

Turinys

Įvadas

Daugiau nei 360 milijonų žmonių pasaulyje kenčia dėl klausos ir kalbos įvairių problemų, o daugiau nei 32 milijonai jų yra vaikai ir šis skaičius vis auga [4]. Gestų kalba yra pagrindinis šių žmonių bendravimo įrankis. Tačiau reiktų pastebėti ir tai, jog dalis jų moka dalinai skaityti iš lūpų. Norint žmogui be šių ydų bendrauti su gestakalbiu (*gestų kalba kalbantis žmogus*), o kartais net dviems gestakalbiams tarpusavyje, reikia vertėjo, kuris išverstų gestų kalbą į įprastinę ir atvirkščiai.

Kiekviena pasaulyje esanti kalba turi ir savo gestų kalbą. Tai reiškia, skiriasi tiek gestų kalbos gramatika, tiek netgi patys gestai. Pasaulyje randama net dialektų pagal regionus, ne tik pagal šalis. Amerikiečių anglų gestų kalba (*toliau - ASL*) šnekančių žmonių pasaulyje priskaičiuojam nuo 500 tūkstančių iki netgi 2 milijonų vien tik Jungtinėse Amerikos Valstijose. Remiantis Census Bereau surinktais duomenimis, kuris domisi kalbų mažumomis, ASL yra pirmoji mažumos kalba, po „didžiojo ketverto“, kurį sudaro ispanų, italų, vokiečių ir prancūzų kalbos [2]. Tad netgi bendraujant dviem žmonėms, mokantiems gestų kalbą neretai iškyla vertimo problema, todėl tenka ieškoti gestų vertimų. Paiešką šiuo metu galima atlikti atsižvelgiant į delno padėtį, vienos ar abiejų rankų judesį, jų padėtį ir gestą atliekančių rankų skaičių. Tuomet pagal gesto išvaizdos nuotraukas ar kartais net vaizdo įrašus, gestakalbiai gali išsiversti gestus. Tam yra skirtos tiek internetinės svetainės - žodynai, tiek įvairūs rašytiniai žodynai.

Darbo tikslas

Šio tyrimo tikslas - ištirti ir išanalizuoti galimybes internetinės kameros pagalba versti gestų kalbą, taip padedant ne tik gestakalbiams tarpusavyje, bet ir žmonėms, nesuprantantiems gestų kalbos bendrauti su gestakalbiais tam pasitelkiant technologijas. Galiausiai, taip suteikiant šiems žmonėms pilnavertį gyvenimą bendraujant su kitais. Šiuo tyrimu siekiama apžvelgti ir įvertinti ar naudojantis įprasta internetine kamera įmanoma paversti gestų kalbą rašytiniu tekstu ar net garsine kalba ir lygiai taip pat versti rašytinę ar garsinę kalbą į gestų kalbą. Taip pat siekiama, kad vėliau būtų sukurtas visiems gestakalbiams prieinamas produktas ar programinė įranga, kurią kiekvienas įsidiegęs į savo įrenginį - kompiuterį, mobilųjį telefoną ar planšetinį kompiuterį galėtų naudotis šiuo vertėju. Vėliau tai galėtų tapti ir mokomąja gestų kalbos priemone.

Naudojamos priemonės

1. Programavimo kalba Python;
2. Vaizdų apdorojimo įrankis OpenCV;
3. Matematinių skaičiavimų biblioteka NumPy;
4. Įrankių moksliniams tyrimams biblioteka SciPy;
5. Savaime apsimokančių sistemų biblioteka scikit-learn;
6. Neuroninių tinklų aplikacijų programavimo sąsaja Keras.

Tyrimo eiga

Šiame darbe bus nagrinėjamos amerikiečių anglų ir lietuvių gestų kalbos.

1. Susipažinimas su gestų kalba ir jos problematika;
2. Susipažinimas su kelių tipų apmokymo sistemomis;
3. Praktinis pritaikymas.

1. Teorija

Šiame skyriuje bus aprašoma teorija apie apsimokančias sistemas ir neuroninius tinklus.

1.1. Apsimokančios sistemos

Apsimokančios sistemos (*angl. machine learning*) - tai mokslas apie tai, kaip kompiuterius užprogramuoti taip, jog jie patys darytų sprendimus be žmogaus įsikišimo neužprogramuojant kiekvienos galimos situacijos. Kitais žodžiais tariant, leisti kompiuteriui pačiam nuspręsti kaip elgtis esant tam tikroms aplinkybėms. Savaime apsimokančios sistemos yra didelis žingsnis į priekį norint sukurti dirbtinį intelektą.

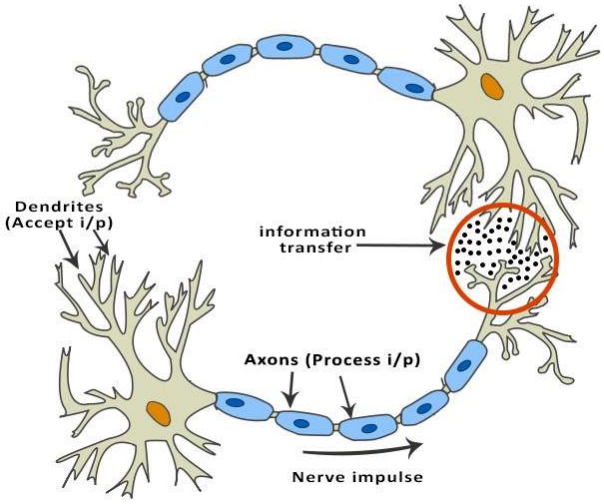
Savaime apsimokančių sistemų ir jų algoritmų sukūrimo dėka gatvėmis pradėjo važinėti patys save vairuojantys automobiliai (*angl. self-driving cars*) arba dar kitaip vadinamos autonominės transporto priemonės. Įvairūs paieškos varikliai tokie kaip „Google“ ar „Yahoo“ taikydami šiuos alogirtmus naudotojams rodo kiekvienam asmeniškai sugeneruotą turinį. Taip pat reikėtų paminėti ir kalbos atpažinimo sistemas tokias kaip „Siri“ ar „Google Assistant“, kurios iš joms duotų komandų atlieka tam tikrus veiksmus.

