VILNIAUS UNIVERSITETAS
MATEMATIKOS IR INFORMATIKOS FAKULTETAS
INFORMATIKOS KATEDRA

Baigiamasis bakalauro darbas

Kalbėtojo atpažinimas naudojantis dirbtiniais neuroniniais tinklais

(Speaker Recognition Using Artificial Neural Networks)

Atliko: 4 kurso, 1 grupės studentas
Nikodemas Žaliauskas
Darbo vadovas:
Andrius Grevys
Recenzentas
Arūnas Janeliūnas

Vilnius
2017

Turinys

Įvadas

Dirbtiniai padarai jau buvo aprašyti graikų mitologijoje. Vienas iš jų yra auksinis Hefaisto robotas. Taip pat viduramžiuose sklido gandai apie paslaptingas mistines ar alchemines būtybes, kurioms buvo įdiegtas protas ir mintys, pavyzdžiui „homunculus“ arba golemas. Taigi, jau nuo seno žmonėms buvo kilus mintis apie esybes, kurios turi žmogaus sukurtą protą.

Žmogaus smegenyse yra milijardai neuronų, kurie tarpusavyje jungiasi ir iš jų susidaro milžiniški neuroniniai tinklai, apdoroja seną bei naują informaciją. Šiais laikais ypač yra populiarus dirbtinio intelekto mokslas, kai žmonės bando sukurti programas, kurios veikia tarsi kaip žmogaus smegenys ir gali atlikti žmonėms pajėgias užduotis, tokias kaip vaizdų atpažinimas, garso atpažinimas, ateities įvykių prognozavimas, ir netgi tobulėti atliekant vis daugiau užduočių.

Šiandien, dirbtinis intelektas yra nuėjęs ilgą kelią ir jau gali atpažinti įvairius objektus nuotraukose, atpažinti žmones pagal jų biometrinius parametrus, prognozuoti rinkos ateities įvykius ar net vairuoti automobilį už žmogų. Tačiau, dirbtinis intelektas nėra tobulas. Programos, kuriamos prognozuoti rinkos ateities įvykius suklysta ir ganėtinai dažnai, nes rinka yra neprognozuojama. Bepilotės mašinos padaro avarijų, nes nenumato, kad šalia važiuojanti mašina stabdys ar rikiuosis į kitą juostą. Telefonuose, pavyzdžiui iPhone, kalbant su kalbos vertimo ir atpažinimo programa „Siri“, ji ne visada teisingai supranta pasakytus žodžius, nes skirtingi žmonės turi skirtingus akcentus, kalba skirtingu garsumu arba aplinkui yra pašalinio triukšmo. Dėl tokių niuansų mokslininkai nuolatos tobulina ir galvoja, kaip pagerinti neuroninių tinklų veikimą, kad būtų pasiekiami geresni rezultatai. Viena iš problemų yra kalbėtojo atpažinimas naudojantis dirbtiniais neuroniniais tinklais.

Kalbėtojo atpažinimas iš jo balso yra ganėtinai sudėtinga problema. Viename įraše vienu metu gali kalbėti keli asmenys arba kalbant vienam žmogui šalia, gali girdėtis važiuojančių automobilių garsai ar dar kokie nors pašaliniai garsai. Žmogui atpažinti žmogų iš balso yra paprasta. Mes net nesusimąstome, kaip tai padarome. Mums atpažinti kito asmens balsą ar kitokius garsus yra įprasta, nes žmogus nuo pat mažumės augo ir tuo pačiu mokėsi ne tik kalbėti, rašyti ar suprasti, kad palietus ugnį galima nudegti, bet ir tai, kad visi objektai gali skleisti įvairiausius garsus, kurie yra skirtingi. Tačiau, kaip priversti mašiną išmokti atpažinti balsus?

Šio darbo tikslas yra atlikti praktinį tyrimą su giliais neuroniniais tinklais.

Darbo metu bus aptarta teorinė dalis apie neuroninius tinklus, kaip žmogaus balsą paversti į kompiuteriui suprantamą kalbą, kaip mašinos gali būti išmokytos atpažinti balsus. Taip pat bus sukurtas gilus neuroninis tinklas, kuris bus apmokomas naudojant skirtingus parametrus. Apmokytas dirbtinis neuroninis tinklas turės gebėti atpažinti skirtingus kalbėtojus. Vėliau apmokyti neuroniniai tinklai bus tikrinami ar jie gerai ar blogai atpažįsta kalbėtojus. Galiausiai iš visų gautų rezultatų bus atliekama analizė. Bus palyginti gauti rezultatai tarpusavyje, bei su kursinio darbo gautais rezultatais.

1. Dirbtinis intelektas

Dirbtinis intelektas (angl. Artificial Intelligence) yra mokslo šaka, kurioje kuriamios protingos programos, analizuojami taip vadinami „protingi agentai“ (angl. Intelligent Agents). Protingi agentai yra mašinos, kurios mąsto ir bando pakelti savo šansus įveikti tam tikrą užduotį. Visos tokios mašinos bando tiesiog atkartoti žmogaus smegenų veiklą. Kitaip tariant, bando atlikti veiksmus, kuriuos gali atlikti ir pats žmogus, pavyzdžiui, atpažinti matomą

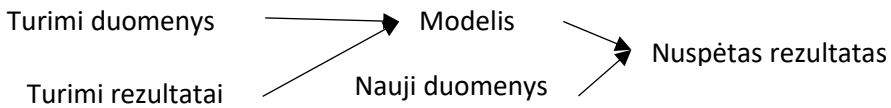
objektą arba atskirti, kada žmogus laimingas, o kada liūdnas. Jau 1950 metais, anglų matematikas, kriptoanalitikas Alanas Turingas (angl. Alan Turing 1912 - 1954) pasiūlė būdą kaip patikrinti ar mašina mąsto. [Hut99] straipsnyje yra aprašyta, kad A. Turingas pasiūlė „Turingo testą“, kuriame dalyvauja du žmonės ir mašina. Kad nebūtų fizinio kontakto, visi yra izoliuoti atskiruose kambariuose ir yra bendraujama per kompiuterius. Vienas žmogus yra teisėjas, o kitas žmogus ir kompiuteris yra dalyviai, kurie bando įrodyti teisėjui, kad jie yra žmonės. Jeigu teisėjas pasako, kad kompiuteris yra žmogus, tai mašina laimi testą.

2. Mašinos mokymasis

Mašinos mokymasis (angl. Machine Learning) yra kompiuterių mokslo atšaka, kurioje kuriamos programos, kurios gerina savo veikimo rezultatus ne jas perprogramuojant ar tobulinant, o jos pačios mokosi, naudodamos vis daugiau gaunamų duomenų.

Per pastarąjį dešimtmetį, mašinos mokymasis mums leido sukurti mašinas, kurios yra bepilotės, žodžių atpažinimo programas, efektyvias naršymo internete galimybes ir be abejo padėjo priartėti prie žmogaus genomo suvokimo. [SD14] straipsnyje yra aprašyti du pagrindiniai mokymosi tipai. Stebimas ir nestebimas mokymasis.

Stebimas mokymasis (angl. Supervised learning) pagal [Mat94a] dokumentaciją yra tada, kai mašinai yra duoti įvesties ir norimos išvesties pavyzdžiai. Tuomet tikslas yra išmokti taisykles, kurios padėtų gauti norimus rezultatus iš įvesties. Stebimą mokymąsi labai lengva atvaizduoti paprasta diagrama.



pav. 1 Stebimojo mokymosi modelis

Pirmame grafike iš turimų duomenų ir rezultatų yra sukuriamas modelis, kuriam yra paduodami nauji duomenys ir finale, mes gauname naujus rezultatus. Taip pat, prižiūrimas mokymasis yra skirstomas į papildomas dvi kategorijas.

1. **Klasifikavimas** – šios kategorijos tikslas yra baigtiniam duomenų sąrašui priskirti po klasę ar etiketę ir tokio tinklo rezultatas paprastai būna klasės ar etiketės pavadinimas.

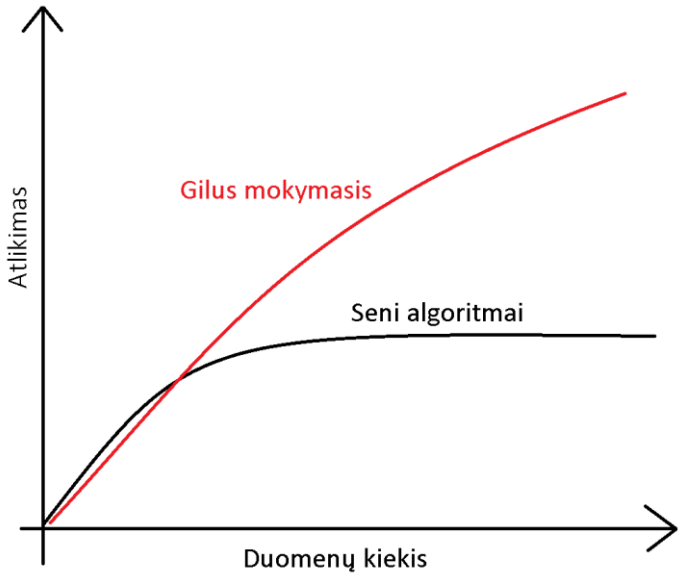
2. **Regresija** – šios kategorijos tikslas yra prognozuoti stebėjimų rodmenis. Tokių tinklų rezultatas paprastai būna realus skaičius, o juo yra bandoma nuspėti rinkos pokyčius, energijos suvartojimą ir panašiai.

Tuomet, kaip anksčiau minėjau yra nestebimas mokymasis (angl. Unsupervised learning) pagal [Mat94b] dokumentaciją egzistuoja tada, kai mašinai nėra paduodamos jokios pavyzdinės įvestys ir algoritmas turi pats atrasti struktūras iš gaunamų duomenų. Dažniausias neprižiūrimo mokymosi metodas yra grupavimo (angl. Clustering) analizė, kurios pagalba galime rasti paslėptus šablonus, stebimuose duomenyse, kuriuose grupės yra modeliuojamos naudojant panašumo vienetus, kurie aprašyti pagal Euklidines arba tikimybinio atstumo

metrikas. Toks mokymasis paprastai naudojamas sekų analizėje, genų grupavime ar medicinoje, vaizdų segmentavime.

