Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

FERI

Sašo Pavlič

Branje in obdelava EEG-signalov – pristop s strojnim učenjem

Diplomsko delo

Maribor, avgust 2019

Univerza v Mariboru

Fakulteta za elektrotehniko,
računalništvo in informatiko

FERI

Branje in obdelava EEG-signalov – pristop s strojnim učenjem

Študent: Sašo Pavlič

Študijski program: Visokošolski študijski program
Informatika in tehnologije komuniciranja

Mentor: dr. Sašo Karakatič

Somentor: mag. Boris Bizjak

Zahvala

Najprej bi se rad zahvalil svojemu mentorju dr. Sašu Karakatiču za konstantno strokovno pomoč in usmeritve pri pisanju diplomskega dela.

Iskreno bi se rad zahvalil svoji celotni družini, ki me je v času študija podpirala in me navdihovala na moji izobraževalni poti.

Branje in obdelava EEG-signalov – pristop s strojnim učenjem

Ključne besede: EEG, možganski valovi, strojno učenje, BCI-naprava, snemanje podatkov

UDK: 004.383.3:612.82(043.2)

Povzetek

Diplomska naloga zajema spoznavanje in predstavitev z osnovami EEG-možganskih valov s pomočjo naprave Emotiv Insight. Zajeti EEG-podatki predstavljajo vhodne podatke v modelu strojnega učenja, s pomočjo katerega se je ugotavljalo, kdaj in kje se pojavljajo iskani vzorci. Eksperiment razvite metode zajema podatkov in uporabe modela se je izvedel tako, da se je testni subjekt izpostavil izmenjujočim izbranim slikam, ob tem pa so se z napravo Emotiv Insight zajeli EEG-možganski valovi. Zajeti EEG-podatki so služili kot zbirka podatkov, iz katere se je učil klasifikacijski model umetne nevronske mreže, ki uspešno razpoznava, kdaj je testni subjekt podvržen eni vrsti slik in kdaj drugi.

Reading and processing EEG signals – machine learning approach

Keywords: EEG, brainwaves ,machine learning, BCI device, recording data

UDK: 004.383.3:612.82(043.2)

Abstract

The diploma thesis covers cognition and presentation, with the basics of EEG brain waves using the Emotiv Insight device. The captured EEG data represents the input data into a machine learning model, which was used to determine when and where the required patterns appear. The experiment of the developed method of capturing data and model usage was carried out by exposing the test subject to the alternating selected images and capturing the EEG brain waves with the Emotiv Insight device. The captured EEG data served as a database from which the artificial neural network classification model learnt to successfully recognize when a test subject was exposed to one type of image and when to another.

Kazalo vsebine

KAZALO SLIK

KAZALO TABEL

SEZNAM UPORABLJENIH SIMBOLOV IN KRATIC

BCI – Brain Computer Interface (Vmesnik možgani računalnik)

COM – Communication port

EEG – Electroencephalography

AI – Artificial inteligence

ML – Machine learning

DL – Deep learning

CSV – Comma seperated values

Cortex – Osrednji razred, ki uporablja knjižnico Cortex API

Data annotations – Opombe za podakte

Insight, Epoc, Epoc+ – Različni modeli BCI-naprav

Backend – Strežniški del aplikacije

DNN – Deep neural network

RNN – Recurrent neural network

1 Uvod

V razvoju informacijske tehnologije smo si vedno za vzor jemali naravo in načine, kako se je spopadala s problemi s pomočjo mehanizma evolucije, kot so na primer preživetje najmočnejšega, dedovanje primernih genov, iskanje najustreznejše poti, in vsekakor tudi naše najmočnejše orodje. To so seveda naši možgani, vendar vse do današnjega dne še vedno možgane le preučujemo, in sicer kako delujejo in kaj upravlja nek določen del možganov v našem telesu. Vsekakor so za vsako našo potezo, akcijo zadolženi le-ti. Ko pridemo do povezave med našo stvaritvijo, ki jo imenujemo računalnik, in našimi možgani, moramo še vedno uporabiti vmesne naprave, ki so po navadi miške, tipkovnice, krmilniki in prsti.
V okviru diplomske naloge bomo prikazali, kako lahko neko napravo (angl: *brain computer interface*) uporabimo za branje določenih možganskih valov in na podlagi tega ugotavljamo razlike v dojemanju, kar lahko v prihodnosti vodi do ugotavljanja želja, čustev, razmišljanj. V praksi bi to pomenilo prepoznavanje in kontroliranje računalnika oz. naprave z našimi mislimi.

Zaradi vseh dejavnikov, ki povzročajo eksplozije misli v vsakdanjem življenju, je zelo težko možgane pripraviti do tega, da bi bili osredotočeni na specifično stvar, ter bi izklopili vse preostale akcije in misli, ki jih ne potrebujemo za analizo našega problema. Prav to bo največji izziv, kako izdelati oz. implementirati stanje, v katerem bomo odstranili vse zunanje dejavnike (šume). Prav tako pa lahko težavo pomeni tudi uporabniški vmesnik, ki bi bil primeren za upravljanje z računalnikom. V nalogi se bomo osredotočili na branje možganskih valov in obdelavo podatkov o možganskih valovih s pomočjo strojnega učenja.

2 Razumevanje in pridobivanje podatkov

2.1 Analiza možganskih valov v preteklosti

Uporaba elektroencefalografije (EEG) je osrednji del tekočih raziskav človeških možganov in se uporablja že več kot 80 let. Kot metoda za beleženje delovanja možganov ostaja ena od redkih metod, ki so neposredno povezane z nevronsko aktivnostjo. Ko nevroni drug drugemu streljajo ali pošljejo signal, proizvajajo električni tok; ko je dovolj električnih tokov, ki jih proizvajajo, jih lahko elektroda zazna, kar nas privede do raziskovanja, kako delujejo možgani. Da bi razumeli, kako je to orodje oblikovalo sodobno raziskovanje, je treba pogledati njegovo preteklost.

Raziskovanje se je začelo z Luigijem Galvanijem (1737−1798), uglednim znanstvenikom, ki je med mnogimi svojimi dosežki in odkritji pokazal, da bi lahko žabje krake z električnimi impulzi stimulirali, da bi trzali in se premikali. Električni tok bi stimuliral živčna vlakna in povzročil krčenje mišic. Najpomembnejše je, da je Galvani prvi pokazal, da je električna energija sila, ki nadzoruje naša telesa. To je bil osrednji trenutek za biološko znanost in danes podpira naše razumevanje nevroznanosti. [1]

Oče EEG-ja Hans Berger (1873–1941), nemški psihiater, ki je bil naklonjen mističnemu in prepričanju v telepatijo. Pomislil je na nadaljnje napredovanje inovacij. Leta 1929 je izdal v knjigi »*On the Electroencephalogram of Man«*, ki je pokazal EEG v obliki, skoraj enaki tehnologiji, ki jo poznamo danes. V članku so predstavili delovanje EEG-ja na najmanj 44 straneh in tako previdno dokumentirali valove električne aktivnosti, ki se pojavljajo v možganih. Ti valovi − nevronska nihanja − so pokazali, kako bi sinhronizirani orkester tisočih nevronov lahko ustvaril električna polja, ki jih lahko zazna in izmeri EEG. To stališče so si delili mnogi v znanstvenem svetu, ki so nazadnje postavili pod vprašaj legitimnost tako nezaslišane ugotovitve.

Do leta 1939 so znanstveniki že govorili o »alfa« in »beta« valovanjih, da bi opisali stopnjo nevronske aktivnosti in razpravljali o nadaljnjih raziskavah epilepsije. Najpogostejša stopnja nevronske aktivnosti v budnem človeku, alfa val, je bila prvotno znana kot "Bergerjev ritem"

V naslednjih desetletjih se je konstantno povečevala uporaba EEG z različnimi aplikacijami. Raziskovalci so dosegli napredek pri proučevanju možganske patologije, delovanja in obnašanja, vse zaradi enostavnosti in vpogleda v EEG-zapise. Ta napredek je bil zlasti razširjen ali pomemben v raziskavah spanja in diagnosticiranju epilepsije [1]

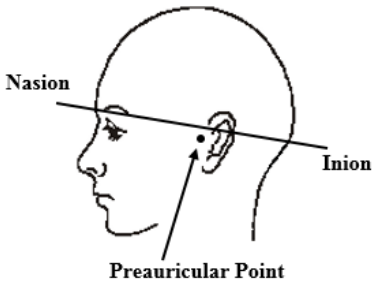
2.2 Karakteristike in frekvence valov

EEG (elektroencefalograf) predstavlja meritve električne aktivnosti večjih skupin nevronov v naših možganih. Da bi lahko te aktivnosti nevronov izmerili, moramo zajeti signale iz različnih mest v naših možganih, ki sinhrono pošiljajo signale, ter ugotoviti, koliko jih pošilja signale naenkrat. Elektroencefalogram se imenuje posnetek, ki ga dobimo iz tega opazovanja, v njem pa lahko tudi razberemo, v kakšnem stanju je osebek glede na EEG-ritmov v posnetku.

Torej EEG združuje podatke o električni aktivnosti tisoče nevronov, ki sinhrono generirajo električno napetost (v voltih). EEG ima tudi dobro časovno resolucijo, vendar sta frekvenčni razpon in prostorska resolucija omejena. Ena izmed slabosti pa so artefakti, ki nastanejo ob drugih bioelektričnih signalih, kot je premikanje mišic ali očesa v bližini elektrod; naša kakovost EEG-signala se s tem poslabša.

Amplituda, ka nastane pri snemanju podatkov, je oslabljena, saj pride do upornosti tkiva, ki se nahaja med generatorji potenciala in elektrode. Da bi zmanjšali ta efekt, moramo elektrode postaviti na čim boljše položaje na glavi (odvisno koliko elektrod ponuja naš BCI). V svetu obstaja mednarodni standardni sistem nameščanja elektrod. Sistem se imenuje 10−20, to je razporeditev elektrod na 10 ali 20 % razdalje od čela do tilnika. Pri takšni postavitvi dobimo podatke, ki jih lahko med seboj primerjamo z različnimi osebami. [2]

Poglejmo še, kako moramo določiti tri referenčne točke, da bi kar se da dobro kasneje namestili elektrode na lobanjo za pridobivanje kar se da kakovostnih podatkov.



Slika 1: Referenčne točke na lobanji [3]

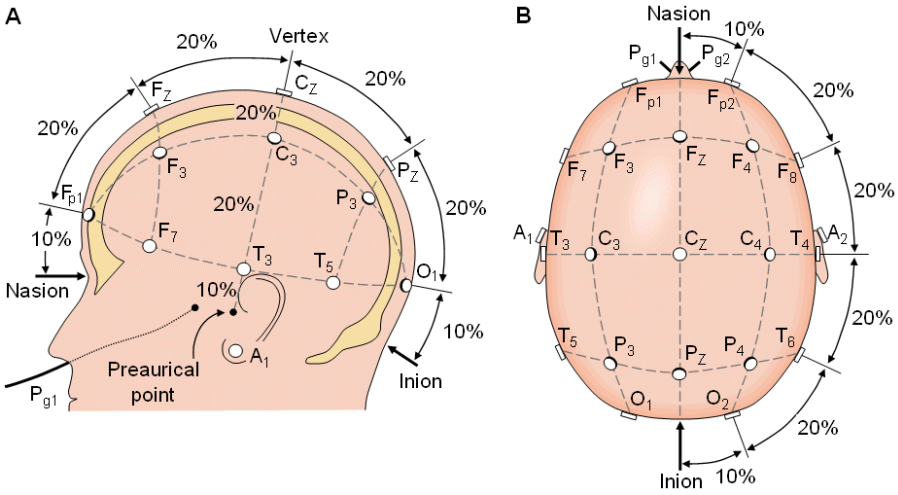
Točke na lobanji:

- predušna referenčna točka (angl. *pre-auricular*),
- zatilna referenčna točka (angl. *inion*),
- nosna referenčna točka (angl. *nasion*).

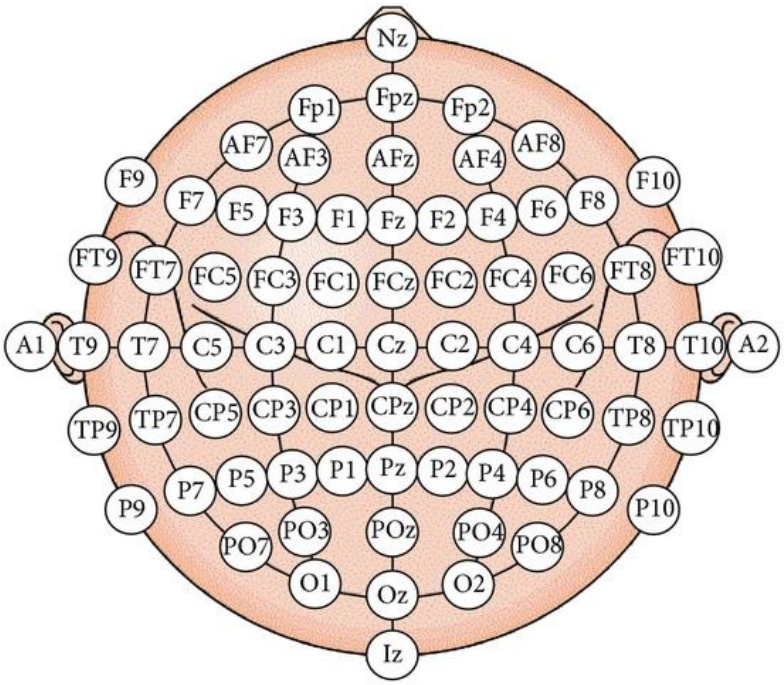
Elektrode so poimenovane na naslednje oznake glede na anatomski položaj referenčne točke.

- Fp – prefrontalni del,
- F – frontalni del,
- T – temporalni del,
- O – temenski okcipitalni del,
- C – centralni del,
- P – parientalni del,
- A – aurikularni ušesni del,
- G – ground elektroda za prizemljitev.

Lihe številke označujejo levo polovico glave, sode pa desno



Slika 2: Razmerja referenčnih točk [4]



Slika 3: Poimenovanja referenčnih točk [5]

Naši možgani delujejo na več različnih frekvencah. Delovanje možganov lahko opišemo z več različnimi možganskimi valovi, ki nastopajo pri različnih funkcijah delovanja možganov. Oglejmo si posamezne frekvenčne pasove in njihove značilnosti.