1.2. Neuroniniai tinklai

Neuronas (*arba - nervinė ląstelė*) - pagrindinės nervų sistemos ląstelės, sukuriančios ir/arba perduodančios elektrocheminius impulsus.

Žmogaus smegenys yra sudėtingas, nelinijinis ir paralelinis kompiuteris[1], kurias sudaro neuronai. Vienas neuronas vienu metu jungiasi su daugybe kitų neuronų per dendritus, ant kurių yra daug sinapsių, per kurias ateina informacija iš kitų neuronų. Dendritus paprasčiau galima pavadinti kaip informacijos priimėjais. Todėl vienas neuronas gali sudaryti iki 100000 sinapsių. Kiekviena sinapsė gali būti jaudinanti arba slopinanti. Visas šis mechanizmas dar nadinamas **neuroniniu tinklu** (*angl. Neural network*).

Vienas neuronas priima informaciją per dendritus, tuomet pats neuronas nusprendžia ar bus siunčiama informacija į kitus neuronus ir kokia ji bus siunčiama (*žr. 1 pav.*)



1 pav. Neuroninio tinklo struktūra[1]

Remiantis šiais principais buvo sukurti dirbtiniai neuroniniai tinklai.

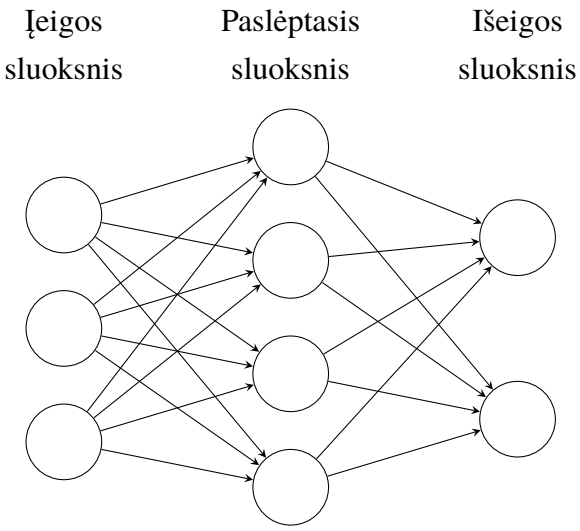
1.2.1. Dirbtiniai neuroniniai tinklai

Dirbtinis neuroninis tinklas (*angl. Artificial neural network*) - struktūra, skirta apdoroti dideliam kiekiui informacijos, sukurta rementis žmogaus nervų sistemos veikimo principu. Kitais žodžiais tariant, skaitmenizuota žmogaus smegenų veikla.

Neuronai dirbtiniame neuroniniame tinkle yra sujungti jungtimis. Taip jie tarpusavyje komunikuoja perduodami vienas kitam informaciją. Kiekvienas neuronas gali priimti atėjusią informaciją, ją apdoroti ir perduoti kitam neuronui. Kiekviena jungtis turi savo svorį, pagal kurį pasirenkama, į kurį neuroną turi būti perduodama informacija toliau. Šių jungčių svorius galime įvertinti naudodamiesi apsimokančių sistemų pagalba.

2 pav. pateikiama supaprastinta schema, kaip veikia dirbtiniai neuroniniai tinklai:

[1] `https://www.tutorialspoint.com/artificial_intelligence/artificial_intelligence_neural_networks.htm/`

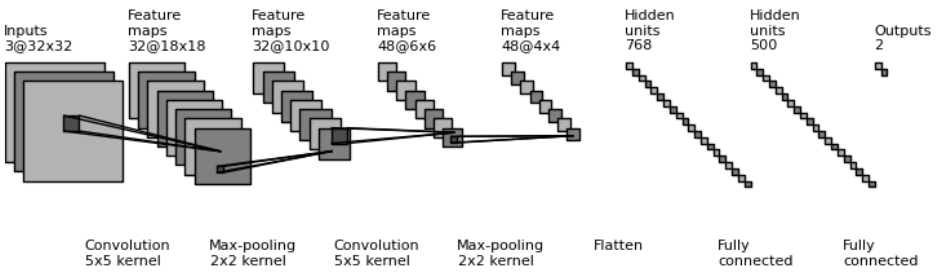


2 pav. Dirbtinio neuroninio tinklo pavyzdys

1.2.2. Konvoliuciniai neuroniniai tinklai

Konvoliuciniai neuroniniai tinklai (*angl. Convolutional neural network*) (*toliau - KNT*) - specilios rūšies „Feed-forward“ neuroniniai tinklai, kurie reamiasi tuo pačiu dirbtinių neuroninių tinklų (*žr. 2 pav.*) principu. Šie tiklai yra skirti atpažinti objektus. Apmokant KNT, pagrindinis principas, kaip ir raktinis šių tinklų žodis, yra konvoliucija. Tai tarsi filtrai, kurie išskaido kadrą į dalis, pritaiko įvairius filtrus ir gliausiai randa pasikartojančius įvairiuose kadruose esančius požymius. Tokius kaip, ratas, langai ar pan., o šio tyrimo atveju - rankos, delnai, pirštai, jų padėtis ir kt.