2.1. Gilus mokymasis

Savo bakalauriniam darbui pasirinkau gilius neuroninius tinklus, nes tai yra populiariausias būdas kuriant dirbtinį intelektą. Gilus mokymasis yra mašinų mokymosi algoritmų klasė, kuri bando modeliuoti aukšto lygio abstrakcijas iš duotų duomenų, naudodama daugiasluoksniškumą. Kiekvienas paslėptas sluoksnis naudoja prieš tai buvusio sluoksnio išvestį – gautus rezultatus, kaip įvestį. Ši įvestis yra naudojama atlikti tam tikrus skaičiavimus, norint apmokyti tinklą. Andrew Ng, kuris yra iš įmonės „Coursera“, taip pat yra vyriausias mokslininkas „Baidu Research“ ir kuris sukūrė „Google Brain“, [Bro16] interviu kalbėjo, kad senos kartos mokymosi algoritmai, anksčiau ar vėliau, pasiekia plynaukštę, tačiau gilus mokymasis yra pirmoji klasė, kurią galima plėsti, nes gilus tinklas gaudamas vis daugiau duomenų grąžina geresnius rezultatus. Šią mintį jis pavaizdavo paprastu grafiku.



pav. 2 Grafikas sudarytas remiantis [Bro16] straipsniu

Antrame paveikslėlyje matome , kad juoda linija yra pavaizduoti seni algoritmai tokie kaip, sprendimų medis (angl. Decision Trees) arba įprasta mažiausių kvadratų regresija (angl. Ordinary Least Squares Regeression). Pagal [Bro16] straipsnį, seni algoritmai, kažkada pasiekia viršūnę ir jų rezultatai nebegerėja. Raudona linija yra pavaizduoti gilaus mokymosi algoritmai, tokie kaip gilūs neuroniniai tinklai (angl. Deep Neural Networks) arba konvoliuciniai neuroniniai tinklai (angl. Convolutional Neural Network). Gilaus mokymosi algoritmai pateikia geresnius rezultatus, kai dirbtinis tinklas gauna vis daugiau duomenų.

3. Neuroniniai tinklai

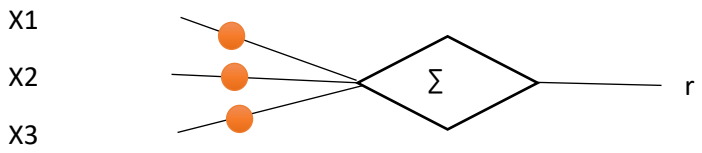
3.1. Dirbtinis neuronas

Dirbtinis neuronas (angl. Artificial Neuron) yra matematinė funkcija, kuri yra suvokiama, kaip biologinio neurono modelis. Dirbtinis neuronas yra pagrindinis vienetas, kuriant

neuroninius tinklus. Dirbtiniai neuronai atlieka pagrindines keturias funkcijas, kaip ir biologiniai neuronai.

1. Įvesties funkcija
2. Svorio keitimo funkcija
3. Vykdymo funkcija
4. Išvesties funkcija

Įvesties funkcija priima apdorotus duomenis arba duomenis gaunamus iš prieš tai buvusių neuronų. Svorio keitimo funkcija, keičia jungties tarp neuronų stiprumą. Paprastai svoris yra keičiamas, kai sumažėja skirtumas tarp apskaičiuoto rezultato ir norimo rezultato. Vykdymo funkcijų yra daugybe. Atsižvelgiant į tai kokią užduotį norime, kad neuronas atliktų, mes pasirenkame vykdymo funkciją. Galiausiai po vykdymo funkcijos, mes ateiname iki išvesties funkcijos, kuri arba parodo rezultatą, arba yra sujungta su kitu neuronu ir yra perduodamas rezultatas, kaip įvestis kitiems neuronams.



pav. 3 Dirbtinis neuronas

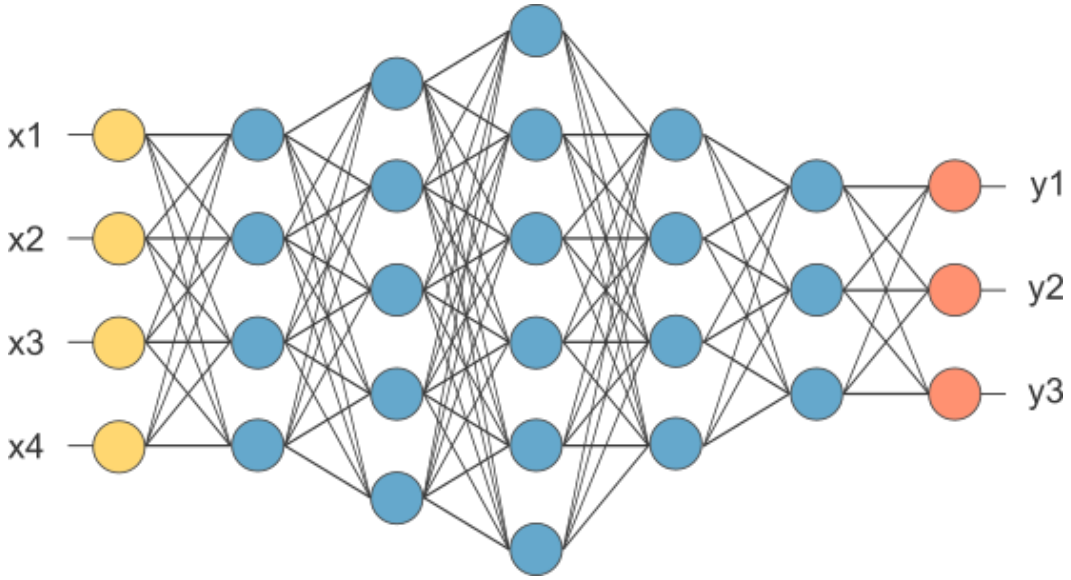
Trečiajame paveikslėlyje matome pavaizduotą dirbtinį neuroną, kur x1, x2, x3 yra įvesties funkcijos, oranžiniai skrituliai yra svorio keitimo funkcijos, ∑ yra Vykdymo funkcija ir r – tai išvesties funkcija.

3.2. Gilūs dirbtiniai neuroniniai tinklai

Dirbtiniai neuroniniai tinklai yra sudaryti iš tarpusavyje sujungtų dirbtinių neuronų. Tinklai gali būti padaryti gilūs. Gilus dirbtinis neuroninis tinklas reiškia, jog toks tinklas turės ne vieną, o kelis paslėptus sluoksnius. Dabar apžvelgsiu iš ko susidaro neuroniniai tinklai.

Pirmasis sluoksnis yra įvesties sluoksnis, kuris priima pradinius duomenis. Trečiasis sluoksnis – išvesties sluoksnis, skirtas atvaizduoti gautus rezultatus. Antrasis sluoksnis yra paslėptas sluoksnis, kuris neturi nieko bendro su išore, kaip įvesties ar išvesties sluoksniai ir yra skirtas atlikti atitinkamus skaičiavimus, gautus iš įvesties sluoksnio. Paslėptas sluoksnis gali būti sudarytas ne iš vieno, bet ir iš kelių sluoksnių.

Naudojant tokius neuroninius tinklus galima sukurti balso atpažinimo sistemą.



pav. 4 Gilių neuroninių tinklų architektūra iš [Ope17] puslapio

Ketvirtajame paveikslėlyje galime pamatyti visus išvardintus sluoksnius. Kur kiekvienas apskritimas reprezentuoja neuroną, o iš neuronų išdėstymo matoma kiekviena vertikali linija yra tinklo sluoksnis. Geltoni neuronai priklauso įvesties sluoksniui, į kurį patenka transformuotos duomenų reprezentacijos. Mėlyni neuronai priklauso paslėptiems sluoksniams. Paslėptame sluoksnyje matome 5 sluoksnius, kur kiekviename sluoksnyje gali būti skirtingas skaičius neuronų. Ir kiekviename paslėptame sluoksnyje paprastai yra skirtingos vykdymo funkcijos. Paskutinis, oranžinis sluoksnis yra išvesties sluoksnis, kuris išveda tinklo rezultatą.

4. Kalbėtojo atpažinimas

Kalbėtojo atpažinimas (angl. Speaker Recognition) – tai asmens atpažinimas naudojantis biometriniais duomenimis. Biometriniai duomenys – tai žmogaus fizinės ir psichologinės charakteristikos, pvz. akies rainelė, piršto antspaudas, liūdesys ar džiaugsmas. Labai svarbu nesumaišyti kalbėtojo atpažinimo su kalbėtojo autentifikavimu. Nors šios dvi sistemos ir skiriasi savo funkcionalumu, kur atpažinimas – tai sistema, kuri iš garso įrašo nustato asmens tapatybę, o autentifikacija – tai sistema, kuri iš garso įrašo nustato, ar kalbantysis asmuo yra duomenų bazėje ir tokios sistemos išvestis paprastai būna loginės reikšmės „true“ arba „false“, tačiau jos abi yra gana glaudžiai susijusios ir iš kalbėtojo atpažinimo sistemos galima nesunkiai perdaryti į kalbėtojo autentifikavimą ir „vice versa“. Šiame darbe galime pamiršti kalbėtojo autentifikavimą, nes kalbėsime tik apie kalbėtojo atpažinimą naudojant neuroninius tinklus.

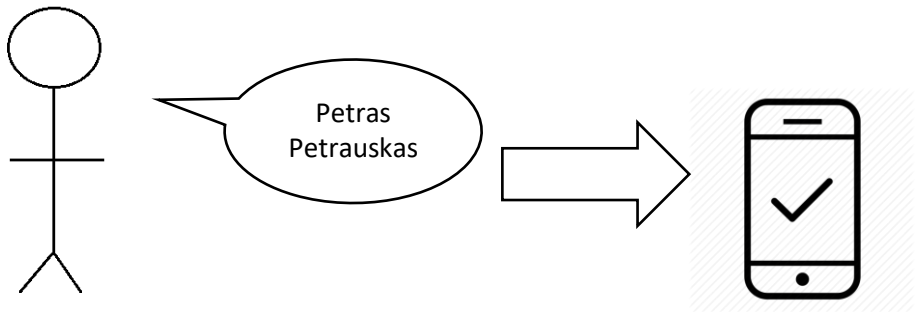
Balso atpažinimo sistemos tikslas, kaip ir minėjau yra atpažinti kalbėtojo balsą. Galbūt, kyla klausimas, kam reikia neuroninių tinklų norint atpažinti balsą? Juk galime balsą įrašyti į duomenų bazę ir palyginti su nauju įrašu. Be abejo, tai būtų paprasta išeitis, norint sukurti tokią sistemą, tačiau aplink mus sukasi įvairiausių aplinkybių, kurios gali pabloginti naująjį įrašą ir palyginus su duomenų bazės įrašu, mes negausime norimo atsakymo. Gali kalbėtojas būti užkimęs ar priduseęs. Gali būti, jog mikrofonas bus sugedęs ir iškreips įrašomą garsą, o galbūt tiesiog kažkas netoliese kalbėsis telefonu ir fone girdėsis pašalinis balsas. Atsiradus tokiems trikdžiams, mums gali pagelbėti gilūs neuroniniai tinklai. Mes galime sukurti neuroninius tinklus, kad jie galėtų atpažinti kalbėtoją iš jo balso, netgi tada, kai yra pašalinių trikdžių ar kalbėtojo balsas yra minimaliai pasikeitęs, dėl jo savijautos ar būklės.