Delta (0,5−3 Hz)

Najnižja frekvenca možganskih valov, ki se giblje pod 3 Hz, se pojavlja se predvsem v globokem spanju. Ta frekvenca prevladuje pri dojenčkih do enega leta starosti. Prisotna je tudi med 3. in 4. fazo spanca.

Delta valove zmanjšujemo pri zelo intenzivni zbranosti in kadar uporabljamo svoje miselne procese zelo aktivno. Zanimivost najdemo pri posameznikih, ki imajo težave z zbranostjo in učenjem. Naravno povečujejo delta valove; kadar se želijo zbrati, jih ne uspejo zmanjšati. Prav zaradi tega pojava omejijo svojo zmožnost usmerjanja zbranosti in učenja.

V tem stanju najdemo možgane v zaklenjenem ponavljajočem se stanju, saj v tem stanju sanjarimo ali smo zaspani. Prav to je želela doseči narava. [4]

Akcija	**Opis**
Subjektivni občutki	Globoki spanec brez sna, nezavest
Vedenje in aktivnosti	Ne pozorno, mirujoče
Fiziološke povezave	Mirovanje
Učinki induciranja	Trans, globoko sproščena stanja

Tabela 1: Delta možganski valovi

Theta (3–8 Hz)

Tudi klasificirana kot počasnejša možganska aktivnost. Povezavo lahko sklenemo s kreativnostjo, intuicijo, sanjarjenje in fantaziranjem. Zajema tudi spomine, čustva in občutke. Theta valovanje je možno izraženo ob molitvi, meditaciji in duhovnim zajemanjem. Lahko rečemo, da se pojavlja med budno zavestjo in spanjem.

Kadar je theta valovanje optimalno, omogoča prilagodljive in kompleksne strukture vedenja, kot sta učenje in spominjanje. Neravnovesje teh valov lahko nakazuje na prisotno bolezen ali stres. [4]

Akcija	**Opis**
Subjektivni občutki	Intuicija, kreacija, spomin, zaspanost
Vedenje in aktivnosti	Kreativnost, fokus
Fiziološke povezave	Zdravljenje, integracija uma in telesa
Učinki induciranja	Trans, globoko sproščena stanja

Tabela 2: Theta možganski valovi

Alfa (8–12 Hz)

Normalno alfa stanje omogoča hitro in učinkovito upravljanje opravil. V tem stanju se večina ljudi počuti sproščeno in umirjeno. Lahko bi rekli, da je to valovanje kot nekakšen most med zavestnim in nezavestnim.

Alfa stanje je povezano z ekstravertiranostjo, kreativnostjo (pri reševanju problema ali pri poslušanju) in ob mentalnem delu.

Kadar so alfa valovi v optimalnem obsegu, doživljamo dobra počutja, vidimo svet pozitivno in prežema nas občutek umirjenosti. To stanje je eno najbolj pomembnih, kadar se učimo in pri uporabi že naučenih informacij, kot sta delo in izobraževanje. [4]

Akcija	**Opis**
Subjektivni občutki	Sproščenost, umirjenost, zavest
Vedenje in aktivnosti	Meditacija, brez aktivnosti
Fiziološke povezave	Sproščenost, zdravljenje
Učinki induciranja	Sproščenost

Tabela 3: Alfa možganski valovi

Beta (12–38 Hz)

Valovanje je značilno za »hitre« aktivnosti. To valovanje jemljemo kot normalen ritem in je dominantno valovanje, kadar je oseba zbrana ali razburjena, pri odprtih očeh.

Valovanje se pojavlja tudi pri poslušanju, razmišljanju, med analitičnim reševanjem problemov, odločanjem, procesiranjem informacij itd.

To valovanje zaradi relativno širokega razpona delimo na nizko, srednje in visoko beta valovanje. [4]

Nizka beta (12–15 Hz)

Akcija	**Opis**
Področja	Spredaj ali zadaj
Subjektivno občutenje stanj	Sproščeno in zbrano, integrirano
Vedenje in aktivnosti	Pomanjkanje usmerjenje pozornosti
Fiziološke povezave	Gibanje
Učinki treniranja	Sproščena zbranost, pozornost

Tabela 4: Nizki Beta možganski valovi

Srednje Beta (15–18 Hz)

Akcija	**Opis**
Področja	Regionalno, po različnih področjih
Subjektivno občutenje stanj	Razmišljanje, zavedanje sebe in okolice
Vedenje in aktivnosti	Mentalna aktivnost
Fiziološke povezave	Buden, aktiven, ne razburjen
Učinki treniranja	Lahko poveča mentalne sposobnosti, zbranost, IQ

Tabela 5: Srednji beta možganski valovi

Visoko Beta (nad 18 Hz)

Akcija	Opis
Področja	Lokalno, lahko je zelo natančno omejeno
Subjektivno občutenje stanj	Budnost, naprezanje, razdraženost
Vedenje in aktivnosti	Mentalna aktivnost, matematika, načrtovanje
Fiziološke povezave	Splošna aktivacija uma in telesnih funkcij
Učinki treniranja	Lahko sproži budnost, ampak tudi razdraženost

Tabela 6: Visoki beta možganski valovi

Gama (38–42 Hz)

Je edinstveno frekvenčno valovanje, ki je prisotno po vseh delih naših možganov. Kadar morajo predelati določene informacije iz različnih delov, je prav frekvenca 40 Hz tista, ki združi potrebne predele možganov za istočasno obdelavo podatkov.

Kadar se dobro spominjamo nečesa, je to pri 40 Hz aktivnosti. [4]

Akcija	**Opis**
Subjektivni občutki	Razmišljanje
Vedenje in aktivnosti	Procesiranje informacij visoke gostote
Fiziološke povezave	Povezano je z informacijsko bogato obdelavo zaznavanja
Učinki treniranja	/

Tabela 7: Gama možganski valovi

2.2.1 Uporaba EEG

Medicina je glavna panoga uporabe EEG-signalov. Uporaba seže vse od diagnosticiranja epilepsije, motenj spanja in možganskih bolezni. Kadar preidemo do diagnosticiranja tumorjev in kapi, pa EEG nadomeščajo nove metode, kot so MRI, CT in PET, ki imajo boljšo topografsko ločljivost, vendar pa EEG odlikuje boljša časovna ločljivost. [5]

2.2.2 Premagovanje artefaktov in izboljšave meritev

Pri meritvah lahko ločimo dve vrsti artefaktov (popačitev) signala. Imamo biološke in tehnične popačitve. Omenili smo že, da do bioloških popačitev lahko pride zaradi trzanja ali premikanja mišic ob elektrodah. Te posledice sprožijo nastanek dipola, saj so ti mišični premiki (dipoli) močnejši kot pa dipoli EEG-signala, do tehničnih pa lahko pride iz več razlogov.

Te težave nam pomaga premostiti več identičnih meritev ob nekakšni miselni nalogi; ob pridobljenih podatkih odstranimo šum, da bi dobili obliko iskanega vzorca možganov. Pri tem imamo vedno enako referenčno točko, ki nam predstavlja nekak začetek dražljaja, od koder začnemo naše meritve spreminjanja signala. Kadar imamo na razpolago več elektrod, lahko izdelamo tudi topografsko mapo aktivnosti možganov. [5]

2.3 Vmesniki za branje možganov

Vmesniki za branje možganskih valov predstavljajo neposredno povezavo med možgani in računalnikom. Razlikujejo se v primerjavi z običajnimi vhodnimi napravami, da nam ni treba fizično premikati stvari, kot so poteg miške, pritisk tipkovnice ali dotik zaslona, ampak merijo signale, ki jih oddajajo možgani, ki morajo biti povezani z osebkovo namero, te pa lahko nato prevedemo v ukaz v napravi. Da bi razumeli te ukaze, mora računalnik pri obdelavi poiskati lastnosti signalov na posnetkih in ponavljajoče se vzorce.
Možganski vmesnik je sestavljen iz naslednjih glavnih komponent:

- merjenje signalov možganske aktivnosti,
- pošiljanje povratnih informacij uporabniku,
- delovanje sistema za razumevanje ukaza.

2.3.1 Vrste vmesnikov za branje možganov

Invazivni BCI

Pri tej metodi se elektrode vstavijo v možgane. Zaradi neposredne bližine elektrode in centra dogajanja aktivnosti je ta metoda najnatančnejša. Ta metoda se večinoma uporablja pri invalidih, da lahko upravljajo z različnimi umetnimi udi ali pripomočki. Vendar pa je takšna uporaba težavna in dolgotrajna, predvsem zaradi raznolikosti misli in reakciji ljudi, težave pa še povzročajo brazgotine v okolici elektrode, kadar telo začne reagirati na tujek v telesu, to pa bistveno poslabša signal, ki lahko celo izgine.

Polinvazivni BCI

Zahteva enako nevrokirurško operacijo, vendar se pa elektrode vsadijo le na površino možganov, ob tem pridobimo še vedno dobro kakovost signalov in manjšo verjetnost nastanka brazgotin.

Neinvazivni BCI

To metodo lahko štejemo med najcenejše in najvarnejše tipi uporabe BCI, ampak zaradi večje razdalje in vmesnih plasti med možgani in elektrodami dobimo slabše signale. Največja ovira za signale predstavlja ravno lobanjska kost. Komercialni produkti so namenjeni ravno tej metodi. [8]

2.4 Izdelava programske opreme za zajem podatkov

Cortex API je nov in vsestranski *API* za interakcijo z izdelki *Emotiv* (to so naprave za komercialno uporabo pri branju EEG možganskih valov), vključno z *Insight* (ki ga bomo uporabljali v nalogi), *Epoc* in *Epoc+*. *API Cortex* je zgrajen na *JSON* in *WebScokets*, kar olajša dostop do različnih programskih jezikov in platform. *Cortex* lahko uporabite za ustvarjanje iger in aplikacij, beleženje podatkov za poskuse in še več.

S to knjižnico lahko dobimo naslednje podatke iz naprave:

- EEG-podatke, ki so sestavljeni iz vsakega senzorja na napravi (5 senzorjev), tako da lahko dobimo vrednosti pulzov magnetnega valovanja na 5 različnih mestih, s katerih lahko razberemo, kateri del možganov je aktiven ob branju;

- gibalne (angl: *Motion)* podatke, ki predstavljajo vrednosti giroskopa, pospeškometra in magnetometra ob premikanju naprave na naši glavi;

- čustvene (angl: *Met*) podatke, ki so že vnaprej izračuni iz surovih podatkov, ti podatki pa predstavljajo procentualno vrednost od interesa, strese, relaksacije, navdušenja, zanimanja, dolgo ročnega zanimanja, fokusa.

Te tri vrste podatkov so najbolj koristne in uporabljene, seveda pa obstajajo še druge vrste podatkov, ki se tudi razlikujejo od modela naprave in naročnine, ki jo imamo.

Cilj aplikacije je zajem oddajanega magnetnega valovanja možganskih valov (EEG) ob prikazovanju različnih slik uporabniku na določenem intervalu. V praksi smo izdelali naslednji tok aktivnosti.

Uporabnika posedemo v umirjen prostor, v katerem si na svojo glavo nadene BCI-napravo, ki naj se čim bolj dotika kože na glavi, brez vmesnega posredovanja lasov. Ko je uporabnik pripravljen, izberemo 2 tipa različnih kolekcij slik. V našem primeru smo se odločili za slike narave in slike hrane, da dosežemo relativno veliko razliko v samem dojemanju, občutkih ob opazovanju teh slik. Vse te slike se po kategorijah in statičnem intervalu izmenjujejo na monitorju v celozaslonskem načinu, da kolikor se le da zmanjšamo motnje. Ob samem izvajanju aplikacije in opazovanju uporabnika se EEG-podatki iz naprave konstantno shranjujejo v *CSV*-datoteko na disku, v katerem so naslednji podatki:

ID Zaporedna št. vnosa

TimeStamp Ob katerem času je bil zabeležen podatek

Raw_cq Surova vrednost kakovosti signala

Af3 Kanal v Hz

T7 Kanal v Hz

P7 Kanal v Hz

T8 Kanal v Hz

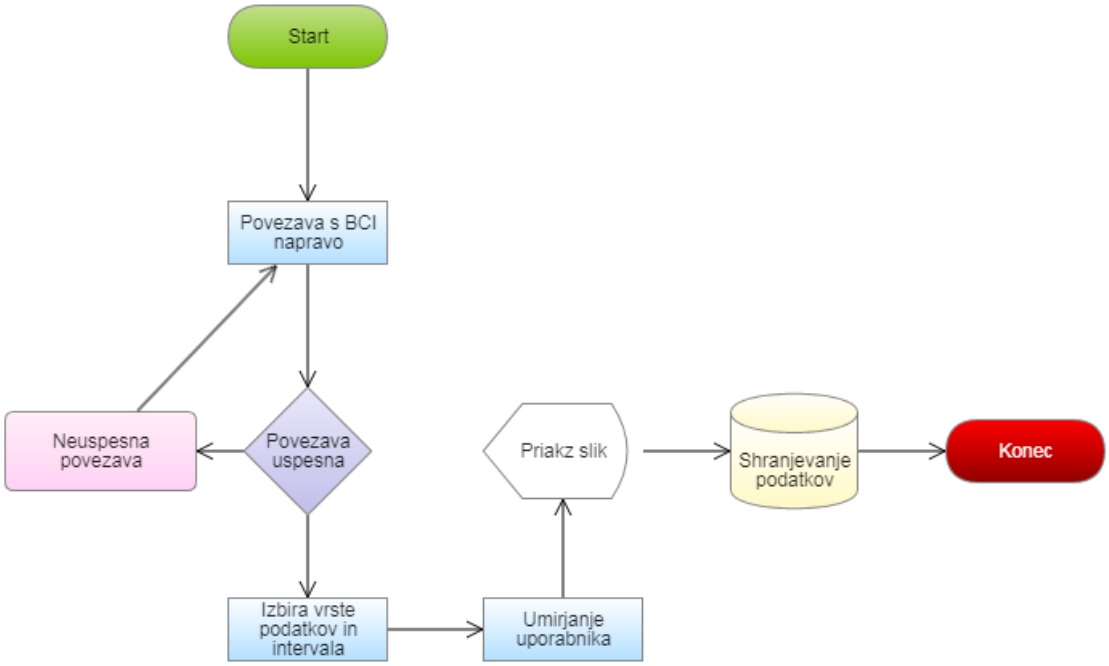
Af4 Kanal v Hz

Marker Označuje stanje uporabnika

- 0 Brez nadzora, lahko dela razmišlja kar koli
- 1 Ima zaprte oči
- 2 Ima odprte oči
- 3 Gleda slike narave
- 4 Gleda slike hrane