Šiame tyrime bus aptariamas ReLU principas, nes konvoliuciniai neuroniniai tinklai, kurie remiasi ReLU principu yra kelis kartus greitesni, nei tie, kurie remiasi kitais principais, pavyzdžiui, tanh [3]. Kuomet kadrui yra pritaikoma konvoliucija (sudauginami visi kadro taškai su konvoliucine matrica), pritaikomas ReLU principas, kuris visus neigiamus skaičius paverčia į 0. Telkimo (*angl. pooling*) fazėje, kadras yra sumažinamas (*žr. 3 pav.*). Plačiau šie etapai bus aptarti 3.3 skyriuje.

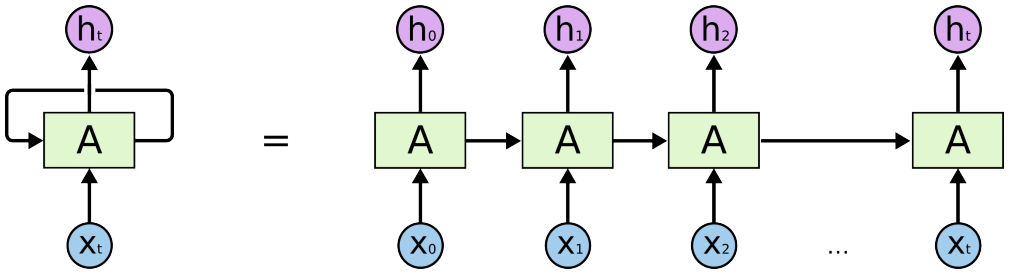


3 pav. Konvoliucinių neuroninių tinklų schema[2]

[2] `http://yann.lecun.com/exdb/lenet/`

1.2.3. Rekurentiniai neuroniniai tinklai

Rekurentiniai neuroniniai tinklai (*angl. Recurrent neural network*) - tai dirbtinis neuroninis tinklas, kuris saugo informaciją apie praeituose žingsniuose (neuronuose) atliktus veiksmus ar skaičiavimus. Remiantis rekurentinių tinklų savybe galima naudoti jau surinktą informaciją, o šiuo atveju tai labai pasitarnautų, kuomet gestas nėra statinis. Todėl kiekviename kadre jeigu fiksuojama kažkokio statinio gesto reikšmė, tai sujungus jas į rekurentinį neuroninį tinklą galima atpažinti dinaminius gestus.



4 pav. Rekurentinių neuroninių tinklų schema[3]

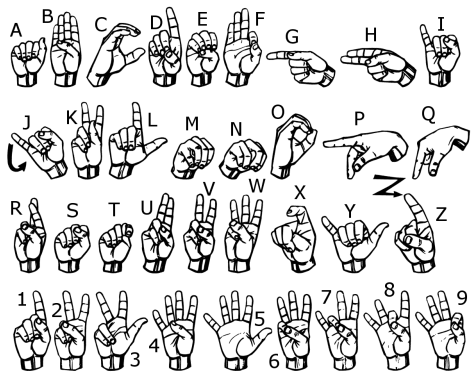
- x_t - įeiga momentu t;
- h_t - išeiga momentu t;
- A_t - būsena momentu t.

[3] http://colah.github.io/posts/2015-08-Understanding-LSTMs/

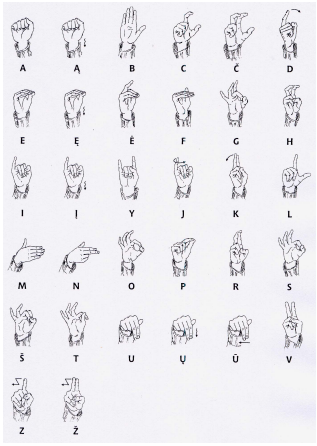
2. Gestų kalba

2.1. Gestų kalbos skirstymas

Gestų kalba susideda iš dviejų dalių - statinių ir dinaminių ženklų. Gestų kalboje kiekviena kalba turi savo abėcėlę. Statiniais ženklais atvaizduojama didžioji abėcėlių raidžių dalis. O dinaminiais - žodžiai ir kai kurios gestų abėcėlių raidės. *Pavyzdžiui*, amerikiečių gestų kalbos abėcėlėje J ir Z raidės atvaizduojamos dinaminiais judesiais (*žr. 5 pav.*), o lietuvių - jau minėtosios J ir Z raidės bei Ą, D, Į, K ir kt. (*žr. 6 pav.*)



5 pav. Amerikiečių gestų kalbos abėcėlė[4]



6 pav. Lietuvių gestų kalbos abėcėlė[5]

2.2. Problematika

Norint atpažinti gestus, paversti juos į raides, žodžius ar sakinius susiduriama su problemomis, kurios susijusios tiek su statinių, tiek su dinaminių gestų atpažinimu.

2.2.1. Statinių ženklų problematika

Pagrindinės problemos iškylančios atpažįstant statinius gestų kalbos ženklus yra:

1. Kiekvienos kalbos abėcėlę sudaro skirtingas raidžių (statinių ženklų) skaičius. *Pavyzdžiui*, lietuvių kalbos abėcėlę sudaro 32 ženklai, o amerikietišką - 26;
2. Gestų panašumai. *Pavyzdžiui*, raidės A, E, N, S, T yra atvaizduojamos sugniaužtus kumštį, o net trijose iš jų (A, E ir S) skiriasi tik nykščio padėtis;
3. Kampas, kuris susidaro atpažįstant gestą. *Pavyzdžiui*, kai A raidė rodoma ne iš priekio, o iš šono;
4. Apšvietimas. *Pavyzdžiui*, gestų atpažinimas esant prieblandai ir dienos šviesai.

