Univerzita
Hradec Králové
Fakulta informatiky
a managementu

Diplomová práce

Uživatelské rozhraní pro osoby s nižší hybností a model pro snímání elektrické aktivity mozku

Bc. Tereza Kvášová

Katedra informačních technologií
Vedoucí práce: Ing. Pavel Kříž, Ph.D.

13. srpna 2022

Poděkování

Poděkování za vedení práce patří Ing. Pavelu Křížovi, Ph.D. Za externí vedení práce a odbornou konzultaci bych chtěla poděkovat Ing. Vladimíru Blažkovi.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval(a) samostatně a že jsem uvedl(a) veškeré použité informační zdroje v souladu s Metodickým pokynem o etické přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 46 odst. 6 tohoto zákona tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou, a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, a za jakýmkoli účelem (včetně užití k výdělečným účelům). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu), licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným způsobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Hradci Králové dne 13. srpna 2022 .....

Univerzita Hradec Králové
Fakulta informatiky a managmentu

Odkaz na tuto práci

Kvášová, Bc. Tereza. *Uživatelské rozhraní pro osoby s nižší hybností a model pro snímání elektrické aktivity mozku.* Diplomová práce. Hradec Králové: Univerzita Hradec Králové, Fakulta informatiky a managmentu, 2022.

Abstrakt

Cílem práce je zhotovit uživatelské rozhraní pro osoby s postižením pohybového aparátu, které bude schopné na základě několika příkazů fungovat jako klávesnice. Součástí práce je průzkum algoritmů strojového učení a oblasti zpracování EEG signálů.

Klíčová slova: uživatelské rozhraní, EEG signál, zpracování EEG signálu, algoritmy strojového učení, mozkové vlny

Abstract

The goal of this work is to create a user interface for disabled people, which will be able to work as a keyboard based on several commands. Part of the work is the research of machine learning algorithms and EEG signal processing area.

Keywords: user interface, EEG signal, EEG signal processing, machine learning algorithm, brain waves

Obsah

Seznam obrázků

Úvod

S každým dnem se technologie posouvají za hranice možností. Vyvíjí se neuvěřitelnou rychlostí. Postupně se staly součástí našich životů a dnes je již nevnímáme jako něco neobvyklého.

V kapsách s sebou nosíme mobilní telefony, v taškách laptopy, které odemkneme pouhým snímáním obličeje. Jezdíme v chytrých automobilech, jež dokážou detekovat okraj vozovky a udržet nás v jednom pruhu. Během pár hodin jsme schopni se dostat na druhou stranu zeměkoule. Ano, přístroje nám opravdu usnadňují každodenní život a díky nim máme možnosti, které by nám naši předci mohli jen závidět. Technologie protknuly všechny sféry lidského života tak, že se bez nich člověk v dnešní době již neobejde.

Pokrok v oblasti interakce s počítačem umožnil, že můžeme v dnešní době se stroji pracovat například pomocí dotyku, vidění, řeči, nebo třeba pomocí mozkových signálů. To nám dává jedinečnou příležitost dalšího rozvoje aplikací například pro osoby s tělesným postižením, kterému se autorka práce bude věnovat.

Podle různých studií v oblasti informačních technologií je dnes vidět, že se do popředí postupně během několika let dostanou hlavně technologie pracující s velkým množství dat (Big Data) nebo právě s využitím machine learning algoritmů. Jedná se o velmi populární téma, na které začíná postupně vznikat větší a větší množství průzkumů.

EEG v souvislosti s použitím algoritmů strojového učení se již využívá v oblastech, jako je zdravotnictví nebo věda. V některých se dokonce zaměřují na napojení zařízení na klávesnici. Tato řešení však nezahrnují implementaci vlastního speciálního rozhraní upraveného pro ovládání mozkovými signály.

Práce je zaměřena právě na vytvoření přívětivého rozhraní na bázi dvou příkazů. Návrhu je věnována samotná kapitola, kde práce čerpá inspiraci z běžného rozhraní, jako je klávesnice pro počítač nebo mobilní telefon. Aplikace je založena na real-time webových technologiích, o kterých se v ní také pojednává.

Část práce je věnována popisu fungování zařízení pro snímání elektrické

aktivity mozku, kontrétně zařízení Emotiv. Nechybí ani popis aplikací poskytovaných touto společností a ukázka implementace s použitím jejich rozhraní pro učení - EmotivBCI.

Součástí je průzkum volby vhodného algoritmu pro učení u dat na bázi EEG signálu. V tomto ohledu se práce odkazuje na již provedené průzkumy v této oblasti a snaží se z nich vybrat ty nejlepší postupy.

Kapitola 1

Cíl práce

Hlavní myšlenkou, díky které vznikla tato práce, je zhotovit rozhraní pro osoby s postižením pohybového aparátu, které bude schopné na základě EEG signálů fungovat jako klávesnice. Toto rozhraní bude však od běžných klávesnic odlišné. Cílem je získat rozhraní, se kterým se lépe pracuje při mentálních příkazech skrze přístroj pro snímání elektrické aktivity mozku.

Důraz je kladen na modulárnost celého projektu. Vytvořit tak fungující systém modulů, který lze rozdělit na několik částí. Jednotlivé moduly jsou schopné fungovat samostatně, nejsou na sobě závislé. K jejich komunikaci zvolit vhodné komunikační rozhraní, aby je v případě potřeby bylo možné snadno nahradit jiným modulem (jinými moduly). Na práci se tak dá jednoduše navazovat v jiných projektech, kde lze použít pouze část systému.

Součástí této práce je rekognoskace machine learning algoritmů a jejich použití v souvislosti s EEG signály. Průzkum oblasti strojového učení povede k výběru vhodného algoritmu pro práci se signálem. V díle se autorka věnuje implementaci a vyhodnocení výsledků pomocí vybraných algoritmů – rozhodovací strom, logistická regrese a rekurentní neuronová síť.

Kapitola 2

Rešerše stávajících řešení

Brain-Computer Interface (BCI) zaměřující se na komunikaci mezi člověkem a počítačem se používají v několika oblastech výzkumu jako je neurorehabilitace, roboti, exoskeletony atd. Rozhraní BCI překládá aplikaci signály mozkové aktivity uživatele na příkazy/zprávy.

Elektroencefalografie (EEG) se obecně používá k měření mozkových aktivit uživatele BCI. Tento systém pak na základě EEG může umožnit svému uživateli například pohybovat kurzorem na obrazovce počítače vlevo nebo vpravo za použití vizualizace pohybů levé a pravé ruky. [11]

Oblast zpracování a využívání EEG signálů pro interakci s aplikacemi není ještě zcela prozkoumána. V posledních letech však narůstá počet výzkumů v souvislosti se zpracováním EEG signálů a strojovým učením.

Na Univerzitě v Maďarsku vznikla studie EEG-based Computer Control Interface for Brain-Machine Interaction [12]. V této studii je demonstrován model na bázi EEG a aplikace pro digitální zpracování signálu založená na technologii NeuroSky. Vizualizace pro vyhodnocování signálů mozkových vln je implementována v objektově orientovaném jazyce C#. Tato aplikace vyhodnocuje signál podle míry soustředění.

Studie EEG Signal Based System to Control Home Appliances [13] se zabývá extrakcí signálu v reálném čase pro řízení domácích spotřebičů pomocí příkazů. Tento projekt je založen na zařízení NeuroSky MindWave. Signály jsou přenášeny do Arduina pomocí modulu Bluetooth HC-05. Pro ovládání je použito signálů zachycujících mrkání očí a meditace. Signál je zpracován s využitím algoritmu CSP pro odstranění šumu z EEG dat.

Diplomová práce Wheelchair Control Using EEG Signal Classification [14] se věnuje konceptu mozkem ovládaného elektrického invalidního vozíku, jejím cílem bylo prozkoumat možnosti ovládání směru vozíku pomocí elektroencefalogramu.

The Thought Translation Device (TTD) for Completely Paralyzed Patient [15] je zařízení na překlad myšlenek pro pacienty, kteří dokáží komunikovat s vnějším světem jen na základě pomalých kortikálních potenciálů za-

znamenaných elektroencefalografem. Výsledky jsou popsány pro pět pacientů s amyotrofickou laterální sklerózou. Pacienti si vybírají písmena, slova nebo piktogramy v počítačovém programu. Na základě sebraných dat je potvrzeno, že jsou tito pacienti schopni využít tohoto zařízení jako alternativního komunikačního kanálu pro přenos myšlenek. TTD je tedy možné používat u úplně paralyzovaných pacientů. Ve výsledcích je uvedeno, že není nutné zavádět invazivní přístup (zavedení do mozku), jelikož je neinvazivní metoda dostačující.

Zrychlením komunikace se zabývá projekt alternativního pravopisného zařízení nazývaný Virtual Keyboard (do češtiny přeloženo virtuální klávesnice, zkráceně VK) [16]. Toto zařízení je založené na Graz-BCI (více o něm v odstavci níže). VK je zařízení pro hláskování písmen ovládané spontánním EEG, přičemž EEG je modulováno mentálním zobrazením motorů rukou a nohou. Tetraplegický pacient, který byl původně vyškolen na řídicím systému ortézy založeném na EEG ovládajícím VK, dosáhl schopnosti v používání VK a dokáže s ním pracovat rychlostí přibližně jedno písmeno za minutu.

The Graz-BCI [17] byl vyvinut skupinou Pfurtscheller na katedře lékařské informatiky Technické univerzity v Grazu. Je to malý a přenosný systém optimalizovaný pro každodenní použití pacientů trpících těžkým motorickým postižením. Systém získává své řídicí informace z on-line klasifikace obrazců EEG souvisejících s motorickým zobrazením. Motorické zobrazení odpovídá pohybu levé a pravé ruky (je možná i kombinace jedné ruky a nohy). Klasifikace dvou EEG vzorů (tříd) odpovídá binárnímu rozhodování, a tedy i kontrolní signály jsou binárního typu.

V průzkumu byly nalezeny studie, které zpracovávají EEG signál a pomocí strojového učení jsou schopny ovládat například domácnost, invalidní vozík nebo ortézy (náhrady). Byla nalezena i signálem ovládaná klávesnice a pravděpodobně se najdou i další práce, které takové řešení obsahují.

Odlišnost od nich tato práce získává díky speciálnímu rozhraní, které je vyvinuto na základě práce s pouze dvěma příkazy (dále a potvrzení). Tato speciální klávesnice je optimalizována tak, aby se uživateli snadněji a rychleji pohybovalo skrz abecedou. Součástí této práce je také prozkoumat oblast učících algoritmů a zjistit, které studie dosáhly nejlepších výsledků v souvislosti s daty na bázi EEG signálů.

Kapitola 3

EEG signal processing

Následující kapitola se věnuje oblasti zpracování EEG signálu. Příprava podkladů o vzniku a zpracování signálů je inspirována studií Analysis of Electroencephalography (EEG) Signals and Its Categorization [18].

3.1 Lidský mozek

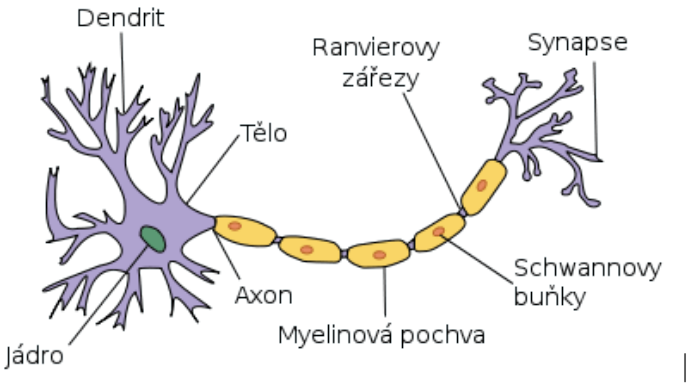
Mozek je jedním z nejdůležitějších orgánů v lidském těle. Náleží do centrální nervové soustavy a je uložen v dutině lebeční, která ho chrání před poškozením.

Vývoj lidského mozku je zdlouhavý proces, který začíná ve třetím gestačním týdnu diferenciací nervových progenitorových buněk (progenitory; buňky, které nemají schopnost sebeobnovy) a prodlužuje se přinejmenším do pozdního dospívání, pravděpodobně po celý život. [19]

Mozek dospělého člověk se skládá z více než 100 miliard neuronů (Pakkenberg a Gundersen 1997). Neuron (viz. obrázek č. 3.1) je nosičem informací mezi tělem a mozkem. Existuje mnoho různých druhů neuronů, které se diferencují svou velikostí, tvarem a vlastní funkcí. Neurony vytvářejí spojení s jinými neurony a formulují tak sítě pro zpracování informací. Ty jsou zodpovědné za všechny naše myšlenky, pocity, city i činy. Jelikož každý neuron může vytvořit spojení s více než tisícem dalších neuronů, odhaduje se, že dospělý mozek má více než 60 bilionů neuronových spojení.

3.2 Neuron

Neurony jsou posly informací. Používají elektrické impulsy a chemické signály k přenosu informací mezi různými oblastmi mozku a mezi mozkem a zbytkem nervového systému. Vše, co si myslíme, cítíme a děláme, by nebylo možné bez práce neuronů a jejich podpůrných buněk (gliových buněk zvaných astrocyty a oligodendrocyty). [20]



Obrázek 3.1: Anatomie neuronu [1]

Na obrázku 3.1 je zobrazen neuron. Má tři základní části: buněčné tělo a dvě rozšíření nazývaná axon a dendrit. Axon vypadá jako dlouhý ocas a přenáší zprávy z buňky. Dendrity vypadají jako větve stromu a přijímají zprávy pro buňku. V těle buňky je jádro, které řídí buněčnou činnost a obsahuje buněčný genetický materiál. Neurony spolu komunikují tak, že posílají chemikálie, zvané neurotransmitery, přes malý prostor, nazývaný synapse [19], mezi axony a dendrity sousedních neuronů. [20]

3.3 EEG signál

Jednotlivé soustavy lidského těla jsou řízeny pomocí specifických frekvencí signálu. Elektrická aktivita mozku (EEG) vykazuje významné komplexní chování se silnými nelineárními a dynamickými vlastnostmi. [21]

K vyšetření kontinuální nervové aktivity mozku lze využít elektrofyziologické metody. Při EEG vyšetření je vždy pozorován potenciálový rozdíl mezi dvěma elektrodami. Tato měření mohou být jak bipolární, kdy je křivka zaznamenaná na dvou různých bodech lebky hodnocena vzájemnou korelací, tak unipolární, kdy je střídání potenciálu porovnáváno s křivkou zaznamenanou indiferentní nebo neaktivní elektrodou. Svody, které nepřenášejí nervovou aktivitu (bod daleko od lebky, např. ušní lalůček), se nazývají neaktivní referenční elektrody, zatímco ty, které detekují změnu napětí v důsledku nervové aktivity, se nazývají aktivní elektrody. [22]

Získávání a analýza signálů elektroencefalografie (EEG) patří mezi nejvýznamnější neinvazivní elektrofyziologické monitorovací metody pro záznam elektrické aktivity mozku. [23]

3.4 Mozkové vlny a jejich kategorizace

Existuje pět široce uznávaných mozkových vln: delta, theta, alfa, beta a gama frekvence v rozsahu od 0,1 Hz do více než 100 Hz. Tyto frekvence zachycuje obrázek č. 3.2.

Následující popis, co které mozkové vlny značí, je čerpán z článku Ovládněte své mozkové vlny [24].

Mozkové vlny Delta (0,5-3 Hz) se objevují například při hluboké meditaci, bezvědomí nebo hlubokém spánku.

Théta (4-7 Hz) značí také meditaci, hluboký spánek nebo odpočinek, ale při těchto vlnách vnímáme vlastní tělo. Při Delta vlnách je vnímání těla sníženo na minimum.

Alfa (7-13 Hz) zachycuje stav při uvolnění. Může to být například chvíle při čtení knihy, kdy necháváme myšlenky volně plynout. Pokud se nachází ve vysokých hodnotách (11-13 Hz), tak se cítíme maximálně uvolněni. Při nízkých hodnotách (7-11 Hz) se jedná o úplnou relaxaci, dostáváme se do svého vlastního vnitřního světa. Alfa vlny přispívají ke koncentraci, snižují stres a podporují kreativitu. Do procesu se zapojují obě hemisféry.

Při běžných aktivitách se zapojuje hlavně Beta (14-40 Hz) vlna. V té se nacházíme po většinu dne, například když třeba řídíme auto. V této chvíli pracuje převážně levá hemisféra, které se připisuje hlavně logické myšlení, kreativita je tedy snížena. Vysoká Beta (21-40 Hz) značí stres a napětí. Pokud v ní člověk setrvává dlouhodobě, může to vést k nemocem a psychickým problémům.

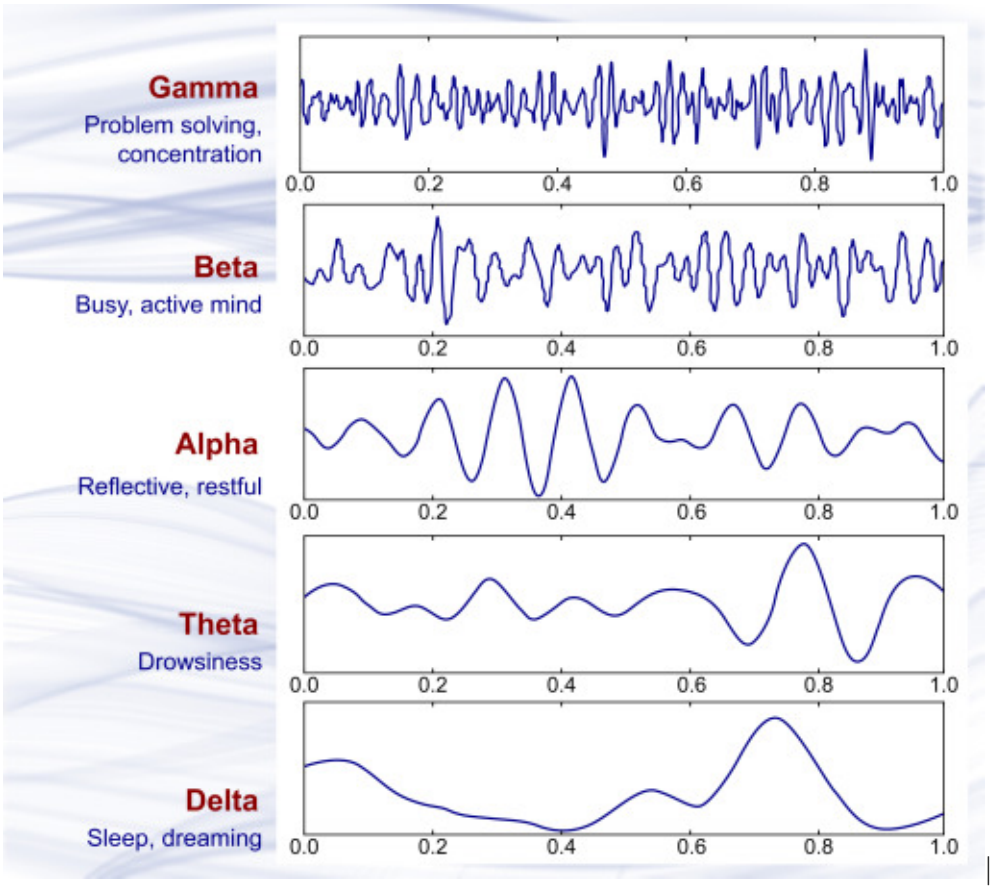
Gama (30 Hz a více) je spojována s krizovými situacemi nebo vysokým napětím. Je zaznamenána při vypjatých situacích jako je například skok padákem nebo nehoda.

EEG normálního dospělého v plně vědomém, ale uvolněném stavu, je tvořeno pravidelně se opakujícími oscilujícími vlnami známými jako alfa vlny. Když je člověk vzrušený nebo vyděšený, jsou alfa vlny nahrazeny nízkonapěťovými rychlými nepravidelnými vlnami. Během spánku jsou mozkové vlny extrémně pomalé. Stejně tomu tak je i v případě, kdy je člověk v hlubokém komatu. Jiné abnormální stavy jsou spojeny s určitými EEG obrazci. Například nepravidelné pomalé vlny známé jako delta vlny vznikají v blízkosti lokalizované oblasti poškození mozku. [25]

Různé oblasti mozku vysílají různé frekvence mozkových vln současně. Vzorce mozkových vln jsou jedinečné pro každého jednotlivce.

3.5 Elektroencefalograf

Elektroencefalograf je přístroj, který se používá k zachycení mozkových funkcí. Zařízení se skládá z elektrod umístěných na EEG čepici, procesoru a zesilovače. EEG signál dosahuje pouze nízkých hodnot napětí, proto je v zařízeních obsažen i zesilovač, který se postará o jeho zesílení.



Obrázek 3.2: Mozkové vlny [2]

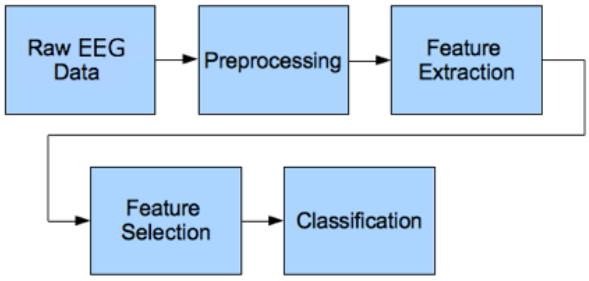
Elektroencefalogram lze zaznamenat dvěma způsoby. Prvním způsobem je invazivní metoda, která se používá při pokusech na zvířatech, kdy se do lebky vyvrtají otvory a do mozkové tkáně se umístí mikroelektrody. Druhá varianta, která se používá u člověka, je neinvazivní. V tomto případě se umisťují elektrody na pokožku hlavy.

Elektroencefalografické (EEG) monitorování bylo jednou z prvních neurofyziologických technik používaných na operačním sále. [26]

V roce 1929 německý vědec Hans Berger zveřejnil výsledky první studie využívající elektroencefalograf, nástroj, který měří a zaznamenává tyto vzory mozkových vln. [25]

Pro práci je zvoleno zařízení Emotiv Epoc-X, které je schopné zaznamenat čtrnách kanálů. Je ho možné vidět na obrázku 3.3. V kapitole Realizace lze nalézt jeho bližší popis.