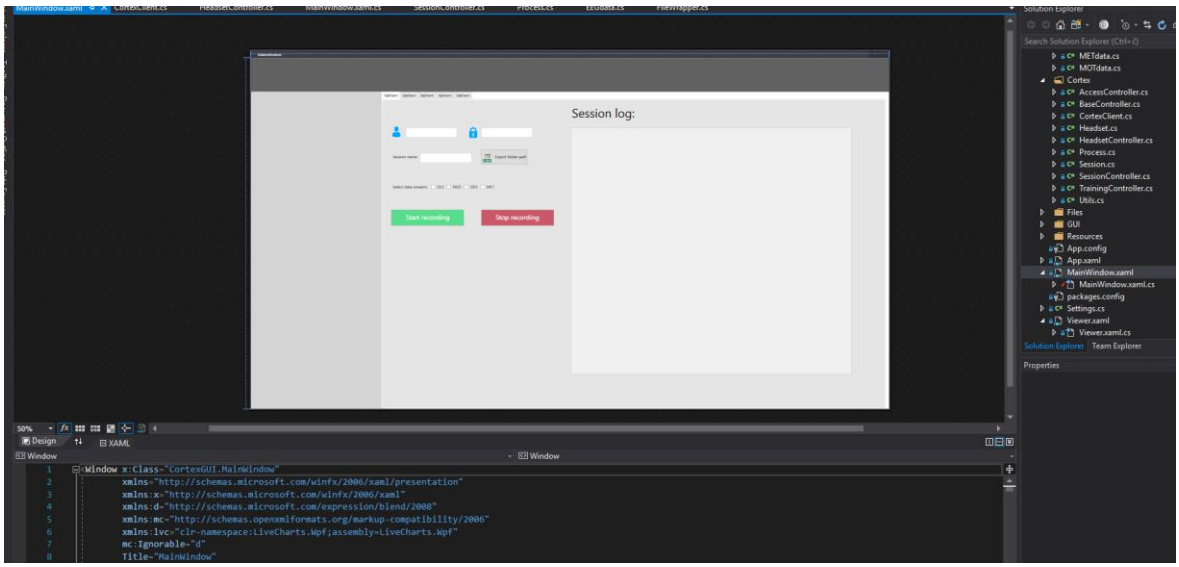
Te podatke smo uporabili za učenje nevronskega modela.

Slika 4 prikazuje diagram aktivnosti za zajemanje podatkov.



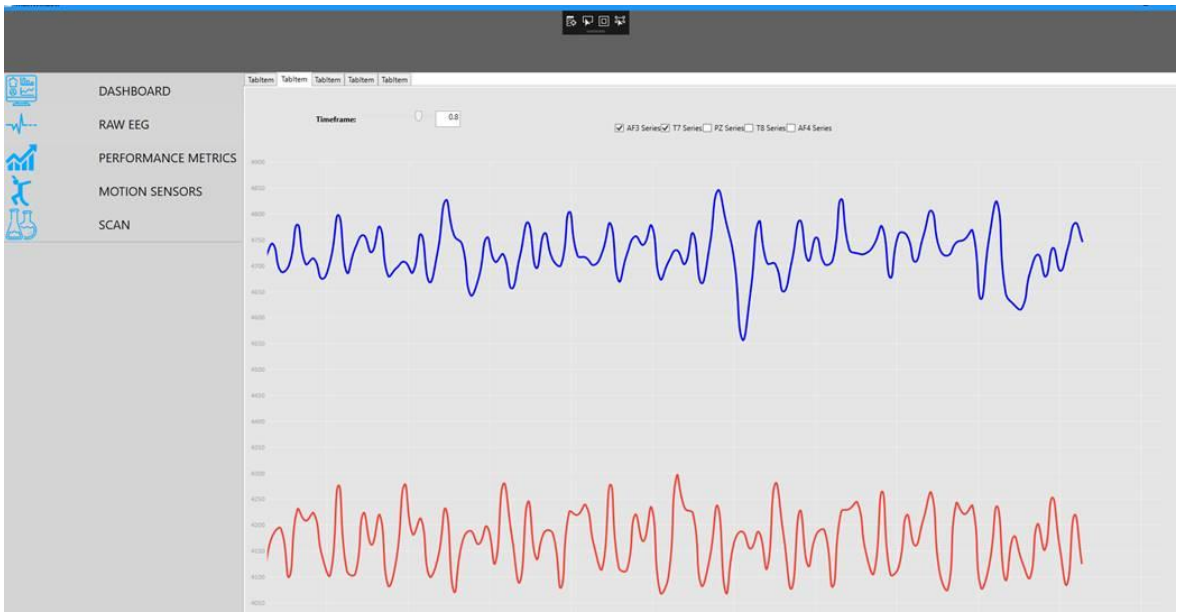
Slika 4: Diagram aktivnosti

Za čim lažjo uporabo naprave in samega skeniranja uporabnika smo izdelali namizno Windows aplikacijo v programskem jeziku C# kot projekt WPF.



Slika 5: Cortex GUI

V aplikaciji smo vgradili knjižnico *LiveCharts*, ki nam je omogočila preprosto in učinkovito prikazovanje grafov ob prejemanju podatkov iz naprave. Tako smo izdelali dva različna prikaza grafov. Prvi prikazuje prejemanje EEG-signalov po specifičnem kanalu.

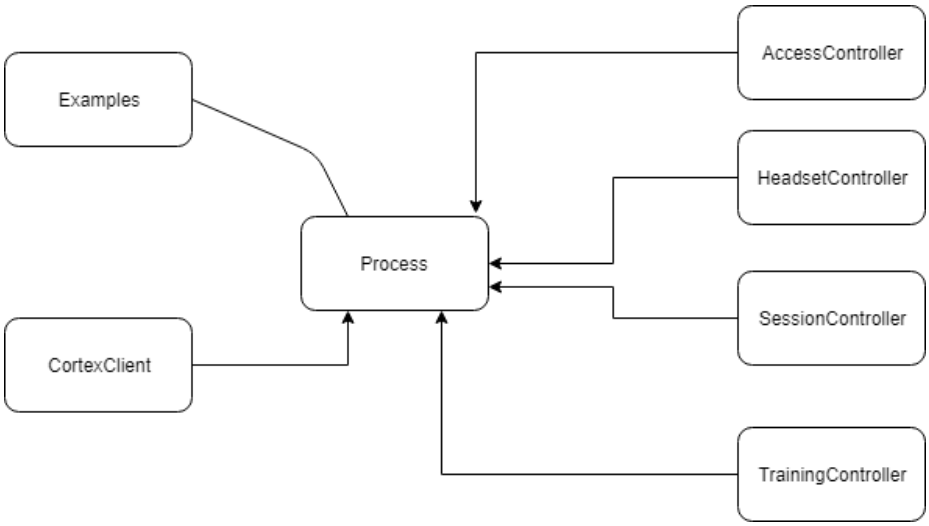


Slika 6: Graf aktivnosti EEG valov

Graf na sliki 7 nam prikazuje gibalne podatke (giroskop, pospešku meter, magnetometer). S temi podatki lahko zraven EEG-podatkov tudi spremljamo, kako je uporabnik premikal svojo glavo med prestajanjem poizkusa, da bi morebiti zavrgli podatke, če ni bil miren in ni gledal slike v monitorju.

Slika 7: Graf aktivnosti giroskopa

Naslednji diagram na sliki 8 prikazuje, kako so razredi med seboj povezani in kako komunicirajo med seboj. *CortexAcces* projekt je jedro in je zadolžen za procesiranje vseh zahtev in odgovorov z vmesnim nivojem med uporabnikom in korteksom.



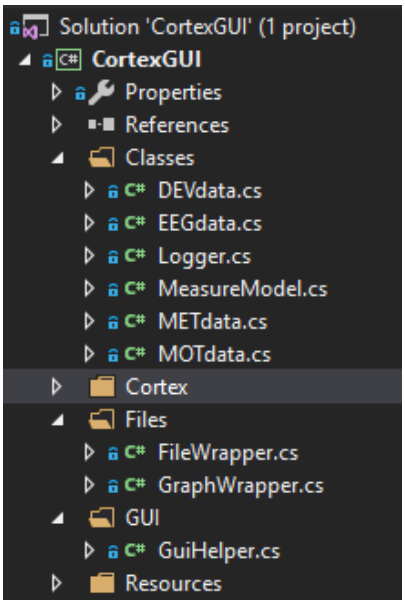
Slika 8: Prikaz strukture razredov

- CortexClient: je zadolžen za pošiljanje zahtev Cortexu in pridobivanje odgovorov, obvestil, podatkov od Cortexa.
- Process: procesira zahteve/pošilja odgovore in preklaplja med potmi do ustreznega kontrolorja.

    - AccessController: odgovoren za prijavo, avtentikacijo in generiranje žetona za uporabo knjižnice s Cortex oblakom.
    - HeadsetController: odgovoren za poizvedovanje po BCI-napravi, kakšna je povezava, vsakih 10 s bo tudi preverjal, ali je naprava še vedno na voljo ali ne.
    - SessionController: odgovoren je za odpiranje seje med napravo in storitvami Cortex, posodobitve, snemanje podatkov, vstavljanje markerja v podatke. Seja je vzpostavljena, kadar je naprava povezana, in prekinjena, kadar je povezava izključena.
    - TrainingController: odgovoren za ustvarjanje računa, posodobitve, nalaganje, shranjevanje in trening (tega razreda nismo uporabljali)

- Examples: vnaprej napisani programi za testiranje knjižnice.

Implementirali pa smo tudi razrede, ki nam pomagajo pri sami obdelavi, prikazovanju in shranjevanju podatkov.



Slika 9: Cortex GUI-razredi

DEV, EEG, MET, MOT podatki služijo za pretvarjanje surovih JSON-podatkov v objekte, da jih lahko nato s C# lažje uporabljamo. Vsak izmed njih ima svojega konstruktorja, ki prejme parametre iz odgovorov Cortexa. Definirali smo tudi *data annotations* za vsako lastnost, ki se bo preslikala v CSV-format.

MesureModel skrbi, da ima vsak podatek vrednost in časovni žig, kdaj je bil podatek prejet, da lahko nato te informacije prikazujemo na grafu.

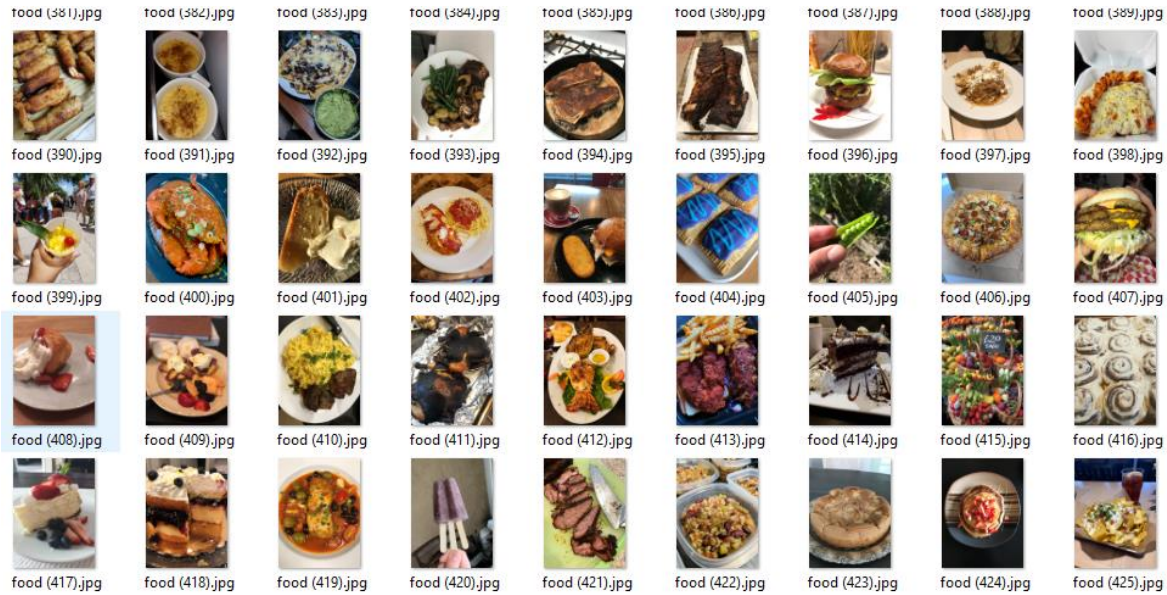
FileWrapper razred je namenjen pridobivanju podatkov iz naprave. Glede na tip podatka se jih dodaja na list podatkov. Ker pa te podatke dobivamo v intervalu 100 ms, se ta list hitro nakopiči. Zato imamo definirano metodo, ki ob določeni velikosti lista ustvari .csv datoteko in vanjo vpiše podatke. Ob vsaki naslednji zapolnitvi lista se podatki le pripišejo v datoteko.

GraphWrapper skrbi za pravilno prikazovanje izbranega tipa podatkov in da se prikaz osveži na nek interval prihoda podatkov.

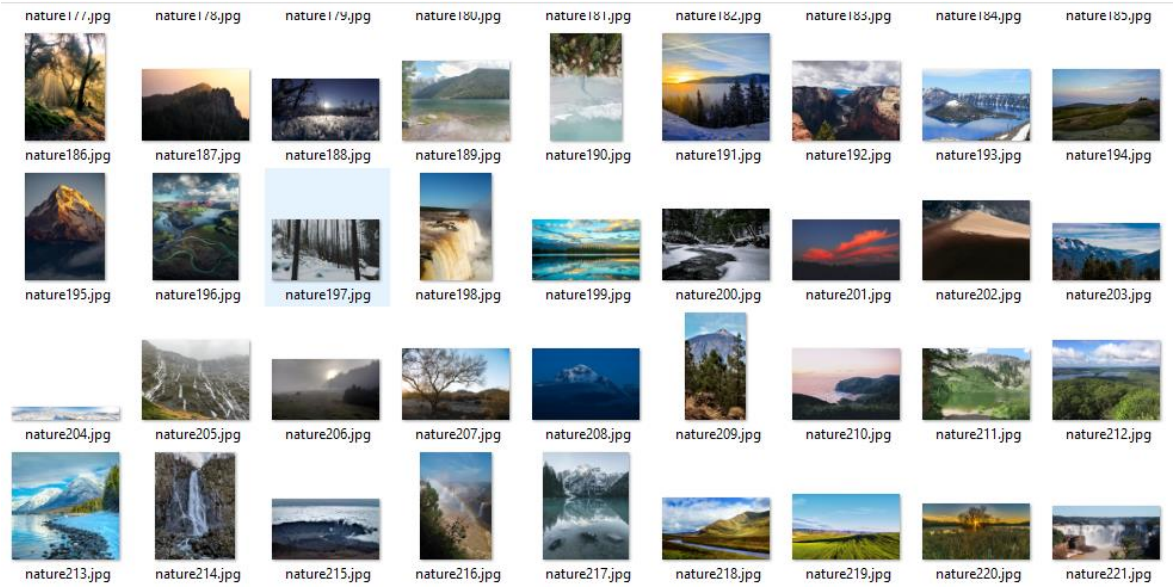
2.5 Priprava testnega okolja in zajem valov iz naprave

Da bi naredili kar se da dobre in kakovostne podatke, smo izločili vse šume iz okolice, svetloba je bila konstantno enaka in uporabnika smo limitirali na eno osebo. Prav tako je moralo lasišče biti sveže umito in ne sme vsebovati kakšnih kozmetičnih sredstev. Na elektrode smo predhodno tudi namestili poseben gel, ki povečuje prevodnost signalov.