4.1. Atpažinimo kategorijos

Kalbėtojo atpažinimas yra skirstomas į dvi kategorijas. Priklausomas ir nepriklausomas nuo teksto. Šiuos punktus aprašysiu remdamasis [Ryd01] darbu.

4.1.1. Priklausomas nuo teksto

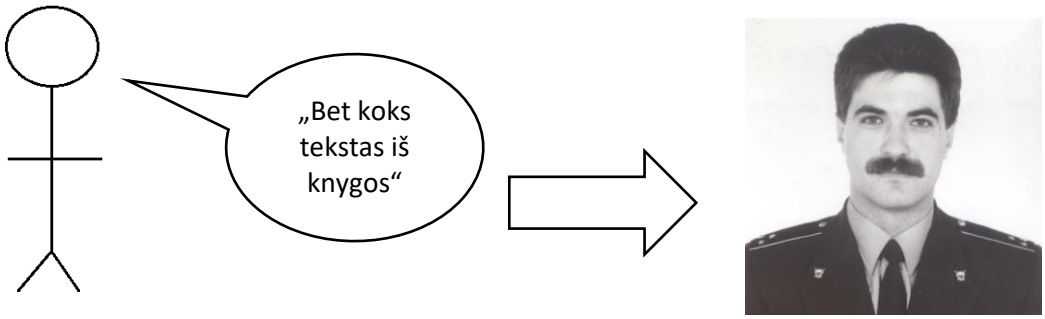
Tokios sistemos yra apmokomos su tam tikra fraze arba galėtume pavadinti slaptažodžiu. Tiek apmokymo, tiek tikrinimo metu yra naudojama viena ir ta pati frazė, kuri yra nustatoma prieš pradedant mokymosi procesą, kad galėtume atpažinti kalbėtoją. Tokios sistemos dažniau yra naudojamos ne balso atpažinime, o funkcijos uždavinyje. „Android“ operacinę sistemą turintys telefonai, turi tokią funkciją. Jeigu telefono naudotojas pasako „Okay google“, telefonas kaip mat įjungia google paiešką ir laukia, kol naudotojas pasakys, ko ieškoti. Manau būtų visai patogu, jei telefonai būtų atrakinami naudojantis žmogaus balsu ir jo pasirinkta fraze. Pavyzdžiui, kaip pavaizduota penktajame paveikslėlyje, žmogus pasako savo vardą ir kalbėtojo atpažinimo sistema jam leidžia naudotis telefonu kaip pavaizduota penktajame paveikslėlyje.



pav. 5 kalbėtojo atpažinimas priklausomas nuo teksto

4.1.2. Nepriklausomas nuo teksto

Nepriklausomos nuo teksto sistemos yra apmokamos su bet kokiu tekstu. Taigi, apmokymo metu gali būti naudojamas vienas pasakytas tekstas, o validacijoms - kitoks. Sistemos tikslas yra atpažinti kalbėtoją, nepaisant to, koks yra pasakytas tekstas. Kadangi, tokios sistemos neturi jokio leksinio suvokimo, teoriškai priklausomos nuo teksto kalbėtojo atpažinimo sistemos pasirodo geriau, nei nepriklausomos nuo teksto. Šiuo metu, nežinau, kur būtų naudojamos kalbėtojo atpažinimo sistemos, kurios yra nepriklausomos nuo teksto, bet manau, jos yra naudojamos slaptosiose tarnybose, ieškomų asmenų paieškai arba tiesiog laboratorijose, norint gerinti dirbtinio intelekto sugebėjimus. Šeštajame paveikslėlyje parodyta kaip žmogus sako bet kokią frazę ir kalbėtojo atpažinimo sistema patikrinus parodo, kas tai per žmogus, kaip parodyta šeštajame paveikslėlyje.



pav. 6 Kalbėtojo atpažinimas nepriklausomas nuo teksto

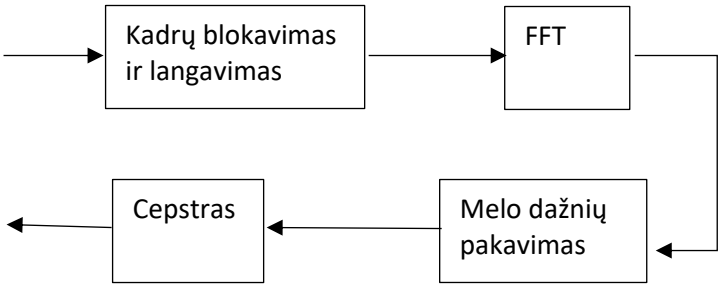
4.2. Kalbėtojo reprezentacija

Žmogus gali išmokti ir atpažinti įvairiausius garsus. Garsas sklinda bangomis ir jeigu mes esame netoliese garso šaltinio, garso bangos pasieks mūsų ausis. Remiantis [Hea13] straipsniu, garso bangos, pasiekusios ausies sraigės plaukų pluoštus, sujudina šiuos pluoštus ir sukuria vibraciją iš plaukelių judėjimo, kurie sukelia elektros impulsus ir pastarieji impulsai yra siunčiami į smegenų klausos centrą.

Neuroniniuose tinkluose tiek apmokymo, tiek tikrinimo metu, kalbėtojo atpažinimo sistemoje, mums reikia garso įrašo. Tačiau, kaip ir žmogus, dirbtinis tinklas negali apdoroti gryno garso, todėl mums yra reikalinga kažkokia kalbėtojo reprezentacija, kurią galėtų suprasti neuroninis tinklas. Viena iš reprezentacijų yra MFCC koeficientai.

4.2.1. Kalbėtojo reprezentacija išreikšta kaip MFCC

Melo dažnių cepstro koeficientas (angl. Mel Frequency Cepstral Coefficient, akronimas - MFCC) – tai yra dažniausiai naudojama garso signalo reprezentacija, kuri yra paremta žmogaus ausies veikimo principu. Žemiau yra parodytas grafikas iš [RA05] darbo, kaip atrodo MFCC procesorius.



Grafikas 1 MFCC procesorius iš [RA05] straipsnio

Pirmasis grafikas yra matematiškai paaiškintas žemiau.

Turime garso signalą $s(n)$, kuris yra ϕ kHZ dažnio. Tada dalijame $s(n)$ į tarkime t = 25 milisekundžių (ms) intervalus, tad iš viso turėsime γ kadrų, kur $\gamma = (t * 10^{-3}) * \phi$. Tuomet turėsime masyvą duomenų $s_i(n)$, kur n – svyruoja nuo 1 iki γ, o i – svyruoja per intervalų skaičių.

Toliau atliekame Diskrečiąją Furjė transformaciją (DFT), kuri atrodo taip,

$$S_i(k) = \sum_{n=1}^{N} s_i(n)h(n)e^{\frac{-j2\pi kn}{N}}, 1 \le k \le K \quad (1)$$

kur $h(n)$ – Hamingo lango funkcija (angl. Hamming Window), atrodo taip,

$$h(n) = \alpha - \beta \cos\left(\frac{2\pi n}{N-1}\right), \alpha = 0.54, \beta = 0.46 \quad (2)$$

K – tai DFT ilgis, $P_i(k)$ – tai i kadro galios spektras, kuris apskaičiuojamas pagal formulę,

$$P_i(k) = \frac{1}{N}|S_i(k)|^2 \quad (3)$$

$P_i(k)$ apskaičiavimas vadinamas periodograminiu galios spektro apskaičiavimu. Toliau atliekame greitąją Furier transformaciją (FFT) ir po jos gauname rinkinį koeficientų, iš kurių pasiemame tiek, kiek mums reikia.

Toliau mums reikia apskaičiuoti Melo pasiskirstymo filtro banką. Tai yra 20 – 40 filtrų rinkinys (standartiškai 26 filtrų rinkinys). Taigi, turime 26 vektorius, kurių ilgis yra, po greitosios Furjė transformacijos atlikimo, pasirinktų koeficientų skaičius. Norint apskaičiuoti filtro banko energijas mums reikia kiekvieną filtro banką padauginti iš galios spektro ir pridėti koeficientus. Kai tai padarome, mums lieka 26 skaičiai, kurie parodo, kiek yra energijos kiekvienoje filtro banko masyvo iteracijoje.

Norint apskaičiuoti Melo pasiskirstymo filtro banką pirmiausia reikia pasirinkti žemą ir aukštą dažnį f_l ir f_h. Tuomet naudodami Melo skalės funkciją,

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \quad (4)$$

paverčiame dažnius į melus m_l ir m_h. Kadangi turime 26 filtrus, tai reiškias, mums reikia 28 taškų, kur 26 iš jų yra tiesiškai pasiskirstę tarp m_l ir m_h ir gauname masyvą $m(i)$. Dabar naudojame atvirkštinę Melo skalės funkciją,

$$M^{-1}(m) = 700(\exp\left(\frac{m}{1125}\right) - 1) \quad (5)$$

ir gauname masyvą $h(i)$. Kadangi neturime dažnių raiškos, reikalingų sudėti filtrus į atitinkamus taškus, tad mums reikia suapvalinti dažnius iš $h(i)$ masyvo iki arčiauso greitosios Furier transformacijos (FFT) ilgio, kur suapvalinus gausime dar vieną masyvą $f(i)$. Norint atlikti suapvalinimą, turime žinoti greitosios Furjė transformacijos dydį ir imties rodiklį

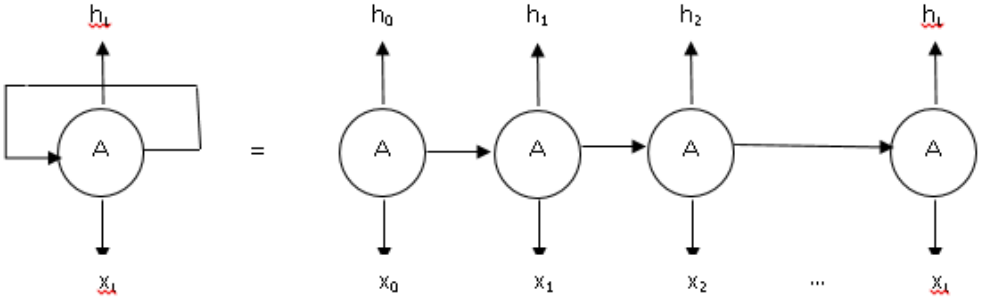
$$f(i) = floor((nfft + 1) * \frac{h(i)}{imties\ rodiklis}) \quad (6)$$

gavus $f(i)$ turime filtro banką, galime apskaičiuoti filtro banko energijas kiekvienam intervalui. Turėdami filtro banko energijas, iš tų 26 energijų pasiliekame pirmas 12 – 13 energijų. Jos ir bus mūsų Melo dažnių kepstro koeficientai. Pagaliau juos galime jas naudoti kaip tinkamus pradinius duomenis įvesties sluoksnyje ir pradėti apmokymą.