Obrázek 3.3: Elektroencefalograf Emotiv Epoc-X [3]



Obrázek 3.4: Jednotlivé kroky nutné pro zpracování EEG dat [4]

3.6 Analýza signálu a jeho zpracování

EEG signál je komplexní vícesložková periodická vlna. Tyto signály pravděpodobně vznikají jako důsledek agregované elektrické aktivity několika tisíc neuronů umístěných na povrchu mozkové kůry. Bioelektrické signály generované v mozku nebo elektrická aktivita neuronů indukují některá elektromagnetická pole. To znamená, že ke změnám napětí dochází aktivitou kůry. [22]

EEG jsou však typicky rušeny šumem, čímž se zhoršuje následná interpretace získaných informací. Pro účely zlepšení signálu zůstává kritickým úkolem odstranění rušivého šumu ze souboru EEG signálu navržením vhodných metod filtrování. [23]

Co se týká samotné analýzy a zpracování EEG signálů, existuje již řada studií a konceptů, které se touto tématikou zabývají. Na obrázku 3.4 je zobrazen celý proces zpracování dat. Nejprve jsou sbírána data, následně se pokračuje krokrem předzpracování dat, poté extrakcí funkcí, výběrem funkcí a následně klasifikací. Jednotlivým krokům se práce věnuje v následujících sekcích této kapitoly.

3.7 Předzpracování signálu

Získaná data musí projít před vlastním zpracováním ještě několika fázemi. Jednou z těchto fází je předzpracování signálu, které spočívá v odstranění artefaktů. Artefaktem rozumíme signály, které nejsou projevem mozkové aktivity např. pohyb očí nebo dýchání. Artefakt může být způsoben i selháním techniky např. narušením sítě. Pro odstranění artefaktů se běžně používá analýza nezávislých komponent (ICA).

Tato část zahrnuje aplikace filtrů, kdy se například aplikuje horní (většinou omezení 1 Hz) nebo dolní propustnost pro odstranění stejnosměrných složek signálů. Frekvence nad 90 Hz (Gamma) se také odstraňují. Existuje korekce technik elektrookulogramem, který zaznamenává pohyb očí. Snímaný subjekt musí mít pro snímání elektrické aktivity mozku oči otevřené. Při mrkání totiž vznikají silná elektrická pole a ovlivňují tak EEG záznam. Záznamy se po očištění stříhají po několika sekundách a z každého záznamu se extrahují jeho rysy. [4]

3.8 Příznaky (extrakční funkce a jejich výběr)

Díky rozvinuté technologii v oblasti počítačů dnes můžeme získat velké množství informací shrnujících znalost o datech aplikováním složitých algoritmů. Jednotlivé části (segmenty) signálu lze charakterizovat sadou příznaků např. minimum, maximum, střední hodnota, koeficient špičatosti, šikmosti, průměr, max. první a druhá derivace, modus, medián, směrodatná odchylka, rozptyl (obrázek č. 8.15).

Pro vizualizaci lze využít i histogram, což je sloupcový graf, kde je u každé třídy znázorněna její četnost. Data lze převést i na graf pro síťovou analýzu, kdy je bodem elektroda a spojení odpovídá EEG signálu elektrod.

Existuje také EEG tomografie, která umožňuje vypočítat, které oblasti uvnitř mozku jsou právě aktivní aplikací tzv. inverzního přístupu. To vyžaduje vysoký počet elektrod (alespoň 32), aby bylo dosaženo dobrého prostorového rozlišení.

Výběr funkcí je volitelným krokem. Používá se v případě velkého množství dat, kdy je snaha omezit počet použitých funkcí na ty, které jsou vhodné pro konkrétní data.

V tomto případě se používají techniky jako analýza hlavních komponent (PCA) nebo třeba Fisherova diskriminační analýza. [27]

3.9 Klasifikace

Dále se pokračuje klasifikací. Pod tímto pojmem se skrývá zařazení částí signálu do určitých tříd. Třídou se rozumí např. spánková fáze mozku nebo stav

oka (otevřené/zavřené). Pomocí technik strojového učení můžeme trénovat klasifikátor, aby z funkcí rozpoznal, do které třídy který signál spadá.

Klasické metody učení s učitelem, jako je lineární diskriminační analýza (LDA), podpůrné vektorové stroje (SVM) a rozhodovací stromy, jsou běžné v neurální klasifikaci. [27]

V následující části této sekce jsou popsány některé vybrané algoritmy využívané pro klasifikaci. Data jsou čerpána z článku Classification of EEG Signals Based on Pattern Recognition Approach [28].

3.9.1 Support Vector Machine (SVM)

Do řady klasifikačních metod patří například SVM. Algoritmus je určen k učení s učitelem, kdy mapuje vstupní data do vyšších dimenzí pomocí množiny nelineárních bázových funkcí.

3.9.2 Multilayer Perceptron (MLP)

MLP je metoda založená na nelineární neuronové síti. Tato síť je tvořena vstupní vrstvou, skrytou vrstvou a výstupní vrstvou. Ve skryté vrstvě se přenáší vstupní data do výstupní vrstvy. Lze v ní upravovat počet neuronů. Na počtu neuronů velmi záleží. Problémem u modelu MLP může být totiž nadměrné přizpůsobení, které je způsobeno nedostatečným nebo nadměrným počtem neuronů.

3.9.3 Naïve Bayes (NB)

Efektivní pravděpodobnostní klasifikací založenou na Bayesově větě je klasifikátor NB. Předpokládá, že extrahované prvky nejsou závislé. Model NB používá algoritmus maximální pravděpodobnosti k určení třídy dřívějších pravděpodobností a rozdělení pravděpodobnosti prvku z trénovací datové sady. Výsledky jsou pak použity s maximalizovaným posteriorním rozhodovacím stromem k nalezení specifického označení třídy pro novou testovací instanci. [29]

3.9.4 K-Nearest Neighbor (k-NN)

K-nejbližší soused (K-Nearest Neighbor) je řízený algoritmus učení, který identifikuje třídu testovacího vzorku podle většinové třídy k-nejbližších trénovacích vzorků; tj. štítek třídy je přidělen nové instanci nejběžnější třídy mezi KNN v prostoru „feature“. [29]

3.9.5 K-fold Cross Validation (Křížová validace)

Křížové validace je běžně používaná technika, která porovnává výkony dvou klasifikačních algoritmů nebo hodnotí výkon a jediný klasifikátor na daném

datovém souboru [30]. Výhodou je použití všech instancí v datové sadě buď pro trénink, nebo pro testování, kde je každá instance použita k ověření přesně jednou.

Kapitola 4

Dataset

Dataset je soubor dat. Obsah takového souboru dat odpovídá jedné tabulce v databázi nebo matici statistických dat.

Pokud tvoříme soubor dat pro strojové učení, vytváříme nejprve tréninkovou sadu. Na této sadě se algoritmus trénuje tak, aby v budoucnu dokázal rozlišovat určité typy EEG signálů (v tomto konkrétním případě, jinak může například rozlišovat obrázky a tak dále).

Soubor dat pro model zahrnuje vstupní data i očekávaný výstup, který se porovnává s výsledkem učícího algoritmu. Vstupní data by měla tvořit přibližně 60% z celkových dat. Druhou sadou je ověřovací sada. Ta se používá k doladění finálního modelu. Třetí sadou je testovací soubor dat. Ten tvoří přibližně 20% dat. Správnost vstupních dat je ověřena většinou člověkem.

Celý proces učení je závislý na datech. Díky nim se umělá inteligence (artificial intelligence - AI) může učit požadovanému chování. S nevhodným souborem dat se nedá dosáhnout dobrých výsledků, proto je tomuto tématu věnována celá kapitola.

Příprava dat není jen fází sběru dat, ale součástí toho je i jejich úprava tak, aby byla vhodná pro strojové učení.

Celý proces vytváření datové sady zahrnuje tři kroky: získání, čištění a označení dat.

4.1 Získání dat

Tento proces zahrnuje hledání datových sad vhodných pro řešenou problematiku. Pokud neexistuje žádná vhodná datová sada, lze si takovou sadu například vygenerovat.

První variantou, jak získat data pro projekt, je si samotný soubor s vhodnými daty vyhledat. Na internetu již existují úložiště, kde uživatelé ukládají nahraná data a je možné je využívat bez nutnosti sám si tyto soubory vytvářet.

V tomto případě se hledají data s EEG signály, které je možné rozdělit na dvě klasifikační skupiny. Jedna skupina signálů bude v tomto projektu

označena jako potvrzení, druhá skupina slouží pro pokyn posunutí (vysvětlení těchto pokynů v kapitole Návrh uživatelského rozhraní).

Jak již bylo zmíněno v předchozím odstavci, za účelem sdílení takových dat existují některá úložiště, jako je například DataHub. Ten poskytuje možnost spolupráce několika lidí. Je inspirován verzovacím systémem Git.

Další variantou sdílených dat je Kaggle. Kaggle je webová aplikace založena opět na možnosti spolupráce skupin lidí. Lze přes ni sdílet a objevovat data. Zároveň se zde lidé dělí o poznatky a postupy práce s těmito daty. Tímto způsobem se dá inspirovat nebo například rozvíjet práce dalších účastníků. Velmi často je Kaggle používán ke konání soutěží nad určitými typy dat za účelem získání nejefektivnějšího, nejzajímavějšího řešení atp. I díky nim se zde dají dohledat zajímavé výsledky postupů jednotlivých účastníků soutěže.

Protože je výroba velkého množství dat vhodných k použití strojového učení časově náročná, jsou sady poskytovány prostřednictvím internetu.
Pro usnadnění prohledávání internetu a hledání datových sad začaly vznikat systémy. V souvislosti s těmito systémy je třeba zmínit pojem datové jezero.

Datové jezero je podle dokumentace Microsoft [31] úložiště s velkým množstvím dat v nezpracované formě. Tím se odlišují od datových skladů. Datový sklad je úložiště, kde se data zpracovávají při vstupu.

Google Data Search (Goods) [32] pracuje s informacemi o databázích na Googlu. Shromažďuje o nich metadata. Vytváří se tak katalog databází. Další takovou kolekcí databází je WebTables systém [33], který pomocí HTML tagů na webu shromažďuje data.

Co dělat v případě, že máme nějaká data, ale není jich dostatek? Pokud nemáme dostatek dat, lze je rozšířit. Taková technika může zahrnovat například úpravu jasu u již připravených dat, která jsou uložena jako nové položky do sady (pokud by se jednalo o obrazová data). V oblasti strojového učení je to běžný způsob rozšiřování trénovacích dat.

A pokud nenalezneme žádná použitelná data? Potom přichází v úvahu generování dat. To lze dělat buď ručně (tato varianta je však pracná) nebo automaticky.

Jednou z možností získání takových dat je crowdsourcing.Tento pojem je složen ze dvou anglických slov. Crowd znamená skupina, sourcing je zdroj. Jak už tyto slova napovídají, jedná se o získání dat od skupiny lidí prostřednictvím internetu.

Na této bázi dat funguje například aplikace Waze, kdy uživatelé aplikace hlásí ostatním uživatelům nehody a další dopravní problémy. Tak je k dispozici obraz o dopravě v reálném čase [34]. Mezi outsourcingové platformy patří například Amazon Mechanical Turk nebo Citizen Science.

Další variantou je získat data pomocí generování syntetických dat tj. dat vytvořených pomocí počítače. Jedná se o velmi rychlou a levnou variantu. Takové generické modely může generovat například Generative Adversarial Network (GAN).

Principem GAN jsou dvě neurové sítě, které mezi sebou soutěží. Jedna síť je generativní, druhá síť je diskriminační. Generativní síť vytváří nová data na základě původních dat a diskriminační se snaží tyto případy nově vytvořených (generativní sítí) rozlišovat od původních dat. Cílem generativní sítě je zvýšit chybovost diskriminační sítě. Jinými slovy, generativní síť se snaží oklamat diskriminační síť. Snaží se ji mystifikovat tak, aby diskriminační síť považovala synteticky vytvořená data jako skutečná (ne syntetická) data.

Příkladem takové GAN sítě je MEDGAN [35], která generuje syntetická data z původních originálních dat pacientů.

4.2 Čištění dat

Když je zásoba dat dostatečná, je potřeba tato data očistit. Čistí se například od šumu. Tuto práci je možné provádět ručně, nicméně je značně zdlouhavá. To je důvod, proč se lze obrátit na některé frameworky a nástroje, které tuto práci udělají za vás. Takovým příkladem je HoloClean.

HoloClean opravuje, čistí a obohacuje data. Systém využívá hodnotové korelace, pravidla kvality a referenční data, aby pomohl vytvořit pravděpodobnostní modely, které zachycují proces generování dat. Pomáhá také datovým vědcům ušetřit čas potřebný k čištění dat. [36]

Dalším systémem, který lze využít je ActiveClean, BoostClean nebo MLClean.

U tohoto bodu je vhodné zmínit, že je potřeba čistit data jen do určité míry. Datová sada by měla být stále reprezentativní pro vybranou populaci, na které chceme algoritmus učit.

4.3 Označení dat (data labeling)

Při předzpracovávání dat se datům přiřazuje jejich význam tzv. označení (label). Obrázek psa je tedy označen jako „pes“. Tento proces může být do značné míry subjektivní a označování celé řady dat je velmi pracné. I v tomto kroku hrají učící algoritmy svou roli a lze je na tuto práci použít.

Pokud máme část dat již označených a část dat ještě neoznačených, tak se při semi-supervised učení druhá půlka dat (neoznačená) předpoví podle již označené části. Algoritmus odvodí další označení.

Pro označení dat lze opět využít crowdsourcingu a již zmíněné platformy jako je Amazon Mechanical Turk (MTurk) a Citizen Science. Alternativou je také generování nedokonalých štítků, kdy se tato nižší kvalita kompenzuje velkým množstvím dat, ve kterých se ztráta ztratí.

Označování dat je závislé i na typu dat. Některé algoritmy se hodí na diskrétní hodnoty, jiné na použití v souvislosti se spojitými.

4.4 Získání dat pro projekt

V předchozích částech kapitoly je představeno několik způsobů získání dat pro projekt. Jednou z variant je i vytvoření vlastního datasetu. Toto řešení je však časově náročné. Podle průzkumu výše, existují na internetu zdroje, kde uživatelé ukládají jimi zpracovaná data. Po prohledání několika takových úložišť byl zvolen dataset na stránce Kaggle. Více k tomuto datasetu v kapitole Realizace.

Kapitola 5

Machine learning algorithm

Strojové učení nebo také automatizované učení je jednou z nejrychleji rostoucích oblastí informatiky s dalekosáhlými aplikacemi. Pojem strojové učení označuje automatizovanou detekci smysluplných vzorců v datech. V posledních několika desetiletích se stal běžným nástrojem pro téměř jakýkoliv úkol, který vyžaduje extrakci informací z velkých souborů dat. [37]

Tato technologie nás již v dnešní době obklopuje. Mnohdy si to ani neuvědomujeme. Kdo v dnešní době z mladší generace ve vyspělých zemích nepoužívá internet? Asi bychom nenašli moc takových jedinců. S jistotou lze tedy tvrdit, že většina mladých lidí přichází do styku např. s vyhledávačem. Tento nástroj nám nabízí na základě předchozích vyhledávání nejlepší výsledek, aby nám samotné vyhledávání výrazů usnadnil. Zároveň je schopen na základě těchto dat směřovat na klienta např. reklamy, které jsou uživateli zobrazovány (tzv. cílená reklama). V obou případech se využívají algoritmy strojového učení.

Dalším příkladem využití těchto algoritmů je schopnost digitálních fotoaparátů rozpoznat obličeje. Proces rozpoznání tváře nahrazuje např. vyplňování hesla před otevřením mobilního zařízení.

Strojové učení vychází z informací o schopnosti učit se, jak tomu je například u nás, lidí. Vezmeme-li si příklad od inteligentních bytostí, mnohé z našich dovedností získáváme nebo zdokonalujeme učením se z našich předchozích zkušeností (než následováním pouhých pokynů). Nástroje strojového učení se zabývají vybavováním programů schopností „učit se“ a přizpůsobovat se. [37]

Cílem učení je naprogramovat stroj tak, aby byl schopen „učit se“ na základě předložených dat (vstupů), získat tak určitou znalost (odbornost) a z tohoto poznatku dokázal čerpat u práce nad dalšími daty. Tudíž se snažíme o to, aby algoritmus dokázal na základě znalosti z tréninkových dat definovat některé aspekty na datech testovacích. Takto „naučený“ algoritmus je zpravidla podkladem pro další aplikaci.

Je potřeba vzít v potaz, že lidé jsou schopni na základě zdravého ro-

zumu rozhodovat o výsledcích učení a zabránit tak volbě nesmyslného závěru. To stroje zatím nedokážou.

Programu je nezbytné jasně definovat zásady pro vyloučení nesprávného výsledku. S tím souvisí téma zabývající se správným začleněním předchozích znalostí. Nýbrž právě zahrnutí předešlé zkušenosti ovlivňuje celý proces učení.

Jinými slovy, čím jsou silnější předchozí znalosti (nebo předchozí předpoklady), se kterými člověk začíná proces učení, tím snazší je učit se z dalších příkladů. Čím silnější jsou však tyto předchozí předpoklady, tím méně flexibilní je učení – je a priori vázáno k těmto předpokladům. [37]

Mezi modelem dostatečně naučeným a modelem přeučeným je mnohdy těžké najít hranici. Patřičné je model na přeučenost testovat a eventuálně využít techniky pro eliminaci přeučení modelu.

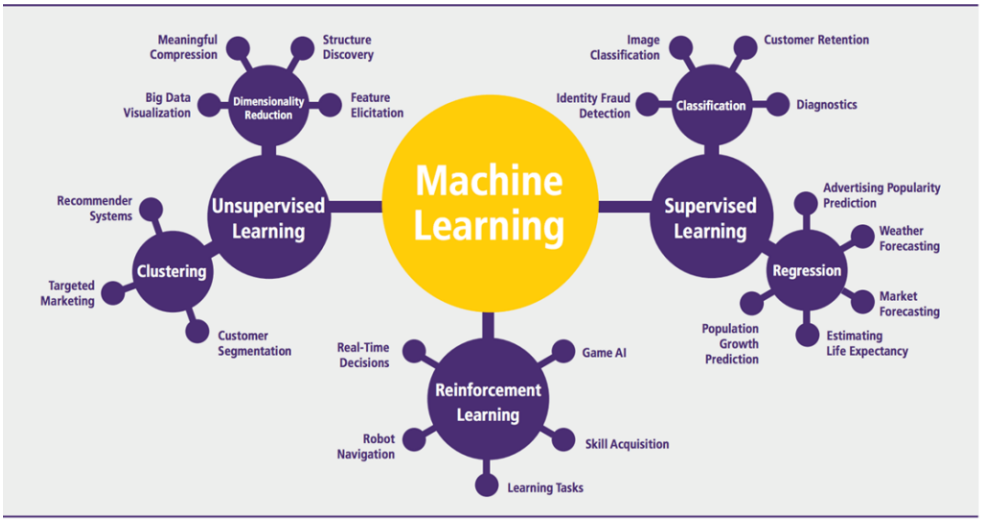
Vyvstává otázka, proč se snažíme naučit stroje chování, které ovládáme my sami? Odpověď na ní je poměrně jednoduchá. Lidé nejsou schopni pracovat s tak obrovským množstvím dat stejně dobře jako stroje. Například pokud jde o rozpoznání řeči, je stroj schopen jednoduše určit, o jaký jazyk jde, a to velmi rychle. Stroje jsou schopny zpracovat velké množství dat, které člověk jen stěží dokáže pojmout.

Big data (přeloženo do češtiny, velká data) jsou podle definice na webové stránce `www.oracle.com` v článku What is Big Data [38] definovány jako data, která obsahují větší rozmanitost, jsou dodávána ve větších objemech a s větší rychlostí. Tři Vs je volume (objem), velocity (rychlost) a variety (rozmanitost). Zjednodušeně řečeno, velká data jsou větší datové sady a jsou tak objemná, že je tradiční software pro zpracování dat nedokáže spravovat, proto se dostávají na scénu algoritmy strojového učení.

Možná by se mohlo zdát, že díky strojovému učení už nás počítače mohou zcela nahradit. To je však mylná domněnka. I při používání chytrých systémů založených na učících algoritmech je neustále potřeba lidská ruka, přesněji lidský mozek, spravující data programu. Počítače zatím nejsou schopny uvažovat a učit se z vlastních chyb.

V dnešní době je velké množství dat digitalizovaných a máme k nim jednoduchý přístup pomocí internetu. Tyto data lze využít k různým analýzám a statistickým předpokladům, které se používají právě pro celý proces.
Nejprve probíhá učení, kdy je tvořen model. Potom, pokud je model již naučený, můžeme rozhodovat o výsledku.

Algoritmy strojového učení se dělí na tři hlavní skupiny - učení s učitelem (supervised learning), bez dozoru (unsupervised learning) a posílené učení (reinforcement learning), jak je zobrazeno na obrázku č. 5.1. Tyto kategorie jsou blíže popsané v následujících třech sekcích.



Obrázek 5.1: Rozdělení učících algoritmů na tři kategorie - učení s učitelem (supervised learning), bez dozoru (unsupervised learning) a posílené učení (reinforcement learning) i s konkrétními příklady algoritmů spadajících pod tyto části [5]

5.1 Učení s učitelem

Učení s učitelem potřebuje trénovací data a zpětnou vazbu na výsledek. Jak je řečeno v knize New Advances in Machine Learning [39], jsou to algoritmy, které mapují vstupy na požadovaný výsledek.

Těmto algoritmům jsou zadána data i s informací o požadovaném výstupu. Například bude do EEG dat zaevidována informace, zda je při snímání mozkové aktivity otevřené nebo zavřené oko. To znamená, že z určitých signálů zaznamenaných v nějakém časovém úseku vyplývá, že má uživatel otevřené (eventuálně zavřené) oko. Takto lze ze vstupních informací čerpat a následně odvozovat výsledky pro další data. Postupně se generuje model, který je možné aplikovat na předpovědi v reálném světě.

Pod toto paradigma spadají dvě kategorie, a to je klasifikace nebo regrese (obrázek č. 5.1).

5.2 Učení bez učitele

Učení bez učitele je založeno na zkoumání vstupní proměnné bez toho, že by byla algoritmu dána explicitně výstupní data. Tento algoritmus se dá např. použít, pokud nevíme, jak data klasifikovat [5]. Model se trénuje tak, aby vyřešil problém bez znalosti odpovědi.

Většinou se tento styl učení volí v případech, kdy neexistuje správná ani špatná odpověď, ale je nějaký lepší a horší způsob řešení.