[5] http://lifeprint.com/asl101/topics/wallpaper1.htm
[5] http://gestai.ndt.lt/pirstu-abecele

2.2.2. Dinaminių ženklų problematika

Pagrindinės problemos iškylančios atpažįstant dinaminius gestų kalbos ženklys yra:

1. Nauji gestų kalbos žodžiai. *Pavyzdžiui*, kiekvienas uraganas turi savo pavadinimą, todėl tai gali reikšti naujo gesto atsiradimą;
2. Gesto kelios reikšmės. *Pavyzdžiui*, vienas gestas gali turėti kelias reikšmes, kaip kad lietuvių kalboje vienas žodis „kasa“ gali turėti net tris skirtingas reikšmes;
3. Kampas, kuris susidaro atpažįstant gestą. *Pavyzdžiui*, kai rodantysis žmogus stovi šiek tiek šonu;
4. Žodžių apjungimas į vieną sakinį. *Pavyzdžiui*, keli gestai einantys vienas po kito gali reikšti vieną žodį, tačiau tuo pačiu būti panašūs į vieną gestą, kuris jau reikš tik vieną žodį.

3. Sistemos apmokymas

Norint, jog sistema atpažintų gestus, svarbiausia ją apmokyti, ką reiškia tam tikri gestai. Tam galime išnaudoti kadrus (*angl. frame*) ir savaime apsimokančių sistemų galimybes. Tad reiktų imti vieną kadrą ir jį paversti į kompiuteriui suprantamą kalbą.

3.1. Kadro transofrmacija

Kiekvienas kadras yra sudarytas iš *h* * *w* * *c* taškų (*angl. pixels*). Čia, *h* - aukštis (*angl. height*), *w* - plotis (*angl. width*), *c* - spalva (*angl. color*), susidedanti iš RGB spalvų paletės. RGB spalvų paletė - tai trijų - raudonos, žalios ir mėlynos - spalvų paletė, kuri yra ypatinga tuo, jog kiekviena spalva, kurią mato žmogus, susideda būtent iš šių trijų spalvų, varijuojant jų ryškumu. 0 - spalva nenaudojama, o 255 - spalva naudojama pilnai ryškiai. Tad kiekvieną kadrą galima išskirti į šių trijų spalvų sluoksnius.

3.2. Sistemos apmokymas, naudojantis įprastinėmis sistemomis

Kompiuterio vaizdų atpažinimo sistemos (*angl. computer vision*) - sistemos, kurias galima apmokyti atpažinti įvairius objektus, dar žinomos kaip *CV*.

Šiame poskyryje bus nagrinėjamos OpenCV atviro kodo (*angl. open-source*) vaizdų atpažinimo aplikacijų programavimo sąsajoje esančios galimybės ir algoritmai.

3.2.1. Kadro apdorojimas naudojant Sobel branduolį

Sobel branduolys (*angl. Sobel operator*) - vaizdų apdorojimo algoritmas, skirtas skirtas paversti kadrą į kontūrų žemėlapį.

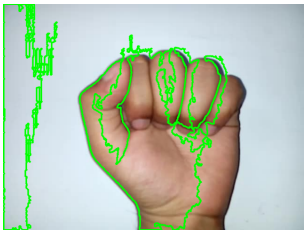
Šis branduolys naudojasi šiomis funkcijomis, kad konvertuotų vaizdą į kontūrus:

$$G_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A \tag{1}$$

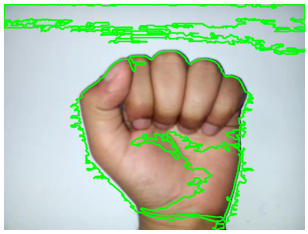
$$G_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A \tag{2}$$

$$G = \sqrt{G_x^2 + G_y^2} \tag{3}$$

7 pav. Orginalus paveikslėlis



8 pav. Pritaikyta G_x



9 pav. Pritaikyta G_y

10 pav. Pritaikyta G

11 pav. Be fono

12 pav. Dviejų spalvų

Prieš kadrui taikant Sobel funkcijas, reikia jį paruošti. Kadangi Sobel funkcijos ima matricas sudarytas iš 3×3 skaičių, todėl reikia sušvelninti šalia esančius skaičius, tam jog jos rastų tikruosius kontūrus. Tą padaryti lengviausia uždedant kadrui, pavyzdžiui, 5% miglą (*angl. blur*). Uždėti miglą reikia tam, jog sumažintume triukšmą (*angl. noise*) kontūrams.

Toliau, atiduodant kadrą Sobel funkcijoms, kadras yra konvertuojamas į skaičių masyvą, kur kiekvienas taškas turi savo skaičių – spalvos kodą. Tuomet norint rasti visus kontūrus esančius kadre, taikome šiuos veiksmus:

1. Einame per visus taškus esančius kadre ir taikome 1 formulę, kur A - kiekvienas 3×3 kadro taško spalvos mastyvas. Gauname 8 paveiksle pavaizduotą vaizdą;
2. Einame per visus taškus esančius kadre ir taikome 2 formulę, kur A - kiekvienas 3×3 kadro taško spalvos mastyvas. Gauname 9 paveiksle pavaizduotą vaizdą;
3. Apskaičiuojame dabartinio taško tikrąją reiškę taikydami 3 formulę. Gauname 10 paveiksle pavaizduotą vaizdą.

Po šių žinsnių yra gauta matrica taškų, kuriose keičiasi spalva. Dabar reikia išimti tuos taškus, kurie yra, pavyzdžiui, ta pati balta spalva, tik kitokio atspalvio. Tam įgyvendinti lengviausia pasitelkus matricos vidurkį ir atmetus visus taškus, kurie yra mažesni už vidurkį. Kitaip tariant, išimant mažo skirtumo taškus. Po šių veiksmų lieka tik tie taškai, kurie jau turėtų priklausyti gesto kontūrui.