5. Kalbėtojo atpažinimas naudojantis rekurentiniais neuroniniais tinklais

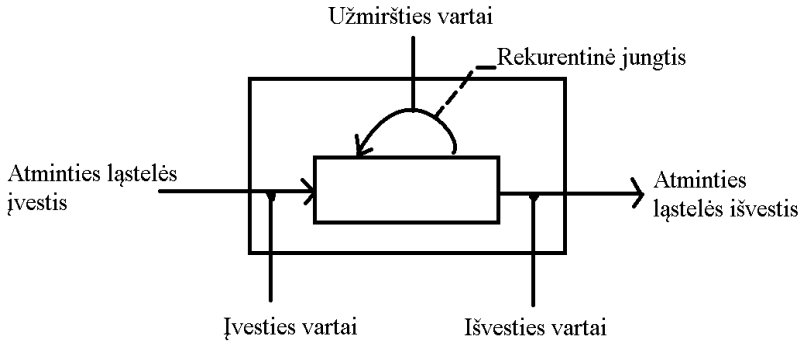
Rekurentiniai neuroniniai tinklai (angl. Recurrent Neural Network, akronimas - RNN) – tai dirbtinių neuroninių tinklų klasė, kur neuronai yra sujungti taip, kad jie sudarytų orientuotą grafą. Kad sistema veiktų geriau, galima naudoti ilgą trumpalaikės atminties architektūrą.

Ilga trumpalaikė atmintis (angl. Long short-term memory, akronimas - LSTM) – tai rekurentinių neuroninių tinklų architektūra. Ji naudojama tam, kad išvengtume ilgalaikės priklausomybės problemos.



pav. 7 Rekurentiniai neuroniniai tinklai

Septintame paveiksle matome išskleistą rekurentinį neuroninį tinką, kur A pasiima kažkokią įvestį x_t ir išveda h_t. Ciklas padeda perdavinėti išvestis iš vieno tinklo žingsnio į kitą. Iš tiesų, iš duoto paveikslo darosi aišku, kad rekurentiniai neuroniniai tinklai turi panašumų su paprastais neuroniniais tinkais. Tačiau, toks tinklas turi ilgalaikės priklausomybės problemą. Problema atsiranda, kai mūsų rekurentiniam neuroniniam tinklui reikia nuspėti kas bus toliau, remiantis praeities įvykiais. Teoriškai, rekurentinis tinklas turėtų turėti ilgalaikę atminį, bet praktiškai [CC16] straipsnyje buvo parodyta, kad toks tinklas negali turėti ilgalaikės atminties. Tam, kad būtų išvengta tokia problema buvo sukurta ilga trumpalaikės atminties architektūra.



pav. 8 Ilgos trumpalaikės atminties ląstelė

Iš aštuntojo paveikslo matome, kaip atrodo ilga trumpalaikės atminties ląstelė. Ji yra sudaryta iš keturių pagrindinių elementų. Įvesties vartų, kurie gali leisti ar blokuoti ateinančiam signalui keisti atminties ląstelės būseną. Neurono su rekurentine jungtimi, kur jungties svoris yra 1.0 ir užtikrina, kad blokuojant išorės interfeisus, atminties ląstelės būsena išliks konstanta nuo vieno žingsnio iki kito. Užmiršties vartų, kurie kontroliuoja rekurentinę jungtį ir leidžia ląstelei prisiminti arba pamiršti prieš tai buvusią būseną. Išvesties vartų, kurie leidžia arba draudžia atminties ląstelės būsenai turėti įtakos kitiems neuronams. Dabar

remdamiesi mokomuoju straipsniu [CC16] straipsniu, išsiaiškinsime, kaip atminties ląstelė keičiasi kiekviename laiko momente t. Pirmiausia reikia apskaičiuoti įvesties vartų i_t ir atminties ląstelių būsenų reikšmę $\widetilde{C_t}$ laiko momentu t.

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i) \tag{7}$$

$$\widetilde{C_t} = \tanh(W_c x_t + U_c h_{t-1} + b_c) \tag{8}$$

Toliau, apskaičiuojame užmiršties vartų aktivacijos reikšmę f_t, laiko momentu t.

$$f_t = \sigma\left(W_f x_t + U_f h_{t-1} + b_f\right) \tag{9}$$

Turėdami tris reikšmes iš 8, 9 ir 10 – os formulių, galime apskaičiuoti naują būseną laiko momentu t.

$$C_t = i_t * \widetilde{C_t} + f_t * C_{t-1} \tag{10}$$

Gavę naują atminties ląstelių būseną, mes galime apskaičiuoti išvesties vartų ir vėliau jų išvestis.

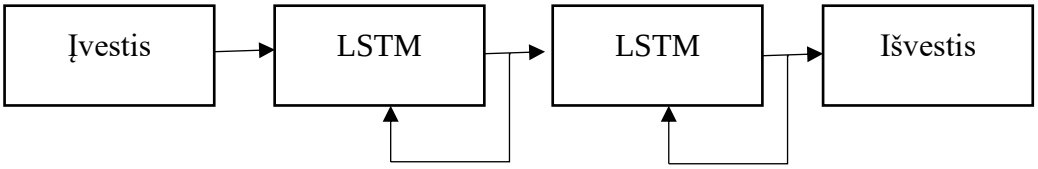
$$o_t = \sigma(W_o x_t + U_o h_{t-1} + V_o C_t + b_o) \tag{11}$$

$$h_t = o_t * \tanh(C_t) \tag{12}$$

Čia x_t – tai įvestis į ląstelę tam tikru laiko momentu t. $W_i, W_f, W_c, W_o, U_i, U_f, U_c, U_o, V_o$ – tai svorių matricos ir b_i, b_f, b_c, b_o – tai poslinkio vektoriai.

Taigi, trumpai susipažinę su ilga trumpalaikės atminties architektūra ir kaip kinta tam tikru laiko momentu, galime sudaryti gilų tinklą. Remiantis straipsnio [HS06] duotomis architektūromis, pasinaudosime vieną iš galimų gilios ilgos trumpalaikės atminties architektūrų.

9 – ame paveiksle matome gilų ilgą trumpalaikės atminties tinklą. Čia įvestis ir įvestis skirtos atitinkamai paduoti tinklui duomenis ir išvesti rezultatus. LSTM blokas – tai ilga trumpalaikės atminties ląstelė (angl. Long short-term memory, akronimas - LSTM). Punktyrinė rodyklė reiškia, kad tarp dviejų LSTM blokų, gali būti ir daugiau tokių blokų, priklausomai nuo to, kiek reikia. Reikia nepamiršti, kad tinklas su vienu LSTM bloku jau yra gilus tinklas, nes jį išskleidus gauname gilų neuroninį tinklą, kur kiekvienas sluoksnis naudoja tuos pačius modelio parametrus. Tačiau didinant LSTM blokų skaičių, mes galime parametrus paskirstyti per kelis blokus. Pavyzdžiui vietoj to, kad dvigubai padidintume standartinio modelio atmintį, mes galime padidinti LSTM blokų kiekį iki 4, su tokiu pat kiekiu parametrų. Toks tinklo patobulinimas padaro, kad įvestys eina per daugiau netiesinių operacijų per tam tikrą laiko tarpą. Tai reiškia, kad toks tinklas galėtų būti greičiau apmokytas.



pav. 9 Gili ilgos trumpalaikės atminties architektūra

Kaip ir giliuose neuroniniuose tinkluose, panaudosime d – vektorių, kaip kalbėtojo reprezentaciją. Kaip įvestį panaudosime Melo dažnių cepstro koeficientus. Priminkime žingsnius, kuriuos reikia atlikti norit gauti cepstro koeficientus.

1. Padaliname signalą į mažus kadrus.
2. Apskaičiuoti galios spektrą kiekvienam kadrui.
3. Pritaikyti Melo filtro banką kiekvienam galios spektrui ir sudėti energijas kiekviename filtre.
4. Logaritmuoti filtro banko energijas
5. Apskaičiuoti diskrečias kosinuso transformacijas.
6. Išmesti visus diskrečios kosinuso transformacijos koeficientus išskyrus pirmus 12 - 13.

Turėdami Melo dažnių cepsto koeficientus, mes galime pradėti apmokymą, kur bandysime išgauti d – vektorių, kaip kalbėtojo reprezentaciją atpažinimui. Taigi mokymo etape, mes vėlgi paduodame didelį kiekį garso įrašų. Visi įrašai pereina per funkciją, kuri gauna kiekvieno kalbėtojo Melo dažnių cepstro koeficientus ir juos paduoda įvesties sluoksniui, kur atitinkamai iš jo duomenys eina į LSTM bloką / -us. LSTM bloke yra atliekamos reikiamos funkcijos, kurios iš duotų koeficientų apskaičiuoja d – vektorių.

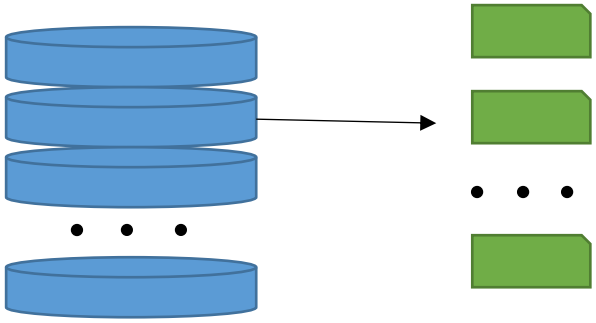
6. Neuroniniu tinklų implementacija naudojantis „Matlab“

Matlab – tai programavimo aplinka ir kalba, skirta manipuliuoti matricomis, implementuoti matematines funkcijas, vaizduoti grafikus ir daug kitų matematinių veiksmų. Pats Matlab turi neuroninių tinklų funkcijų rinkinį, kuris padeda kurti neuroninius tinklus. Šias funkcijas, galima susirasti [Mat94c] dokumentacijoje.

6.1. Duomenų bazė

Mano neuroninio tinklo duomenų bazę sudaro garso įrašai. Išviso buvo surinkti 921 skirtingi kalbėtojai iš [Pov12] duomenų bazės. Ši duomenų bazė yra sudaryta Meino informatikos universiteto laboratorijos (pranc. Laboratoire d‘Informatique de l‘Universiteté du Maine, akronimas - LIUM). Duomenų bazės turinys surinktas iš „TED“ (angl. Technology Entertainment Design) konferencijos įrašų. Kiekviename surinktame įraše kalbėtojas kalba anglų kalba. Šiame darbe, mums nesvarbu ar kalbėtojas yra vyras ar moteris. Įrašų trukmės svyruoja nuo 3 iki 13 sekundžių. Įrašai yra trumpi, nes buvo siekiama išbandyti kaip neuroniniai tinklai veikia su trumpais įrašais. Taip pat trumpesni įrašai teoriškai reiškia ir mažesnius skaičiavimus, kas paspartina neuroninių tinklų mokymąsi ir naudoja mažiau serverio resursų. Duomenų bazė, kurią parsisiunčiau nebuvo tinkamai paruošta. Kiekvienas garsinis failas buvo „sph“ formato. Kiekvieną failą konvertavau į „wav“ formatą naudodamas „SoX“ komandines eilutes, nes „sph“ formato failai yra paprastai naudojami kalbos

atpažinimui. Tuomet pasidariau naudodamas C# programavimo kalbą pasidariau man reikiamą duomenų bazės struktūrą, kuri vizualiai atrodė taip.



pav. 10 Kalbėtojų duomenų bazė

Septintame paveikslėlyje mėlynomis figūromis yra pavaizduoti n aplankalų, kur kiekviename aplankale yra po m garsinių failų. Taigi, maksimaliai buvo 921 kalbėtojas, tai išviso duomenų bazėje yra 921 aplankalas, kur kiekvienas iš jų turi po 3 garsinius failus. Maksimaliai trys garsiniai failai yra pasirinkti dėl laiko ir resursų stokos mokymosi metu, kas bus aptarta analizės dalyje.