Slike, ki so se prikazovale na monitorju, so bile vedno naključne iz kolekcije 721 slik narave in 771 slik hrane.



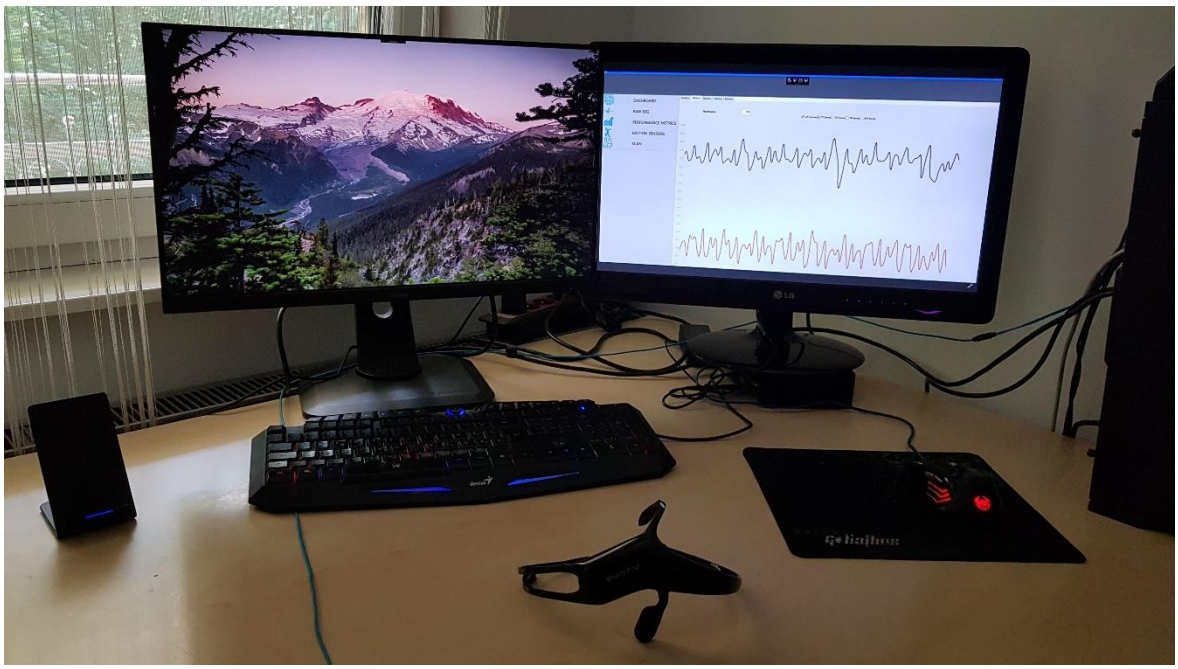
Slika 10: Kolekcija slik hrane



Slika 11: Kolekcija slik narave

Pridobivanje EEG-možganskih valov uporabnika je potekalo tako, da smo najprej opravili snemanje slik hrane v skupni dolžini ene ure. Seveda celotno snemanje ni bilo opravljeno naenkrat, ampak v razponu po 15 min, saj je težko ostati povsem miren in osredotočen na to, kaj se prikazuje na monitorju. Pri tem poskusu pa mora uporabnik ostati povsem zbran in razmišljati le o tem, kaj je na zaslonu. Vsi nenadni gibi ali plod podzavestnega razmišljanja o drugi tematiki pokvarijo kakovost podatkov.

Pridobivanje podatkov smo ponovili še s skupnim enournim snemanjem slik narave. Naslednja slika prikazuje, kako je potekalo snemanje možganov uporabnika: na levem zaslonu so se izmenjevale slike iz kolekcije, na desnem monitorju (ki naj bi bil namenjen opazovalcu snemanja) se pa je izrisoval graf toka EEG-podatkov iz naprave po kanalih.



Slika 12: Prikaz delovanja programa in BCI-naprave

Vsi ti podatki so se na intervalu 100 ms zapisovali v datoteko tipa CSV z vrednostmi posameznih kanalov (elektrod) in časovnim žigom ter kakšna vrsta akcije je bila izvedena pri uporabniku (ima zaprte / odprte oči, gleda slike narave / hrane ali prosto dejanje)

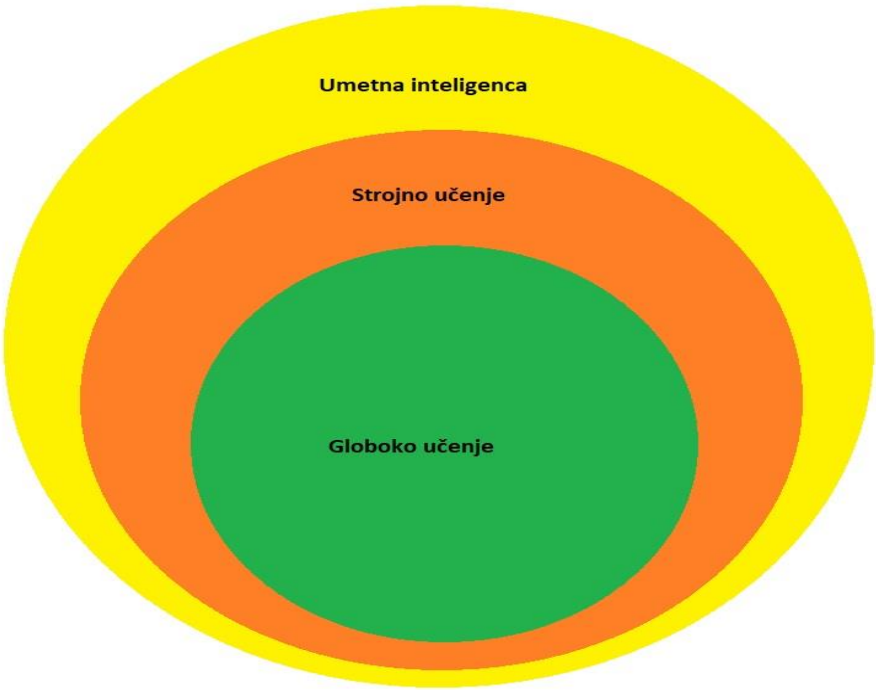
	A	B	C	D	E	F	G	H	I
1	ID	Timestamp	Raw_cq	Af3	T7	P7	T8	Af4	Marker
2	0	115	0	4279.487	4250.256	4254.872	4225.641	4291.795	0
3	1	230	0	4252.821	4209.231	4249.231	4165.641	4246.667	0
4	2	345	0	4267.692	4235.897	4192.308	4167.179	4249.231	0
5	3	460	0	4253.333	4229.744	4276.41	4169.744	4240.513	0
6	4	575	0	4262.564	4203.077	4183.077	4165.641	4246.154	0
7	5	690	0	4335.385	4231.795	4235.385	4207.179	4307.179	0

Slika 13: Prikaz posnetih EEG-podatkov

3 VPELJAVA STROJNEGA UČENJA

Globoko učenje je podzvrst strojnega učenja v sklopu umetne inteligence, ki se ukvarja z algoritmi izpeljanih iz biološke strukture in funkcionalnosti možganov, s katerimi nato vodimo stroj z nekakšno inteligenco.

Da bi si boljše predstavljali to strukturo, si poglejmo spodnjo sliko (Slika 14).



Slika 14: Struktura umetne inteligence

Velikokrat pride do pomote in se vsi ti koncepti obravnavajo kot eno, vendar so razlike velike. Razložimo si razliko na primeru: prepoznavanje obraza bi bila umetna inteligenca za prepoznavanje čustev na slikah, strojno učenje bi bilo namenjeno za vstavljanje na stotine slik s človeškim obrazom v sistem, globoko učenje pa bi prepoznalo vzorce v obrazih in čustva, ki se ponavljajo na slikah.

Umetna inteligenca	Strojno učenje	Globoko učenje
Mimika ali replikacija človeške inteligence	Dovoljuje strojem, da se učijo iz zbirke podatkov in ustvarjajo napovedi glede na scenarij.	Algoritmi v tem konceptu poskušajo modelirati visoko konceptne abstrakcije v podatkih, da bi ugotovili višji pomen. Je učinkovitejše v primerjavi s strojnim učenjem.

Tabela 8: Opis pojmov

Umetna inteligenca (AI) je zelo obsežno raziskovalno področje, kjer stroji kažejo kognitivne sposobnosti, kot so učna vedenja, proaktivna interakcija z okoljem, sklepanje, računalniški vid, prepoznavanje govora, reševanje problemov, predstavitev znanja, zaznavanje in mnogi drugi. AI označuje vsako dejavnost, kjer stroji posnemajo inteligentno vedenje, ki ga tipično prikazujejo ljudje. Umetna inteligenca črpa navdih iz elementov računalništva, matematike in statistike.

Strojno učenje (ML) je podskupina umetne inteligence, ki se osredotoča na poučevanje računalnikov, kako se učiti, brez potrebe programiranja za posebne naloge. Pravzaprav je ključna zamisel, da je mogoče ustvariti algoritme, iz katerih se učijo in napovedujejo iz podatkov. Obstajajo tri različne široke kategorije ML. Pri nadzorovanem učenju je stroj predstavljen z vhodnimi podatki in želenim rezultatom, cilj pa je, da se učijo iz teh primerov usposabljanja na tak način, da se lahko naredijo smiselne napovedi za sveže nevidne podatke. Pri nenadzorovanem učenju je stroj predstavljen samo z vhodnimi podatki, stroj pa mora poiskati neko smiselno strukturo brez zunanjega nadzora. Pri učenju z okrepitvijo stroj deluje kot posrednik, ki komunicira z okoljem in spozna, kakšna so vedenja, ki ustvarjajo želene/dobre rezultate.

Globoko učenje (DL) je posebna podskupina metodologij ML, ki uporabljajo umetne nevronske mreže (ANN), ki so rahlo navdihnjene s strukturo nevronov v človeških možganih. Beseda globoko se nanaša na prisotnost več plasti v umetni nevronski mreži, vendar se je ta pomen sčasoma spremenil. Medtem ko je pred štirimi leti 10 plasti že zadoščalo za upoštevanje globine omrežja, je danes običajno, da je omrežje globoko, če ima na stotine plasti.

Ko pridemo do globokega učenja, moramo vedeti, da obstaja več vrst strojnega učenja. V tabeli 15 si poglejmo razlike med vrstami strojnega učenja.

	Supervised learning	Unsupervised learning	Reinforcement learning
Podatki	Podatki za učenje vsebujejo napovedovalce in napovedi.	Podatki za učenje vsebujejo le napovedovalce	Iz podatkov se lahko spozna osnova in se iz tega dalje uči.
Algoritem	Linearna in logistična regresija, metoda podpornih vektorjev, naivni Bayes …	Algoritmi gručenja, Algoritmi za redukcijo dimenzije.	Q-Learning, State-Action-Reward-State-Action (SARSA), Deep Q Network (DQN)
Uporaba	Prepoznava slik, govora in napovedovanja	Preprocesiranje podatkov in iskanja logično podobnih podatkov.	Skladišča, vodenje inventarja, upravljanje dostave, finančni sistemi.

Tabela 9: Vrste strojnega učenja

3.1 Pregled ustreznih tehnik za izdelavo modela

Kadar pride do odločitve, za katero ogrodje se bomo odločili pri implementaciji modela, moramo najprej vedeti, da se ogrodja delijo predvsem na dva tabora. Tako da imamo na voljo nižje- ali višje nivojska DL-ogrodja. Ta izbira je ključnega pomena, odločitev, ki jo sprejmemo, pa vpliva na številne dejavnike in potrebna znanja za uporabo le-te. Vsaka izbira ima svoje prednosti in slabosti, večinoma pa bi lahko razdelili ta dva tabora na naslednje lastnosti.

3.1.1 Nižje nivojska DL-ogrodja

Namenjeni so predvsem za profesionalno in strokovno uporabo izdelave nevronskih modelov, kjer stvari temeljijo na nižjenivojski implementaciji, kar v praksi pomeni, da moramo zelo dobro poznati vse komponente, ki jih potrebujemo za izdelavo. Pri tej izbiri imamo proste roke. Primeri teh ogrodij so predstavljeni v nadaljevanju.

Theano[1]

Gre za enega izmed prvih ogrodij za DL-knjižnice. Je odprtokodna Python knjižnica. Prednosti uporabe te knjižnice so:

- Testna integracija z NumPy,
- transparenta uporaba grafične kartice,
- učinkovito simbolično razlikovanje (Theano naredi vaše derivate za funkcije z enim ali več vhodi).

Torch[2]

Prav tako popularno ML- in DL-ogrodje, napisano v Lua programskem jeziku. Prvotno so ga spisali trije inženirji, nato pa ga je izboljšal Facebook z raznimi dodatki, kot je odprtokodna programska oprema.

PyTorch[3]

Je odprtokodna ML- in DL-knjižnica za Python, ki so jo izdelali programerji v Facebook AI raziskovalni skupini. Postala je bolj popularna kot Torch, saj lahko že z osnovnim razumevanjem Pythona začnemo s programiranjem DL-modelov. PyTorch je veliko lažji, pregleden in uporaben za razvoj DL.

[1] http://deeplearning.net/software/theano/.
[2] http://torch.ch/.
[3] https://pytorch.org/.

TensorFlow[4]

Je nedvomno najbolj popularno in razširjeno DL-ogrodje. Izdelano je bilo kot odprtokodna rešitev Googla. Ogrodje podpira uporabo modela na procesorju, grafični kartici, mobilnih napravah na robu. Prednost uporabe teh ogrodij je fleksibilnost pri načrtovanju modela.