Ieškant kontūro taip pat reikėtų atmesti ir tuos kontūrus, kurie, pavyzdžiui, neužima daugiau nei 5% viso ploto ir tuomet nubrėžti kontūrą.

Kitas žingsnis – paversti kadrą į kompiuteriui suprantamą ir kuo paprastesnę kalbą. Turėdami kontūrus, galime kadrą paversti į dvispalvį kadrą, kur viskas, kas yra kontūre bus baltos spalvos, o viskas kas už kontūro ribų – juodos. Tam įgyvendinti reiktų pasidaryti kaukę (*angl. mask*), kurioje viskas, kas už kontūro ribų bus juodos spalvos, tai kas mūsų atveju yra fonas (*žr. 11 pav.*), o paskui ištrinti tai, kas yra kontūre ir padaryti baltos spalvos, tai kas mūsų atveju bus ženklas (*žr. 12 pav.*).

Paskutinis žingsnis – prieš apmokant modelį reikia kadrą paversti duomenimis, iš kurių modelis galėtų mokytis. Šioje vietoje buvo pasirinktas plačiausiai naudojamas metodas - kadrą paversti į skaičių matricą, kur balta spalva atitinka 255, o juoda - 0. Ši konvertacija pasirinkta pagal spalvų kodus. Jau pavertus kadrą į skaičių matricą, suplokštiname ją, kad gautume vienmatę matricą. Ir galiausiai į vieną bendrą duomenų rinkmeną (*angl. file*) surašome duomenis tokia seka: pirmas eilutės langelis - gestą atitinkanti abėcėlės raidė, o visi likę šios eilutės langeliai užpildomi vienmatės kadro matricos duomenis skaičių pavidalu, apie kuriuos buvo kalbėta šios pastraipos pradžioje.

3.2.2. Apsimokančių sistemų apmokymas iš Sobel branduoliu apdoroto kadro

Turėdami rinkmeną, kurioje yra surašyti visi apmokymams skirtų kadrų duomenys, aptarsime, kaip iš šių duomenų apmokyti sistemą.

Šiame poskyryje aptarsime keletą objektams atpažinti populiarių technikų, kuriomis remiantis galima apmokyti sistemą atpažinti gestus.

Aptarsime visas technikas pasinaudodami ASL abėcėlės aibe, kurioje yra 24 statiniai gestai.

$$z = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1^2 + \theta_4 x_1 x_2 + ... \tag{4}$$

4 formulė apibrėžia, kaip bus apskaičiuojamas parametras *z*, kai norima susidaryti pasirinktos formos plotą, pagal kurį gestas priklausys arba nepriklausys išmoktai abėcėlės raidei. x_1 ir x_2 yra koordinačių ašys, kuriose yra išsidėstę taškai.

3.2.2.1. Logistinės regresijos algoritmas

Logistinės regresijos algoritmas dar žinoma kaip „logistic regression“ algoritmas. Tai – vienas populiariausių šiuo metu naudojamų klasifikavimo algoritmų.

$$0 \leq h_\theta^i(x) \leq 1; i = 1, 2, ..., 24 \tag{5}$$

Remiantis 5 formule, apsibrėžiama, jog tikimybės kiekvienai raidei bus ne mažesnės nei 0 ir ne didesnės nei 1, o *i* pasako kelinta raidė tyrinėjama, kur, tarkime, A = 1, B = 2, ir t.t.

$$h_\theta^i(x) = P(y = i|x; \theta) \tag{6}$$

Remiantis 6 formule, apsibrėžiama tikimybės formulė, kuriai kaip parametras paduodamas gesto numeris (tarkime, kad raidė A = 1, B = 2 ir t.t.), bei bus naudojamasi parametrais θ, kur θ - svorių vertės, kuriomis remiantis apskaičiuojama tikimybė *P*. θ reikšmes savaime apsimokančios sistemos apsiskaičiuoja skirtingais algoritmais, todėl šiame darbe jie nagrinėjami nebus.

$$h_\theta(x) = g(z) \tag{7}$$

Remiantis 7 formule, galima apsibrėžti ne tik tiesės atskyrimą (pagal įprastinę šios formulės

reikšmę), bet ir įvairias formas, kaip šiuo nagrinėjamu atveju, tarkime, delno formą.

$$h_\theta(x) = \frac{1}{1 + e^{-z}} \tag{8}$$

Galiausiai, 8 formulėje galime matyti, jog tikimybė yra apskaičiuojama remiantis 7 formulėje apsibrėžtos formos pavidalu.

3.2.2.2. Linijinis palaikančiųjų vektorių algoritmas

Linijinis palaikančiųjų vektorių algoritmas dar žinomas kaip „Linear Support Vector Machine“ algoritmas.

Iš esamų duomenų aibės taškų, šiuo atveju – skirtingų gestų raidžių taškų, yra sudaroma matrica, kurioje tarp skirtingų gestų yra nubrėžiamas vektorius, kuris nusako, kurioje vietoje bus traktuojamas vienas gestas, kurioje - kitas. Iki vektoriaus yra parenkamas didžiausias galimas atstumas nuo artimiausių prie vektoriaus esančių duomenų taškų.

Pats principas θ parinkimui yra labai panašus kaip loginėje regresijoje, bei z yra paskaičiuojama pagal tą pačią formulę.