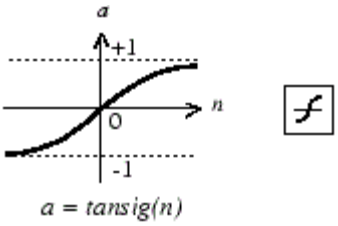
Duomenų bazę išskirsčiau į mokymosi ir tikrinimo. Atitinkamai mokymosi duomenų bazė buvo naudojama neuroninio tinklo mokymosi metu, o tikrinimo – tinklo testavimo metu. Duomenų bazes išskirsčiau, kad apmokytas tinklas nebūtų testuojamas su mokymosi duomenų baze, nes kitaip tinklas atspėtų visus kalbėtojus. Tikrinimo duomenų bazė yra sudaryta iš tų pačių kalbėtojų, tik kalbėtojų įrašuose yra kalbamas kitas tekstas ir kiekvieno įrašo trukmė gali skirtis nuo mokymosi duomenų bazės įrašų trukmės.

6.2. Tinklas

Neuroninis tinklas yra sudaromas iš kelių sluoksnių. Pirmasis yra įvesties sluoksnis. Įvesties sluoksnis yra sudaromas iš 36 neuronų, nes yra 12 – a Melo cepstro dažnio koeficientų, 12 – a delta koeficientų ir 12 – a delta – delta koeficientų. Sudėjus visus koeficientus, mes gauname 36 koeficientus, kurie atitinkamai yra įvesties neuronai.

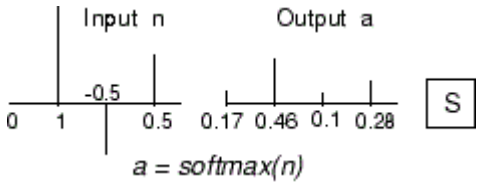
Paskutinis sluoksnis yra išvesties sluoksnis, kuriame yra x neuronų. Šiuo atveju x – tai kalbėtojų skaičius. Priklausomai nuo to, kiek kalbėtojų dalyvauja mokymosi metu, tiek išvesties sluoksnyje yra neuronų.

Tarp įvesties ir išvesties sluoksnių, mes galime susikurti norimą kiekį paslėptų sluoksnių. Mano atveju, aš nepersistengiau su tinklo gilumu, dėl laiko ir resursų stokos. Aš pasirinkau, kad būtų pora paslėptų sluoksnių, kur pirmasis yra mažas su 64 neuronais. Tokį skaičių pasirinkau, nes dažnas atvejis, jog programavime yra naudojamas skaičius 2^n, kur n – tai laipsnis, kuris priklauso natūraliųjų skaičių aibei. Šio sluoksnio funkcija yra hiperbolinės liestinės riestinės perdavimo funkcija (angl. Hyperbolic tangent sigmoid transfer function, akronimas - tansig) rasta [Mat94d] dokumentacijoje. Tansig yra neurono vykdymo funkcija. Šiai funkcijai paprastai yra paduodama *S x Q* dydžio matrica *M*, kuri atėjo per neurono įvestį, o išvestis grąžina *S x Q* dydžio matricą *A*, kur kiekvienas matricos *M* narys yra suspaustas į skaičių, kurio intervalas [-1; 1].



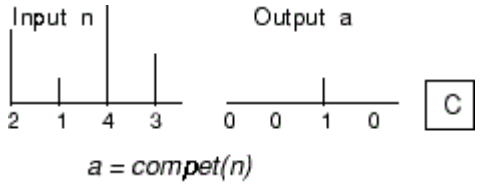
pav. 11 „Tansig“ funkcija iš [Mat94d] dokumentacijos

Antras paslėptas sluoksnis, buvo sudarytas jau iš kitokio skaičių neuronų. Šis sluoksnis turėjo x neuronų. Kaip ir išvesties sluoksnyje, šiame paslėptame sluoksnyje yra tiek neuronų kiek yra kalbėtojų. Toks neuronų skaičius yra todėl, kad šis paskutinis paslėptas sluoksnis yra sluoksnis, kuris naudoja prieš tai buvusio sluoksnio sąvybes ir todėl į kiekvieną sluoksnio neuroną yra dedamas galimai atpažintas kalbėtojas. Šio sluoksnio funkcija yra softmax perdavimo funkcija (angl. Softmax transfer function) rasta [Mat94e] dokumentacijoje. Ši funkcija taip pat yra neurono vykdymo funkcija, kuri per neurono įvestį gauną *S x Q* dydžio matricą *M* ir išveda *S x Q* dydžio matricą *A*, kur kiekvienam matricos *M* stulpeliui buvo pritaikyta softmax konkurencinga funkcija (angl. Softmax Competitive function) paimta iš [Mat94f] dokumentacijos.



pav. 12 „Softmax“ funkcija iš [Mat94e] dokumentacijos

Kompetentinga funkcija irgi priima *S x Q* dydžio matricą *M* ir išveda tokio pat dydžio matricą *A*, kur kiekviename stulpelyje reikšmė yra 1, jei matricos *M* stulpelis turi savo maksimalią reikšmę ir 0 kitu atveju.



pav. 13 „Compet“ funkcija iš [Mat94f] dokumentacijos

6.3. Serveris

Neuroninių tinklų mokymosi metu yra atliekami didelis skaičiavimai. Pradžioje, naiviai tikėdamasis, kad pavyks, aš pabandžiau pasileisti tinklo mokymąsi ant savo asmeninio kompiuterio, kuris turi pirmoje lentelėje pavaizduotus parametrus.

Procesorius	2,40 GHz i7-5500U
Vaizdo plokštė	Integruota
Operatyvi atmintis (RAM)	16 GB

lentelė 1 Asmeninio kompiuterio parametrai

Paleidus tinklą su 500 kalbėtojų, kompiuteris iškarto pradėjo naudoti visus resursus ir po kurio laiko tinklas tiesiog nulūždavo ir mesdavo klaidą, kad neužtenka operatyvios atminties. Tuomet, neturėdamas kur dėtis, užsisakiau serverį. Serveris buvo gautas iš interneto paslaugų tiekėjo „Baltneta“. Gavau serverį, kuris turi antroje lentelėje pavaizduotus parametrus.

Procesorius	10 virtualių procesorių po 2 GHz
Operatyvi atmintis (RAM)	40 GB

lentelė 2 Serverio parametrai

Tačiau, net gavus tokį serverį, pasileidus tinklą su pilna duomenų bazė (921 kalbėtojais), tinklas vis tiek lūždavo ir mesdavo tą pačią klaidą. Tuomet, pabandžius su 800 kalbėtojų, tinklas nenulūžo ir sėkmingai pradėjo mokymosi procesą. Klaida dėl per mažos operatyvios atminties buvo gaunama, nes programa, kai būdavo paleidžiama, visus įrašus užkraudavo į operatyviąją atmintį ir viso mokymosi metu buvo ji apkrauta.

7. Tyrimo analizė

Šioje dalyje aprašysiu savo tyrimo analizę. Analizė susidarė iš trijų dalių. Tinklo sudarymo, tinklo apmokymo ir tinklo tikrinimo. Tinklo sudarymas susideda iš kalbėtojų skaičiaus, kalbėtojų reprezentacijos, tinklo gilumo, kiekvieno sluoksnio neuronų skaičiaus, apmokymo funkcijos bei gradiento skaičiaus parinkimo. Tinklo apmokymas susidaro tiesiog iš laukimo. Laukimas užtrunka, nuo kelių minučių iki kelių parų ar net gi kelių savaičių. Tinko tikrinimas susidaro iš apmokyto tinklo rezultato tikrinimo, lyginant tikrinimo duomenis, kartu su apmokytu modeliu.

Toliau bus aptarti keli apmokyti neuroniniai tinklai su skirtingais parametrais. Vėliau pastebėsite, kad visuose tinkluose įvesties sluoksnyje yra po 36 neuronus. Toks skaičius, kaip anksčiau minėjau, yra parenkamas nes yra trys vektoriai. MFCC, delta ir delta-delta vektoriai, kurie turi po 12 koeficientu. Sudėjus kartu visus koeficientus, mes gauname 36 neuronus.

Kiekvienas iš aptariamų neuroninių tinklų bus matuojamas tikslumu, kurio dimensija yra procentai. Tinklo tikslumą aš skaičiuoju paprastai. Apmokės tinklą aš patikrinu tinklą paduodamas duomenis iš tikrinimo duomenų bazės. Tuomet aš gaunu rezultatą kiek kalbėtojų buvo atpažinta teisingai ir kiek klaidingai. Iš to nesunku suprasti, kad tikslumą paskaičiuoju, teisingai atpažintų kalbėtojų skaičių padalindamas iš visų kalbėtojų skaičiaus.

7.1. Net50_1

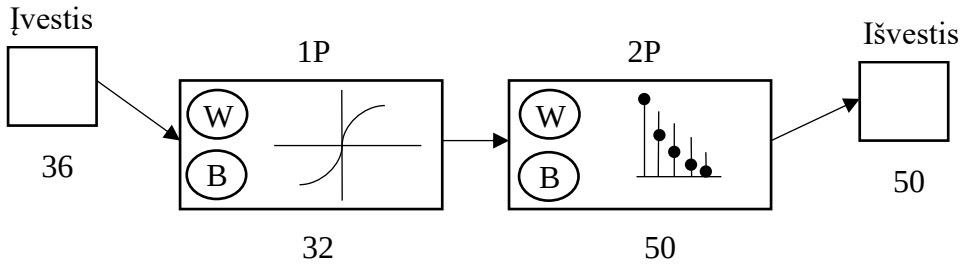
Šis tinklas buvo apmokytas naudojant 50 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turi po 1 garsinį įrašą. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis yra sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 50 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 64 ir 50 neuronų. Garsinis įrašas kaip minėta anksčiau svyruoja nuo 3 iki 16 sekundžių. Šiam tinklui prireikė 3 minučių ir 37 sekundžių, kad apsimokytų. Per šį laiką praėjo 331 epochas ir baigė darbą, nes tinklas praėjo validacijų patikrinimus. Validacijos tikrinimas tai procesas vykstantis viso mokymosi metu, kur yra tikrinama ar klaidos lygis nekinta per tam tikra epochų skaičių. Šio tinklo tikslumas yra 64%.