Pro představu, algoritmu dodáme EEG data, která neobsahují informaci o stavu oka. Tyto data algoritmus rozdělí do několika skupin podle podobných znaků charakteristických pro určité signály, tj. snaží se na základě předchozích zkušeností z dat odhadnout chybějící charakteristiky,

Není zde rozdíl mezi tréninkovými a testovacími daty.

5.3 Posílené učení

Reinforcement Learning (RL, česky posílené učení) umí určit co dělat, jak mapovat situace, aby se maximalizovala odměna. Program nemá informaci o tom, jaké akce má provést. Místo toho musí zjistit, jaké akce přinášejí největší odměnu.

Proces učení probíhá tak, že se vyzkouší možnosti a zhodnotí se odměna za ně [40]. Pro lepší představu si můžeme uvést příklad o výběru šachového tahu z knihy Understanding machine learning [37]. V knize je uvedeno, že lze algoritmus naučit hodnotovou funkci, která popisuje pro každé nastavení šachovnice stupeň, o který je pozice bílého lepší než pozice černého. Jedinou informací, kterou má algoritmus v době tréninku k dispozici, jsou pozice, které se vyskytly během skutečných šachových partií, označené podle toho, kdo danou hru nakonec vyhrál. Na základě těchto informací je pak schopen rozhodovat o výsledku.

5.4 Overfitting

Velmi známým problémem při učení modelu je overfitting. Jedná se o chybu při modelování, kdy se algoritmus přeučí tj. funkce je příliš blízko k omezené sadě datových bodů nebo jí přímo odpovídá. V tomto případě je algoritmus perfektně naučen na danou sadu, ale na jiných datech nefunguje.

Kapitola 6

Webové aplikace pracující v reálném čase

Aplikace pracující v reálném čase neboli real-time aplikace, jak její název napovídá, je aplikace pracující v ideálním případě bez zpoždění. Podle anglického článku [41] je definice takové aplikace následující:*„Real-Time Applications are applications that operate within an immediate time frame“* , tj. aplikace pracuje bez časového rámce. Je to program, u kterého má uživatel pocit, že se vše děje „teď“, v tomto okamžiku. Zpoždění takové aplikace je v ideálním případě nulové. Reálně se se však může pohybovat okolo 2 vteřin.

Kde se lze s real-time aplikacemi střetnout? Jsou běžnou součástí našeho života. Aplikace v reálném čase se využívají všude kolem nás: při komunikaci pomocí sociálních sítí např. při používání aplikace Facebook nebo třeba při hraní online her. S real-time aplikacemi se pracuje také v zdravotnictví, kde si vyžaduje řešení situace velmi rychlé reakce a rozhoduje o osudu pacienta.

U konvenční webové aplikace je zahájena komunikace na straně klienta. Webové aplikace fungující v reálném čase oproti tomuto způsobu komunikace umožňují zahajovat komunikaci na straně serveru a to s minimálním zpožděním.

U real-time aplikace lze očekávat aktuální data. Technologicky lze real-time komunikace docílit například prostřednictvím HTTP komunikace u webových aplikací, které využívají tzv. pooling, forever frame, http streaming atd. Jednotlivým přístupům se věnují následující části této kapitoly, kde je proveden průzkum možností pro samotnou realizaci aplikace. Součástí práce je totiž tvorba real-time webové aplikace s uživatelským rozhraním pro osoby s nižší hybností.

6.1 Pooling

Prvním řešením, které se nabízí pro vývoj aplikace je pooling. Pooling je založen na periodickém zasílání požadavků klienta na server. Pokud nemá server data, vrací prázdnou odpověď, v opačné případně data na danou událost odešle.

Server není schopen posílat požadavky na klienta, pouze odpovídá na klientovi požadavky. Doba jeho odezvy závisí na frekvenci odesílání požadavků z klientské aplikace a době nutné ke zpracování daného požadavku.

Nevýhodou je právě vytváření spojení i v případě, že server ještě nemá odpověď. Server je tak zatížen velkým množstvím požadavků.

Tento problém řeší long pooling. Long pooling se odlišuje od normálního poolingu tím, že klient ponechá otevřené spojení serveru a ten odpoví, až na danou událost. HTTP spojení má danou maximální délku trvání a ta určuje, jak dlouho bude spojení navázáno. Tato komunikace je ukončena i v případě, že server neodeslal zpátky na klienta žádná data a vytváří se opět nové spojení s maximálním intervalem.

Přístupy, jako je long pooling, vyžadují mnoho přeskoků mezi servery a zařízeními. Brány mají často různé představy o tom, jak dlouho může být typické spojení otevřené. Pokud zůstane otevřená příliš dlouho, může se tak snadno stát, že je komunikace ukončena, i když se vykonává důležitý krok.

6.2 Http streaming

Http streaming je založen na jedné otevřené komunikaci mezi serverem a klientem. Spojení se udržuje po celou dobu otevřené.

Umožňuje serveru nepřetržitě odesílat data bez nutnosti uzavírat a znovu otevírat spojení. Server data pošle, jakmile jsou připravena.

Server uzavírá spojení jen v případě, že je o to požádán. Není nutné posílat HTTP hlavičky, odpadá dotazování.

6.3 Server-sent events

Alternativně mohou být data streamována prostřednictvím metody Server-Sent Events (SSE), pro kterou existuje standardizované HTML5 API s názvem EventSource.

S SSE jsou data v záhlaví kódována jako text/stream-stream. Data jsou prostřednictvím prohlížeče zpracována jako DOM události. Odesílá se pomocí HTTP protokolu.

Server data odesílá sám automaticky, takže se klient nemusí na data dotazovat. Při ztrátě spojení je spojení opět navázáno klientem. Pro zprávy odeslané touto komunikací je stanoven formát MIME.

6.4 Web sockets

Protokol WebSockets se objevil s verzí jazyka HTML a to HTML5. Byl standardizován v roce 2011 organizací Internet Engineering Task Force a ve stejném roce také aplikační rozhraní tohoto protokolu konsorciem W3C [42]. WebSockets se používají, pokud potřebujeme oboustranné plně duplexní spojení klienta a serveru. Plně duplexní znamená, že spolu klient a server mohou komunikovat nezávisle na sobě.

Funguje na protokolu TCP tzn. je zaručeno, že spojení nebude přerušeno a všechna data se odešlou na druhou stranu neporušena a ve stejném pořadí, jako byla odeslána. Klient a server si vyměňují datový rámec, oba mají 2B.

Komunikace probíhá na portu 80 nebo 443 pro šifrovanou komunikaci (HTTPS). Komunikace je zahájena přes rozhraní WebSockets, do instance je přidán listener čekající na událost (zprávu).

Komunikace má dobrou podporu. Oboustranná komunikace je vhodná pro hry, aplikace se zasíláním zpráv, či aktualizace v obou směrech téměř v reálném čase. Je podporována ve většině prohlížečích například v Google Chrome, Firefoxu nebo třeba Safari.

Podpora prohlížečů je jeho výhodou a proto byl také zvolen mezi technologie použité při tvorbě aplikace. Další výhody jsou jmenovány v odstavcích níže. Aplikace pomocí WebSockets posílá na server nepřetržité datové toky EEG a server je zpracovává pro klasifikace v reálném čase.

Rozhraní WebSocket disponuje následujícími instrukcemi:

- open - vytvoření spojení mezi klientem a serverem
- message - příjem zpráv ze serveru
- error - pokud někde nastane nějaká chyba
- close - slouží pro uzavření komunikace

WebSockets udržuje jediné trvalé připojení otevřené a zároveň eliminuje problémy s latencí, které vznikají u metod založených na požadavku/odpovědi HTTP. Obecně nepoužívá XMLHttpRequest, a proto se hlavičky neodesílají pokaždé, když klient potřebuje získat další informace ze serveru. To také snižuje zatížení serveru daty. Techniky založené na HTTP bývají na serverech mnohem náročnější na zdroje, zatímco WebSockets mají na serverech extrémně nízkou stopu.

Nevýhodou WebSockets je, že se po ukončení připojení automaticky neobnoví. Tato část musí být naprogramována, nicméně existuje mnoho knihoven, které tento problém na straně klienta řeší, a tak není potřeba žádná práce navíc.

Obecně budou WebSockets lepší volbou v kontextu probíhající komunikace v reálném čase. WebSockets jsou voleny v aplikacích fungujících na bázi EEG signálu. Využívá je i společnost Emotiv pro své vlastní aplikace.

Kapitola 7

Návrh uživatelského rozhraní

Uživatelské rozhraní (UI - user interface) je bodem interakce člověk-počítač. Jedná se například o obrazovku, klávesnici nebo myš. Je to také způsob, jakým uživatel komunikuje s aplikací nebo webem.

Dobře provedené uživatelské rozhraní usnadňuje interakci mezi uživatelem a programem, aplikací nebo strojem prostřednictvím čistého designu.

Uživatelské rozhraní může být například grafické (GUI - Graphic User Interface) nebo lze počítač ovládat skrze rozhraní příkazového řádku (CLI - Command Line Interface). Existují také rozhraní ovládaná dotykem nebo třeba hlasem.

Tato práce se zaměřuje na tvorbu neurálního rozhraní (BCI - Brain-Computer Interface) fungujícího na principu klávesnice. Uživatel prostřednictvím tohoto rozhraní může psát textové zprávy.

Návrh uživatelského rozhraní je proces, kdy návrhář vytváří vizuální obsah s cílem najít vzhledově nebo stylově nejvhodnější řešení. Tvůrce se snaží vytvořit pro zákazníka příjemný a jednoduchý design.

Existuje také pojem uživatelská zkušenost (UX - User Experience), který s uživatelským rozhraním úzce souvisí, ale jejich specifika se liší. Uživatelské rozhraní je navrženo podle zamýšleného vzhledu a chování systému, zatímco UX pokrývá celý proces konceptu, vývoje a poskytování aplikace uživateli. UX se zabývá hlavně účelem a funkčností produktu.

Jak se píše v knize The Elements of User Experience: User-Centered Design for the Web and Beyond [43], ani ten nejlepší obsah a nejsofistikovanější technologie nepomohou vyvážit cíle bez soudržného a konzistentního uživatelského zážitku, který je podporuje.

7.1 Kritéria návrhu

Při tvorbě uživatelského rozhraní je nutné stanovit si konkrétní cílovou skupinu uživatelů, kterou by měla aplikace primárně oslovit. Této skupině potom primárně přizpůsobit její vzhled.

Uživatele lze kategorizovat například podle věku, úrovně znalostí, pohlaví, kultury, zdravotního stavu. Cílovým zákazníkem této aplikace budou osoby přibližně od deseti let výše (dokážou psát a interagovat s počítačem). Převážně je cíleno na osoby od 20 do 60 let.

Rozhraní je navrženo pro osoby s omezenou tělesnou hybností. V projektu se počítá s tím, že tito lidé nemohou využívat k interakci s počítačem horních končetin a nejsou tak schopni jednoduše stroj ovládat.

Systém pro snímání elektrické aktivity mozku umožňuje psát zprávy pouze za pomocí myšlenky. Navržená aplikace slouží k psaní textů. Dále ji lze rozšířit například jako chatovací aplikaci ovládanou EEG signály.

Na vznik systému a jeho vývoj má vliv zvolená kategorie uživatelů. To v konečném důsledku ovlivňuje jejich potřeby a požadavky. V tomto případě uživatelé očekávají od aplikace velmi jednoduchou práci s počítačem tj. přehledné a snadno ovladatelné rozhraní, které jim umožní psát bez použití končetin.

Celkový návrh by měl být decentní. Nemá na sebe poutat přílišnou pozornost. Uživatele by neměla grafická stránka jakkoli při práci (při výběru tlačítka nebo znaku) rušit. Grafika je navržena tak, aby se mohl zákazník soustředit na dokončení úkolu. Většina uživatelů ocení také jednoduchost a snadnou ovladatelnost.

Důležitým aspektem je počet příkazů, které rozhraní přijímá. Ten se má držet na co nejmenším počtu. V ideálním případě snížit toto číslo na dva příkazy, kterými bude uživatel schopen s aplikací komunikovat. Právě počet příkazů k ovládání je stěžejním bodem ovlivňujícím celkový vzhled. Je nutné vymyslet také způsoby ovládání tak, aby se s rozhraním dobře pracovalo. Rozhraní je napojeno na aplikaci pracující s mozkovými signály.

Proč je počet příkazů v ovládání rozhraní tak důležitý? Systém se jednotlivé příkazy musí naučit pomocí učících algoritmů. Aby byl algoritmus schopen rozlišovat jednotlivé požadavky uživatele od sebe (odlišit jednotlivé EEG signály, které zachytí), potřebuje k procesu učení velké množství dat.

Čím více příkazů, tím více dat je potřeba. Čím více dat je pro projekt nutné shromáždit, tím je práce na systému časově náročnější. Při rozlišování dvou příkazů stačí menší množství dat a méně času na samotné strojové učení. Pro urychlení procesu učení si tedy systém vystačí pouze s řešením pro dva příkazy.

Rozhraní vzniká jako náhrada alfabetické části klávesnice. Uživatel s ním interaguje jen za pomoci dvou signálů. Jeden z těchto příkazů může sloužit jako potvrzení. Druhý umožňuje pohyb mezi písmeny abecedy (nebo dalšími ovládacími prvky). Pro pochopení možností ovládání je uveden tento příklad, jak by mohlo rozhraní fungovat. Bližšímu popisu způsobu ovládání a možnostem rozhraní je věnován prostor v následujících sekcích.

V návrzích se autorka zabývá i počtem signálů, které musí uživatel vyslat, aby vybral určité písmeno. Tedy kolik kroků je třeba podniknout k dosažení cíle. Jinými slovy, kolik signálů potřebuje k výběru písmene. Je zde snaha tento počet minimalizovat tak, aby byl uživatel schopen rychle napsat jednotlivá

slova a nemusel zdlouhavě procházet celou řadu prvků, než se k vybranému znaku dostane.

7.2 Příprava návrhu

Jak je zmíněno výše v podkapitole 7.1 ve specifikaci uživatelského rozhraní, aplikace disponuje dvěma vstupy, kterými je rozhraní ovládáno. To značně ovlivňuje i vzhled výsledného návrhu uživatelského rozhraní.

V následujících odstavcích je čerpána inspirace v již člověkem vytvořených uživatelských rozhraních k ovládání počítače, jako je například běžná klávesnice na počítači nebo na mobilním telefonu.

7.2.1 Inspirace v počítačové klávesnici

Historie moderní počítačové klávesnice se datuje od vynálezu psacího stroje. Byl to Christopher Latham Sholes, kdo si nechal patentovat v roce 1868 první moderní psací stroj. O 11 let později začala společnost Remington Company masově prodávat první stroje. V roce 1878 bylo Sholesem a jeho partnerem Jamesem Densmorem vyvinuto rozložení QWERTY klávesnice. [44]

Pokud se podíváme na tradiční počítačovou klávesnici viz. obrázek č. 7.1 určenou pro psaní všemi deseti (tj. disponuje deseti vstupy), zjistíme, že je tvořena až 48 klávesami. Z toho je 26 kláves vyčleněno pro abecedu (bez diakritiky).

Pro práci s deseti prsty je klávesnice optimalizovaná, ale pokud by se měla použít pro vstupy dva, nastává problém. Dlouhá řada tlačítek s jedním znakem uvnitř se stává překážkou a psaní je velmi zdlouhavé.

Než se uživatel dostane několika příkazy na druhou stranu řady znaků, trvá to velmi dlouho. Pokud se navíc uživatel omylem posune dále, než původně zamýšlel, musí se dostat přes všechny znaky zpátky na začátek a znovu se zkusit posunout na požadované písmeno.

Jak se tedy rychle dostat až k poslednímu tlačítku v pořadí pomocí dvou vstupních signálů? Při snaze se dostat k poslednímu písmenku je třeba poslat 26 stejných signálů, což není optimální řešení.

Alternativou ke QWERTY klávesnici se stala v roce 1936 patentovaná Dvorakova klávesnice (obrázek 7.2). Dvorakova klávesnice je ergonomickou alternativou k běžnému rozložení klávesnice QWERTY.

Umožňuje rychlejší psaní a je snadnější se naučit rozložení jednotlivých písmen. Proč tomu tak je? Pokud se podíváme pořádně, má klávesnice na pravé straně souhlásky a na levé straně samohlásky. Je to z důvodu, že je lepší funkci pravé a levé ruky střídat.

Navíc je klávesnice Dvoraka navržena tak, aby se postupně při psaní písmen střídaly prsty a nemusel se využívat jeden prst dvakrát po sobě. To činí rozhraní opět efektivnější.

Pro člověka je zároveň jednodušší postupně stlačovat prsty z vnější strany na vnitřní. Tomu je Dvorakova klávesnice také přizpůsobena.

Dalším faktorem, který ovlivňuje pohodlnost při psaní, je umístění nejvíce používaných znaků na domovské řadě. Domovská, neboli prostřední řada, je série znaků, na které spočívají prsty, pokud uživatel nepíše. Ostatní, také hojně používaná písmena, jsou posunuta na horní řádek. Zejména používané znaky jsou umístěny pod prostřední řadou.

Proč tato klávesnice nenahradila QWERTY klávesnici, když by se podle mínění jejího autora měla lépe ovládat? Jedním z důvodů je, že tento návrh přišel již v době, kdy by bylo dost náročné a drahé nahrazovat klávesnice tolika strojů. [45]

Důležité je také zmínit, že Dvorakova klávesnice nepočítá s diakritikou. V projektu, který vzniká na základě této práce je tomu stejně. Na diakritiku se nebere zřetel, a proto se dá inspirovat i v tomto rozhraní.

V případě aplikace napojené na EEG headset není potřeba přemýšlet nad rozložením prstů nebo se ohlížet na možnost střídání rukou při psaní. Zařízení je ovládáno signálem vyslaným z mozku a ruce (eventuálně prsty) nejsou ke kontrole systému potřeba.

Pokud jde o kombinaci dvou signálů (dále, potvrdit), není nutné zvažovat jejich střídání. Tuto myšlenku lze později ověřit při napojení na aplikaci transformující signály na příkazy a eventuálně se zamyslet nad kombinací těchto příkazů. V aktuálním návrhu tedy není zahrnuta varianta kombinace příkazů.

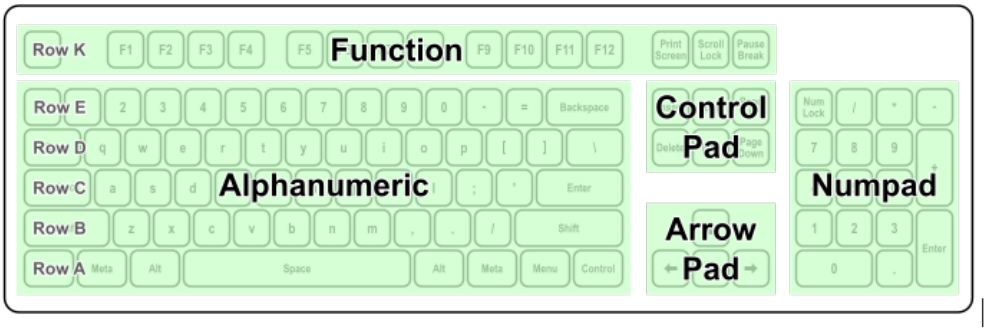
Čím se však lze inspirovat, je pořadí souhlásek a samohlásek nebo vhodné umístění nejvíce využívaných písmen podle Dvoraka. Proto se touto klávesnicí práce zabývá podrobněji.

Na prostředním (domovském) řádku jsou v návrhu Dvoraka umístěna písmena v tomto pořadí: A, O, E, U, I, D, H, T, N, S. Jak je psáno na začátku této kapitoly, střídání levé (samolásky) a pravé (souhlásky) ruky zefektivňuje psaní všemi deseti.

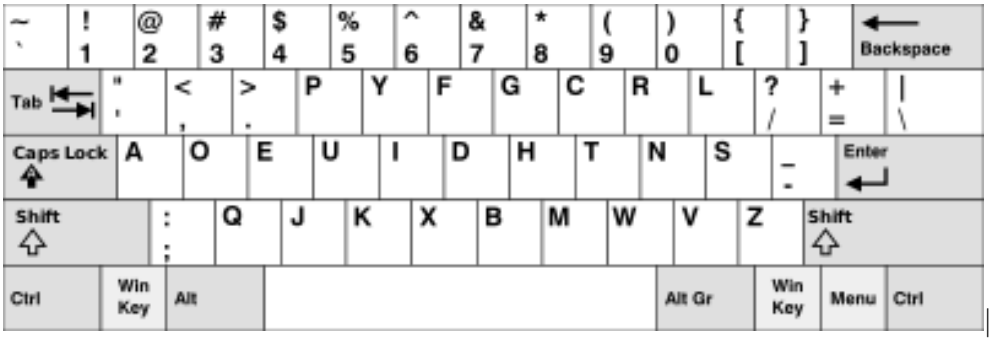
Při vytváření rozhraní pro psaní písmen pomocí mozkových vln se lze zaměřit na pořadí znaků. V ideálním případě by se měla kombinovat souhláska se samohláskou. Je potřeba mít na mysli, že je nutné počítat v návrhu s omezením týkajícím se ovládání. Znaky lze procházet pouze jedním směrem (kvůli omezenému počtu signálů k ovládání aplikace). Na kombinaci samohlásek a souhlásek se zaměřuje návrh č. 3 7.7 a návrh č. 4 7.8.

Při zařazení samohlásek mezi určitý počet souhlásek tak, aby se tyto bloky střídaly, by se dalo docílit rychlejšího psaní. Při napsání souhlásky, bude následující písmeno ve slově pravděpodobně samohláska. Při tomto řazení písmen je větší pravděpodobnost, že se následující písmeno, které chce uživatel vybrat, nachází někde za již napsaným znakem.

Druhou inspirací v návrhu Dvoraka je umisťování znaků od nejvíce využívaných písmen po nejméně používaná. Pokud jsou rozmístěna nejvíce používaná písmena rovnoměrně v řadě písmen, potom je možné, že je uživatel bude využívat postupně při průchodu řadou až na konec sekvence.



Obrázek 7.1: Počítačová QWERTY klávesnice [6]



Obrázek 7.2: Dvorakova počítačová klávesnice [7]

K zamyšlení je způsob ovládání u varianty seřazené podle frekvence použití. V tomto případě může rozhraní po napsání písmene vracet uživatele na první znak. Tím pádem jsou nejvíce používaná písmena na začátku řady a nejméně používaná písmena na konci. Takto je schopen uživatel opět používat nejvíce frekventovaná písmena ve slovech jako první.