Šis metodas yra geresnis, kai yra ganėtinai didelis skirtumas tarp duomenų aibės taškų ir lengvai nubrėžiami vektoriai, nes nesikerta duomenų aibės. Dar vienas šio metodo privalumas yra tas, jog nubrėžus vektorių esant didžiausiam atstumui tarp duomenų taškų, esančių arčiausiai vektoriaus, ganėtinai nesunkiai yra nustatoma, kuriai pusei (šiuo atveju gestui), priklauso duotasis taškas.

Tačiau iš kitos pusės, šis metodas nėra ypač lengvai apdorojantis duomenis, jei duomenų aibės yra persidengiančios, todėl gestų kalbos atpažinimo atveju šis metodas nėra labai tikslus, nes gestų duomenų aibės yra persidengiančios didžiaja dalimi ir skiriasi tik pirštų padėtys.

3.2.2.3. K artimiausių kaimynų algoritmas

K artimiausių kaimynų algoritmas dar žinomas kaip „k-nearest neighbors“ algoritmas.

Viskas yra suskirstoma į aibes ir pagal tašką ieškomas artimiausias kaimynas arba kaimynų aibė ir, priklausomai nuo to, kurių kaimynų daugiausia, parenkama reikšmė, kokia bus naujojo taško reikšmė.

Šis algoritmas kaip ir prieš tai aptartas linijinis palaikančiųjų vektorių algoritmas nėra idealus pasirinkimas gestų kalbos atpažinimui, nes, tarkime, jei būtų imamas delno vidurio taškas galimi visi 24 variantai ir tik kampuose, žiūrint į atlenktų pirštų skaičių būtų galima nuspręsti, kuris gestas rodomas.

3.3. Sistemos apmokymas, naudojantis konvoliucinio neuroninio tinklo modeliu

Prieš apmokant sistemą, naudojantis konvoliucinio neuroninio tinklo modeliu, pirma aptarsime, kas yra konvoliucinis bei telkimo sluoksniai, o vėliau - kaip apmokyti sistemą.

3.3.1. Konvoliucinis sluoksnis

Konvoliucinis sluoksnis turi tokius parametrus:

- Priima matricą, sudarytą iš $W_1 \times H_1 \times D_1$, kur W_1 - plotis, H_1 - aukštis ir D_1 - gylis;
- Reikalingi papildomi parametrai:
  - Filtrų skaičius K (dažniausiai naudojamas 2-to laispnis);
  - Filtro dydis F (dažniausiai naudojami 3×3 filtrai);
  - Žingsnis, per kiek paslenkamas filtras S;
  - Papildomas rėmelis duotajai matricai, sudarytas iš 0, P;
- Sudaroma matrica $W_2 \times H_2 \times D_2$, kur W_2 - plotis, H_2 - aukštis ir D_2 - gylis, kur:
  - $W_2 = (W_1 - F + 2P)/S + 1$;
  - $H_2 = (H_1 - F + 2P)/S + 1$;
  - $D_2 = K$.

Sluoksnis, kuris pagal filtrą nusprendžia, kokia išeities matricą bus gaunama. Tarkime, jog bus imamas filtras, kuris yra *3 × 3* matrica. Kadrą, kaip jau buvo minėta teorijos skyriuje, galima išskirstyti į tris skirtingus sluoksnius - raudonos, žalio ir mėlynos spalvos. Pavertus šiuos sluoksnius skaičių matricomis, kur 0 – spalva nenaudojama, o 255 – spalva naudojama pilnai ryškiai, gaunamos matricas, kurios nusako, kaip kiekviename taške ryškiai yra naudojama tam tikra spalva.

Einant taškas po taško per kadro kiekvienos spalvos matricą, imamos *3 × 3* matricos, kurios yra sudauginamos su filtro matricomis. Taip gaunama matrica, kuria remantis bus galima išskirti savybes.

Pavyzdys:

$$\begin{bmatrix} 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \\ 1 & 0 & 1 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 0 & 1 & 0 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 3 & 1 & 3 & 1 & 3 & 0 \\ 3 & 2 & 5 & 2 & 4 & 2 & 2 \\ 2 & 4 & 3 & 3 & 3 & 2 & 2 \\ 2 & 4 & 3 & 4 & 2 & 5 & 0 \\ 3 & 2 & 5 & 2 & 4 & 2 & 2 \\ 2 & 4 & 3 & 3 & 3 & 2 & 2 \\ 3 & 2 & 2 & 2 & 1 & 3 & 0 \end{bmatrix} \tag{9}$$

$$0 * 1 + 0 * 0 + 0 * 1 + 0 * 0 + 1 * 1 + 1 * 0 + 0 * 1 + 1 * 0 + 0 * 1 = 1 \tag{10}$$

$$0 * 1 + 0 * 0 + 0 * 1 + 1 * 0 + 1 * 1 + 0 * 0 + 1 * 1 + 0 * 0 + 1 * 1 = 3 \tag{11}$$

9 formulėje yra pateikiama *7 × 7* matrica sudauginama su *3 × 3* filtru. Panagrinėsime pirmos eilutės pirmąjį (*žr. 10 formulę*) ir antrąjį (*žr. 11 formulę*) narius. Pirmasis narys yra kampinis skaičius 1. Todėl jis dauginaas su filtro viduriniuoju nariu. Kadangi dabar filtras „išlenda“ iš matricos rėmų, tai įsivaizduojama, kad yra kadro matricai rėmelis, kuris yra sudarytas iš 0. 10 ir 11

formulėse pirmasis sandaugos narys - *7* $\times$ *7* kadro matricos narys, antrasis - *3* $\times$ *3* filtro matricos narys.

$$M = \frac{F-1}{2} \tag{12}$$

12 formulėje yra pateikta, kokio dydžio papildomą išorinį sluoksnį sudarytą iš 0 konvoliuciniai neuroniniai tinklai dažniausiai naudoja. Čia *M* yra sluoksnio plotis pridedamas prie kiekvienos matricos kraštinės, o *F* – filtro matricos dydis. Tai naudojama tam, kad būtų gaunama tokio pačio dydžio matrica, kokia buvo paduota filtrui apdoroti.