3.1.2 Višje nivojska DL-ogrodja

Prejšnja omenjena ogrodja so prvi nivo abstrakcije za DL-model. Čeprav lahko z njimi tako rekoč naredimo, kar koli želimo, vseeno zahtevajo veliko znanja in pisanja programske kode za izdelavo DL modela, a vseeno veliko manj, kot pa bi vse skupaj pisali samo v Pythonu ali C++.

Da bi poenostavili proces izdelovanja DL-modelov, imamo ogrodja, ki uporabljajo drugi nivo abstrakcije. Nižje nivojska DL-ogrodja lahko uporabimo kot *backend,* nad katerimi pišemo kodo v višje nivojskih DL-ogrodjih, kar pa še dodatno poenostavi razvoj.

Keras[5]

Najbolj uporabljen višjenivojski nevronski model API je napisan v Pythonu. Prednost uporabe le-tega je enostavnost izdelave modela. Internetna skupnost je zelo velika, kar pa nam omogoča učenje te knjižnice na veliko različnih primerih. Omogoča uporabo več različnih *backend* ogrodij. Med najbolj uporabljenimi kombinacijami sta Keras in TensorFow za *backend*. V praksi to pomeni, da lahko kodo napišemo v Kerasu, le-ta pa je prevedena v TensorFlow različico.

Gluon[6]

Obratuje nad MxNet. Izdelala sta ga AWS in Microsoft. Gluon skuša skrajšati čas in kompleksnost tega procesa usposabljanja tako, da tesno integrira model nevronske mreže z algoritmom usposabljanja. Ta struktura lahko razvijalcem pomaga tudi pri ustvarjanju in

[4] https://www.tensorflow.org/.
[5] https://keras.io/.
[6] https://gluon.mxnet.io/.

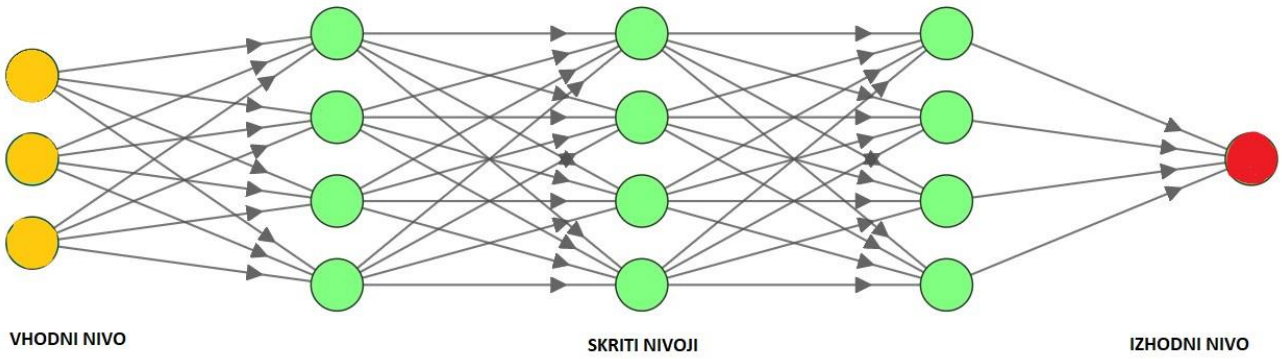
konfiguriranju bolj izpopolnjenih globokih modelov učenja ter racionalizaciji procesa razhroščevanja in posodabljanja za svoje nevronske mreže.

Lasagne[7]

Je lahka knjižnica za izgradnjo in usposabljanje nevronskih mrež v Theanu, zadnje čase njena uporabljenost sovpada s padcem Theano knjižnice.

Če bi hoteli izpostaviti eno značilnost iz vseh naborov različnih knjižnic, bi lahko rekli, da vsi poskušajo doseči nekakšen nivo izdelavo nevronskega modela, ki bi bil kar se da lahek, preprost in časovno ne potraten, saj vse znanje, ki ga potrebujemo za izdelavo takšne naloge, obsega različne tematike znanja, ki pa so vse prej kot lahke. Torej lahko govorimo o borbi, kdo bo iznašel najboljši način implementacije modela, da bi lahko bil uporabljen v globalni skupnosti razvijalcev.

Za razumevanje postopka strojnega učenja moramo najprej razumeti, kaj sploh predstavlja model in iz katerih komponent je sestavljen, da lahko svojo nalogo upravi tako, kot smo ga načrtovali. Sedaj ko imamo razumevanje različnih ogrodij za DL, lahko pogledamo, zakaj je Keras tako poseben in v prednosti v primerjavi z ostalimi. Prav zaradi njegovih lastnosti ga bomo uporabljali v našem projektu.

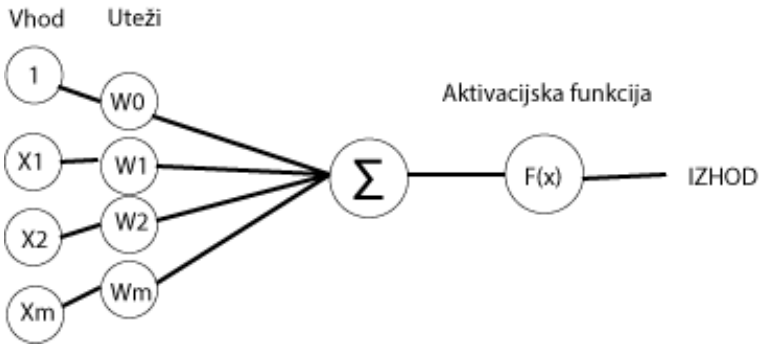


Slika 15: Nevronska mreža

[7] https://github.com/Lasagne/Lasagne.

Na sliki 16 vidimo osnovno zgradbo nevronskega modela, da si lahko bolje predstavljamo njegovo delovanje.

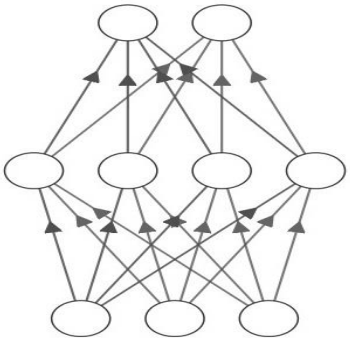
Posamezni nivoji so zgrajeni iz vozlišč (nevron). Nevron je mesto kjer se zgodi vso računanje, primerjava z možganskimi nevroni kjer ko pride do določena dražljaja se le ta sproži. Nevron združuje vhodne podatke z nizom koeficientov ali uteži, ki bodisi ojačujejo ali ublažujejo ta vhod, s čimer dodeljujejo pomen vnosom glede na nalogo, ki jo algoritem poskuša naučiti; npr. kateri vnos je najbolj koristen pri razvrščanju podatkov brez napak. Ti vhodni produkti se seštevajo in nato se vsota prenese skozi tako imenovano aktivacijsko funkcijo vozlišča, da se ugotovi, ali in v kolikšnem obsegu mora signal napredovati po mreži, da bi vplival na končni rezultat: npr. če signali preidejo skozi, je
nevron »aktiviran«.



Slika 16: Vizualizacija delovanja nevrona

Teža predstavlja moč povezave med enotami. Če je teža od vozlišča 1 do vozlišča 2 večja, to pomeni, da ima nevron 1 večji vpliv na nevron 2. Teža zmanjšuje pomen vhodne vrednosti. Teža blizu nič pomeni, da sprememba tega vhoda ne bo spremenila izhoda. Negativne uteži pomenijo, da povečanje tega vnosa zmanjša rezultat. Teža določa, koliko vpliva ima vhod na izhodu.

Nivoji so skupek nevronov oz. so kot stikala, ki se prižigajo in izklapljajo, ko prihajajo novi podatki skozi vhod. Izhod vsakega nivoja je hkrati tudi vhod v naslednji nivo.

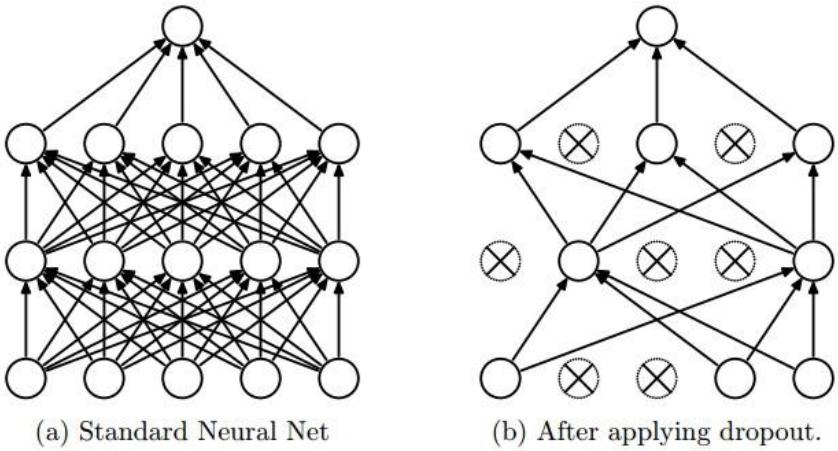


Slika 17: Vizualizacija nivojev

Združevanje nastavljivih uteži modela z vhodnimi značilnostmi pomeni, kako pripisujemo pomen tem funkcijam, glede na to, kako nevronska mreža klasificira naše podatke.

Dense nivo je običajen DNN-nivo, ki povezuje vsak nevron v definiranem nivoju na vsak nevron v prejšnjem nivoju. Na primer, če ima prvi nivo pet nevronov in drugi nivo (*dense* nivo) tri nevrone, je skupno število povezav med prvim in drugim nivojem 5 x 3, kar predstavlja vse možne povezave med prvim in drugim nivojem.

V DL *dropout* nivo pomaga zmanjšati prenapolnitev z uvajanjem možnosti regularizacije in generalizacije v model. V dobesednem pomenu *dropout* nivo izpusti nekaj nevronov ali jih nastavi na 0 in zmanjša izračunavanje v fazi učenja. Proces poljubno padajočih nevronov dobro deluje pri zmanjševanju prenapetosti.



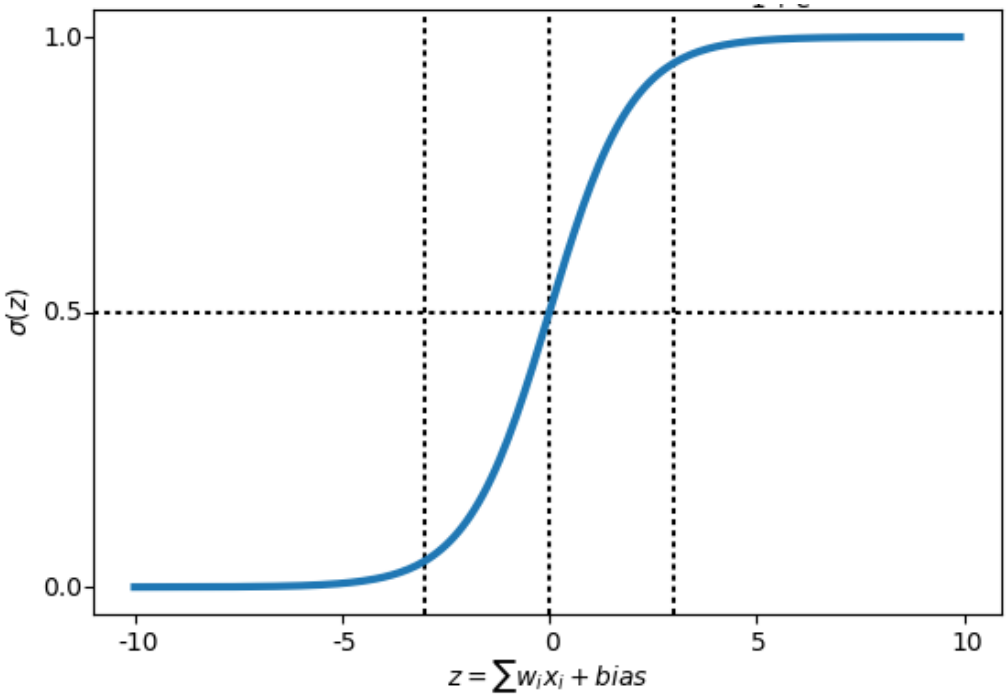
Slika 18: Vizualizacija, kako deluje *dropout* nivo [8]

Aktivacijska funkcija določa stanje nevrona z izračunom aktivacijske funkcije na podlagi združenih vhodov. Pomembna opomba je, da rezultat funkcije ne more biti 0, sicer DNN nima smisla. Pri izbiri aktivacijske funkcije imamo več možnosti, najpopularnejši izbiri pa sta:

- Simgoid aktivacijska funkcija

  Je matematična funkcija, ki ima karakteristiko »S« oblike. Tako uporabljena v strojnem učenju je predvsem zato, ker pri neki določeni vrednosti X postopoma preslika vrednost Y, kar je zelo praktično pri klasificiranju podatkov, za katere vemo, da imajo zgolj dva pomena. Izhod je omejen na vrednosti med 0 in 1.

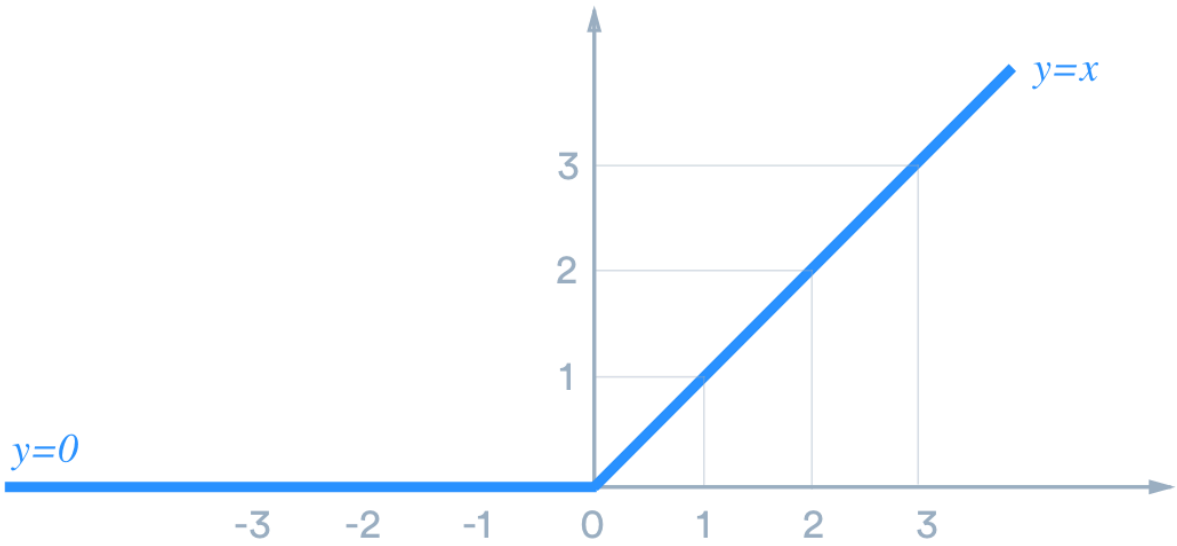
$$f(x) = \frac{1}{1 + e^{-x}}$$



Slika 19: Graf Sigmoid funkcije [9]

-ReLU-aktivacijska funkcija

Je sestavljena iz funkcije F(z) = max (0,z), kar pomeni da če je rezultat pozitiven, bo izpisala isto vrednost, sicer pa bi izhod bil 0. Na spodnji sliki si lahko ogledamo graf funkcije.