7.2.2 Inspirace v mobilní klávesnici

Na obrázku č. 7.3 je vidět klávesnice navržena pro mobilní zařízení. To je navrženo tak, aby ho byl člověk schopen ovládat i v jedné ruce za pomocí jednoho prstu - palce.

Na obrázku (č. 7.3) má klávesnice jedenáct tlačítek, která stačí k napsání textu zprávy. Pod tlačítky se, jak je vidět na obrázku, skrývá více znaků. Při výběru tlačítka je zpřístupněna volba znaku ve vybrané sekci. Uživatel teď může měnit znak. Má na to určitý čas. Po vypršení intervalu je písmeno napsáno. Tento princip je využit v jedné z variant v realizaci návrhu.

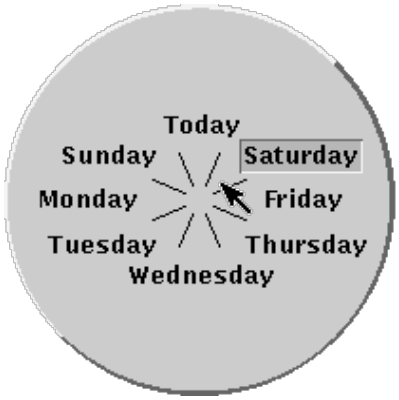
Pod prvním tlačítkem, umístěným vlevo nahoře, není žádné písmeno abecedy. V rozhraní jsou písmena skryta až pod tlačítkem číslo dva a dále. Posledním tlačítkem je tlačítko pro mazání. Tlačítko nula má pod sebou skrytou

Obrázek 7.3: Klávesnice na mobilním zařízení [8]

funkcionalitu - mezeru, která slouží k oddělení slov mezi sebou. Na návrhu není tato mezera znázorněna graficky. Vzhledem k tomu, že je mobilní zařízení a jeho klávesnice již známa, může si výrobce dovolit odstoupit od znázornění mezery. Uživatel už ví, že je mezera umístěna na tomto místě, proto je v tomto případě od grafického znázornění upuštěno. Stejně tomu je i u prvního tlačítka, které slouží k napsání interpunkce.

Při používání klávesnice mobilního zařízení je tedy návrh přizpůsoben pohodlnému držení telefonu. Pokud je to nutné, je klávesnice uzpůsobena možnosti ovládat zařízení palcem. V případě použití jen jedné ruky, tedy psaním palcem, se vlastníkovi mobilního zařízení s touto klávesnicí nejlépe mačká tlačítko na středu, kdy je palec v rovné poloze. Pro napsání znaku Z, je nutné ohnout prst a víckrát stlačit tlačítko s číslem devět. Naopak pro napsání nějakého písmene na tlačítku s číslem čtyři, je potřeba prst natáhnout, eventuálně trochu změnit úchop mobilního telefonu, pokud je ruka malá a prsty na ní menší.

Rozložení tlačítek není nutné v návrhu přizpůsobovat podle držení zařízení v ruce, protože se bude jednat o software ovládaný mozkovými vlnami. Co lze ale v návrhu použít, je shluk znaků v jednom tlačítku. Umožňuje to uživateli rychlejší pohyb abecedou. Pokud si všimneme, na klávesnici mobilního zařízení není brán ohled na více frekventovaná písmena ve slovech, jako tomu je například u klávesnice Dvoraka (popsané v části 7.2.1). U mobilní verze jsou seřazena písmena abecedně.



Obrázek 7.4: Koláčová nabídka podle Dona Hopkinse z článku [9]

7.2.3 Radial menu

Radial menu nebo také pie menu (koláčová nabídka) je kontextová nabídka s tlačítky uspořádanými do kruhu.

První zdokumentované radiální menu je připisováno systému zvanému PIXIE v roce 1969. V roce 1986 Mike Gallaher a Don Hopkins společně nezávisle dospěli ke konceptu kontextového menu. [46]

Často se objevuje ve hrách, jako jsou hry RPG. Je používána na herních konzolích, kdy se v menu uživatel pohybuje analogovou páčkou, která umožňuje rychlý pohyb s malým počtem stisknutí. Koncept se začal využívat v konzolových RPG sériích Mana, počínaje Secret of Mana v roce 1993. [47]

Většinou se používá s myší nebo stylusem [48]. Pro představu, jak takové menu může vypadat, je přiložen obrázek č. 7.4.

Dialogové kolečko je formou radiálního menu určeného pro dialogové stromy. Dialogové kolo zpopularizovala série Mass Effect. [47]

Tento návrh stojí za zvážení i v případě použití aplikace pracující s EEG daty. Správně rozdělená koláčová nabídka může snížit počet vysílaných signálů ze zařízení a usnadnit tak uživateli výběr písmen a zrychlit psaní.

7.3 Vlastní návrh

U všech návrhů níže se autorka snažila pracovat s co nejmenší znakovou sadou:

- všechna písmena abecedy bez diakritiky
- mezera
- tečka
- čárka (jen v některých)

- otazník
- vykřičník
- znak pro mazání posledního psaného písmene (například šipka zpět)

Celkové řešení tedy obsahuje písmena abecedy potřebná k psaní zpráv bez diakritiky, která není nutností. Tím se zmenšuje počet znaků a zrychluje se práce s rozhraním. Dále se nepočítá např. s možností volby malého nebo velkého písmene. Text bude zobrazen velkými písmeny. Tato funkcionalita je další nadstavbou, kterou lze případně v budoucnu realizovat.

Dále se autorka rozhodla v některých návrzích pro jednoduchost nepřidávat čárku (interpunkční znaménko), bez které lze komunikovat. Počet potřebných znaků je minimalizován, protože má být rozhraní co nejjednodušší. K poslednímu znaku se uživatel musí dostat co nejkratším počtem signálů, aby byl schopen napsat text za poměrně krátkou dobu.

Podle požadavků sepsaných výše byl sestaven první návrh rozhraní, který je možné vidět na obrázku č. 7.5. Obsahuje všechny písmena abecedy, mezeru, tečku, otazník, vykřičník a šipku zpět, kterou lze již napsaný text mazat.

Je navržen podobně, jako tomu je například u mobilních zařízení, kdy je pod tlačítky schován určitý počet znaků. Tímto způsobem se dá rychle posouvat po sekcích v abecedě.

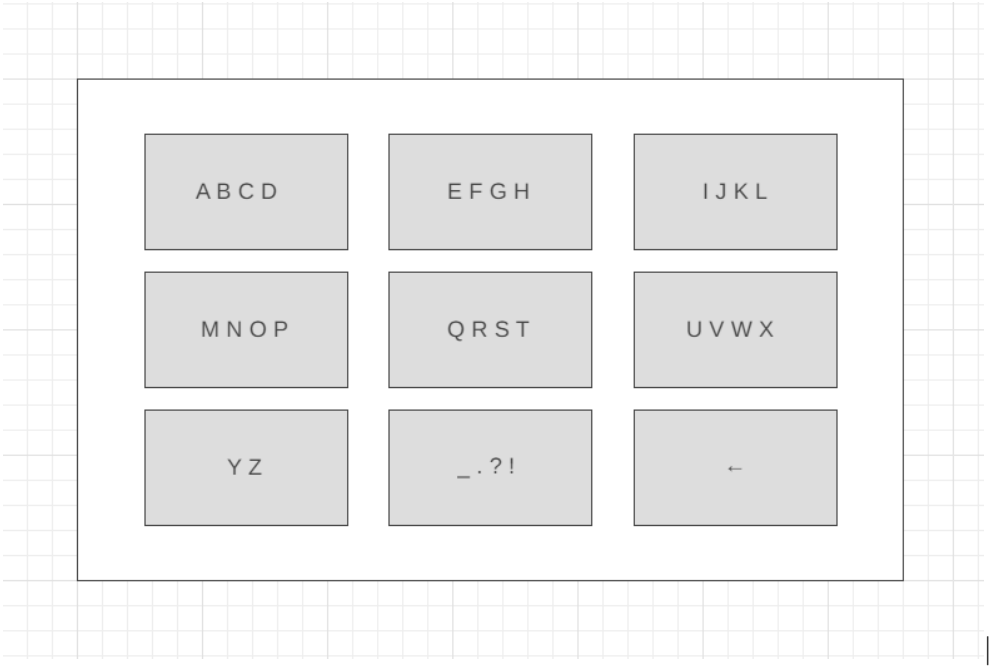
Abeceda má 26 písmen. Pokud k tomu připočteme znak mezery, tečky, otazníku, vykřičníku a znaku pro mazání (tedy 5 znaků), dostaneme se na prvočíslo 31. Z toho je patrné, že se nedá docílit stejného rozdělení prvků na jednotlivá tlačítka. Pro takové rozdělení bychom museli dostat násobek dvou čísel a tomu se tak v tomto případě nestalo.

Aby návrh vypadal dobře a zároveň splňoval požadavky, jsou rozdělena písmena po čtyřech znacích na jedno tlačítko. Na šest tlačítek je využito dvacet čtyři znaků abecedy. Písmena Y a Z (zbývající prvky) jsou pod samostatným tlačítkem.

Další potřebné znaky jsou rozděleny mezi dvě zbývající tlačítka. První obsahuje čtyři znaky a poslední tlačítko pro mazání pouze jeden znak. Tlačítko zpět je umístěno samostatně. Uživatel se nemusí postupně dostávat přes další znaky v tlačítku, pokud by potřeboval některý znak vymazat a pokračovat v psaní.

Pokud se k prvočíslu 31 přičte ještě jeden znak (konkrétně čárka), potom se dostaneme na číslo 32. Vyjádřeno nejmenším společným násobkem je to 2^5. Jedná se o násobky dvojky. S tím se dá při rozdělování skupin znaků pracovat.

Po vydělení čtyřmi (počet znaků na tlačítko) vyjde osm tlačítek. Toto rozhraní je tedy tvořeno dvěma řádky po čtyřech tlačítkách. Lze se na něj podívat na obrázku č. 7.6. Mísí se zde však interpunkční znaménka a písmena. V poslední části znaků je kombinováno tlačítko pro mazání s interpunkčními znaménky.



Obrázek 7.5: Návrh č. 1 inspirovaný klávesnicí na mobilním zařízení

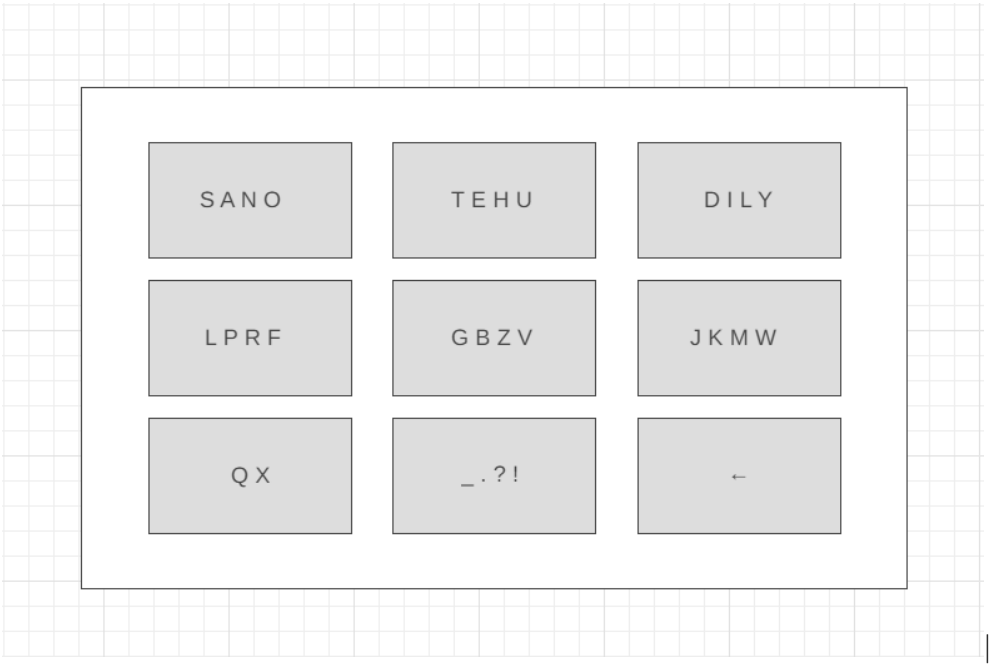
Toto řešení se nezdá být dosti přehledné. V prvním návrhu je znaku pro mazání vyhrazeno samostatné tlačítko. Podle subjektivního názoru autorky je první řešení s devíti tlačítky graficky lepší a je vybráno pro výsledné řešení.

Před implementací samotného rozhraní se práce zaměří na pořadí znaků. V podkapitole 7.2 se rozebírala Dvorakova klávesnice. Pokud se s ní uživatelé naučí, slibuje vyšší rychlost při psaní. Při tvorbě dalšího návrhu 7.7 se zohledňuje pořadí nejvíce používaných písmen. V tomto náčrtu je střídána souhláska a samohláska v samotném tlačítku. U dalšího návrhu 7.8 jsou pro samohlásky vyčleněny dvě tlačítka střídající se na začátku s tlačítky pro souhlásky. Zároveň je opět zohledněno pořadí písmen podle Dvoraka. Začíná se prostřední řadou, pokračuje vrchní a končí písmeny poslední řady. U obou návrhů se počítá s ovládáním, kdy je uživatel po napsání písmene vrácen zpátky na první tlačítko.

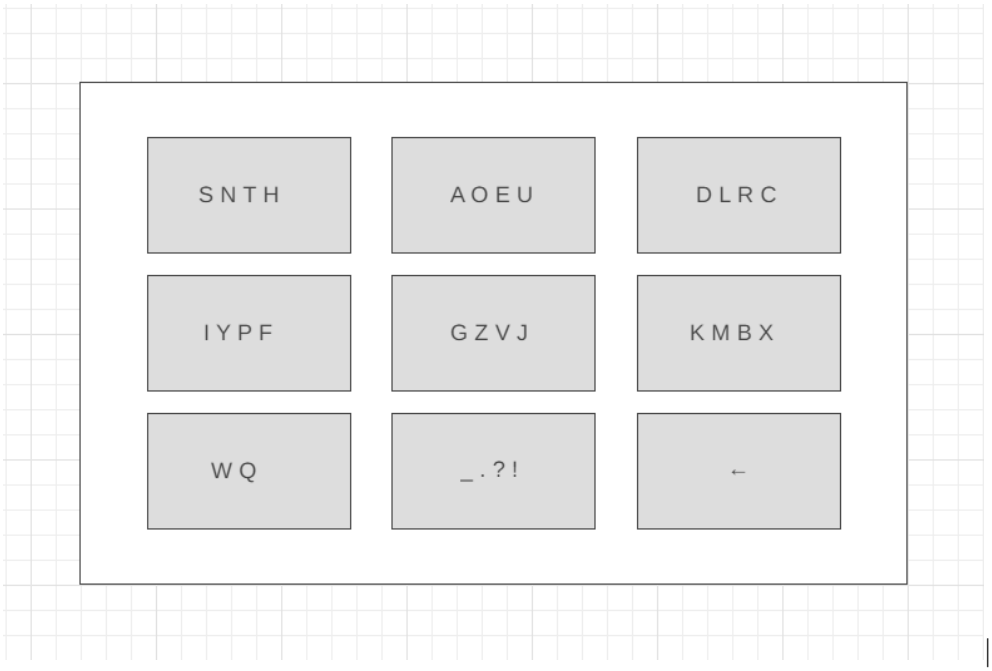
Tyto návrhy zohledňující Dvoraka mají jeden nedostatek. Vytvořené pořadí písmen se musí každý naučit. Uživatel tak na začátku neustále hledá jednotlivá písmena. Při tvorbě rozhraní založeného převážně na mobilní verzi tento nedostatek nevzniká. K dispozici jsou písmena seřazena podle abecedy. Jejich výběr není tak náročný. Pro implementaci byl z tohoto důvodu vybrán první návrh.

Tyto dva návrhy (inspirované klávesnicí Dvoraka) lze později například vyzkoušet na skupině uživatelů, zda se jim s nimi pracuje lépe. Mohlo by se takto

Obrázek 7.6: Návrh č. 2 inspirovaný klávesnicí na mobilním zařízení



Obrázek 7.7: Návrh č. 3 inspirovaný klávesnicí na mobilním zařízení



Obrázek 7.8: Návrh č. 4 inspirovaný klávesnicí na mobilním zařízení

dosáhnout rychlejšího výsledku. Tato myšlenka však není ničím podložena.

Následující návrh (obr. č. 7.9) je vytvořen jako koláčové menu používané u RPG her. Oproti ostatním návrhům obsahuje navíc tlačítko pro čísla, které urychlí jejich psaní. Celkem se skládá z šesti kláves rozdělujících abecedu na čtyři části. Pod dalšími tlačítky se skrývá znak pro mazání nebo třeba interpunkční znaménka.

Tento návrh by mohl být využit nejen pro klávesnici, ale také pro další nabídky ovládané mozkovými signály, pokud bude jejich používání zefektivňovat práci s počítačem.

V návrhu je vidět koláčové menu, ve kterém se aktuálně uživatel nachází v sekci „Samohlásky“.

Obrázek 7.9: Návrh č. 5 koncipovaný jako koláčové menu

Kapitola 8

Realizace

8.1 Implementace a návrh celého systému

Vizí celé práce je zhotovení rozhraní pro osoby s nižší hybností a modelu pro snímání elektrické aktivity mozku. Součástí práce je implementace systému, který je schopen na základě EEG dat detekovat jednotlivé příkazy ovládající rozhraní podobné klávesnici.

Původní myšlenka pracovala i s tvorbou samotné aplikace pro vytváření modelu strojového učení. Pro časovou náročnost celého řešení je nakonec využita aplikace poskytovaná od Emotivu - EmotivBCI, ve které je již vytvoření takového modelu implementováno.

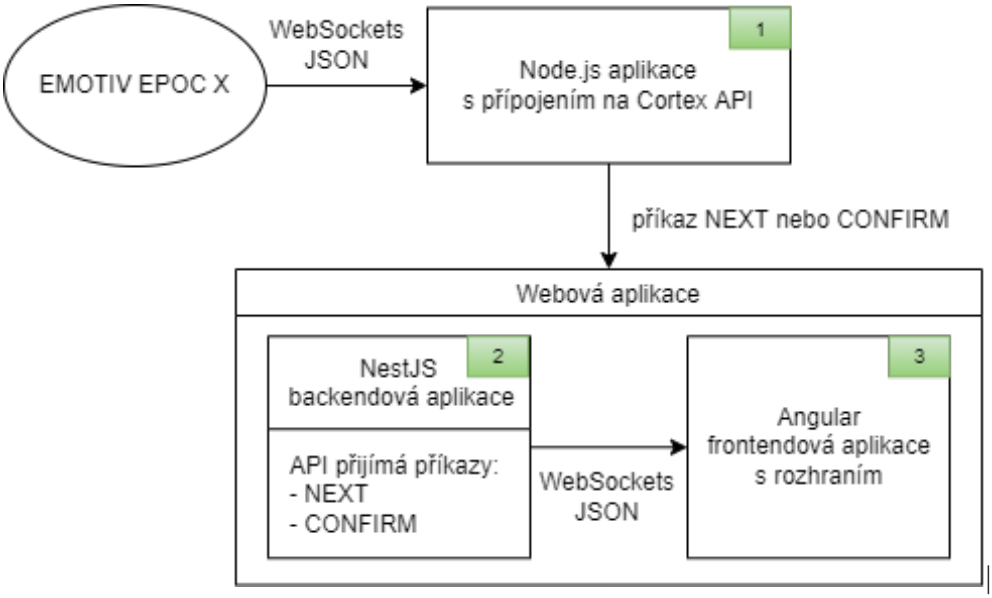
V EmotivBCI se model trénuje na jednotlivé příkazy a pomocí Cortex API, které je poskytované od Emotivu, se systém připojuje k zařízení a dostává data pro ovládání rozhraní.

Součástí implementační části je i průzkum učících algoritmů vhodných pro vytvoření modelu s EEG daty. Ten je popsán v podkapitole 8.5.

Po prozkoumání fungování Emotiv headsetu a jeho příslušenství je navržen systém pro ovládání rozhraní (vytvořeného v této práci 8.3). Navržený systém je možné vidět na obrázku 8.1.

Zařízení (Emotiv Epoc X headset) je propojeno s počítačem pomocí USB a technologie Bluetooth. Přes Emotiv Launcher a vytvořenou aplikaci (Node.js aplikace; na obrázku 8.1 označena číslem 1) pracující s Cortex API je poslán ze zařízení příkaz pomocí WebSockets popsané v práci v kapitole výše. Ten se v této aplikaci rozezná na základě zaslaných JSON dat. Podle obsažené informace v JSON datech se zavolá endpoint na backendové části další vytvořené webové aplikace (na obrázku aplikace označená číslem 2). Zde se vytvoří zpráva s příkazem. Ten je zachycen na frontendové části webové aplikace (obrázek 8.1, aplikace číslo 3) za pomoci technologie WebSockets. Tímto způsobem je ovládána speciální klávesnice (obrázek 8.3).

Moduly (jednotlivé aplikace) jsou vytvořeny tak, aby se daly v budoucnu jednoduše nahradit. Například aplikace komunikující s Emotiv headsetem pro-



Obrázek 8.1: Návrh systému pracujícího se zařízením Emotiv Epoc X prostřednictvím Cortex API a WebSockets technologie

střednictvím Cortex API bude nahrazena vlastním řešením. V něm se se upotřebí znalosti z průzkumu učících algoritmů specializujících se na EEG signály, který je obsažen v této práci v podkapitole 8.5.

Vysvětlení jednotlivých částí aplikace a bližší popis jejich fungování je v následujících sekcích.

8.2 Implementace webové aplikace

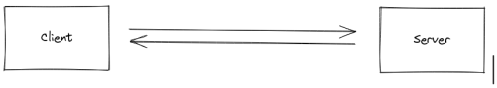
Webová aplikace je program, který se nachází na vzdáleném serveru a je poskytován prostřednictvím Internetu přes rozhraní prohlížeče.

Velkou výhodou takových aplikací je, že není nutné stahovat program na klienta. Aplikace je přístupná prostřednictvím sítě přes URL adresu. Uživatel se k aplikaci snadno dostane skrz webové prohlížeče, jako je třeba Safari, Mozilla Firefox či Google Chrome.

Funkčnost aplikace je závislá na databázi, webovém a aplikačním serveru. Webový server přijímá požadavky od klienta a aplikační server zajišťuje dokončení této úlohy. Databáze slouží k ukládání dat potřebných k fungování webové aplikace.