3.3.2. Telkimo sluoksnis

Telkimo sluoksnis turi tokius parametrus:

- Priima matricą, sudarytą iš $W_1 \times H_1 \times D_1$, kur W_1 - plotis, H_1 - aukštis ir D_1 - gylis;
- Reikalingi papildomi parametrai:
  - Filtro dydis F (dažniausiai naudojami 2 $\times$ 2 filtrai);
  - Žingsnis, per kiek paslenkamas filtras S (dažniausiai naudojams žingsnis - 2);
- Sudaroma matrica $W_2 \times H_2 \times D_2$, kur W_2 - plotis, H_2 - aukštis ir D_2 - gylis, kur:
  - $W_2 = (W_1 - F)/S + 1$;
  - $H_2 = (H_1 - F)/S + 1$;
  - $D_2 = D_1$.

Kaip buvo minėta anksčiau, telkimo sluoksnis yra skirtas sumažinti matricą arba kitais žodžiais tariant - kadrą. Telkimo sluoksnis dažniausiai taiko arba vidutinės arba didžiausios reikšmės atrinkimo algoritmą. Kaip pavyzdį paimkime, jog telkimo sluoksnio filtro dydis bus *3* $\times$ *3*. Bus taikomas maksimalaus telkimo principas, kuris nusako, jog paėmus matricos poaibį yra išrenkama didžiausia reikšmė.

Pavyzdys:

$$\begin{bmatrix} 1 & 3 & 1 & 3 & 1 & 3 \\ 3 & 2 & 5 & 2 & 4 & 2 \\ 2 & 4 & 3 & 3 & 3 & 2 \\ 2 & 4 & 3 & 4 & 2 & 5 \\ 3 & 2 & 5 & 2 & 4 & 2 \\ 2 & 4 & 3 & 3 & 3 & 2 \end{bmatrix} = \begin{bmatrix} 5 & 4 \\ 5 & 5 \end{bmatrix} \tag{13}$$

$$max(\begin{bmatrix} 1 & 3 & 1 \\ 3 & 2 & 5 \\ 2 & 4 & 3 \end{bmatrix}) = 5 \tag{14}$$

$$max(\begin{bmatrix} 3 & 1 & 3 \\ 2 & 4 & 2 \\ 3 & 3 & 2 \end{bmatrix}) = 4 \tag{15}$$

13 formulėje parodyta, kaip pritaikius 3×3 telkimo filtrą gaunama beveik 90% mažesnė matrica. Mažinimas vyksta taip: imama filtro dydžio matrica, pradėdedant nuo kairiojo viršutinio kampo. Tuomet iš tos matricos išrenkama didžiausia reikšmė (*žr. 14 formulę*) ir į naująją matricą ji įrašoma. Tuomet, kadangi jau iš pirmosios 3×3 matricos yra išrinkta didžiausia reikšmė (jei būtų taikomas vidutinės reikšmės atrinkimo algoritmas, būtų imama šios matricos vidutinė reikšmė), imama kitą į dešinę esanti matrica, kuri nepersidengia su pirmąja ir iš jos taip pat išrenkama didžiausia reikšmė (*žr. 15 formulę*). Taip filtras yra pritaikomas ir likusioms matricos dalims. Jei filtras „užeina“ už kadro matricos ribų, interpretuojama, jog tose vietose yra 0.

3.3.3. Apmokymas naudojantis konvoliucinio neuroninio tinklo modeliu

Kiekvieną sistemą galima apmokyti naudojantis kovoliuciniu neuroniniu tinklo modeliu arba nuodugniai (kuomet apmokomas modelis neturi jokių duomenų) arba dalinai (kuomet pasinaudojama jau veikiančiu modeliu ir nutrinami paskutinį ar paskutinius sluoksnius). Apmokymas nuo pat pradžių dažniausiai reikalauja labai daug laiko resursų, nes užtrunka labai daug laiko. Šis modelis retai naudojamas praktikoje. Dažniausiai naudojamas dalinis apmokymas, pasinaudojus jau sukurtu modeliu galima jį pakartotinai apmokyti norimam rezultatui gauti.

Keras - giliojo mokymosi biblioteka, naudojama su Theano arba TensorFlow.

Theano - Python biblioteka, skirta darbui su matematinėmis išraiškomis, jų optimizavimui, apibrėžimui, darbui su n-mačiais masyvais.

Naudojantis Keru, sistema buvo apmokyta naudojantis konvoliucinio neuroninio tinklo modeliu. Tai buvo atlikta trimis etapais.

Pasinaudota dvejomis duomenų aibėmis - pirmoji skirta mokymuisi (*data/train*) ir antroji - pasitvirtinimui - validacijai (*data/validation*).

Buvo naudojamasi VGG16 (*Visual Geometry Group*) architektūra (*žr. 15 pav.*) - konvoliucinių ir telkimo sluoksnių pasikartojančios grandinės. Telkimui naudojama 2×2 matrica, o konvuliacijai - 3×3 matrica.

3.3.3.1. Pirmasis etapas

Pirmame etape buvo pritaikytas duomenų dauginimo (*angl. data augmentation*) principas iš duotųjų mokymuisi ir validacijai. Kiekvienam kadrui buvo pritaikytos įvairios modifikacijos apmokant - kadro priartinimas, pakreipimas, horizontalus apvertimas.