Slika 20: Graf ReLU funkcije [9]

Funkcija lahko na prvi pogled izgleda kot linearna funkcija, vendar ni. ReLU je nelinearna funkcija in dejansko deluje kot aktivacijska funkcija. Izboljša zmogljivosti in tudi bistveno pomaga pri zmanjševanju izračunov med fazo učenja. To je neposredna posledica vrednosti 0 v izhodu, ko je Z negativen, s čimer se deaktivira nevron. Zaradi horizontalne črte z rezultatom 0 lahko pride do resnih težav. Namreč, kadar odvajamo horizontalno črto, dobimo odvod 0, ki pa predstavlja ozko grlo v fazi učenja, kajti uteži ne bomo mogli enostavno prilagoditi. Da bi se izognili temo problemu, uporabimo novo aktivacijsko funkcijo Leaky ReLU, kjer negativne vrednosti predstavljajo rezultat s črto z rahlim klancem (prej s horizontalno črto), kajti s tem rezultatom lahko vsaj malo spremenimo uteži za naslednji cikel učenja s tako imenovanim izrazom *Back-propagation*

Leakly ReLU je definirana kot:

-f(z) = z; ko je z > 0,

-f(z) = ∝z; ko je z < 0 in ko je ∝ parameter definiran kot majhna konstanta, recimo 0.003.

Celotna struktura DNN je razvita z uporabo objekta (model) v Kerasu. To omogoča preprost način za ustvarjanje kupa plasti z dodajanjem novih plasti eno za drugo. Najlažji način za definiranje modela je z uporabo sekvenčnega modela, ki omogoča enostavno ustvarjanje linearnega niza plasti. Naslednji primer prikazuje ustvarjanje enostavnega sekvenčnega modela z enim slojem, ki mu sledi aktivacija. Sloj bi imel 10 nevronov in bi prejel vhod s 15 nevroni in se aktiviral z aktivacijsko funkcijo ReLU.

```
1. from keras.models import Sequential
2. from keras.layers import Dense, Activation
3. model = Sequential()
4. model.add(Dense(10, input_dim=15))
5. model.add(Activation('relu'))
```

Izvorna koda 1: Primer definiranja sekvenčnega modela

Funkcija izgube je matematična funkcija, ki meri izgubo rezultata do cilja s trenutnimi nastavljenimi parametri (teža in bias). Najbolj priljubljene funkcije izgube so:

- Povprečna kvadratna napaka

  Povprečna kvadratna razlika med dejansko in predvideno vrednostjo rezultata. Razlika na kvadrat omogoča lažje kaznovanje modela za večjo razliko. Torej bi razlika 3 pomenila izgubo 9, razlika 9 pa bi pomenila izgubo 81.

$$\sum_{n=1}^{k} \frac{(Dejansko - Predvideno)^2}{k}$$

- Povprečna absolutna napaka

  Povprečna absolutna napaka med dejanskim in napovedanim.

$$\sum_{n=1}^{k} |Dejansko - Predvideno|$$

- Binarna navzkrižna entropija

  Definira izgubo, ko so kategorični rezultati binarna spremenljivka, to je z dvema možnima izidoma: (Prestalo / Odpovedalo) ali (Da / Ne)

  Izguba = − [y * log(p) + (1−y) * log(1−p)]

Kategorična navzkrižna entropija

Definira izgubo, ko je kategorični rezultati ne binarni, to je x > 2
Možni rezultati: (Da / Ne / Mogoče) ali (Tip 1 / Tip 2 /… Vrsta n)

$$Izguba = -\sum_{i}^{n} y_i' log_2 y_i$$

DL je iterativen proces, ki temelji na veliko funkcijah in parametrih. Parametri morajo biti učinkovito nastavljeni, da lahko dobimo vedno boljše rezultate. Da bi to dosegli, si pomagamo s t. i. optimizacijskimi funkcijami, ki omogočajo, da se naš model na nek način »uči« z vsako iteracijo.

To so matematične funkcije, ki uporabljajo derivate, delne derivate in verižno pravilo, da bi lahko uvideli, koliko sprememb bo model imel, če naredimo malo spremembo v utežeh nevronov. Sprememba v funkciji izgube, ki je lahko povečanje ali zmanjševanje, pomaga pri določanju smeri zahtevane spremembe uteži.

Dva najpomembnejša optimizacijska algoritma:

- Stochastic Gradient Descent (SGD)

  SGD opravi iteracijo z vsakim novim učnim vzorcem, kar pomeni, da se po vsaki predaji vzorca skozi model izračuna izguba, ki ji sledi posodobitev uteži.

  Zaradi tega postopka se uteži preprosto prevečkrat posodabljajo, kar pa predstavlja zelo omotično izgubno krivuljo. Je pa sama optimizacija relativno hitra v primerjavi z drugimi optimizatorji.

  Uteži = Uteži – Stopnja učenja * Izguba
  (Stopnja učenja je parameter ob definiranju nevronske arhitekture)

- Adam

  (Angl: *Adaptive moment estimation*) je daleč najbolj uporabljen popularen in razširjen optimizator v DL. Ta tehnika izračuna prilagodljivo stopnjo učenja za vsak parameter z načinom, da dodaja moment v izračun in varianco od izgube, s tem izkoristi efekt dvojice za posodabljanje vrednosti uteži.

  Uteži = Uteži – (Moment in varianca združena)

Paket podatkov (angl. batch) in epoha učenja (angl. epoch)

Paket podatkov je zbirka vzorcev faze treninga iz celotnega vnosa. DNN posodobi svoje uteži po obdelavi vseh vzorcev v paketu. To se imenuje ponovitev (tj. kadar so uspešno prestali test vsi vzorci v paketu, ki mu sledi posodobitev uteži v DNN). Izračunanje vseh vzorcev treninga, ki so na voljo v vhodnih podatkih, s posodobitvami uteži po batch-by-batch, se imenuje epoh. V vsaki ponovitvi DNN izkorišča funkcijo optimizatorja za majhno spremembo uteži (ki so bile na začetku naključno inicializirane), da bi izboljšali napoved z zmanjšanjem funkcije izgube.

3.2 Analiza in priprava modela

3.2.1 Pregled podatkov

Za začetek izdelave nevronskega modela, ki bo pripomogel k prepoznavanju podatkov za naša dva scenarija (uporabnik gleda sliko narave ali narave), smo se namenili zgraditi preprost nevronski model s pomočjo knjižnice Keras. S to knjižnico bistveno poenostavimo izdelavo in pripravo modela, kajti vse, kar moramo pripraviti pred izdelavo modela, je le način kategorizacije podatkov, predno bodo prišli v naš vhodni nivo v nevronskem modelu.

Pridobljen nabor podatkov, ki smo ga dobili z uporabo naše GUI Cortex namizne aplikacije, pred obdelavo izgleda tako (v *CSV*-obliki):

```
Af3,T7,P7,T8,Af4,Marker
4480,4196.923,3937.949,4074.872,4415.897,0
4540,4217.436,3938.974,4090.256,4473.846,0
4554.872,4237.949,4037.436,4129.744,4477.949,0
4520,4225.641,4315.385,4129.231,4445.128,0……
```

Izvorna koda 2: Posneti podatki iz Emotiv Insight naprave

Vsi naslednji podatki si sledijo tako, kot so tudi bili posneti. Da bi ob učenju nevronskega modela dobili boljše rezultate, smo podatke prebrali v list in jih premešali. Ob tem pa smo še podatke razdelili v dve skupini po dva lista.

Prva skupina (80 % podatkov) je torej namenjena treningu modela; ta kopica podatkov pa se še deli na dva lista: prvi služi vrednostim podatkom (EEG), drugi pa pripadajoči rezultat (slika narave ali hrane).

Druga skupina (20 % podatkov) pa se na enak način deli, vendar je namenjena samemu testiranju modela kasneje.

3.2.2 Izdelava DNN-modela

S pomočjo knjižnice smo izdelali naslednji nevronski model. Tip modela, ki smo ga izdelali, se imenuje sekvenčni model in omogoča kreiranje več nivojev enega za drugim. Omejen je s tem, da ne omogoča kreiranja modelov, s tem, da si delijo različne nivoje med seboj, ali da ima več vhodov in izhodov, kakor je to značilno v funkcijskem modelu.

V vhodnem nivoju je zahtevano, da definiramo dimenzijo podatkov v prvem nivoju, kajti na začetku model ne more vedeti, kateri podatki bodo prišli na vhod, za vse ostale nivoje pa to ni potrebno, saj se avtomatsko ugotovi.

```
1. model = Sequential()
2.  model.add(Dense(5, input_dim=5, init='normal',
3.  activation='relu'))
4.  model.add(Dropout(0.2))
5.  model.add(Dense(3, init='normal', activation='relu'))
6.  model.add(Dropout(0.2))
7.  model.add(Dense(1, init='normal', activation='sigmoid'))
```

Izvorna koda 3: Definiranje nivojev DNN-modela

Za inicializacijo uteži smo uporabili normalno distribucijo, ki nam je inicializirala uteži glede na rezultate funkcije. Aktivacijskem parametru smo definirali dobro uveljavljeno in relativno preprosto *RELU*-funkcijo v vhodnem in skritem nivoju modela. Za izhodni nivo pa smo morali definirati *Sigmoid* funkcijo, ki nam omogoča, da dobimo rezultat med 0 in 1. Med posamezne nivoje smo dodali tudi t. i. *dropout* nivoja, ki služi namenu, da so naključno izbrani nevroni izpuščeni v fazi učenja. To pomeni, da so rezultati aktivacijske funkcije pri *forward pass* odstranjeni, kakor tudi vsaka sprememba uteži ni uporabljena pri *backward pass*. Vsi ti postopki pripomorejo k temu, da model generaliziramo.

Ko smo končali z definiranjem modela, smo model še morali kompilirati s funkcijo, ki bo definirala izgubo (angl. *loss*) optimizatorjem, ki bo znal posodobiti uteži glede na rezultat in

funkcijo, ki nam bo vrnila rezultat, s kakšno natančnostjo v odstotkih, je model napovedal rezultat s trenutnimi parametri (uteži in bias).

```
model.compile(loss="binary_crossentropy",optimizer="adam",
metrics=['accuracy'])
```

Izvorna koda 4: Definiranje argumentov za kompilacijo modela

V naslednjem koraku smo morali še nastaviti parametre, kako se bo model učil.

```
model.fit(x_train,y_train,batch_size=124,epochs=1000,
validation_data=(x_test, y_test))
```

Izvorna koda 5: Definiranje argumentov za učenje modela

Kot prva dva parametra smo poslali podatke, ki so predvideni za učenje skupaj z rezultati, prav tako pa smo definirali, koliko bomo imeli velikosti paketov podatkov v eni epohi. V praksi je to pomenilo več ali majn, da smo večkrat zagnali učenje z različnimi vrednostmi velikosti paketa podatkov in epohe, kot validacijske podatke, ki smo na začetku predvideli za testiranje. Spomnimo se delitve podatkov v skupini 80-20.

Za ugotovitev, kako dobro se je obnesel naš nevronski model, smo še uporabili naslednjo funkcijo, ki vrne vrednosti izgube in pregledano vrednost (angl. *metric value*) za model v testiranju.

```
model.evaluate(x_test, y_test)
```

Izvorna koda 6: Klicanje funkcije, ki oceni, kako točen je model

Iz rezultatov smo dobili rezultat, ki je pokazal, da je naš model pri predvidevanju rezultat dosegel točnost 55 %, kar predstavlja slabi rezultat, saj bi pri tej vrednosti lahko rekli, da je uporaba našega modela neuporabna.

3.2.3 Razlaga RNN-modela

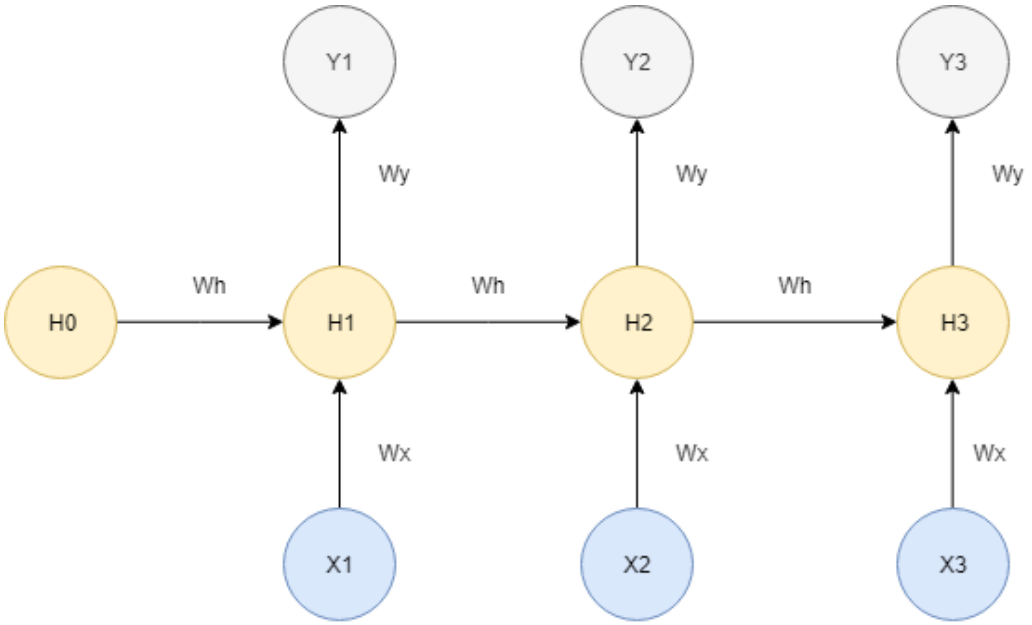
Ta tip nevronskega modela si lahko lažje predstavljamo tako, da pomislimo, kako delujejo naše misli, te se vedno navezujejo na naše prejšnje misli, lahko bi rekli, da nikoli ne začnemo razmišljati iz ničle. Tako nekako deluje tudi RNN-tip nevronskega modela, ki ob vsaki iteraciji ne začne povsem znova. Tradicionalni DNN ne počnejo tega, kar pa predstavlja pomanjkljivost.