Většina webových aplikací je na klientské části napsána v jazyce JavaScript, HTML a CSS (kaskádové styly), na straně serveru se pak využívají skripty v jazyce Python nebo třeba Java.

Výhodou webové aplikace je tedy to, že jí není nutné instalovat (jak je již zmíněno v odstavcích výše). Stejná verze programu je dostupná všem uživa-



Obrázek 8.2: Oboustranná komunikace klienta a serveru [10]

telům. Právě pro dostupnost webové aplikace byla zvolena tato technologie. Lze se k ní dostat jak přes počítač, tak třeba přes tablet nebo mobilní zařízení. Cílovým zařízením systému je ale pouze počítač. O dalších rozšířeních se zatím neuvažuje.

Implementovaná webová aplikace je rozdělena na dvě samostatné části: frontendovou (klientská strana) a backendovou (serverová).

8.2.1 Socket.IO knihovna pro komunikaci mezi frontendovou a backendovou částí

Socket.IO je knihovna, která umožňuje oboustrannou komunikaci (viz. obrázek 8.2) klienta a serveru, založenou na událostech s minimálním zpožděním (použití pro real-time aplikace), jak je také napsáno v dokumentaci [10].

Knihovna je postavena na protokolu WebSocket popsaném v kapitole Webové aplikace pracující v reálném čase. Socket.IO knihovna je nadstavba WebSocket technologie [10]. Socket.IO přidává každému packetu navíc ještě metadata. To je rozdíl mezi ním a samotným využitím WebSocket.

Knihovna se nedá využít jako služba na pozadí u mobilních aplikací, protože je potřeba neustále udržovat spojení na základě TCP. Při použití na mobilním zařízení by způsobila u telefonu rychlé vybíjení baterie. Výsledná práce je určena pouze na počítač, proto může být pro práci použita. S rozšířením na mobilní zařízení se nepočítá.

Socket.IO umožňuje implementaci s pomocí jazyka JavaScript. To znamená, že ji lze použít právě spolu s Angularem a s NestJS frameworky.

Bez použití knihovni Socket.IO (tedy při psaní kódu za využití obyčejného WebSocketu) je nutné si napsat větší část kódu sám např. připravit kód pro znovupřipojení. To je v Socket.IO knihovně vyřešeno za vývojaře. Nástroj tak usnadňuje programátorovi práci. Výhodou je i přehledná dokumentace [49].

8.2.2 Angular a frontendová část

Frontendová část aplikace je naprogramována pomocí frameworku Angular. Angular byl vyvinut společností Google. Framework patří mezi jeden z nejoblíbenějších frameworků pro tvorbu frontendu ve světě, tak jako Vue a React.

První verzí tohoto frameworku je Angular.js. Ten se od dalších verzí Angularu značně odlišuje. Angular.js například podporuje jazyk JavaScript. Oproti tomu Angular 1 a výše podporuje jazyk TypeScript.

Obecně je Angular o něco složitější na naučení než například Vue, kde je učící křivka značně příjemnější. Nicméně autorka je na práci v tomto frameworku zvyklá, proto je také vybrán jako technologie, ve které je aplikace vytvořena.

V úvodu je zmíněno, že se tyto aplikace píší v jazyku JavaScript. Angular však pracuje s jazykem TypeScript, který je nadstavbou jazyka JavaScript. Odlišuje se od JavaScriptu například deklarací typů, kdy je kompilátor schopen navíc kontrolovat tento datový typ a upozornit na nevhodné použití.

Frontendová aplikace se nachází na adrese `https://github.com/terez2/keyboard`. Autorka pracuje s yarn package managerem (správcem balíčku). Po stažení repozitáře do vlastního zařízení je potřeba si nainstalovat závislosti projektu pomocí příkazu „yarn install“.

Ve složce package.json jsou definovány skripty, které lze nad aplikací spustit. Jedním z nich je skript pro start, který se spustí příkazem „yarn start“. Po spuštění běží aplikace na adrese `http://localhost:4200/`.

V repozitáři se nachází složka src, ve které jsou všechny komponenty, service a další třídy. Mimo složku src se nachází další konfigurační soubory jako je například karma.conf.js (Karma je nástroj, který nám umožňuje spouštět v prohlížeči testy Jasmine z příkazového řádku) nebo soubor tsconfig.json, který určuje kořenové soubory a možnosti kompilátoru potřebné ke kompilaci projektu.

V souboru src/index.html je definována webová stránka. Pomocí elementu <app-root></app-root> se v ní mohou měnit jednotlivé komponenty/stránky (kód níže).

```
1  <!doctype html>
2  <html lang="en">
3  <head>
4    <meta charset="utf-8">
5    <title>Keyboard</title>
6    <base href="/">
7    <meta name="viewport" content="width=device-width, initial-
         scale=1">
8    <link rel="icon" type="image/x-icon" href="favicon.ico">
9  </head>
10 <body>
11   <app-root></app-root>
12 </body>
13 </html>
```

Ve složce src/app se nachází vytvořené komponenty (ve složce components) nebo třeba service (ve složce services) pro komunikaci s API.

V souboru src/styles.scss jsou definované globální styly použité pro celou aplikaci. Další soubory se styly jsou umístěny u každé z komponent. Třída RadialComponent má tedy styly definované v souboru radial.component.scss. Styl se komponentě přidá v dekorátoru @Component ve styleUrls viz. kód níže.

```
import {ChangeDetectionStrategy, ChangeDetectorRef, Component,
    EventEmitter, Input, Output} from '@angular/core';
import {ChatService} from "../../sevices/chat.service";
import {BaseKeyboardComponent} from "../base-keyboard.component";

@Component({
selector: 'app-radial',
  templateUrl: './radial.component.html',
  styleUrls: ['./radial.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush,
})
export class RadialComponent extends BaseKeyboardComponent {

  constructor(protected chatService: ChatService, protected cd:
      ChangeDetectorRef) {
       super(chatService, cd);
  }
}
```

Ve složce services se nachází ChatService. Tato třída slouží pro komunikaci s backendovou částí. V konstruktoru je pomocí dependency injection definována proměnná typu Socket, díky které lze emitovat (vyvolávat, posílat) zprávy (v metodě sendMessage) na server nebo je přijímat (metoda getMessage) ze serveru, jak je vidět v kódu níže.

```
import { Injectable } from '@angular/core';
import {Socket} from "ngx-socket-io";

@Injectable({
  providedIn: 'root'
})
export class ChatService {
  constructor(private socket: Socket) {}

  sendMessage(msg: string) {
    this.socket.emit('message', msg);
  }

  getMessage() {
    return this.socket.fromEvent('message');
  }
}
```

Metoda sendMessage je zatím jen předpřipravená a nevyužívá se. Metoda getMessage se používá v BaseKeyboardComponent třídě v souboru src/components/base-keyboard.component.ts v konstruktoru. Kód, kdy se tato metoda volá, je níže.

```
this.chatService.getMessage().pipe(untilDestroyed(this)).
    subscribe((data: any) => {
  this.updateDataWithoutTimer(data);
  this.cd.detectChanges()
})
```

Metoda getMessage vrací Observable, takže se s využitím subscribe čeká na zprávy ze serveru. Po přijetí události se spustí kód, který zajišťuje ovládání tlačítek.

Při použití subscribe() metody je potřeba i unsubscribe(). Aby se nemusela data ukládat do proměnné a následně pak na nich volat unsubscribe() v ngOnDestroy() lifecycle funkci komponenty, je využita knihovna @ngneat/until-destroy. Díky ní stačí pouze v pipe() zavolat untilDestroyed(this). To znamená, že se po odstranění této komponenty zavolá metoda unsubscribe(). Při využití tohoto kódu v jakékoli komponentě je nutné přidat před dekorátor komponenty dekorátor @UntilDestroy().

Aplikace napsaná pomocí NestJS a klientská část napsaná v Angularu spolu komunikují na základě WebSocketů. Tato komunikace je zajištěna využitím knihovny Socket.IO popsané v části 8.2.1. V klientské aplikaci je naistalována knihovna ngx-socket-io. Po instalaci je potřeba vložit inicializaci modulu SocketIoModule do složky app.module.ts i s konfigurací. Zde se například nastavuje adresa serveru. Konfigurace serveru je níže v kódu na řádku č. 8. Modul je přidán do imports na řádku č. 17.

```
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppComponent } from './app.component';
5  import {AppRoutingModule} from "./app-routing.module";
6  import {SocketIoConfig, SocketIoModule} from "ngx-socket-io";
7
8  const config: SocketIoConfig = { url: 'http://localhost:3000',
       options: {} };
9
10 @NgModule({
11   declarations: [
12     AppComponent
13   ],
14   imports: [
15     BrowserModule,
16     AppRoutingModule,
17     SocketIoModule.forRoot(config)
18   ],
19   providers: [],
20   bootstrap: [AppComponent]
21 })
22 export class AppModule { }
```

8.2.3 Způsoby ovládání aplikace

Aplikace je kontrolována pouze dvěma příkazy. V nich je uživatel schopen určit, zda se chce posunout na další tlačítko nebo potvrdit výběr.

V případě, že uživatel potřebuje přemístit ukazatel z tlačítka na další tlačítko, pošle na klientskou aplikaci data ve tvaru {command: “next”} tj. další.

V situaci, kdy chce něco potvrdit, zašle zprávu {command: “confirm”}
tj. potvrdit.

Těmito dvěma příkazy je schopna aplikace rozlišit, zda má posunout ukazatel na další tlačítko, eventuálně na další znak nebo zda chce uživatel písmeno vybrat, tedy napsat.

Ve frontendové části aplikace je implementováno rozhraní, pro které jsou vytvořeny dva typy ovládání. Prvním typem je ovládání s časovačem. Při inicializaci aplikace je ukazatelem vybráno první tlačítko. Z tohoto tlačítka se lze posunout na další tlačítko příkazem NEXT.

Pokud se rozhodne uživatel vybrat tlačítko, musí jeho výběr potvrdit příkazem CONFIRM. Hodnota času je nastavena na 2s. V tuto chvíli se započne odpočet a vybere se první znak v tlačítku. Pokud nepřijde žádný další signál, je po vypršení časového intervalu znak vybrán. Zbývající čas se nastaví opět na iniciální hodnotu 2s a znovu se spustí odpočet.

V případě, že aplikace zachytí příkaz CONFIRM, posune ukazatel na další znak ve vybrané části. Pokud zachytí příkaz NEXT, zruší se výběr znaku a posune se ukazatel tlačítek. Znaky nejsou vybrány a je umožněno se pohybovat na další sekce. Odpočet se zastaví a pokračuje opět až při vybírání znaků v tlačítku.

Po otestování aplikace spolu s EEG headsetem se došlo k závěru, že je lepší druhý způsob ovládání bez časovače, který je popsán v odstavci níže. Z tohoto důvodu je kód pro ovládání rozhraní s časovačem možné dohledat, ale pro jeho spuštění je třeba úprava kódu. Ovládání aplikace tedy nelze jakkoli za běhu přepínat (například tlačítkem).

Druhým způsobem je varianta bez časovače, kdy se mezi tlačítky dá pohybovat pomocí příkazu NEXT a potvrzovat výběr pomocí příkazu CONFIRM. Pro přechod na další tlačítko je potřeba zavolat příkaz NEXT, pro výběr jednoho z tlačítek je nutné zaslat CONFIRM. Pro pohyb v sekci tlačítka je ovládání totožné. Lze se v ní pohybovat mezi jednotlivými znaky pomocí příkazu NEXT a po dosažení cílového znaku lze potvrdit výběr příkazem CONFIRM.

Původně bylo ovládání navrženo tak, že ukazatel zůstal po napsání písmene na vybraném znaku. Uživatel tak pokračoval ve vybraném tlačítku. Ze sekce znaků se musel dostat tak, že vyslal příkaz NEXT tolikrát, jako je počet znaků v tlačítku plus jeden příkaz NEXT navíc. Následně se mohl opět rozhodovat, který segment znaků dále vybere. Po sérii testů bylo toto chování upraveno (popis úprav v části 8.2.4).

Tento způsob ovládání je defaultně nastaven v kódu aplikace

8.2.4 Výhody a nevýhody navržených typů ovládání a výběr výsledného ovládání

Výhodou ovládání s časovačem je rychlá možnost opuštění tlačítka příkazem NEXT. Na druhou stranu je uživateli trochu nepříjemné, že se spouští odpočet pro výběr tlačítka. Je potom nutné rychle reagovat, než čas vyprší. Časovač se

spouští jen v případě, že už je rozhodnuto o výběru tlačítka a vybírá se pouze znak, který bude napsán. Interval zbývající do výběru znaku lze upravit podle reakční doby uživatele.

Jelikož se však při testování aplikace s EEG headsetem musí uživatel soustředit na své myšlenky, je časovač rušivým elementem a dostává zákazníka pod tlak. Z tohoto důvodu je v řešení implementována verze bez časovače.

Při práci s rozhraním bez časovače není kladen důraz na rychlou reakci. Původně bylo ovládání nastaveno tak, že uživatel musel po výběru znaku projít všechny ostatní znaky ležící za tímto znakem a zaslat jeden signál navíc pro vyskočení z nabídky. Až poté se mohl opět dále pohybovat mezi tlačítky.

Vybraný návrh ovládání (bez časovače) rozhraní prošel testy na menším počtu respondentů. Výsledkem tohoto testu bylo vylepšení ovládání, které umožňuje rychlejší psaní a přechod mezi sekcemi. Nově je po výběru znaku uživatel vždy vrácen z výběru mezi znaky v tlačítku do výběru tlačítek, kdy se opět může pohybovat po sekcích abecedy (tlačítkách). Tím se odstranil problém s nutností projít zbylé znaky a psaní pomocí rozhraní se stalo rychlejším.

8.2.5 Uživatelské rozhraní

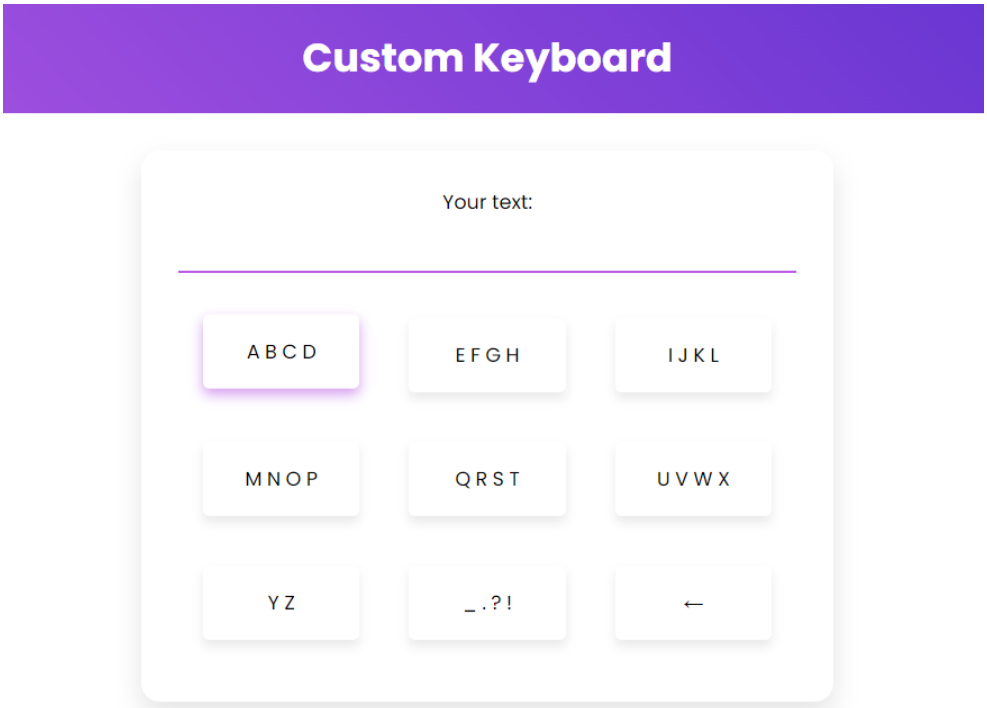
Vzhled výsledného rozhraní (obr. č. 8.3) je uzpůsoben samotnému ovládání. Návrh celého rozhraní vychází z klávesnice na mobilním zařízení (obr. č. 7.3). Po spuštění frontendové aplikace ho lze vidět na adrese `http://localhost:4200/`. V porovnání s normální počítačovou klávesnicí 7.1 se lze v tomto rozhraní rychleji pohybovat - po úsecích 4 znaků.

V rozhraní se nepočítá s číslicemi jako s dalšími symboly na tlačítkách. Psaní těchto znaků není možné, uživatel musí číslo napsat slovy. V původním návrhu bylo zapracováno i tlačítko pro číslice. To se však ukázalo v tomto návrhu jako nadbytečné a bylo odstraněno.

Přidání číslic je jedno z možných rozšíření rozhraní na obrázku č. 8.3. Možná by nebylo ani potřeba přidávat speciální tlačítko pro číslice, jako spíše přidat funkcionalitu, kdy se za předpokladu obdržení, například rychlého sledu dvou potvrzujících signálů, píše místo písmene v tlačítku jeho číslo (pořadí tlačítka).

Druhý implementovaný vzhled výsledného produktu je možné vidět ve frontendové aplikaci na adrese `http://localhost:4200/radial`, zde na obrázku č. 8.4. Celá nabídka je upravena do koláčového menu. V menu jsou znaky rozdělené na šest sekcí a to: tvrdé souhlásky, měkké souhlásky, samohlásky, interpunkce, další možnost (sekce aktuálně obsahuje symbol pro mazání textu) a oproti první implementaci obsahuje navíc čísla.

Design je navržen na zařízení počítače. Je částečně responzivní, ale pouze do určité šířky displeje. V tuto chvíli se nepočítá s rozšířením aplikace na tablet ani na mobilní zařízení.



Obrázek 8.3: Vytvořené uživatelské rozhraní podobné klávesnici mobilního zařízení

8.2.6 NestJS a backendová část

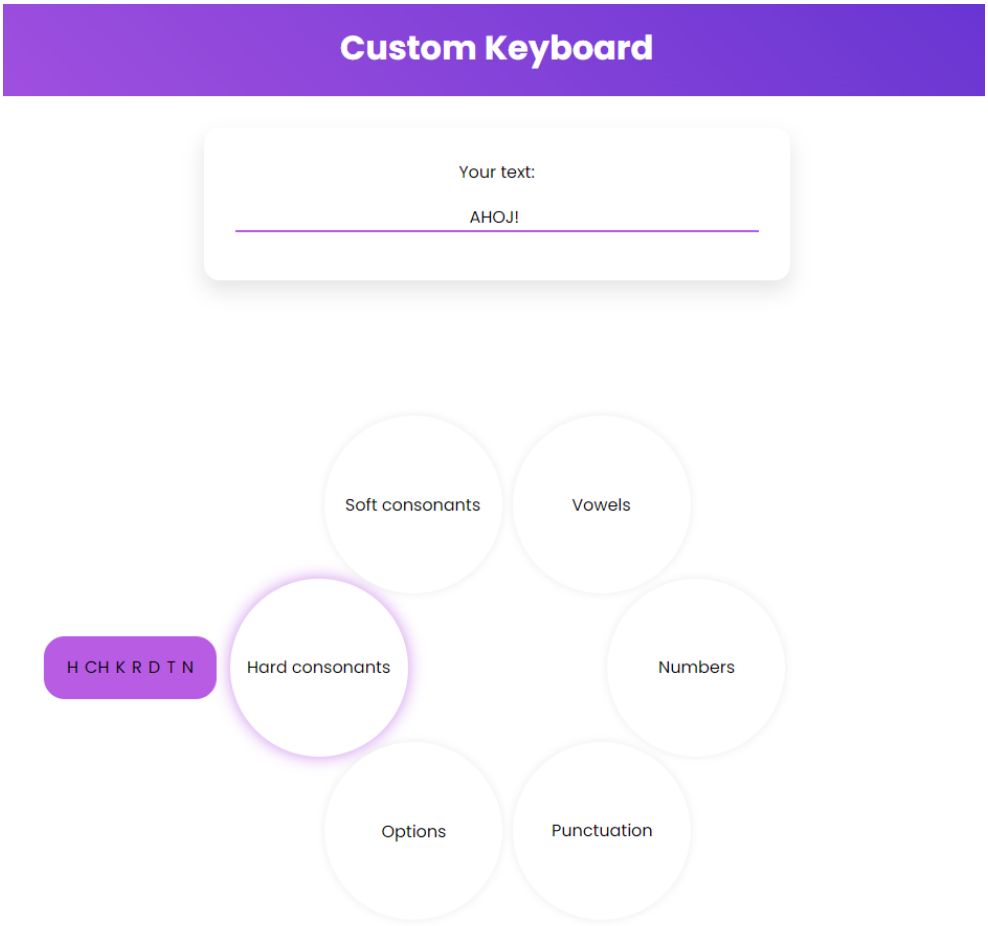
Jak je definováno v dokumentaci NestJS [49], jedná se o framework pro vytvoření Node.js server-side aplikací. NestJS využívá progresivní JavaScript, ale podporuje i TypeScript a kombinuje prvky OOP (objektově orientovaného programování), FP (fundamentálního programování) a FRP (funkcionálního reaktivního programování).

NestJS využívá HTTP Server jako je Express, ale je možné ho nakonfigurovat tak, aby používal také Fastify. Umožňuje používat nesčetné množství modulů třetích stran.

Framework poskytuje předpřipravenou aplikační architekturu, která je snadno testovatelná, škálovatelná a snadno udržovatelná. Celá architektura je inspirovaná frameworkem Angular, který je použit k tvorbě frontendové části. To je také jedním z důvodů, proč je NestJS zvolen pro serverovou část.

Se znalostí frameworku Angular se s frameworkem NestJS pracuje velmi dobře. Frontendový vývojář je schopen si ve velmi krátkém časovém úseku vytvořit funkční backendovou aplikaci.

Na adrese `https://github.com/terez2/keyboard-backend` se nachází



Obrázek 8.4: Vytvořené uživatelské rozhraní jako koláčové menu

backendová část aplikace. V repozitáři jsou konfigurační soubory a složka src. V ní jsou uloženy soubory se třídami jako jsou AppGateway, AppContoller a další. Projekt se spustí příkazem „yarn start“.

Pro spuštění se dá využít i příkaz „yarn start:dev“, který má navíc „–watch“ flag. Díky tomuto flagu jsou zdrojové soubory, které se uloží se změnami, automaticky kompilovány, aniž by bylo nutné aplikaci znovu spouštět. Příkaz se hodí hlavně při vývoji.