7.2. Net50_2

Šis tinklas buvo apmokytas naudojant 50 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turi po 2 garsinius įrašus. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis yra sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 50 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 64 ir 50 neuronų. Šiam tinklui prireikė 11 minučių ir 42 sekundžių, kad apsimokytų. Per šį laiką praėjo 548 epochas ir baigė darbą, nes tinklas praėjo validacijų patikrinimus. Šio tinklo tikslumas yra 66%.

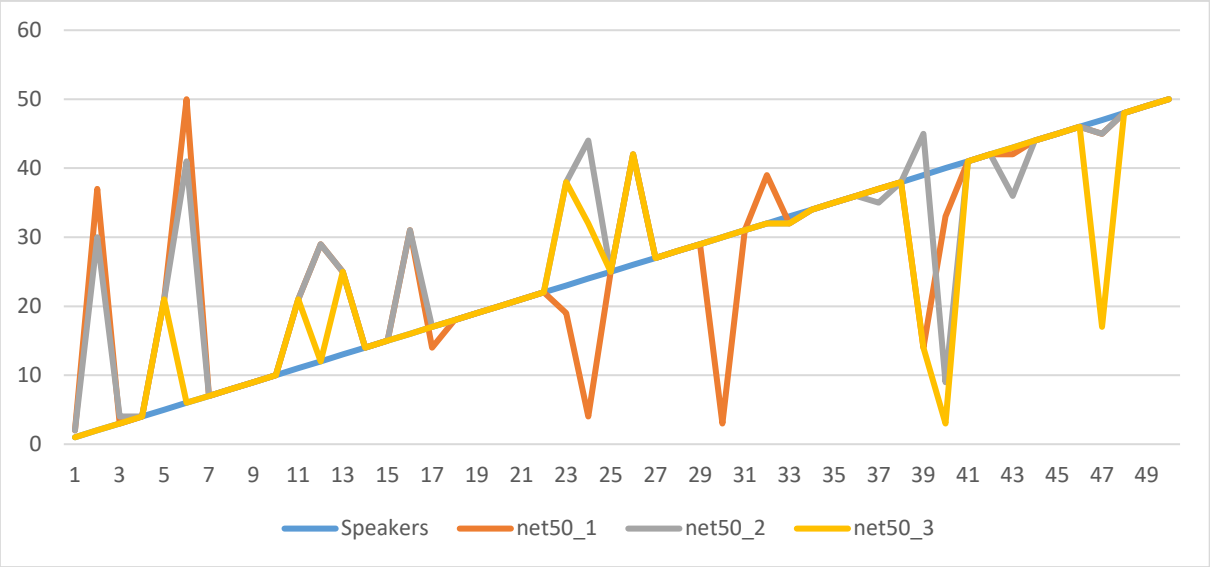
7.3. Net50_3

Šis tinklas buvo apmokytas naudojant 50 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turi po 3 garsinius įrašus. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis yra sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 50 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 64 ir 50 neuronų. Šiam tinklui prireikė 31 minučių ir 49 sekundžių, kad apsimokytų. Per šį laiką praėjo 647 epochas ir baigė darbą, nes tinklas praėjo validacijų patikrinimus. Šio tinklo tikslumas yra 72%.



pav. 14

Penktame paveikslėlyje matome, kaip vizualiai atrodė visų net50 tinklų architektūra. Visuose trijuose tinkluose architektūra buvo vienoda, tačiau keitėsi tik vienas dalykas - kalbėtojų duomenų kiekis. Pirmame tinkle kalbėtojai turėjo po vieną garsinį failą, antrame tinkle - po du garsinius failus, o trečiame tinkle - po tris garsinius failus. Nekeičiant tinklo architektūros, o tik didinant kiekvieno iš kalbėtojų duomenų kiekį, matomas akivaizdus pokytis, kur prasčiausiai pasirodė net50_1 tinklas, kurio tikslumas buvo 60 %. Geriausiai pasirodė net50_3, kuris pasiekė net 72 %.



pav. 15 net50 tinklų atpažinimo tikslumai

Devintajame paveikslėlyje yra pavaizduota kaip skirtingi net50 tikslai po apmokymo sugebėjo teisingai ar klaidingai atpažinti kalbėtojus.

Toliau, buvo atliktas tyrimas su daugiau kalbėtojų, norint pasižiūrėti, kaip kinta tinklo tikslumas, padidėjus kalbėtojų skaičiui.

7.4. Net100_1

Šis tinklas buvo apmokytas naudojant 100 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turi po 1 garsinį įrašą. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis yra sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 100 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 64 ir 100 neuronų. Šiam tinklui prireikė 44 minučių ir 34 sekundžių, kad apsimokytų. Per šį laiką praėjo 728 epochas ir baigė darbą, nes tinklas praėjo validacijų patikrinimus. Šio tinklo tikslumas yra 65%.

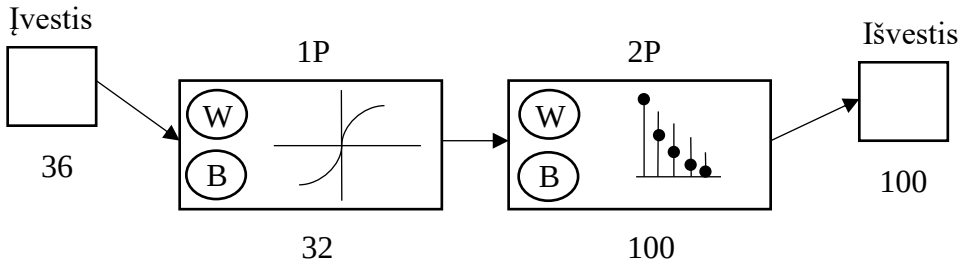
7.5. Net100_2

Šis tinklas buvo apmokytas naudojant 100 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turi po 2 garsinius įrašus. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis yra sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 100 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 64 ir 100 neuronų. Šiam tinklui prireikė 1 valandos 41 minutės ir 53 sekundžių, kad apsimokytų. Per šį laiką praėjo 814 epochas ir baigė darbą, nes tinklas praėjo validacijų patikrinimus. Šio tinklo tikslumas yra 66%.

7.6. Net100_3

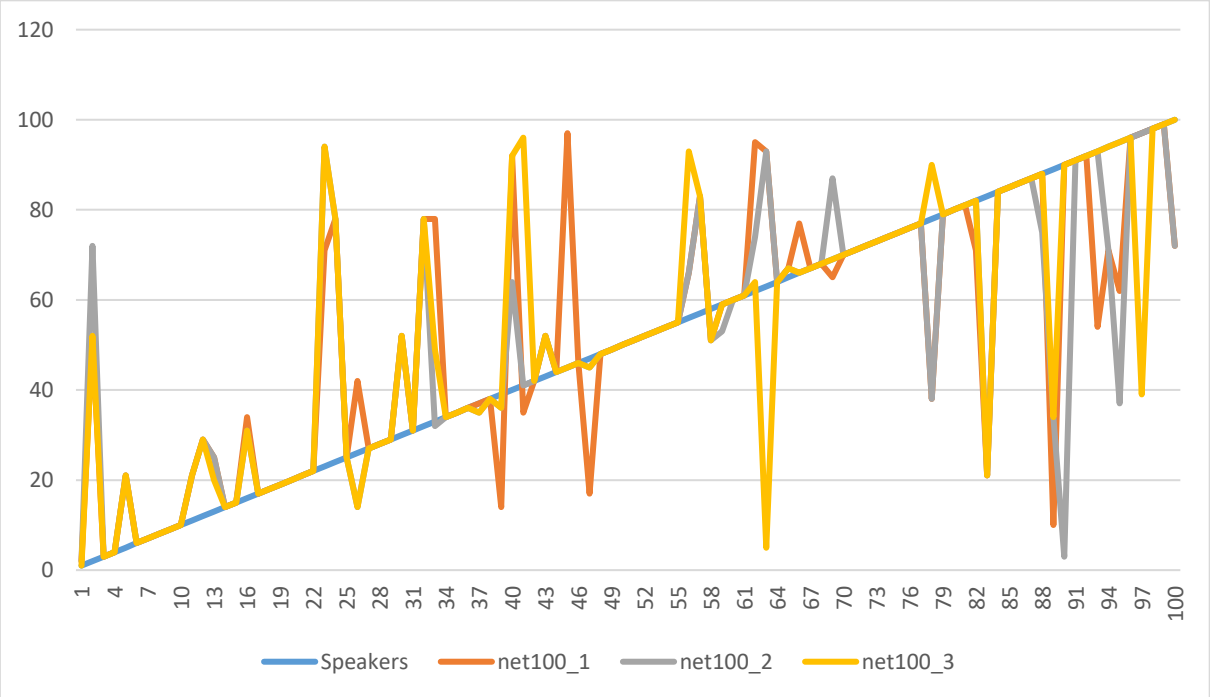
Šis tinklas buvo apmokytas naudojant 100 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turi po 3 garsinius įrašus. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis yra sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 100 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 64 ir 100 neuronų. Šiam tinklui prireikė 3

valandų 4 minučių ir 33 sekundžių, kad apsimokytų. Per šį laiką praėjo 980 epochas ir baigė darbą, nes tinklas praėjo validacijų patikrinimus. Šio tinklo tikslumas yra 73%.



pav. 16

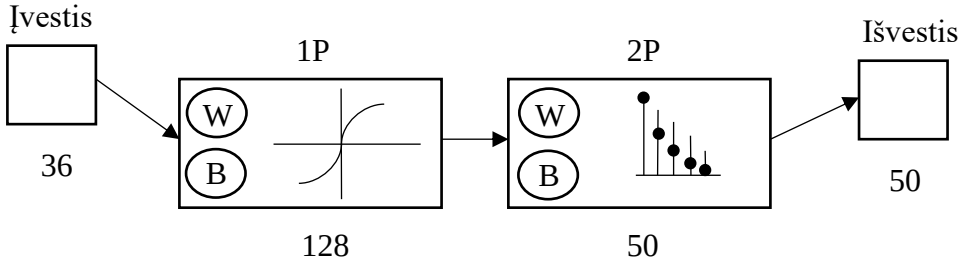
Kaip ir net50 tinkluose, taip ir aukščiau aprašytuose net100 tinkluose architektūra buvo vienoda. Vienintelis skirtumas nuo net50 tinklų architektūros yra, kad antrame paslėptame ir išvesties sluoksniuose yra nebe po 50, o po 100 tarpusavyje sujungtų neuronų. Net100 tinkluose buvo taip pat didinami kiekvieno iš kalbėtojų duomenų kiekiai. Prasčiausiai pasirodė net100_1 su 65 % tikslumu, o geriausias buvo net100_3 su 73 % tikslumu. Iš visų 6 tinklų tyrimo, matome, kad rezultatai gerėja, kai yra didinamas, kiekvieno kalbėtojo duomenų kiekis, tačiau ar tinkle yra apmokoma 50 kalbėtojų ar 100 skirtumo nelabai yra, nes rezultatai skiriasi vos per 1 %.



pav. 17 net100 tinklų atpažinimo tikslumai

Vienuoliktame paveikslėlyje yra pavaizduota kaip skirtingi net100 tikslai po apmokymo sugebėjo teisingai ar klaidingai atpažinti kalbėtojus.