3.3.3.2. Antrasis etapas

Antrajame etape, VGG16

3.3.3.3. Trečiasis etapas

4. Praktinis gestų kalbos apmokyto modelio pritaikymas

Praeituose skyriuose buvo aptarta, kokie būdai yra galimi apmokyti sistemą atpažinti gestų kalbą. Šiame skyriuje bus aprašyta, kaip šie aprašyti žingsniai buvo pritaikyti praktiškai.

4.1. Duomenys

Duomenys (programinis kodas ir kadrai) buvo parsisiųsti iš jau įgyvendintų pavyzdinių gestų kalbos sistemų. Viename iš duomenų rinkinių buvo po 150-250 gesto kadrų kiekvienai raidei (pavyzdys pateiktas 7 paveikslėlyje). Taip pat pasinaudojus šių kadrų paprastumu dėl šviesaus ir vienspalvio fono, buvo uždėti skirtingi fonai.

13 pav. A raidės gestas su pakeistu fonu

14 pav. A raidės gestas su pakeistu fonu

Kiekvienam iš 150-250 kadrų kiekvienai raidei buvo uždėta po du atsitiktinai sugeneruotus fonus. Gesto dydis, spalva, pokrypis keičiamas nebuvo. Todėl buvo sudaryta po 400 gesto kadrų kiekvienai raidei su 21 skirtingu fonu.

4.2. Duomenų suskirstymas

Duomenys buvo suskirstyti į aplankus, kur aplanko pavadinimas rodo raidę, kurią atitinka gestas. Taip buvo sukurti 24 aplankai kiekvienai iš 24 ASL raidžių. Visi duomenys buvo suskirstyti į tris aplankus, kurie buvo skirti apmokymui, patikrinimui ir testavimui. Duomenų imta labai nedaug - apmokymui skirtų gestų kadrų buvo po 500 kiekvienai raidei, patikrinimui - 200 kiekvienai raidei ir dar 100 testavimui skirtų kadrų. Bendrai:

$$500 * 24 + 200 * 24 + 200 = 17000 \tag{16}$$

4.3. Sistemos apmokymas

Buvo pasirinkta apmokyti sistemą konvoliucinio neuroninio tinklo modeliu. Buvo pasirinkta sistemą apmokyti naudojantis „Keras“ giliojo mokymosi biblioteka, kuri naudojosi „Te“ biblioteka (*plačiau - 3.3.3. skyriuje*).

Išvados

Išvadose ir pasiūlymuose, nekartojant atskirų dalių apibendrinimų, suformuluojamos svarbiausios darbo išvados, rekomendacijos bei pasiūlymai.

Literatūra

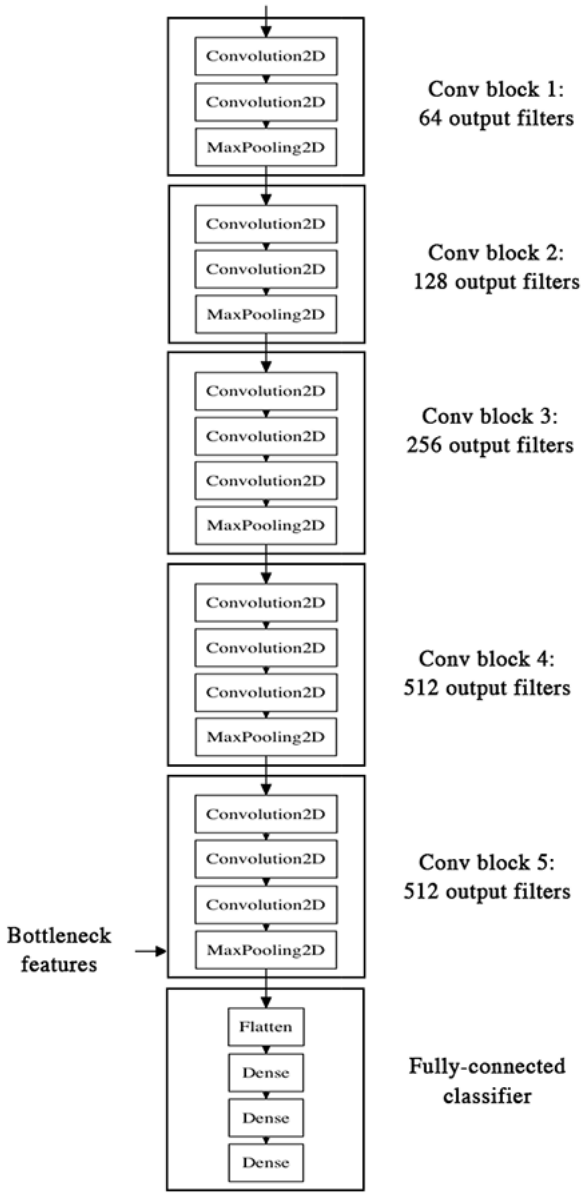
[1] Simon Haykin. *Neural Networks and Learning Machines*. 1 psl. Upper Saddle River, New Jersey 07458: Pearson Education inc., 2009.

[2] Tom Harrington. *ASL: Ranking and number of users*. http://libguides.gallaudet.edu/content.php?pid=114804&sid=991835/. 45 KB, atnaujinta 2016-07. 2004.

[3] Alex Krizhevsky, Ilya Sutskever ir Geoffrey E Hinton. „ImageNet Classification with Deep Convolutional Neural Networks“. *Advances in Neural Information Processing Systems 25*. Red. F. Pereira ir k.t. Curran Associates, Inc., 2012, p.p. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[4] World Health Organization. *Deafness and hearing loss*. http://www.who.int/mediacentre/factsheets/fs300/en/. 2017-02.

Priedas Nr. 1
VGG16 architektūra



15 pav. VGG16 architektūros modelis