8.2.7 Backendová část a komunikace s frontendem

Tato podkapitola obsahuje popis práce s NestJS a Socket.IO knihovnou.

Pro WebSockets v NestJS vznikne brána vytvořením třídy, která se anotuje dekorátorem @WebSocketGateway. V této aplikaci ji lze nalézt v souboru

app.gateway.ts. Z technického hlediska jsou brány nezávislé na platformě, díky čemuž jsou po vytvoření adaptéru kompatibilní s jakoukoli knihovnou WebSockets.

Ve výchozím nastavení jsou podporovány dvě platformy WS: socket.io a ws (viz. dokuementace [49]). V aplikaci je vybrána knihovna Socket.IO, nicméně si lze napsat i vlastní adaptér.

Brána naslouchá na portu 80 stejně jako ostatní HTTP servery. Příchozí zprávy jsou obslouženy v metodě handleMessage s dekorátorem @SubscribeMessage('message'). Tato metoda proběhne vždy, když je zpráva serverem zachycena.

```
1  @SubscribeMessage('message')
2  handleMessage(@MessageBody() message: string): any {
3    this.logger.log('Message received', message);
4
5    this.server.emit('message', message);
6
7    return this.app.message$.pipe(
8      map((message) => ({ event: 'message', data: message })),
9    );
10 }
```

Aplikace je tvořena AppModulem v app.module.ts, kde se definují jednotlivé části aplikace. Je zde registrována také brána.

Brána se umisťuje do sekce providers. Pro přístup k nativní instanci serveru se používá dekorátor @WebSocketServer s názvem a typem proměnné.

```
1  @WebSocketServer()
2  server: Server;
```

V souboru app.controller.ts se nachází Controller pro posílání zpráv. Jsou zde definovány dvě GET metody s cestou next a confirm. Ty se dají použít zavoláním localhost/next nebo localhost/confirm. Každá metoda pak zavolá appService definovaný v konstruktoru třídy AppController (instance se získá na základě dependency injection - DI).

```
import { Controller, Get } from '@nestjs/common';
import { AppService } from './app.service';

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getHello(): string {
    return this.appService.getHello();
  }

  @Get('next')
  next(): string {
    this.appService.sendMessage('next');
    return 'next';
  }

  @Get('confirm')
  confirm(): string {
    this.appService.sendMessage('confirm');
    return 'confirm';
  }
}
```

Přesněji se zavolá metoda sendMessage s typem zprávy. Pokud se podíváme na třídu AppService v souboru app.service.ts, tak v ní nalezneme messageSubject proměnnou, která je inicializovaná jako RxJS Subject. Proměnná message$ je inicializována jako RxJS Observable a lze ji tedy odebírat (poslouchat, výstižněji anglicky - subscribe).

```
import { Injectable } from '@nestjs/common';
import { Subject } from 'rxjs';

@Injectable()
export class AppService {

  private readonly messageSubject = new Subject<{ command: string
      }>();

  readonly message$ = this.messageSubject.asObservable();

  sendMessage(value: string): void {
    this.messageSubject.next({ command: value });
  }
}
```

Je zde definována i metoda sendMessage. Ta je volána z AppController třídy, jak je zmíněno výše. Pomocí ní je všem odběratelům (ti, co poslouchají skrz bránu) poslána zpráva. V tomto případě je sem poslán buď příkaz NEXT nebo CONFIRM.

Po spuštění obou aplikací - aplikace vytvořené pomocí NesjJS frameworku a aplikace frontendové, je při zavolání endpointu GET /confirm poslána zpráva

na frontendovou aplikaci. Zde se podle vybraného typu ovládání vybere daný znak a vypíše se na obrazovku nebo se posune ukazatel tlačítka/znaku. Tímto způsobem je rozhraní ovládáno.

Při zavolání GET /next je na frontendu přijata zpráva s informací {command: next} a v rozhraní se například posune ukazatel z jednoho tlačítka na druhé.

8.3 Práce s Emotiv headsetem a trénink příkazů

Zvoleným EEG zařízením pro práci je přístroj Emotiv Epoc X (obr. č. 3.3), ke kterému má autorka přístup.

Společnost Emotiv nabízí i vlastní software, např. aplikaci Emotiv BCI. V aplikaci Emotiv BCI je uživateli umožněno pohybovat kostkou prostřednictvím myšlenky. Celá aplikace je rozdělena na dvě části. V první části uživatel vybere pohyb kostky, který chce trénovat. Po dostatečném počtu opakování daného pohybu je možné přejít do druhé části. Ta funguje jako zkušební mód, ve kterém lze naučené příkazy vyzkoušet – pohybovat s kostkou pouhou myšlenkou. Kostkou lze rotovat, posouvat ji dozadu, dopředu atp. Tento systém autorku inspiroval k vytvoření vlastní aplikace.

Dále Emotiv nabízí software pro 3D vizualizaci mozku v reálném čase - BrainViz. Tato aplikace ukazuje, která část mozku je aktivní. Je na 7 dní volně dostupná a po vypršení tohoto času je zpoplatněna.

EmotivPro je kompletní sada nástrojů pro neurovědní výzkum. Umožňuje získat a analyzovat EEG data v jednom softwarovém prostředí. U aplikace je nutné si zaplatit licenci. EmotivLabs je platforma, ve které se můžete zúčastnit zábavných neurovědních her.

8.3.1 Emotiv Epoc X

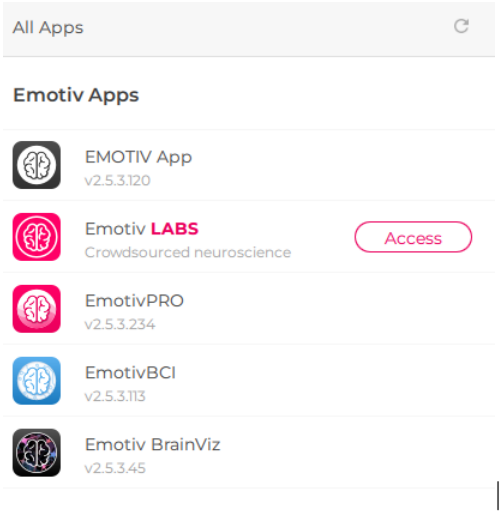
Emotiv Epoc X 3.3 je zařízení, které je schopné pracovat na čtrnácti EEG kanálech: AF3, F7, F3, FC5, T7, P7, O1, O2, P8, T8, FC6, F4, F8, AF4. Obsahuje šestnáct elektrod, které snímají mozkovou aktivitu.

Pro propojení zařízení a počítače je použita technologie Bluetooth® SMART (také známý jako Bluetooth 4.0 Low Energy Protocol), podpora pro Bluetooth® 5.0 bude k dispozici s nadcházejícími aktualizacemi softwaru a firmwaru.

Zařízení není potřeba připojovat kabelem k počítači. Balení obsahuje USB přijímač s pásmem 2,4GHz, který se zapojí do počítače a prostřednictvím něj je zařízení schopné se připojit.

Baterie zařízení Emotiv Epoc X podle výrobců vydrží více jak devět hodin a dokáže snímat pohyb hlavy v devíti různých úhlech. Šířka pásma EEG signálů je 0,16 - 43 Hz. Váží pouze 170g.

Levnější variantou tohoto zařízení je Insight 2.0, ten však disponuje pouze pěti kanály EEG signálu.



Obrázek 8.5: Seznam dostupných aplikací společnosti Emotiv ve spouštěcím programu

Data pro popis zařízení jsou čerpána z dokumentace zařízení na stránce společnosti Emotiv [3]

8.3.2 Registrace a instalace aplikací od společnosti Emotiv

Pro každé zařízení je součástí jeho balíčku i unikátní klíč - EmotivID. Pomocí něj se uživatel přihlašuje do systému spolu s ním vytvořeným heslem. Po přihlášení je možné si stáhnout pět aplikací: Emotiv App, Emotiv Lab, EmotivPRO, EmotivBCI, Emotiv BrainViz. Jsou zachycené na obrázku 8.5.

Některé z těchto aplikací jsou dostupné zdarma na pouze omezený čas a je tak nutné si u společnosti koupit roční předplatné, aby se s nimi dalo dále pracovat. Pro účely této práce je využita pouze volně dostupná verze.

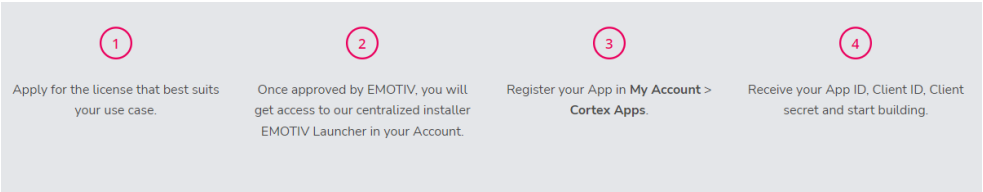
Emotiv App je launcher (spouštěcí program), který je potřeba pro celou práci s Emotiv zařízením. Veškerá konfigurace a připojení headsetu probíhá skrze tuto aplikaci.

Potřebnou aplikací pro tento projekt je EmotivBCI pro trénování jednotlivých příkazů.

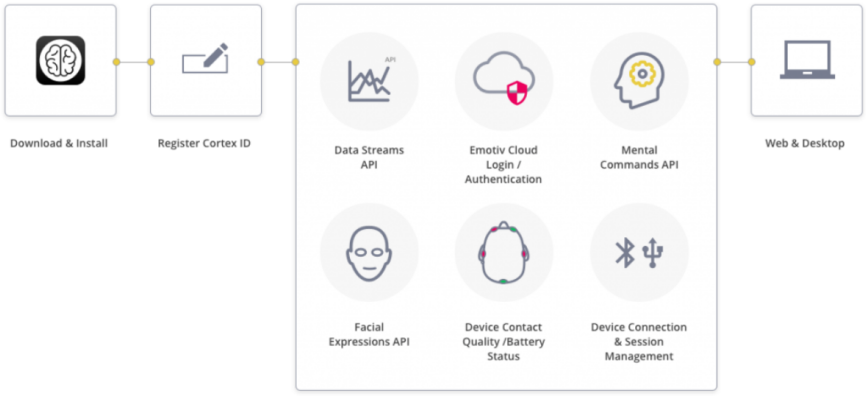
Vzhledem k tomu, že se pracuje pouze s volně dostupnou verzí, není možné pracovat s čistým EEG signálem. Ten je přístupný pouze ve verzi PRO, která je placená. Pro volně dostupnou verzi je k dispozici JSON s informací o vysílaném mentálním příkazu.

Prostřednictvím EmotivBCI aplikace se lze spojit s webovou aplikací a ovládat tak uživatelské rozhraní pomocí mentálních příkazů.

Pro vývoj aplikací se zařízeními Emotiv je třeba postupovat podle následujících kroků:



Obrázek 8.6: Kroky potřebné k možnosti vývoje aplikací se zařízením Emotiv



Obrázek 8.7: Schéma fungování Emotiv Cortex API [3]

1. vybrat a požádat o licenci
2. po schválení společností Emotiv je možné nainstalovat si potřebné aplikace
3. registrovat aplikaci pro přístup ke Cortex API
4. po obdržení ID aplikace, klientského ID a klíče začít vyvíjet

Tento manuál lze nalézt i na stránkách společnosti viz. obrázek 8.6.

8.3.3 Emotiv a učící algoritmy

Společnost Emotiv svým uživatelům a programátorům používajícím jejich produkty doporučuje používat Cortex (Software Development Kit - SDK).

Cortex lze používat tak, že se v kódu volá API pro učení a trénuje se prostřednictvím vlastního volání v kódu. To je první způsob, jak si lze pomocí programování natrénovat vlastní profil na určité příkazy. Další možností je použít na trénování příkazů samotnou aplikaci EmotivBCI a následně z ní zachytit pomocí Cortex API mentální příkazy.

Tato aplikace je poskytována zdarma. Dá se v ní pomocí učících algoritmů pod dohledem trénovat jednotlivé příkazy, výrazy obličeje atp. Pro samotného uživatele je potřeba založit profil. V aplikaci je možné vytvořit více profilů. Jeden profil patří jednomu uživateli. Po založení je možné začít trénovat jednotlivé příkazy.

8.3.4 Připojení zařízení

Před spuštěním aplikace se nejprve spustí Emotiv launcher aplikace. Tato aplikace spolu s USB konektorem propojí zařízení s počítačem (na zařízení je potřeba stisknout tlačítko pro zapnutí).

Po propojení zařízení s počítačem přichází fáze úpravy zařízení na hlavě. Headset se musí umístit správně, aby mohl správně snímat mozkovou aktivitu prostřednictvím všech elektrod.

Elektrody je potřeba před použitím zavlažit. Zařízení funguje na bázi solného roztoku. Pro správnou polohu elektrod je k dispozici obrázek zařízení.

Každá elektroda se zbarví podle toho, jak správně je umístěna. Pokud zachytává signál bezchybně, zabarví se zeleně. To znamená, že je elektroda správně umístěna a už s ní není třeba dále hýbat.

Pokud je signál z větší části zachycen, ale připojení není dostatečné, zbarví se elektroda v aplikaci oranžově. U oranžové barvy je potřeba elektrodu jen trochu přitisknout k hlavě nebo mírně posunout. Občas tato barva problikává v důsledku pohnutí hlavy a tedy posunutí samotného zařízení na hlavě. V tuto chvíli stačí už jen málo k dosažení zelené barvy.

Při zaznamenání špatného kontaktu má elektroda barvu červenou. Červená barva se zobrazuje, pokud se elektroda špatně dotýkala pokožky hlavy. Například je umístěna na vlasech a nemůže tak snímat signál.

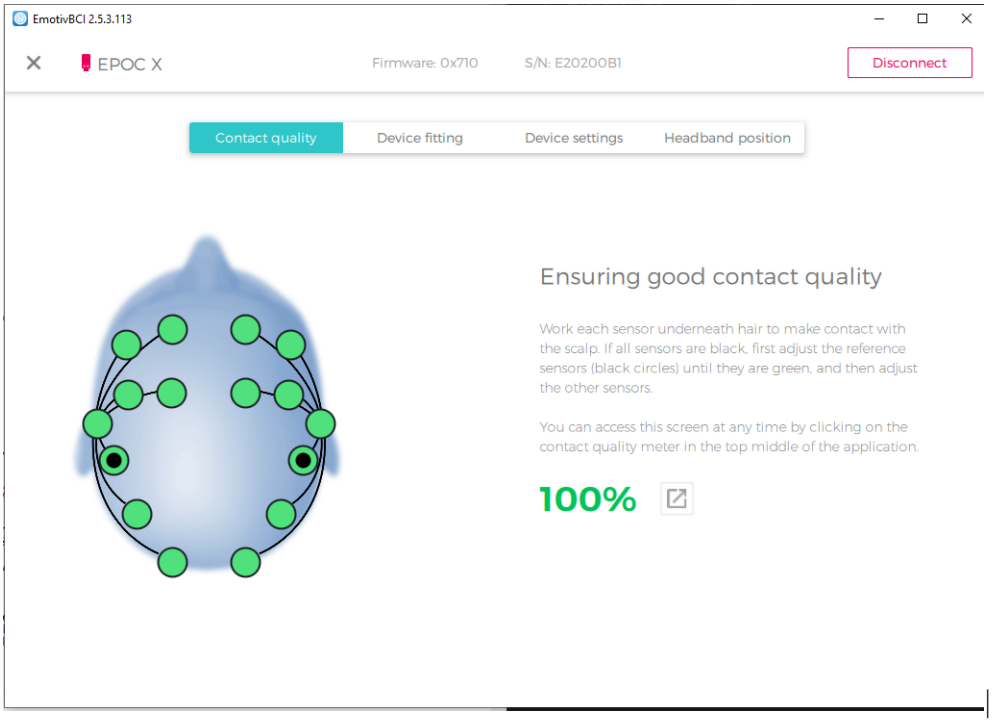
V případě, že se nedaří připojit elektrodu vůbec, zůstává její obraz v aplikaci bez barvy (šedá výplň). Šedé zbarvení je způsobeno převážně tím, že se elektroda nedostatečně zavlažila roztokem a je nutné na ni tento roztok nalít.

Na obrázku 8.8 je vidět správně umístěné zařízení. Oproti tomu je na obrázku 8.9 zařízení ve stavu, kdy je potřeba ještě některé elektrody zavlažit nebo správně umístit na pokožku hlavy.

8.3.5 Trénování příkazů

V aplikaci EmotivBCI se založí profil daného subjektu 8.10. Po vytvoření profilu je možné začít trénovat jednotlivé příkazy. K dispozici jsou předdefinované příkazy: PUSH, PULL, LEFT, RIGHT, LIFT, DROP, ROTATE LEFT, ROTATE RIGHT, ROTATE FORWARDS, ROTATE CLOCKWISE, ROTATE BACKWARDS, ROTATE ANTICLOCKWISE a DISAPPEAR.

Pro použití jednotlivých příkazů je třeba je natrénovat. Prvním stavem, který je v aplikaci doporučeno trénovat, je neutrální postoj. Tento příkaz se



Obrázek 8.8: Ukázka propojení Emotiv Epoc X s počítačem - jeho správné nasazení

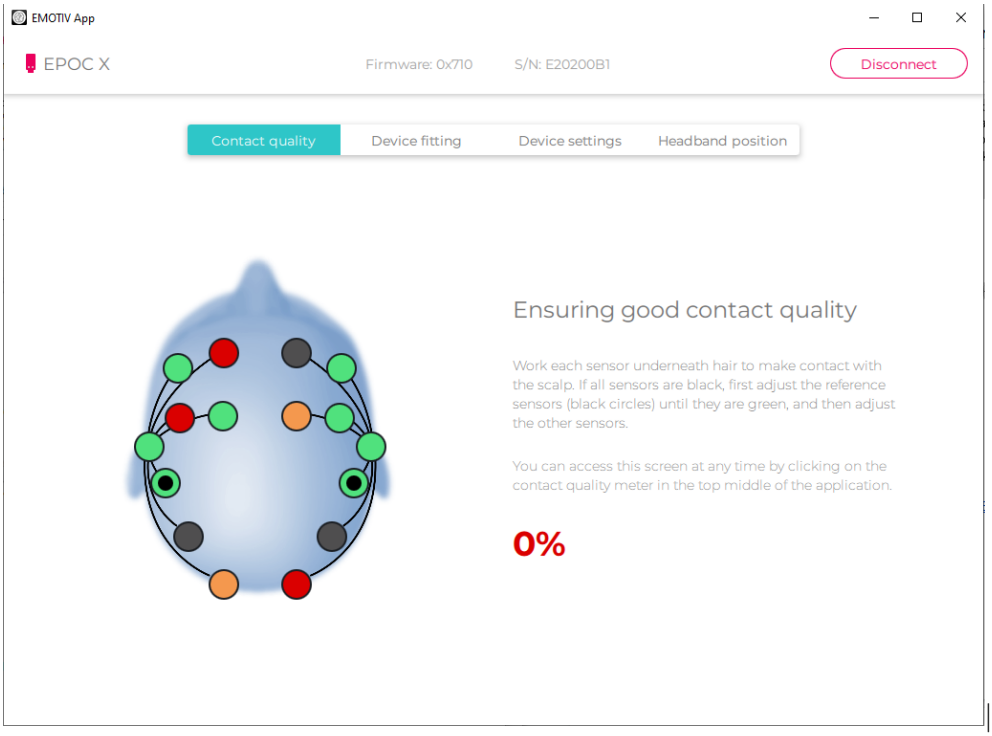
trénuje jednoduše tak, že uživatel v klidu sedí, snaží se nehýbat, nad ničím nepřemýšlet. Sledovaný objekt by měl setrvávat v uvolněné poloze těla i mysli.

Po dostatečném nacvičení příkazu NEUTRAL se může přidat do trénování další příkaz. V tomto případě je součástí aplikace trénink příkazu PUSH. Na obrázku 8.11 je zdokumentovaný průběh práce ve stavu, kdy bylo uskutečněno 28 cvičení příkazu NEUTRAL a 38x byl trénován povel PUSH.

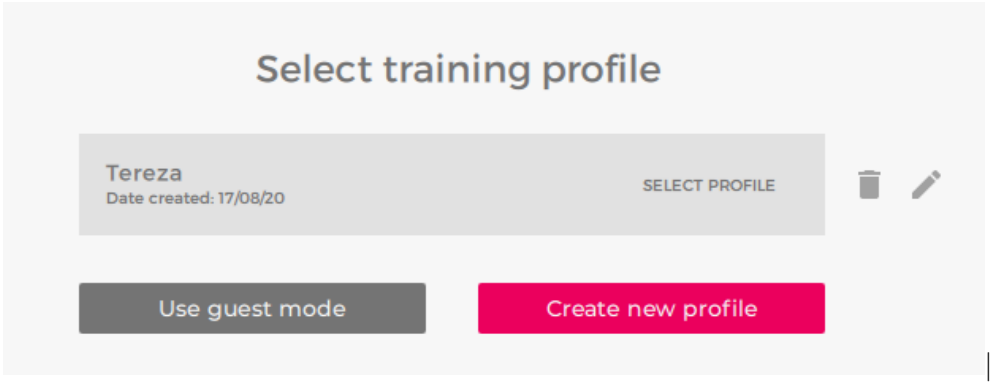
Trénuje se na kostce, která je v aplikaci zobrazena ve 3D scéně a se kterou se v tomto rozhraní pohybuje. Po skončení odpočtu se při spuštění trénování uživatel soustředí na kostku a myslí na něco, při čem se bude kostka posouvat směrem dozadu tj. příkaz PUSH.

Na co bude uživatel při posunování kostky (tréninkovém módu) myslet, je pouze na něm. Nutné je však myslet při stejném příkazu stále na stejnou věc, aby se nestalo, že jednou při trénování příkazu PUSH subjekt myslí na určitý pohyb a jindy na něco diametrálně odlišného.