Naši podatki o prebranih možganskih valovih ob dani sliki so kot seznam, v katerem lahko ob danem trenutku preko različnih kanalov (Af3, T7, P7, T8, Af4) razberemo, katera slika je bila prikazana uporabniku. Vendar pa nam te vrednosti ne morejo zagotavljati visoke zanesljivosti rezultata, saj je bistveno preveč faktorjev bilo prisotnih ob snemanju podatkov, kot so lahko ne zanesljivost elektrod, dekoncentracija uporabnika, nenadni premiki glave uporabnika, fiziološki procesi v telesu ipd. Zaradi teh vzrokov bi bilo mogoče bolje, da bi ta iskani rezultat želeli pridobiti skozi več zapisov podatkov, kajti znotraj teh zapisov lahko določimo vzorec v naših podatkih, ki bi nam mogoče dal bolj zanesljiv rezultat. Kajti tudi ob prisotnosti zunanjih faktorjev, ki motijo našo zanesljivost podatkov, se skozi veliko količino zapisov te vrednosti omejujejo na dano zalogo vrednosti.

3.2.4 Podrobno spoznavanje delovanja RNN-modela

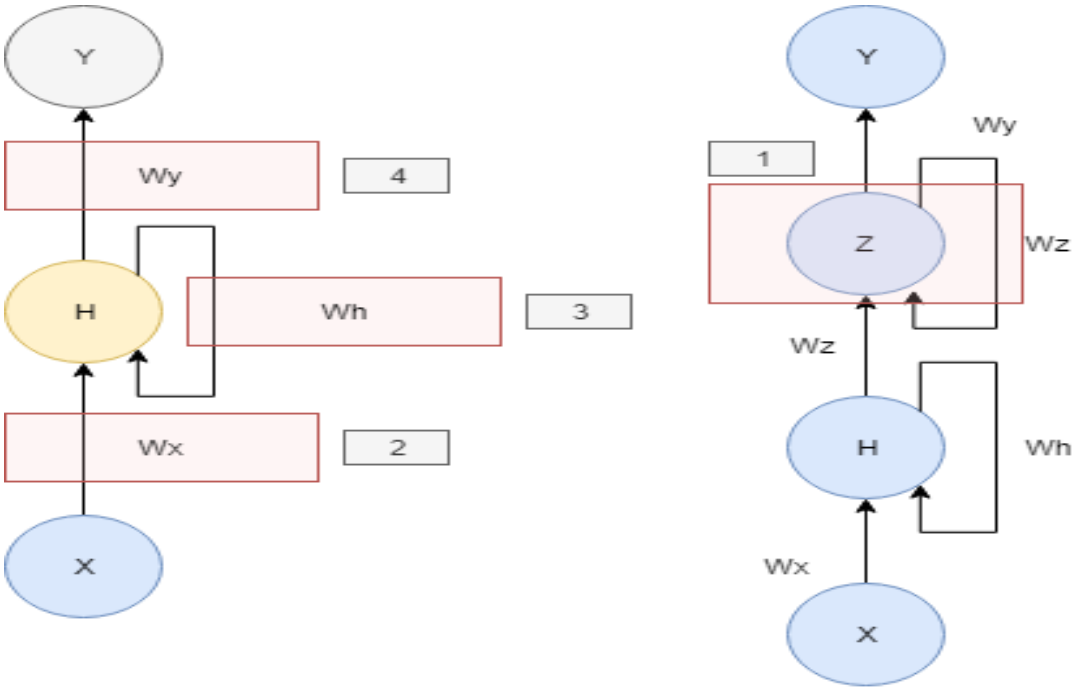
Nevronska mreža s povratno zanko si zapomni iz preteklosti (prejšnje iteracije) naučeno stanje in ga uporabi za prihodnost (naslednjo iteracijo). Tukaj se moramo spomniti, da tudi navadni DNN uporabi svojo naučeno stanje za v prihodnje, vendar to naučeno znanje izhaja iz celotnega predhodno dokončanega treninga.

Medtem pa RNN deluje enako, vendar še za dodatek si zapomni, stanje, ki je bilo naučeno iz prejšnjega vhoda, medtem ko se generira izhod. RNN lahko sprejme enega ali več vhodnih vektorjev in ustvari enega ali več izhodnih vektorjev, na izhod pa vplivajo ne le uteži, ki se uporabljajo na vhodih kot pri običajnem DNN, temveč tudi »skrit« vektor stanja, ki predstavlja kontekst, ki temelji na predhodnih vhodih, izhodih. Torej, isti vhod lahko ustvari drugačen izhod, odvisno od predhodnih vhodov v seriji.



Slika 21: Skriti vektor v RNN

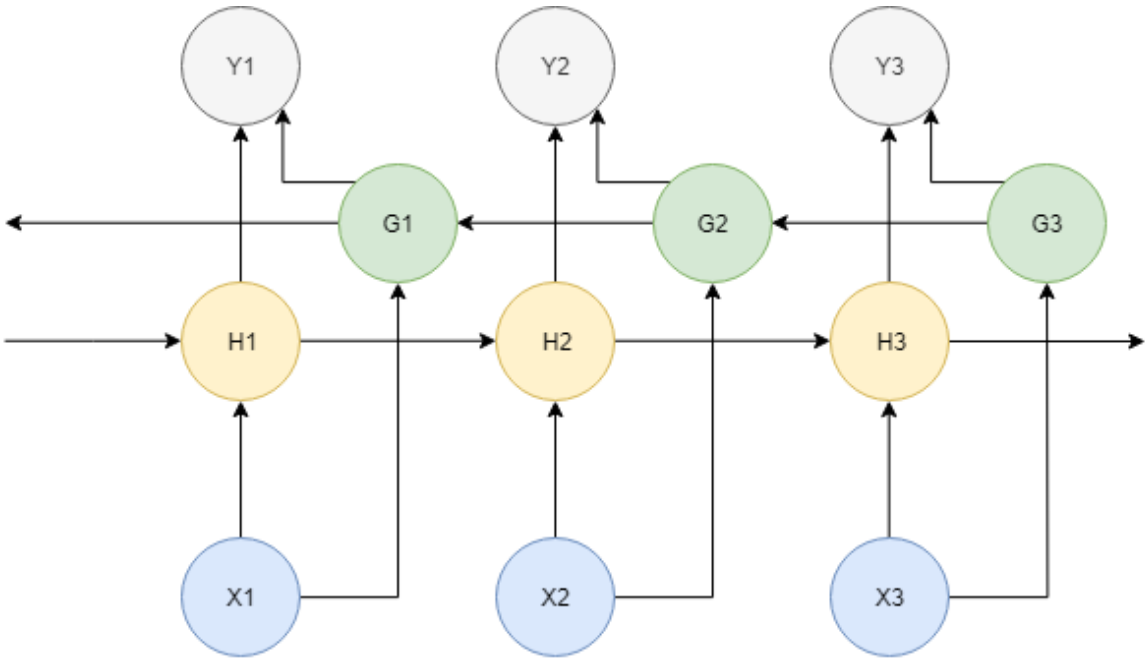
Poleg opisanega RNN-modela poznamo še nekatere druge variacije te zvrsti modela, kot je »Deep« RNN, v katerem lahko dosežemo globino s štirimi načini.



Slika 22: Globoka RNN

V nadaljevanju so predstavljeni štirje možni načini za dodajanje globine. (1) Morda najbolj očitno od vseh je, da dodamo skrita stanja, ena na drugo, ki ustvarjajo izhod enega v drugega. (2) Prav tako lahko dodamo dodatne nelinearne skrite plasti med vhodom in skritim stanjem. (3) Globino lahko povečamo s spreminjanjem povratne uteži v skritem stanju. (4) Povečamo lahko globino med skritim stanjem in izhodom.

Druga variacija modela je dvosmerna RNN, ki deluje tako, da ne opazuje le sprememb stanj v preteklosti, da bi napovedala prihodnost, ampak tudi spreminja preteklost, da bi dosegla boljše rezultate v prihodnosti. Pri nalogah za prepoznavanje govora in prepoznavanje ročne pisave, pri katerih je končni rezultat odvisen od dvoumnosti le enega vnosa, moramo pogosto vedeti, kaj sledi, da bolje razumemo kontekst in zaznamo sedanjost.



Slika 23: Dvosmerna RNN

3.2.5 Izdelava RNN modela

Za izdelavo RNN modela smo si prav tako pomagali s Keras API, pri tem smo uporabili SimpleRNN nivo kateri je popolnoma povezan RNN, kjer se izhodna vrednost iz prejšnjega koraka pošlje nazaj na vhod za novi korak učenja.

RNN model sprejme za vhod 3-dimenzionalni vhodni tip, naši podatki pa so v 2-dimenzionalnem tipu, tretji podatek bo tukaj predstavljal korak. Preoblikovanje podatkov smo dosegli z naslednjo metodo.

```
1. numpy.reshape(a, newshape, order='C')
```

Izvorna koda 7: Preoblikovanje podatkov

Ko smo imeli podatke v zahtevanem tipu in obliki smo začeli z izgradnjo RNN modela.
Za katerega smo definirali tip modela kot sekvenčni. V model smo dodali SimpleRNN nivo z 32 dimenzijami in dva *Dense* nivoja, prvega z 8 dimenzijami in izhodnega z eno. Aktivacijsko funkcijo smo zaradi preprostosti uporabili *Relu*.
Na koncu ko smo model zgradili smo še definirali optimizacijsko funkcijo kot RmsProp katera je predvidena za RNN modele ter funkcijo izgube kot povprečno kvadratno napako.

```
1. model = Sequential()
2. model.add(SimpleRNN(units=32, input_shape=(1, 5),
3. activation="relu"))
4. model.add(Dense(1))
5. model.compile(loss='mse', optimizer='rmsprop',
6.  metrics=['accuracy'])
```

Izvorna koda 8: Definiranje nivojev RNN modela

Po končanem učenju modela, smo dobili za začetek dober rezultat (cca. 62% točnosti), ki je predstavljal dobro izhodišče za optimizacijo našega modela. V naš model smo vstavili še en *Dense* nivo kateri bi naj, pomagal pri vmesnih korakih učenja, saj želimo z RNN modelom pridobiti boljše rezultate kot pa pri prejšnjem DNN, moramo pa vedeti, da je tukaj prisotnih

veliko več faktorjev (parametri, skriti nivoji, razumevanje toka podatkov), ki bistveno otežijo razumevanje in izboljšave modela.

```
1. model.add(Dense(8, activation="relu"))
```

Izvorna koda 9: Dodajanje dodatnega *dense* nivoja

Pridobljeni rezultati po dodajanju novega nivoja v model so bili očitno boljši.

```
Epoch 1000/1000
51000/51000 [==============================] - 0s - loss: 0.1442 -
acc: 0.8074 - val_loss: 0.1520 - val_acc: 0.7842
12704/12904 [============================>.] - ETA: 0s
[0.15197601089437629, 0.78417544947303164]
```

Izvorna koda 10: Rezultati učenja RNN modela

Na koncu smo dobili natančnost ki definira točnost rezultata 80%, kar v našem primeru pomeni zadovoljiv rezultat, v praksi pa na žalost še ne bi bilo dovolj, za uporabljanje modela v komercialne namene ali raziskovanje. Zato smo se odločili, da poizkusimo spremeniti naš model tako, da odstranimo predzadnji nivo z osmimi nevroni, saj je izboljšal rezultat učenja v primerjavi z predhodnim modelom vendar, pa še vedno nismo nikakor uspeli, dobit rezultat, ki bi bil še boljši. Tukaj se je zgodilo to, da je model postal prekompleksen in zaradi tega, aktivacijske funkcije niso uspele, popeljat faze učenja do boljših nastavljanj uteži, ki bi dala boljše rezultate učenja. Na koncu je naš model izgledal enako kot pa na začetku (Izvorna koda 8) definiranja modela. Tukaj smo nato ponovno začeli in pred *SimpleRnn*, vstavili nov vhodni nivo (*Embeding)*, kateri pozitivne podatke s številkami spremeni v *dense* vektorje primer: [[4], [20]] -> [[0.25, 0.1], [0.6, -0.2]]. Ta nivo zato na začetku ne potrebuje podatkov v 3D, zato smo tudi iz naše kode odstranili vrstico(Izvorna koda 7). Izhodni tip iz *Embeding* nivoja pa je v 3D, kar nam je ustrezalo v podajanju podatkov do našega *SimpleRNN* nivoja.

Naša model je sedaj izgledal tako.

```
1. model = Sequential()
2. model.add(Embedding(10000, 124))
3. model.add(SimpleRNN(32))
4. model.add(Dense(1, activation='sigmoid'))
5. model.compile(loss = "mse", optimizer = "rmsprop" ,
6.  metrics=['accuracy'])
```

Izvorna koda 11: Izboljšan RNN model

Po končanem učenju modela, smo dosegli odličen rezultat, ta je sedaj predstavljal točnost z 98.7% (Izvorna koda 12), kar je bilo bistveno boljše kot pa prejšnji poskusi.