Toliau tyrimams pasirinkau tinklus su 50 kalbėtojų, kadangi matome, kad rezultatai yra praktiškai identiški, kur yra tinklai su 100 kalbėtojų, todėl taupydamas laiką, pasirinkau toliau atlikti tyrimus su mažiau kalbėtojų.



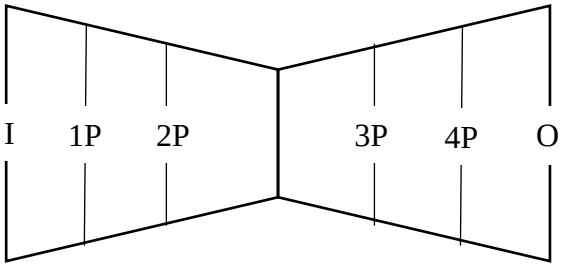
pav. 18

Septintame paveikslėlyje matome, minimaliai pakoreguotą net50 architektūrą. Ši architektūra turi vienintelį pakeitimą, kad pirmajame paslėptame sluoksnyje yra nebe 32, o 128 neuronai. Naudodamas šią architektūrą, atlikau pora tyrimų, pakeisdamas tik kiekvieno kalbėtojo duomenų kiekį nuo 2 iki 3. Pirmas tyrimas su 2 garsiniais failais kiekvienam kalbėtojuj užtruko vos 12 minučių ir 14 sekundžių arba 416 epochas , po kurių tinklas praėjo validacijų patikrinimą. Šio tinklo tikslumas buvo 76 %. Toliau buvo apmokomas tinklas jau su 3 garsiniais failais kiekvienam kalbėtojuj. Šis apmokymas truko 29 minutes ir 12 sekundžių arba 647 epochas , po kurių buvo praeitas validacijų patikrinimas. Šio tinklo tikslumas buvo 80 %.

Šių abiejų testų rezultatai yra geresni už net50 ar net100 tinklus, nes pirmasis paslėptas sluoksnis turi daugiau neuronų, kas leidžia tinklui išmokti atpažinti daugiau struktūrų, ir tai veda prie daugiau atpažintų kalbėtojų. Ar gali būti taip, kad daugėjant neuronų skaičiui, neuroninis tinklas bus vis geresnis ir geresnis? Į šį klausimą galima lengvai atsakyti. Taip, neuroninio tinklo atpažinimo tikslumas gerės, didėjant neuronų skaičiui, tačiau gerės tik iki tam tikros ribos. Jeigu mes vis didinsime ir didinsime neuronų skaičių, tokioje architektūroje, kuri yra pavaizduota 7 paveikslėlyje, mes gausime geresnius rezultatus iki kol pasieksime tam tikrą tašką, nuo kurio pasidarys nebesvarbu kiek bus neuronų, nes rezultatai išliks tokie patys. Tačiau, taip pat gali rezultatai ir prastėti, nes įvyks persimokymas (angl. overfitting). Tai įvyksta, kai yra sudaromas per gilus neuroninis tinklas duotam duomenų kiekiui. Todėl, atlikau papildomus du testus, su kuriais norėjau parodyti, kad dar labiau didinant neuronų skaičių, rezultatas nebekis ir kad padidinus neuronų kiekį per daug, tinklas persimokys ir jo rezultatai bus prastesni. Testą atlikau naudodamas 7 paveikslėlyje pavaizduotą architektūrą, išskyrus pirmame sluoksnyje padvigubinau neuronų skaičių iki 256. Šis tinklas apsimokė per 1 valandą ir 33 sekundes, o šio tinklo tikslumas buvo 80 %. Iš tokio rezultato akivaizdžiai matome, kad jau su 128 neuronais pirmame paslėptame sluoksnyje gavome geriausią atpažinimo rezultatą ir vėliau didinant neuronų skaičių, gauname tą patį rezultatą. Toliau buvo apmokytas neuroninis tinklas, kuris turėjo tokią pat architektūrą, kuri pavaizduota 18-ame paveikslėlyje. Vienintelis tinklo skirtumas buvo tas, kad pirmajame paslėptame sluoksnyje vietoj 128 neuronų buvo 1024 neuronai. Tokiam tinklui prireikė daugiau nei 6 valandų, kad baigtų mokymosi procesą. Atlikus tinklo tikrinimą, buvo pastebėta, kad rezultatai yra prastesni ir tinklui pavyko atpažinti 32 kalbėtojus iš visų 50, kurie buvo panaudoti mokymosi procese. Taigi, išrinkus per didelį neuronų kieki, buvo gautas persimokymas ir tinklo tikslumas buvo 72 %.

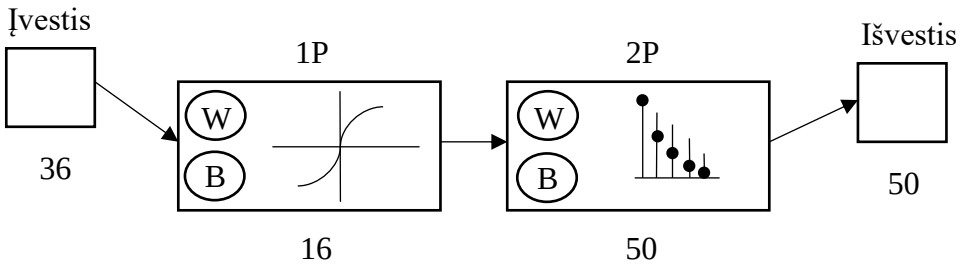
7.6. „Bottleneck“ tinklai

Kas yra „bottleneck“ (liet. Butelio kaklelis) neuroniniuose tinkluose? Trumpai tariant, butelio kaklelis yra tokia neuroninių tinklų architektūra, kur vienas paslėptas sluoksnis turi mažiau neuronų nei prieš tai buvęs sluoksnis. Paprastai butelio kaklelis yra naudojamas, kai yra norima sumažinti matricų dimensijas, kad būtų lengviau ir greičiau atliekami skaičiavimai.



pav. 19 Butelio kaklelio architektūra

Kaip vieną iš galimų eksperimentų pagalvojau, kad būtų įdomu atlikti tinklo apmokymą naudojant butelio kaklelio architektūrą, kuri yra vizualiai pavaizduota 8 paveikslėlyje. Kaip matome 9 paveikslėlyje, architektūra yra beveik panaši į net50, tačiau pirmasis paslėptas sluoksnis yra sudarytas vos iš 16 neuronų



pav. 20

Toks tinklas, kuris pavaizduotas 8 paveikslėlyje, apsimokė per 564 epochas arba 14 minučių ir 12 sekundžių. Šio tinklo duomenų bazė buvo sudaryta iš 50 kalbėtojų, kur kiekvienas kalbėtojas turi po 3 garsinius įrašus. Atlikus atpažinimo testus, buvo apskaičiuota, kad šis tinklas atpažįsta net 70 % kalbėtojų. Taip pat buvo atliktas identiškas testas, tik pirmame paslėptame sluoksnyje buvo ne 16, o 8 neuronai. Šis tinklas baigė mokymosi procesą per 16 minučių ir 33 sekundes. Atlikus atpažinimo testus, buvo gauta, kad tinklas atpažįsta 68 % kalbėtojų.

Lyginant su anksčiau aprašytais net50 neuroniniais tinklais šie butelio kaklelio architektūros tinklai pasirodė prasčiau. Rezultatai yra prastesni, nes pirmajame sluoksnyje yra mažiau neuronų ir tai priverčia tinklą išmokti mažiau struktūrų, nei yra, kas lemia tinklo atpažinimo prastėjimą.

7.7. Net800

Šis tinklas buvo apmokytas naudojant 800 skirtingų kalbėtojų, kur kiekvienas kalbėtojas turėjo po 1 garsinį įrašą. Tinklas susidaro iš 2 paslėptų ir 2 išorinių sluoksnių. Įvesties sluoksnis sudarytas iš 36 neuronų, išvesties sluoksnis sudarytas iš 800 neuronų, pirmas ir antras paslėpti sluoksniai atitinkamai sudaryti iš 128 ir 800 neuronų. Šis tinklas pasileido sėkmingai, po maksimalių 1000 epochų arba po daugiau nei 450 valandų apmokymo, tinklas persimokė ir nepraėjo validacijos patikrinimo, todėl tinklas taip ir nepabaigė mokytis. Ką reikia daryti, kad tinklas, su tokiu kiekiu duomenų baigtų mokytis? Praktiškai to parodyti negalėjau dėl laiko stokos, nes tinklas apsimokinėjo apytiksliai 19 parų. Teoriškai, norint pagerinti tokį tinklą ir padaryti, kad jis baigtų mokymosi procesą, reiktų padidinti visą tinklo architektūrą. Reikėtų padaryti daugiau paslėptų sluoksnių, teisingai paskirstyti neuronų skaičių kiekviename sluoksnyje.

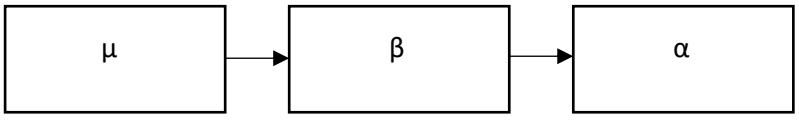
7.8. Kiti tinklai

Be anksčiau minėtų tinklų, bandžiau ir tinklus su 200, 300 kalbėtojų, tinklus kur yra 3 ir daugiau paslėptų sluoksnių, tačiau šie bandymai buvo nesėkmingi, nes jie arba nepraeidavo validacijų patikrinimo arba mokymosi procesas nulūždavo, kadangi reikėdavo daugiau resursų, negu turėjau, o laikas, kurį užtrukdavo tinklas apsimokyti, buvo per ilgas, kad galėčiau bandyti keisti parametrus ar koreguoti tinklo gilumą ir paleisti mokytis iš naujo. Mokymosi laikas svyruodavo nuo apytiksliai 48 valandų iki 100 valandų. Tačiau įsitikinau, kad norint greičiau apmokyti neuroninius tinklus, serveris su vaizdo plokšte (-ėmis) yra būtinas. Remiantis [KEA10] straipsniu kompanija „Intel“ konferencijoje „International Symposium on Computer Architecture“ paskelbė, kad naudojant „NVIDIA GeForce GTX 280“, skaičiavimai buvo atlikti 14 kartų greičiau nei naudojant „Intel Core i7 960“. Taigi, kaip pavyzdį galime paimti net800 tinklą, kuris užtruko 450 valandų nesėkmingai praeiti 1000 epochų. Jeigu, būčiau naudojęs vaizdo plokštę ir ji 14 kartų greičiau atliktų skaičiavimus, tai būčiau sužinojęs, kad padariau blogą tinklą po apytiksliai 32 valandų.