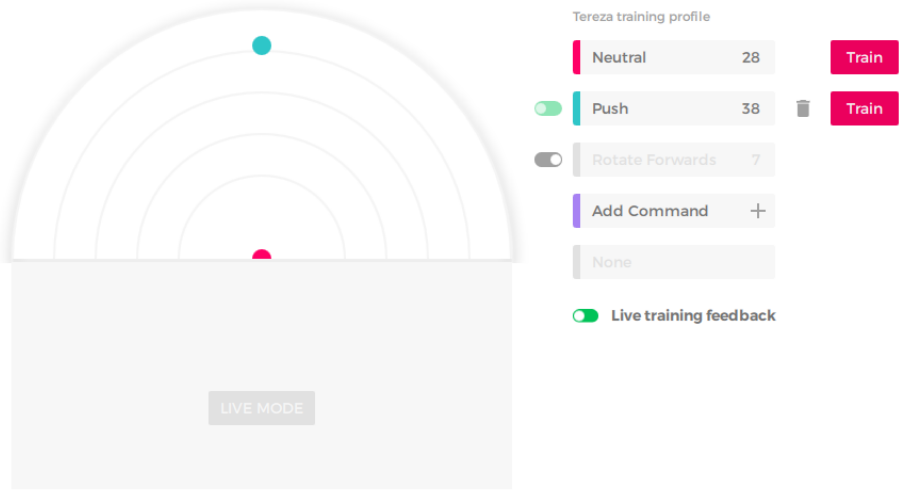
Doporučuje se však volit různé věci pro různé příkazy, aby se od sebe příkazy snadněji odlišily. Pro jeden příkaz zvolit třeba myšlenku „posuň se dozadu“, při dalším myslet na určitou vůni nebo na pohyb končetiny, aby se u každého příkazu aktivovalo jiné centrum a snadněji se odlišilo, co uživatel od aplikace chce.



Obrázek 8.9: Ukázka propojení Emotiv Epoc X s počítačem - nedostatečný signál z elektrod



Obrázek 8.10: Založení profilu v aplikaci EmotivBCI



Obrázek 8.11: Souhrn naučených příkazů v aplikaci EmotivBCI

Po skončení tréninkové lekce aplikace ukáže, jak dobře v ní byl příkaz nacvičen. Měření lze přidat k dalším záznamům, které uživatel již vytvořil. Pokud byl trénink neúspěšný, může ho uživatel zahodit a vytvořit nový. V případě, že je lekce hodnocená dostatečným počtem bodů (ukazatel úspěšnosti lekce), může být přidána do série trénovaných příkazů.

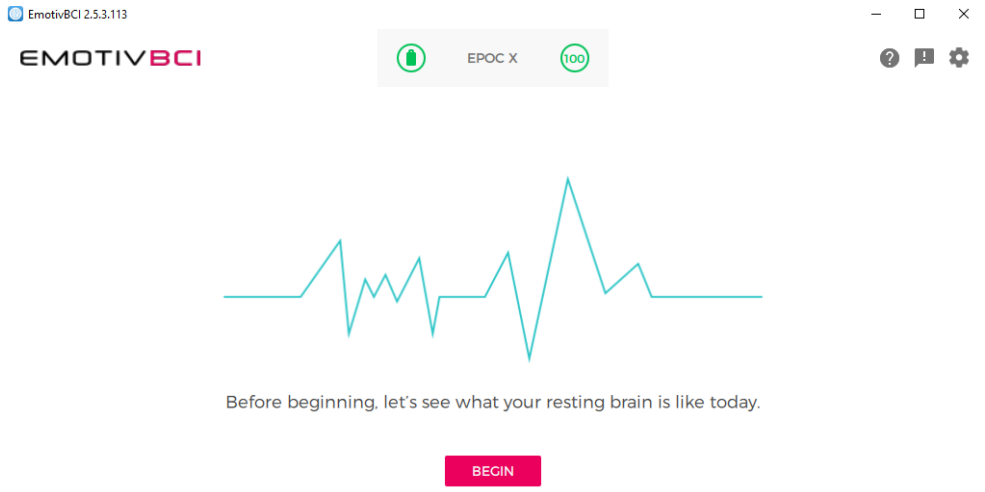
Každý den se při prvním spuštění aplikace zobrazí krátké cvičení pro nahrávání stavu mysli (viz. obrázek č. 8.12). Snímá se stav pro otevřené a zavřené oči. Toto cvičení se dělá z důvodu rozlišení odlišností v signálech mezi různými dny. Jak vypadá cvičení je vidět na obrázku č. 8.13.

V této aplikaci se tedy natrénují jednotlivé příkazy podle potřeby. Pro implementované rozhraní se trénují tři příkazy, které v něm budou hrát svoji roli. Po několika trénincích je možné spustit LIVE MODE, kde se dají využívat natrénované příkazy a hýbat tak kostkou.

Při trénování příkazů se musí počítat s příkazem NEUTRAL. Ten se ze zařízení posílá v době, kdy uživatel žádný příkaz nechce spustit. Jeho mysl se na nic nesoustředí a je v klidu. V tuto chvíli se nic neděje. Trénink pro NEUTRAL začíná jako první. Po dostatečném počtu tréninků lze přidat další příkaz.

Pro fungování celého systému jsou jako další příkazy vybrány PUSH a DROP. V souvislosti s použitím klávesnice je jednodušší si spojit s tlačítkem ENTER (potvrdit) příkaz PUSH, kdy se uživatel snaží tlačit kostku dozadu.

Příkaz DROP je vybrán na základě odlišné myšlenky, která se u něj dá použít. Při přidání třetího příkazu je trénink již obtížnější. Pro nacvičení více



Obrázek 8.12: Po spuštění aplikace EmotivBCI vybídne k nahrání stavu mysli při otevřených a zavřených očích

příkazů je vhodné příkazy od sebe značně odlišit. Tím lze usnadnit jejich trénink.

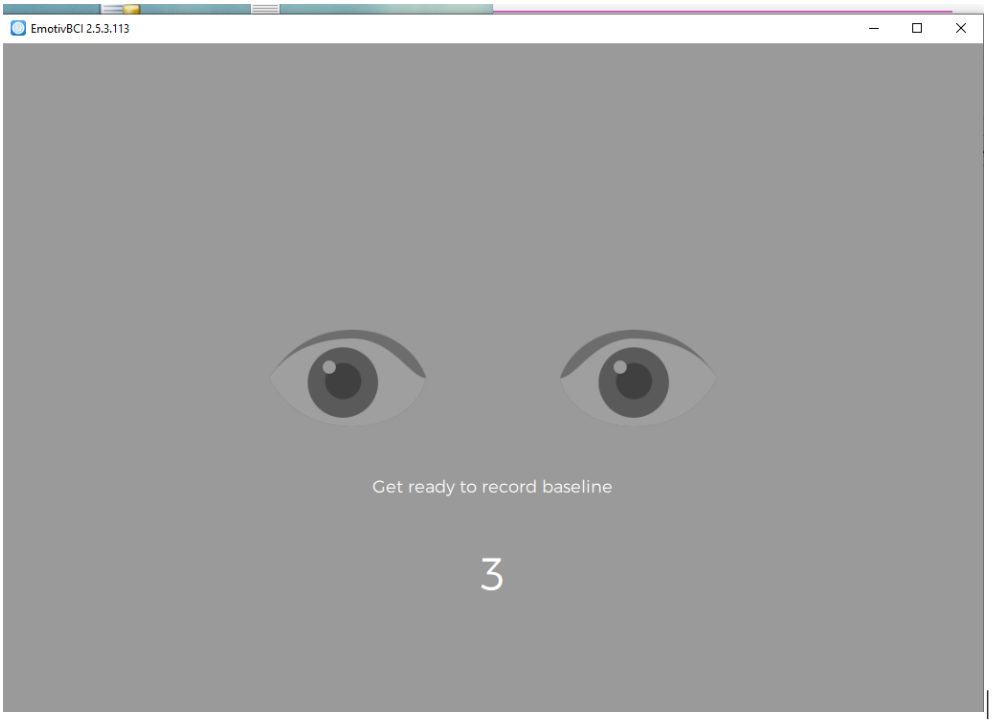
Pro příkaz DROP si autorka například představuje propadání se směrem dolů (proto příkaz DROP, který tuto představu dokresluje). Naopak u příkazu PUSH myslí autorka na pohyb směrem dopředu, odtlačení kostky dále (ve vizualizaci se kostka posune dozadu).

Pro otestování správnosti natrénovaných mentálních příkazů si lze v aplikaci spustit LIVE MODE. V něm se nachází opět kostka, se kterou lze pohybovat myšlenkou. K ovládání kostky uživatel využívá natrénované sady příkazů. Pokud je některý z příkazů špatně rozpoznán, je třeba ho ještě cvičit.

Aplikace si při používání hlídá stav elektrod. Pokud se některá z nich posune a celkový signál není dostatečný, zobrazí se okénko (viz. obrázek 8.14) upozorňující na nízkou kvalitu přijatého EEG signálu. Vždy je třeba dbát na 100% signál, aby se model správně trénoval a nebyla tím narušena nasbíraná data.

8.3.6 Emotiv Cortex API

Emotiv Cortex API je WebSocket server založený na protokolu Json-RPC. Umožňuje spojení s daty pro pohyb, výkonnostní metriku, kvalitu kontaktu,



Obrázek 8.13: Úvodní cvičení v aplikaci EmotivBCI pro otevřené oči

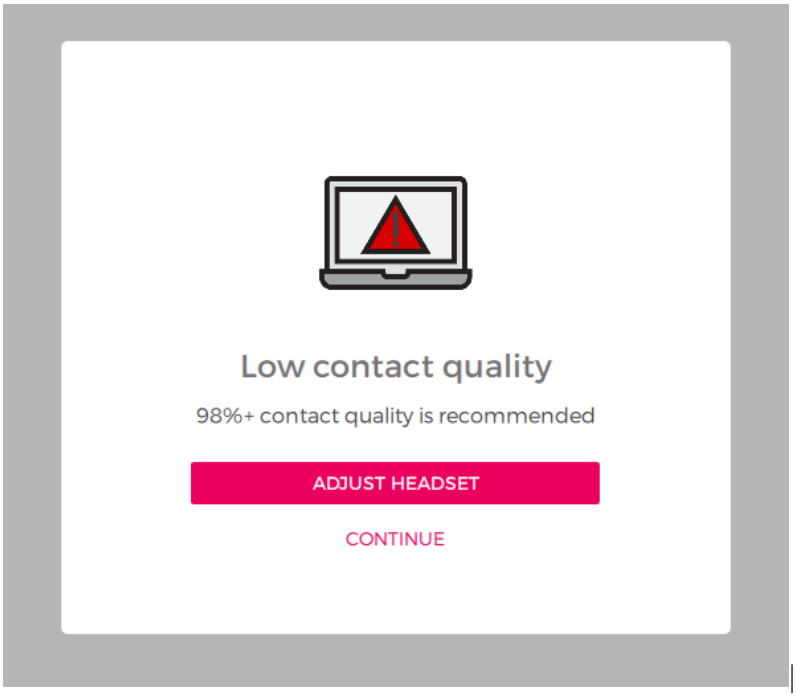
frekvenční pásma, výraz obličeje, úroveň baterie či mentálních příkazů. Jak se píše v dokumentaci [50].

Pro přístup k EEG datům je nutné zaplatit si předplatné. Existují Cortex verze 2 a nově 3, pro které jsou v dokumentaci příklady použití. API podporuje C++, Python, C#, ale i NodeJS. Pro systém se hodí NodeJS verze rozhraní, která je implementována.

Jak Cortex API funguje je možné vidět na obrázku 8.7 v dokumentaci pro vývojáře. Pomocí API se dá vytvořit profil, trénovat příkazy nebo spustit si LIVE MODE. Vývojář tak nemusí využívat aplikaci EmotivBCI.

První, co je potřeba při implementaci připojení skrz Cortex API je registrace aplikace na samotné stránce Emotivu. Po přihlášení aplikace je obdrženo klientské ID, které se vkládá do aplikace spolu s tajným klíčem pro ověření uživatele a začátek komunikace se zařízením.

Při používání zařízení je potřeba mít spuštěnou aplikaci Emotiv Launcher, bez které nelze pokračovat ani po spuštění vlastního kódu na komunikaci s API.



Obrázek 8.14: EmotivBCI aplikace varící uživatele před nízkou kvalitou kontaktu elektrod

8.4 Implementace Node.js aplikace připojené na Cortex API

Jakýmsi propojovacím můstkem je aplikace obsahující kód pro připojení se skrz Cortex API s Emotiv headsetem. Kód ve třídě Cortex je použit podle repozitáře poskytovaného společností Emotiv s příkladem použití API [51]. Pro implementaci této aplikace byla vybrána technologie založena na Node.js.

Repozitář s aplikací je na adrese `https://github.com/terez2/keyboard-cortex`. Pro spuštění aplikace slouží příkaz „node app.js“. V tomto souboru (app.js) se volá živý mód zaznamenávající příkazy v reálném čase. Ve třídě Cortex v metodě live je implementováno připojení se na zařízení v tomto živém módu. Tato třída se nachází v app/cortex.js.

Před samotným spuštěním kódu je potřeba zapnout Emotiv Launcher, zapnout a připojit headset. Zavlažit jednotlivé elektrody roztokem a upravit je do správné polohy podle návodu v Launcheru. Po získání 100% signálu lze spustit kód.

Po inicializaci proměnné typu Cortex se propojí aplikace s headsetem. Zde se používá ID aplikace a secret poskytnutý od společnosti Emotiv, jak je vidět v kódu dále.

```
let socketUrl = 'wss://localhost:6868'
let user = {
"clientId": "LylN2YDoWth7egFBU4UXy0YpTh91FoJZHrewLCef",
"clientSecret": "y7kDi4I6sUt3qeDo9cMP3yeOU4jltngobAcGDo25VQ5
RStiTJrknLOsEdUrbMWmWLTOCh81l5CyOS4qBfEerp7Ea1q6CXARZKdG3TbUemd
6BTTu0JiiWfhMLmGKkRuOx",
"debit": 1
}

let c = new Cortex(user, socketUrl)
```

Ve skriptu se v metodě live otevře spojení kódem níže (pro ilustraci vybrána jen část kódu).

```
this.socket.on('open', async () => {
  // is opened
}
```

Otevře se spojení a v něm se nejprve ověří, zda má aplikace práva na připojení se ke Cortex API, zda odpovídá klientské ID a ID zařízení a další. Tento kód je poskytnut v příkladu od Emotivu a kód je vidět níže.

```
/**
* - check if user logined
* - check if app is granted for access
* - query session info to prepare for sub and train
*/
async checkGrantAccessAndQuerySessionInfo() {
    let requestAccessResult = ""
    await this.requestAccess().then((result) => {
        requestAccessResult = result
    })

    let accessGranted = JSON.parse(requestAccessResult)

    // check if user is logged in CortexUI
    if ("error" in accessGranted) {
        console.log('You must login on CortexUI before request
            for grant access then rerun')
        throw new Error('You must login on CortexUI before
            request for grant access')
    } else {
        console.log(accessGranted['result']['message'])
        if (accessGranted['result']['accessGranted']) {
            await this.querySessionInfo()
        } else {
            console.log('You must accept access request from this
                 app on CortexUI then rerun')
            throw new Error('You must accept access request from
                this app on CortexUI')
        }
    }
}
```

Poté se načte uživatelský profil „Tereza“. Na tomto profilu je uložený model natrénovaný s příkazy NEUTRAL, PUSH a DROP. Kód v live metodě pro načtení profilu je níže.

```
let loadProfileResult = ""
let status = "load"
await this.setupProfile(this.authToken,
    this.headsetId,
    profileName,
    status).then((result) => {
    loadProfileResult = result
})
```

Po načtení profilu se spustí zbytek kódu v metodě live a v tuto chvíli se zaznamenávají myšlenky uživatele. Z API chodí data o tom, na jaký příkaz uživatel myslí. V kódu níže je vidět, jak se přijímá zpráva z Cortex API a přidává se do proměnné result.

```
this.socket.on('message', (data) => {
    result.push(JSON.parse(data));
})
```

Tato data chodí ve tvaru znázorněném v kódu níže.

```
{
  "com":["push",0.564],
  "sid":"79cc669b-af2e-465a-bdc2-0e9bd4aebe80",
  "time":1559903099.348
}
```

V datech typu JSON přijde informace o jaký typ proudu dat se jedná - zde „com“ tzn. command (příkaz). Typ dat se nastavuje v kódu zavoláním socketu s parametrem stream. Typ toku dat se posílá v poli. Aby přišel tok dat typu „com“, pošle se pole "stream":["com"]. V příkladu přišel z Cortex API příkaz PUSH.

Jak je psáno v dokumentaci Cortex API [50], obsahují přijatá data informaci o síle akce, ta se nachází v poli za stringovou hodnotou názvu příkazu. Je to desetinné číslo mezi 0 a 1, kdy nula znamená „nízký výkon“ a 1 značí „vysoký výkon“. Jinými slovy, jak moc (rychle) chce uživatel například tlačit kostku dozadu.

Pro aplikaci je důležitá informace, na který příkaz uživatel myslí. Součástí přijatých dat je i SID - identifikační číslo relace a čas (time), kdy byl tento příkaz zaznamenán.

Z přístroje chodí přibližně tři zprávy s daty za sekundu. Pro použití s klávesnicí jsou data po dobu intervalu 1,5s sbírána. Po tomto časovém úseku se vypočítá, kolikrát byl který příkaz zaznamenán a ten s největším počtem výskytů je vybrán a odeslán webové aplikaci.

Data přijatá z Cortex API je možné kromě výpisu na konzoli také po úpravě kódu uložit pomocí metody writeDataToFile(data) v app/cortex.js do složky outputs. Pro ukázku jsou v ní uložena některá nasbíraná data.

	AF3	F7	F3	FC5	T7	P7	O1	O2	P8	T8	
count	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	14980.000000	149
mean	4321.917777	4009.767694	4264.022433	4164.946326	4341.741075	4644.022379	4110.400160	4616.056904	4218.826610	4231.316200	42
std	2492.072174	45.941672	44.428052	5216.404632	34.738821	2924.789537	4600.926543	29.292603	2136.408523	38.050903	
min	1030.770000	2830.770000	1040.000000	2453.330000	2089.740000	2768.210000	2086.150000	4567.180000	1357.950000	1816.410000	32
25%	4280.510000	3990.770000	4250.260000	4108.210000	4331.790000	4611.790000	4057.950000	4604.620000	4190.770000	4220.510000	41
50%	4294.360000	4005.640000	4262.560000	4120.510000	4338.970000	4617.950000	4070.260000	4613.330000	4199.490000	4229.230000	42
75%	4311.790000	4023.080000	4270.770000	4132.310000	4347.180000	4626.670000	4083.590000	4624.100000	4209.230000	4239.490000	42
max	309231.000000	7804.620000	6880.510000	642564.000000	6474.360000	362564.000000	567179.000000	7264.100000	265641.000000	6674.360000	68

Obrázek 8.15: Příznaky použitých dat

Komunikace s Cortex API je založena na WebSocketech, pro spuštění aplikace je tedy potřeba si knihovnu ws nainstalovat. Další použitou knihovnou je axios. Ta je použita v app/service.js pro komunikaci s webovou aplikací.

8.5 Implementace vybraných algoritmů strojového učení

Pro výběr vhodného učícího algoritmu se autorka rozhodla využít někým již vytvořený soubor dat. Na stránce `www.kaggle.com` je k dohledání několik datasetů, které obsahují záznamy EEG signálů. Zvolený dataset je vybrán na základě stanovených kritérií níže. Lze ho použít k analýze vybraných machine learning algoritmů.

Kritéria pro výběr datasetu:

- typ dat – EEG signály
- v ideálním případě data z headsetu Emotiv
- transformace signálů na dva příkazy, díky kterým bude ovládána klávesnice

Algoritmus rozhodovacích stromů je zvolen kvůli jeho jednoduché implementaci pro osvojení celé techniky. Logistická regrese je vybrána, protože se dá dále kombinovat např. s neuronovými sítěmi. Rekurentní neuronovou síť autorka zvolila jako jeden z typů neuronových sítí, které budou podle některých studií pravděpodobně pro celé řešení nejvhodnější variantou.

8.5.1 Použité technologie a popis repozitáře

Repozitář s implementovanými algoritmy se nachází na githubu na adrese `https://github.com/terez2/signal-trans`.

V repozitáři lze nalézt tři složky:

- rnn (recurrent neural network) - obsahuje algoritmus pro rekurentní neuronové sítě
- logistic_regression - obsahuje soubor s logistickou regresí a soubor s výstupy logistické regrese
- desicion_tree - obsahuje implementaci algoritmu rozhodovací stromy

V těchto složkách jsou spustitelné soubory Jupyter Notebook. Nástroj Jupyter Notebook poskytuje interaktivní prostředí ve webovém rozhraní. Používá hlavně programovací jazyk Python. Tyto soubory mají koncovku .ipynb.

Pro samotné spuštění souborů je třeba si nainstalovat například volně dostupný program Anaconda. Pomocí Anaconda Prompt (příkazové řádky programu Anaconda) se dohledá stažený repozitář a v něm se spustí příkaz „jupyter notebook“.

Nyní lze nalézt složky repozitáře na adrese `http://localhost:8888/tree`, jak je vidět na obrázku č. 8.16. V nich lze otevírat soubory s koncovkou .ipynb a kód spustit. Například kód pro neuronovou síť se nyní nachází na adrese `http://localhost:8888/tree/rnn`. Po otevření souboru rnn.ipynb lze jednotlivé části kódu (buňky) spouštět.

Na začátku každého souboru Jupyter Notebook jsou v jedné z prvních buněk definovány importy knihoven. Tyto knihovny jsou potřebné pro spuštění kódu v dalších sekcích a bez spuštění tohoto bloku nebudou ostatní fungovat. Tuto buňku je potřeba si spustit jako první. Potom lze spustit ostatní části souboru.