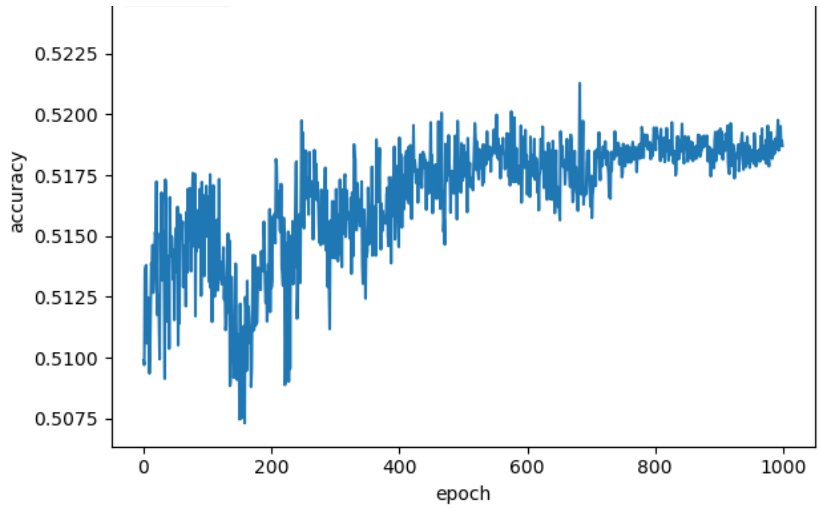
```
Epoch 1000/1000
51000/51000 [==============================] - 2s - loss: 0.0130 -
acc: 0.9870 - val_loss: 0.1604 - val_acc: 0.8302
12416/12904 [============================>..] - ETA:
0s[0.012810457602811636, 0.98721568627450984]
```

Izvorna koda 12: Izboljšan rezultat RNN modela

4 PREGLED IN ANALIZA PRIDOBLJENIH PODATKOV

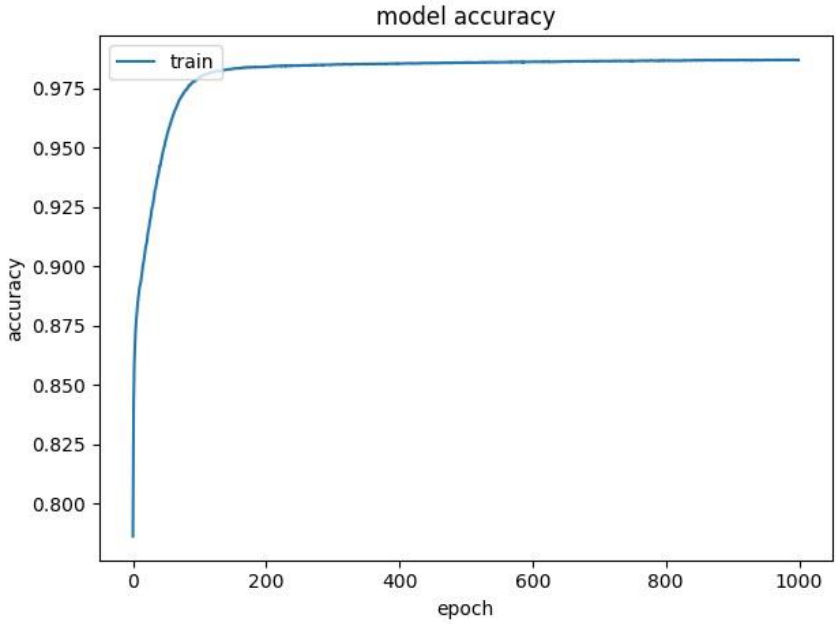
V našem scenariju smo imeli množico podatkov, kateri so predstavljali očitane vrednosti iz BCI naprave po posameznem kanalu in markerju, ki je označeval katero sliko je uporabnik opazoval ob prebrani vrednosti. Imeli smo torej 6 lastnosti, ki smo jih razdelili v dva seznama. Prvega z vrednosti iz BCI naprave in drugega z markerjem, ki je predstavljal rezultat. Vse te podatke pa smo še razdelili v učni in testni seznam.

V našem postopku smo uporabili dva različna modela za strojno učenje. Pri klasičnem DNN modelu smo dobili slabše rezultate, saj je bilo očitno, da imamo premalo podatkov, kljub temu da smo snemali podatke več ur na uporabniku. Prav iz tega razloga se je DNN odrezal slabše, ker spreminja vrednosti uteži le na koncu učenja. Medtem pa je RNN spreminjal uteži že med samim korakom učenja, in na koncu prišel do bistveno boljših predvidevanj, kljub malemu število podatkov za učenje, saj je uporabljal notranji spomin, da je lahko iz prejšnjega izhoda uporabil rezultat za izboljšanje novega.



Slika 24: Učenje DNN

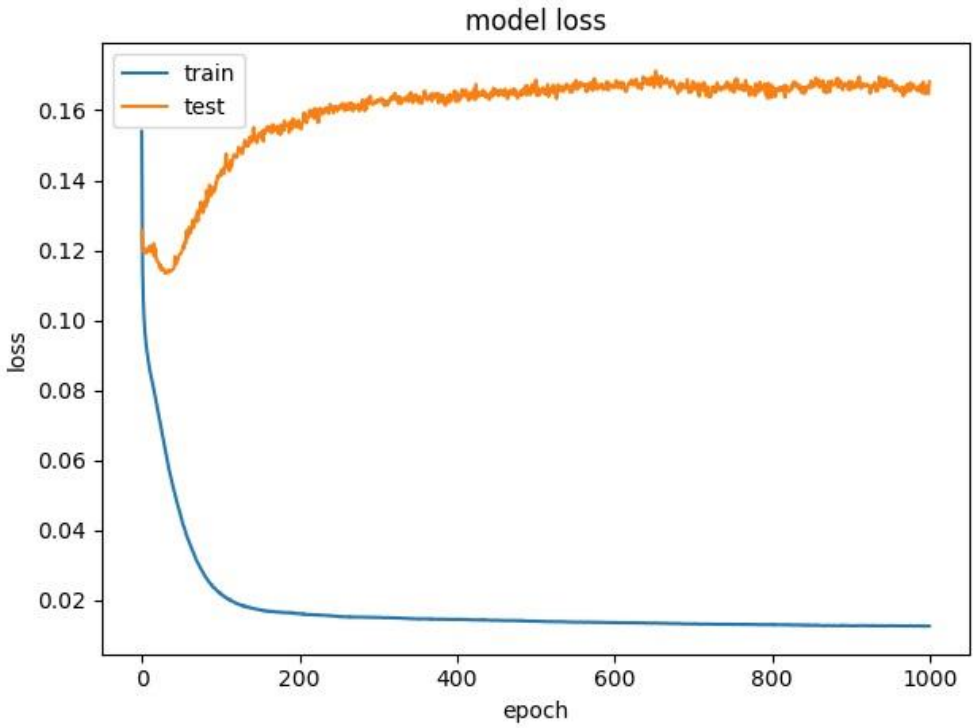
Na sliki (Slika 24) lahko vidimo kako, se je DNN model učil skozi čas. Vidimo, da je model izboljševal svojo točnost v prvi polovici učenja, vendar pa se je proti koncu več ali majn približal enakim vrednostim. V primeru da smo bistveno povečali število epoh, se rezultat s časoma ni izboljševal.



Slika 25: Učenje RNN

Pri učenju modela RNN (Slika 25) pa lahko vidimo, da je točnost modela sunkovito narasla že na začetku učenja. K tej spremembi je pripomogel novi tip podatkov (tokrat v vektorjih), pri katerih pa je nato imel proces učenja, boljšo možnost za nastavljanje uteži med učenjem, pomnimo tudi da RNN model naučeno stanje iz prejšnjega izhoda pošlje na vhod, kar pripomore k »sprotnemu učenju«. Zaradi vseh tek faktorjev, se je ta model odrezal odlično v primerjavi s navadnim DNN. Tukaj govorimo o točnosti, katera v svetu strojnega učenja predstavlja mejo, za uporabo modela v dejanskih aplikacijah.

Prav tako lahko vidimo izgubo na sliki (Slika 26), kako je s časom upadala, kar je pomenilo, da je naš model postajal vedno boljši pri na povedanju rezultata ali uporabnik gleda slike narave ali hrane.



Slika 26:Prikaz izgube skozi učenje RNN

Vsi ti pridobljeni rezultati, pa so še vedno bili močno odvisni od vseh zunanjih faktorjev, ki so bili prisotni pri prvotnem zajemanju podatkov. V začetku moramo vzeti v obzir kakovost in natančnost naprave BCI, kajti namenjena je komercialnim uporabnikom in ne profesionalni uporabi nato moramo vedeti da je že samo snemanje uporabnika je zelo zahtevno, na napravi mora biti ves čas prisoten poseben gel, ki pomaga poveča prevodnost elektrod pri zajemanju elektro magnetnih valov, ob enem morajo elektrode biti čim bližje lasišču in s karseda večjo površino. Naslednji faktor je tudi, razpoloženje samega uporabnika pri skeniranju, ali je bil vedno enako naspan, sit, sproščen, osredotočen, so vse argumenti, ki bi naj bili vedno enaki. Vsi ti faktorji vplivajo na kakovost podatkov in kasnejšo klasifikacijo s strojnim učenjem.

5 SKLEP

V diplomski nalogi, smo si na začetku ogledali in spoznali kaj so možganski valovi in kako se razlikujejo, da smo si lahko boljše predstavljali, kakšen pomen so že imeli na pretekle raziskave in kaj vse nas še čaka v prihodnje.

S tem osnovnim znanjem smo nato se lotili izdelave Cortex GUI aplikacije, s katero smo tudi sami snemali oddajanje možganskih valov, takrat ko je uporabnik gledal določene slike. Slike smo razdelili v 2 skupini, da bi lahko kasneje našli razliko pri oddajanju možganskih valov uporabnika, kadar opazuje sliko narave ali hrane. Pri tem so na našo kakovost podatkov najbolj vplivali zunanji dejavniki, ki so otežili našo snemanje. Te podatke smo nato izvozili iz naše aplikacije in jih nekoliko formatirali da so bili primerni za kasnejšo uporabo za učenje nevronskega modela. Ob tem smo spoznali nekaj različnih knjižnic za izdelavo nevronskih modelov in razlike med njimi. V našem primeru smo si izbrali za backend knjižnico CNTK in Keras knjižnico za izdelavo modela. Ker je za izgradnjo navadnega DNN modela, potrebno predhodno razumevanje vseh parametrov mreže in funkcij, ki so namenjene za izboljšavo učenja modela skozi čas smo si tako ogledali najbolj uporabljene optimizacijske funkcije, funkcije izgube, različne matrike kakovosti podatkov ter aktivacijske funkcije, ter povedali, kdaj so primerne za željeno uporabo.

S tem znanjem smo se lotili na začetku izdelave DNN modela, kateri je na koncu dosegel kar slabe rezultate, saj smo imeli bistveno premalo količino podatkov in tudi sami podatki med seboj se ne močno razlikujejo med seboj. V praksi to pomeni, da verjetno možganska aktivnost ni bila bistveno drugačna med opazovanjem slik narave in hrane. Zato smo se lotili še izdelave RNN modela, katerega bistvena lastnost je da, v času učenja v vsaki iteraciji izhod mreže vpliva na optimizacijo uteži na vhodu.

Z drugimi besedami bi lahko rekli, da ima ta model možnost spomina, in si zapomni kaj je bilo v prejšnjih fazah nastavljeno za doseganje rezultata, da lahko to izboljša v naslednjem koraku. Prav zaradi tega smo z tem modelom na koncu dobili odličen rezultat z našimi podatki. Ob teh spoznanji lahko rečemo da bi ob naslednjih raziskavah v prihodnosti, morali

veliko napora vložiti v karseda kvalitetnejšim snemanjem podatkov na eni ali celo več osebah. Kakor tudi nabava kvalitetnejše in bolj profesionalne naprave in dolžina snemanja podatkov na osebah. Pri tem bi imeli še boljše in kvalitetnejše izhodišče za doseganje še boljših rezultatov s strojnim učenjem. Osebno pa mislim, da ima še to področje raziskav velik potencial za raziskovanje in komercialne produkte v prihodnosti, kateri lahko odprejo nova vrata v svetu povezovanju človeka in stroja.

6 VIRI

[1] [Elektronski]. Available: https://en.wikipedia.org/wiki/Electroencephalography. [Poskus dostopa 16 8 2019].

[2] „imotions.com,“ 7 6 2016. [Elektronski]. Available: https://imotions.com/blog/eeg/.

[3] C. F. Study. [Elektronski]. Available: https://sleepdata.org/datasets/cfs/pages/manuals/polysomnography/17-08-03-01-identify-landmarks.md. [Poskus dostopa 17 6 2019].

[4] C. V. Hagerty-Hoff. [Elektronski]. Available: https://www.semanticscholar.org/paper/MODIFICATION-AND-EVALUATION-OF-A-BRAIN-COMPUTER-TO-Hagerty-Hoff/2673497f822d4c2c37fd2d472d2f2acf8865ccef/figure/1.

[5] S. P.-N. P. I. Suwicha Jirayucharoensak. [Elektronski]. Available: https://www.semanticscholar.org/paper/EEG-Based-Emotion-Recognition-Using-Deep-Learning-Jirayucharoensak-Pan-Ngum/fc7ed93fd8239c6383cf4dfdd58d585c132a32d7. [Poskus dostopa 22 5 2019].

[6] „integral-psy.eu,“ 5 5 2019. [Elektronski]. Available: http://www.integral-psy.eu/sl/integral-psy/znanost-in-mogani.html.

[7] T. Strle, „sinapsa.org,“ 25 2 2009. [Elektronski]. Available: http://www.sinapsa.org/rm/poljudno.php?id=70.

[8] S. Črepinšek, „Trendi v razvoju BCI in,“ p. 71, 2016.

[9] J. Brownlee, „machinelearningmastery,“ 20 6 2016. [Elektronski]. Available: https://machinelearningmastery.com/dropout-regularization-deep-learning-models-keras/. [Poskus dostopa 15 7 2019].

[10] „datasciencefortoday,“ [Elektronski]. Available: https://datasciencefortoday.wordpress.com/tag/logistic-regression/. [Poskus dostopa 20 8 2019].

[11] „tinymind,“ [Elektronski]. Available: https://www.tinymind.com/learn/terms/relu. [Poskus dostopa 22 8 2019].

[12] D. Neubauer, Izbrana poglavja iz elektroencefalografije, Ljubljanja: Ustanova za otroško nevrologijo, 2006.

[13] „brainworksneurotherapy.com,“ 20 6 2019. [Elektronski]. Available: https://brainworksneurotherapy.com/what-are-brainwaves.

[14] M. Venkatachalam, „Medium,“ [Elektronski]. Available: https://towardsdatascience.com/recurrent-neural-networks-d4642c9bc7ce. [Poskus dostopa 8 8 2019].

[15] S. Banerjee, „Medium,“ 23 5 2018. [Elektronski]. Available: https://medium.com/explore-artificial-intelligence/an-introduction-to-recurrent-neural-networks-72c97bf0912. [Poskus dostopa 20 7 2019].

[16] J. Moolayil, Learn Keras for Deep Neural Networks, Vancouver: apress, 2019.

[17] N. Khandwala. [Elektronski]. Available: https://deepgenerativemodels.github.io/assets/slides/session3.pdf. [Poskus dostopa 22 7 2019].