7.9. Palyginimas su kursiniu darbu

Kursiniame darbe buvau aprašęs rekurentinius neuroninius tinklus, bei rekurentinius neuroninius tinklus su ilga trumpalaike atmintimi. Šių tinklų geriausias rezultatus buvo 57,4 % tikslumo. Bakalauriniame darbe, geriausias rezultatas buvo net50_3 tinklo su padidintu paslėptu pirmu sluoksniu. Pastarasis pasiekė net 80% tikslumą.

Kyla klausimas, kodėl gilių neuroninių tinklų rezultatas yra geresnis nei rekurentinių neuroninių tinklų? Visų pirma, kursiniame darbe buvo naudotas neuroninis tinklas kuris buvo sudarytas iš trijų sluoksnių, kuris vizualiai atrodė taip.



pav. 21 Kursinio darbo neuroninis tinklas

Tryliktame paveikslėlyje matome 3 sluoksnius, kur μ yra įvesties sluoksnis, kuris turi 36 neuronus savyje, β tai pirmas ir vienintelis paslėptas sluoksnis , kuriame yra 128 neuronai, o α yra išvesties sluoksnis, turintis 774 neuronus. 10-ame paveikslėlyje yra pavaizduota

bakalaurinio darbo geriausio rezultato neuroninio tinklo architektūra, kuri turi 2 paslėptus sluoksnius, o tai reiškia, kad bakalaurinio darbo tinklas yra gilesnis ir tai praktiškai parodo, kad rezultatai yra geresni. Deja, dėl prarastų, kursinio darbo, rezultatų, negalėjau atkartoti tikslaus tyrimo, todėl negalėjau tiksliai palyginti tinklų su dabartine duomenų baze, ir dėl šios priežasties rezultatai yra nekorektiški.

Net50_1	64 %
Net50_2	66 %
Net50_3	72 %
Net100_1	65 %
Net100_2	66 %
Net100_3	73 %
Net50_2 su daugiau neuronų	76 %
Net50_3 su daugiau neuronų	**80 %**
Net50_3 su dar daugiau neuronų	80 %
Net50_3 overfit	72%
Net50_3_bottleneck_16	70 %
Net50_3_bottleneck_8	68 %

lentelė 3 Visų sėkmingų neuroninių tinklų rezultatai

Trečioje lentelėje matome, visų atliktų dirbtinių neuroninių tinklų rezultatus, kurie visi buvo aprašyti aukščiau.

Išvada

Bakalaurinio darbo pagrindinis tikslas buvo sukurti tekstui nepriklausomą gilių neuroninių tinklų sistemą, kuri gebėtų atpažinti kalbėtojus iš jų balsų. Kad, pasiekčiau savo tikslą aš aptariau teorinę dalį apie dirbtinį intelektą, neuroninius tinklus ir iš ko jie susideda. Buvo aprašyta, kaip iš garso įrašo galima gauti kalbėtojo reprezentaciją, kaip Melo dažnio cepstro koeficientus, kuri yra suprantama kompiuteriui. Taip pat išanalizavau gilių neuroninių tinklų architektūrą, kurią naudojau savo bakalauriniame darbe ir trumpai aprašiau, kas tai per architektūra. Priminiau, kad, praeitų metų, kursiniame darbe naudojau rekurentinių neuroninių tinklų architektūrą ir minimaliai aprašiau teorinę dalį apie šią architektūrą.

Naudojantis „Matlab“ programavimo aplinka ir kalba, buvo sukurtas gilus neuroninis tinklas, kurio tikslas buvo panaudoti kalbėtojų garsinius įrašus, juos išmokti ir gebėti atpažinti išmoktus kalbėtojus.

Bakalauriniame darbe buvo naudojama duomenų bazė, kuri buvo per didelė, nes gautas serveris neturėjo pakankamai resursų, kad galėtų būti paleistas dirbtinis neuroninis tinklas su 921 kalbėtojais. Tačiau nepaisant resursų stokos, galėjau panaudoti mažiau kalbėtojų.

Dažniausiai buvo mokomi tinklai, kurie turėjo po 4 sluoksnius. Vieną įvesties ir išvesties bei du paslėptus sluoksnius. Buvo mėginama daryti ir gilesnius tinklus, tačiau visi gilesni tinklai užtrukdavo daug laiko apsimokyti ir galiausiai dėl blogai parinktų parametrų jiems nepavykdo sėkmingai užbaigti mokymosi proceso.

Iš viso buvo apmokyti 12 tinklų su skirtingais parametrais ir kurie sėkmingai baigė mokymosi procesą. Visus dvylikos dirbtinių neuroninių tinklų rezultatus galime rasti, trečioje lentelėje. Daugiausia tinklų, buvo apmokyti su 50 kalbėtojų. Akivaizdžiai matome, kad geriausias rezultatas gavosi net50_3 tinklo, kuris savo pirmame paslėptame sluoksnyje turėjo 128 neuronus, vietoje 64 ir pasiekė 80 % tikslumą.

Taip pat buvo keletas tinklų, kurie nesėkmingai baigė mokymosi procesą. Net200, net300, net800, ar net200 su daugiau paslėptų sluoksnių. Šių tinklų mokymosi procesas užtrukdavo nuo poros dienų iki poros savaičių, ir kadangi neturėjau serverio su GPU, negalėjau atlikti toliau tyrimų su daugiau kalbėtojų ar gilesniais tinklais, kadangi mokymosi procesas užtrukdavo tiesiog per ilgai.

Aprašęs teorinę dalį ir atlikęs praktinę dalį galiu prieiti prie tokios išvados. Tyrimas pavyko. Buvo sukurtas gilus neuroninis tinklas, kuriam pavyko atpažinti kalbėtojus. Taip pat, pavyko pagerinti kursinio darbo rezultatus. Visi dvylika bandytų tinklų pralenkė kursinio darbo rezultatus, o geriausias bakalaurinio darbo neuroninis tinklas pralenkė beveik 25 – iais procentais. Žinoma duomenų bazės, kaip anksčiau minėjau, skiriasi, tačiau atpažinimo tikslumas bakalauriniame darbe gavosi geresnis.

Summary

In my bachelor, I analyzed the problem of text-independent speaker recognition using artificial neural networks. The main goal was to create deep neural networks that could recognize speakers from their voices.

To reach my terminus, I analyzed theoretical part of artificial intelligence and I made practical analysis of creating a deep neural network, that can recognize speakers from their voices. I summed up neural networks and what do I need to create ones. I investigated how mathematically convert sound files to Mel frequency cepstral coefficients (MFCCs) as a speaker representations, which are understandable by software. I reminded that last year in my course work I made a research about recurrent neural networks.

Using “Matlab” programming IDE and language, I created fully connected deep neural network, which had to learn speakers by their voices from sound files and recognize then afterwards.

The database which I used contained 921 different speakers. Sadly, but by the lack of the resources, that I have got, the maximum number of speakers I could use was 800 and less.

I ran 12 different neural networks that were successful. The success of these networks was because of the correctly chosen number of neurons in each layer, depth of the networks, time of frames, number of gradient, number of speakers in training database. Net50_3 network reached the best result of all practical analysis with the accuracy of 80 %.

I also had several networks, like net200, net300, net800 or net200 with more hidden layers, that were not successful because of the overfitting or because the network did not reach minimum gradient. These networks took from a few days to a few weeks to train and since I had server without GPU the time to train was just too long, to deal with.

Finally, after reviewing theoretical part about artificial intelligence and neural networks I can say that the experiment was successful. I created deep neural network, that was able to recognize speakers from their voices.

Literatūros sąrašas

[Hut99] Jason Hutchens, The Turing Test, URL: http://www.psych.utoronto.ca/users/reingold/courses/ai/turing.html.15.3KB, 1999.

[SD14] Shai Shalev-Shwartz, Shai Ben-David Understanding Machine Learning. From Theory to Algorithms, Cambridge University Press, p. 22, 2014.

[Mat94a] Mathworks, Supervised Learning Workflow and Algorithms, URL: https://se.mathworks.com/help/stats/supervised-learning-machine-learning-workflow-and-algorithms.html?searchHighlight=supervised%20learning&s_tid=doc_srchtitle. 91.9KB, 1994-2017.

[Mat94b] Mathworks, Machine Learning Techniques for Finding Hidden Patterns or Intrinsic Structures in Data, URL: https://se.mathworks.com/discovery/unsupervised-learning.html. 60.2KB, 1994-2017.

[Bro16] Jason Brownlee, What is Deep Learning?, URL: http://machinelearningmastery.com/what-is-deep-learning/. 170KB, 2016 – 08 – 16.

[Ryd01] Sara Rydin, Text dependent and text independent speaker verification systems. Technology and applications, Centre for Speech Technology, KTH, Stockhold. 2001-12-21.

[Hea13] hear-it, How the brain processes auditory signals, URL: http://www.hear-it.org/How-the-brain-processes-auditory-signals. 83.1KB, 2013 – 05 – 02.

[Ope17] OpenNN, Deep Architecture, URL: http://www.opennn.net/. 13KB, 2017.

[RA05] Adjoudj Reda, Boukelif Aoued, Artificial Neural Network & Mel-Frequency Cepstrum Coefficients-Based Speaker Recognition, 3rd International Conference: Sciences of Electronic. 2005.

[Mat94c] Mathworks, Neural Network Toolbox Functions, URL: https://se.mathworks.com/help/nnet/functionlist.html. 90.5KB, 1994-2017.

[Pov12] Daniel Povey, TED-LIUM, URL: http://www.openslr.org/7/. 4.71KB, 2012.

[Mat94d] Mathworks, tansig, URL: https://se.mathworks.com/help/nnet/ref/tansig.html. 66.2KB, 1994-2017.

[Mat94e] Mathworks, softmax, URL: https://se.mathworks.com/help/nnet/ref/softmax.html. 65KB, 1994-2017.

[CC16] Pierre Luc Carrier and Kyunghyun Cho. LSTM Networks for Senriment Analysis. http://deeplearning.net/tutorial/lstm.html#id1. 24 KB, 2008 – 2016.

[Mat94f] Mathworks, compet, URL: https://se.mathworks.com/help/nnet/ref/compet.html. 64.59KB, 1994-2017.

[KEA10] Andy Keane, „GPUS ARE ONLY UP TO 14 TIMES FASTER THAN CPUS“ SAYS INTEL, URL: https://blogs.nvidia.com/blog/2010/06/23/gpus-are-only-up-to-14-times-faster-than-cpus-says-intel/. 99.3KB, 2010 – 06 – 23.

[HS06] Hinton and R. R. Salakhutdinov. Reducing the Dimensionality of Data with Neural Networks ScienceMag, 313(7), 2006, pp. 504-507.