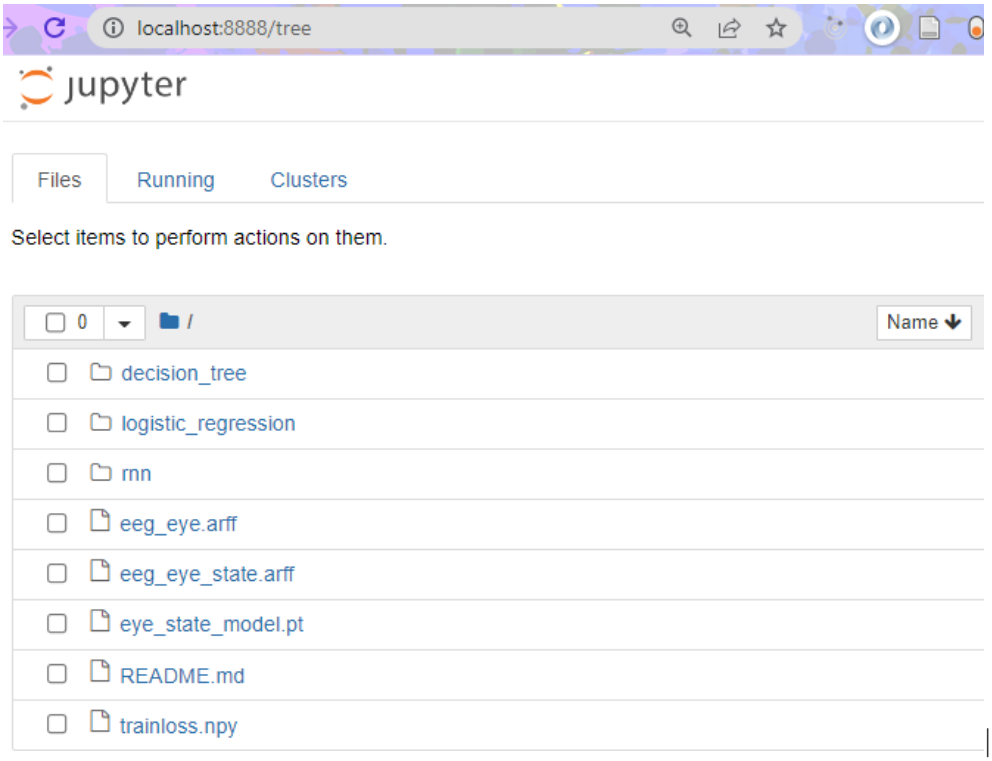
Pro analýzu dat je použita knihovna pandas. Dále se používá například knihovna numpy pro práci s polem. Knihovna joblib je zde importována pro ukládání modelu do souboru. Takový soubor má koncovku .joblib.

Pro spuštění buněk ve složce rnn.ipynb (neuronové sítě) je potřeba si nainstalovat do programu Anaconda Tensorflow (knihovnu pro strojové učení a umělou inteligenci) a Keras (rozhraní Pythonu pro umělé neuronové sítě). Pokud instalace proběhne v pořádku, lze tyto knihovny vidět v seznamu Anaconda Navigator viz. obrázek č. 8.17.

8.5.2 Dataset

Pro implementaci a vyzkoušení si jednotlivých algoritmů je použit dataset EEG signálů pro otevírání a zavírání oka. Tento dataset se skládá ze 14 kanálů a jednoho sloupce s informací o stavu oka. Pokud je oko otevřené, je zde 0, pokud je zavřené, potom je v tomto sloupci číslo 1. Tento dataset je stažen z Kaggle viz. [52]. Dataset je vytvořen pomocí zařízení Emotiv.

Složky z datasetu jsou uloženy v repozitáři `https://github.com/terez2/signal-trans`. Je to soubor trainloss.npy, eye_state_model.pt a eeg_eye.arff. Pro projekt je použit soubor eeg_eye.arff, ve kterém jsou uložena EEG data se stavem oka zaznamenaným číslem 0 nebo 1. Pro projekt byl vytvořen ještě



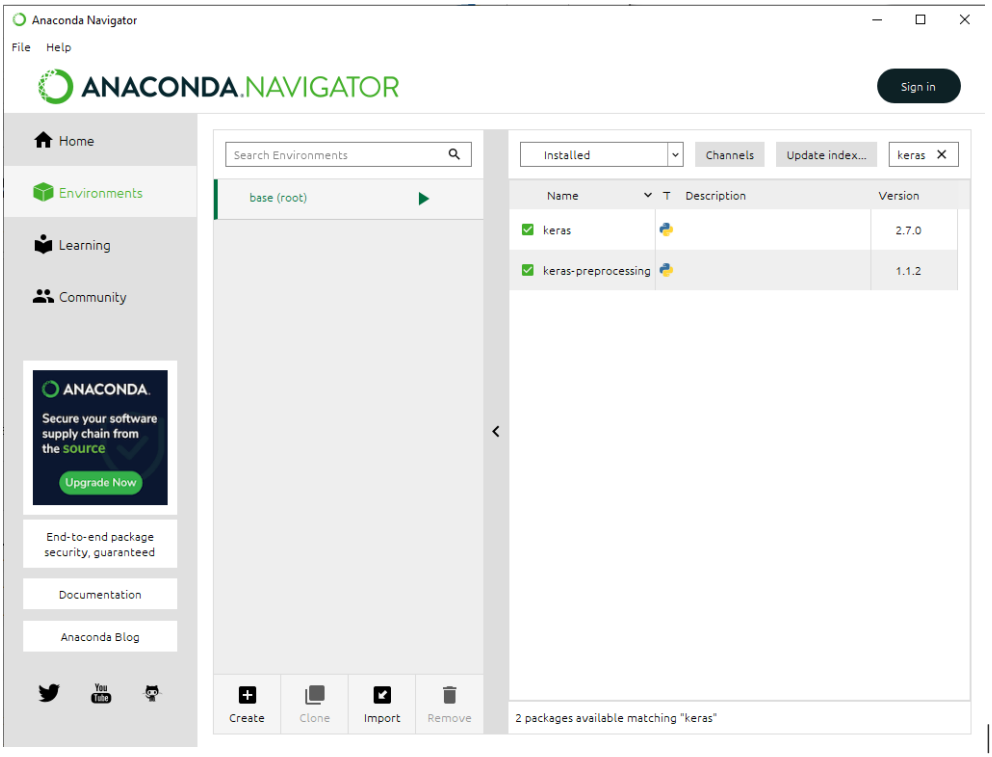
Obrázek 8.16: Ukázka repozitáře spuštěného pomocí Anaconda Prompt na adrese `http://localhost:8888/tree`

jeden soubor eeg_eye_state.arff s EEG daty, kde je otevřené nebo zavřené oko znázorněno slovem OPEN nebo CLOSE.

8.5.3 Decision tree (rozhodovací strom)

Prvním vyzkoušeným algoritmem je algoritmus rozhodovacího stromu, který je velmi oblíbenou metodou. Řadí se do třídy řízeného strojového učení tzn. s učitelem. Jak je vidět na ukázce (obrázek č. 8.20), před vytvářením takového stromu s pomocí Jupyter Notebooku, je nutné data rozdělit na vstupní hodnoty x a výstupní hodnoty y. Ty se dále rozčlení na data pro trénink modelu a data pro testování modelu. Stromy obsahují sadu pravidel a vytvářejí tak stromové struktury. Tyto pravidla jsou založena na základě dělení na další skupiny (větve) podle hodnoty jednoho z atributů.

Tento princip se stal nejrozšířenějším algoritmem v produkčním prostředí, jak se píše v článku [53]. Svou popularitu obdržel i za to, že je intuitivní a dá se jednoduše vizualizovat. Díky svému vnitřnímu fungování zpracovávají rozhodovací stromy velké datové sady a zvýšené objemy dat, aniž by se zastavila jejich rychlost predikce nebo ztratila přesnost. Díky tomu jsou extrémně uži-



Obrázek 8.17: Anaconda Navigator - vyhledání knihovny Keras

tečné pro problémy s velkými daty. Podrobnější popis implementace je obsažen v repozitáři `https://github.com/terez2/signal-trans`.

8.5.4 Logistic regression (logistická regrese)

Logistická regrese náleží do oblasti matematické statistiky. Umožňuje analýzu dat, kdy je odezva frekvenční (mezi hodnotami 0 a 1) nebo binární (nabývá jen dvou hodnot). Pravděpodobnost je modelována pomocí logistického modelu v závislosti na proměnné x.

$$\pi(x) = \frac{exp(\alpha + \beta x)}{1 + exp(\alpha + \beta x)}$$

Tato metoda je implementována s Limited-memory Broyden–Fletcher–Goldfarb–Shanno algoritmem (L-BFGS). Spadá do optimalizačních algoritmů s limitovaným množstvím paměti počítače. Algoritmus se snaží minimalizovat hladkou nelineární funkci, v případě, že je počet proměnných n velký a jsou k dispozici analytické výrazy pro funkci f a gradient g.

$$f : Nn \sim N, minf(x)$$

This trained model is stored using the joblib library for future use.

```
# Save model to external file
joblib.dump(model, 'logistic-regression-eye-state-prediction.joblib')
# Load model from this file
# model = joblib.load('logistic-regression-eye-state-prediction.joblib')
```

Obrázek 8.18: Joblib knihovna a její použití

Je vhodný pro problémy velkého rozsahu, protože velikost úložiště vyžadovaného algoritmy (a tím i náklady na iteraci) může řídit uživatel. Alternativně lze na metody s omezenou pamětí pohlížet jako na implementace kvazi-Newtonových metod, ve kterých je úložiště omezeno. Jejich jednoduchost je jedním z jejich hlavních výhod - nevyžadují znalost struktury Hessia ani znalost separability účelové funkce a jsou jednoduché na naprogramování [54].

Po natrénování modelu lze tento model pomocí knihovny joblib uložit do externího souboru a při přístím pokračování v práci ho jen načíst bez nutnosti spouštět jednotlivé buňky k jeho vytvoření. Ukládání/načtení tohoto modelu je vidět na obrázku č. 8.18.

8.5.5 Recurrent neural network (rekurentní neuronová síť)

U rekurentní sítě oproti vpředné (feedforward) je síť reprezentována neurony s rekurentním propojením, historie není omezená. Umí vytvářet krátkodobou paměť, takže se snadno vypořádá s neměnnou pozicí. Metoda je často používána pro zpracování jazyka, jako je třeba překlad – Google translator.

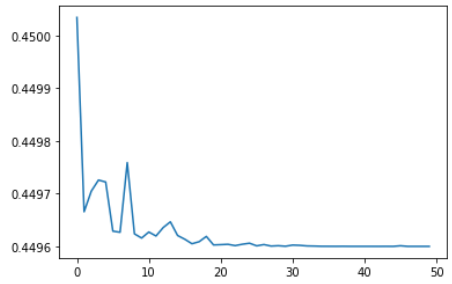
Pro implementaci rekurentní neuronové sítě (RNN) je potřeba nainstalovat Keras a Tensorflow. Použitým optimalizátorem je Adam. Další možností by mohl být třeba RMSprop, ten je však vhodný pro menší množství dat. V projektu je počítáno s velkým množstvím dat.

Jak je psáno v článku Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition [55], Long Short-Term Memory (LSTM) je specifická architektura neuronové sítě, která je použita při její implementaci. Podle této studie taková dvouvrstvá LSTM RNN, kde každá vrstva LSTM má lineární rekurentní projekční vrstvu, může překonat nejmodernější výkon rozpoznání řeči. Tato architektura efektivněji využívá parametry modelu než ostatní, rychle konverguje a překonává hlubokou dopřednou neuronovou síť mající řádově více parametrů.

8.5.6 Testování implementovaných řešení

Na obrázku č. 8.20 je vidět výstup pro hodnotu score - míra přesnosti tohoto modelu, která je přibližně 82%. Pro zobrazení modelu lze využít graphviz. V tomto případě však model nelze zobrazit pro jeho velikost. V tuto chvíli je

```
plt.plot(history.history['val_loss'])
plt.show()
```



Obrázek 8.19: RNN

```
x = eye_data.drop(columns=['eye'])
y = eye_data['eye']

# Divide the data into training data and test data
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2)

# Create decision tree
model = DecisionTreeClassifier()
# Train it
model.fit(x_train, y_train)
prediction = model.predict(x_test)

# Accuracy classification score
score = accuracy_score(y_test, prediction)
score
```

```
0.8361148197596796
```

Obrázek 8.20: Rozhodovací strom

potřeba přidat algoritmus pro prořezávání stromu. To je technika komprese dat v algoritmech strojového učení a vyhledávání, která snižuje velikost rozhodovacích stromů odstraněním částí těchto stromů. Oproti tomu výsledek přesnosti metody pro logistickou regresi s maximálním počtem iterací 15000 se pohyboval kolem 60%. Tuto metodu samu o sobě není vhodné pro řešení použít. Jak je vidět z obrázku č. 8.19, model s rekurentní neuronovou sítí se postupně po 50 epochách naučí minimalizovat ztráty. Pro lepší výsledek jsou data před učením modelu normalizována, aby se zvýšila rychlost učení.

Shrnutí výsledků

Na základě této práce je vytvořen systém pracující na bázi EEG dat. Pro uživatele s nižší hybností je vytvořeno příjemné uživatelské rozhraní fungující jako webová aplikace.

Samotnému návrhu rozhraní je věnována část práce, kdy je kladen důraz na menší počet signálů vysílaných ze zařízení do aplikace pro pohyb v klávesnici. Na základě návrhů vznikla dvě řešení, která byla implementována.

Jedno řešení se inspiruje v klávesnici na mobilním zařízení. Rozděluje znaky na devět tlačítek a uživatel je schopný poměrně rychle psát. Druhé vychází z koláčového menu, kdy se znaky dělí do šesti sekcí.

Rozhraní inspirované telefonem je rozměrově menší oproti koláčové nabídce, která zabírá více prostoru. Funkčnost obou rozhraní je podobná, pro porovnání je nutné provést testování na několika respondentech. Na jeho základě pak vyhodnotit finální design klávesnice.

Implementovaná real-time webová aplikace je schopna díky další Node.js aplikaci komunikace se zařízením (přes Cortex API) a snímat tak elektrickou aktivitu mozku.

Uživatel pouze pomocí myšlenky aktivuje příkaz CONFIRM (potvrdit) a vybere se písmeno, které se vypíše na obrazovku. Při pomyšlení na něco jiného je schopný se pohybovat příkazem NEXT (dále) mezi tlačítky a znaky a vybírat tak další možnosti rozhraní.

Jednotlivé moduly (aplikace) jsou navrženy tak, aby se daly jednoduše nahradit jiným řešením. Výsledkem je systém s ovládáním optimalizovaným pro přijímání mozkových vln.

Model pro snímání elektrické aktivity mozku je vytvořen pro jednoho uživatele (autorku práce), kde funguje obstojně. Na dalších respondentech je třeba jeho funkčnost ověřit. Pro lepší běh aplikace u jiných osob je vhodné provést další měření a trénink. Model pak bude schopen lépe rozlišovat příkazy od jiných uživatelů.

Model je trénován prostřednictvím poskytované aplikace EmotivBCI, která usnadnila značnou část práce. V budoucnu bude její použití nahrazeno vlastní

aplikací pro trénink a učení modelu. Pro aplikaci nahrazující EmotivBCI je předpřipraven průzkum vhodných algoritmů pracujících s EEG daty.

Trénink příkazů je poměrně časově náročný. Myšlenky vstupující do mysli pro vyvolání příkazu musí být odlišné pro každý z příkazů. Přidáváním příkazů se jejich nácvik stává stále obtížnějším. Model od sebe musí odlišovat i menší rozdíly. Natrénování tři příkazů (NEUTRAL, PUSH, DROP) vyžadovalo v součtu alespoň 100 cvičení, aby je model poměrně dobře rozlišoval v krátkém časovém úseku.

Při tréninku příkazů nebyl zvolen pravděpodobně vhodný postup. Nejprve se cvičil perfektně jeden příkaz, až potom se přidal další. Tímto postupem se stalo přidání dalšího příkazu složitější a jeho trénink vyžadoval větší soustředění.

Správný postup je takový, že se hned na začátku přidají všechny příkazy, kterými má aplikace disponovat. Následně se rovnoměrně tyto příkazy trénují, jak se také píše na stránce `https://emotiv.gitbook.io/emotivbci/mental-commands/tips-and-tricks`.

Průzkum strojového učení pracujícího s EEG daty obsahuje některé vybrané algoritmy a jejich realizaci na datech z Kaggle. Zvolený dataset s informací o stavu oka byl pro pochopení učících algoritmů a práce s nimi uspokojující. Avšak pro celou práci je vhodné si vytvořit vlastní dataset.

Vytvořený dataset by měl zahrnovat data od alespoň pěti respondentů. Příkazy by měly být založeny například na myšlení na pohyb kostky, na pocit, vůni atp. Je to lepší volba, než reagovat na stav oka, kdy je možné, že je jeho pohyb zachycen např. posunem elektrod na hlavě, nikoliv signálem jako takovým.

Implementaci vybraných učících algoritmů lze brát jako první průzkum machine learning algoritmů a vhled do dané problematiky. Pro ucelený pohled je nutné vyzkoušet připravené metody na vlastních datech (po vytvoření datasetu). V implementační části jsou vyzkoušeny pouze tři metody, a proto stále nelze říct, jaký algoritmus bude ten nejlepší. Nicméně slouží pro představu, jak asi budou na datech fungovat.

Podle mnohých studií je výhodná i kombinace několika metod. Podle dosažených výsledků se pravděpodobně ve vlastním řešení objeví neuronové sítě. Zdají se být pro tento systém vhodným kandidátem. V projektu se pojednává o rekurentní neuronové síti, ta by se dala propojit např. s konvoluční neuronovou sítí nebo logistickou regresí.

Závěr

Práce slouží jako průzkum jednotlivých oblastí, do kterých zasahuje, jako jsou například EEG signály a mozkové vlny, zpracování samotného signálu (signal processing) nebo algoritmy strojového učení (machine learning algorithms).

Výsledkem práce je systém pro ovládání uživatelského rozhraní pomocí EEG signálů. Obsahuje tři aplikace, které spolu pracují v reálném čase.

Jedna aplikace pracuje jako komunikační rozhraní mezi webovou aplikací a zařízením Emotiv. Další dvě aplikace tvoří webovou aplikaci. Ta je rozdělena na frontendovou a backendovou část, kdy je součástí frontendu speciální uživatelské rozhraní podobné klávesnici.

Toto rozhraní je optimalizováno pro přijímání EEG signálů, tak aby se minimalizoval počet vyslaných příkazů při výběru písmene.

Verze celého systému není konečná. Pro trénování pomocí učících algoritmů je využito rozhraní EmotivBCI, které usnadnilo značnou část práce. Toto rozhraní bude však v budoucnu nahrazeno vlastním rozhraním pro trénování učících algoritmů.

Systém vznikl tak, aby byl dostatečně modulární a jeho jednotlivé části se daly jednoduše nahradit. Webovou aplikaci lze jednoduše napojit na jiný systém pracující například na základě snímání obrazu.

Pro vytvoření modelu pracujícího na základě strojového učení byl proveden průzkum dané oblasti pro nalezení vhodného algoritmu. V něm byly vyzkoušeny některé algoritmy na datasetu ze stránky Kaggle. Podle výsledků testů je vhodné využít neuronové sítě v kombinaci s dalším algoritmem strojového učení.

Literatura

[1] Neuron a jeho stavba | Mentem.cz. [cit. 2022-01-02]. Dostupné z: https://www.mentem.cz/blog/neuron/

[2] Abhang, P. A.; Gawali, B. W.; Mehrotra, S. C.: Chapter 2 - Technological Basics of EEG Recording and Operation of Apparatus. In *Introduction to EEG- and Speech-Based Emotion Recognition*, editace P. A. Abhang; B. W. Gawali; S. C. Mehrotra, Academic Press, ISBN 978-0-12-804490-2, s. 19–50, doi:https://doi.org/10.1016/B978-0-12-804490-2.00002-6. Dostupné z: https://www.sciencedirect.com/science/article/pii/B9780128044902000026

[3] EMOTIV EPOC X | 14 Channel Mobile EEG Headset. Dostupné z: https://www.emotiv.com/epoc-x/

[4] EEG Signal Processing for Dummies - Neuroelectrics Neuroelectrics Blog - Latest news about EEG & Brain Stimulation. Prosinec 2014. Dostupné z: https://www.neuroelectrics.com/blog/2014/12/18/eeg-signal-processing-for-dummies/

[5] Pareek, P.: Machine learning Algorithms and where they are used? Dostupné z: https://techsavvypriya.wordpress.com/2019/12/19/machine-learning-algorithms-and-where-they-are-used/

[6] UI Events KeyboardEvent code Values. Dostupné z: https://www.w3.org/TR/uievents-code/

[7] Dvorak keyboard layout. Červenec 2022, page Version ID: 1097687980. Dostupné z: https://en.wikipedia.org/w/index.php?title=Dvorak_keyboard_layout&oldid=1097687980

[8] Onscreen keyboards - Selection And-input - Components - Human Interface Guidelines - Design - Apple Developer. Dostupné z: https:

//developer.apple.com/design/human-interface-guidelines/components/selection-and-input/onscreen-keyboards/

[9] Hopkins, D.: The Design and Implementation of Pie Menus. Srpen 2021. Dostupné z: https://donhopkins.medium.com/the-design-and-implementation-of-pie-menus-80db1e1b5293

[10] Introduction | Socket.IO. Dostupné z: https://socket.io/docs/v4/

[11] Al Duhayyim, M.; Alshahrani, H. M.; Al-Wesabi, F. N.; aj.: Intelligent Machine Learning Based EEG Signal Classification Model. *CMC-COMPUTERS MATERIALS & CONTINUA*, ročník 71, č. 1, 2022: s. 1821–1835, ISSN 1546-2218, doi:{10.32604/cmc.2022.021119}.

[12] Katona, J.; Kővári, A.: EEG-based Computer Control Interface for Brain-Machine Interaction. *International Journal of Online Engineering (iJOE)*, ročník 11, 11 2015: s. 43–48, doi:10.3991/ijoe.v11i6.5119.

[13] Kumar, A.; Jangir, G.: EEG Signal Based System to Control Home Appliances. 05 2018: s. 2454–9150, doi:10.18231/2454-9150.2018.0178.

[14] Malý, L.: Wheelchair Control Using EEG Signal Classification. 201.

[15] Birbaumer, N.; Kübler, A.; Ghanayim, N.; aj.: The thought translation device (TTD) for completely paralyzed patients. *Rehabilitation Engineering, IEEE Transactions on*, ročník 8, 2000: s. 190–193, doi:10.1109/86.847812.

[16] Obermaier, B.; Müller, G.; Pfurtscheller, G.: 'Virtual Keyboard' Controlled by Spontaneous EEG Activity. In *Artificial Neural Networks — ICANN 2001*, editace G. Dorffner; H. Bischof; K. Hornik, Lecture Notes in Computer Science, Berlin, Heidelberg: Springer, 2001, ISBN 978-3-540-44668-2, s. 636–641, doi:10.1007/3-540-44668-0_89.

[17] Pfurtscheller, G.; Neuper, C.; Muller, G.; aj.: Graz-BCI: state of the art and clinical applications. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*, ročník 11, č. 2, Červen 2003: s. 1–4, ISSN 1558-0210, doi:10.1109/TNSRE.2003.814454, conference Name: IEEE Transactions on Neural Systems and Rehabilitation Engineering.

[18] Kumar, J. S.; Bhuvaneswari, P.: Analysis of Electroencephalography (EEG) Signals and Its Categorization–A Study. ročník 38: s. 2525–2536, ISSN 1877-7058, doi:https://doi.org/10.1016/j.proeng.2012.06.298. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1877705812022114

[19] Stiles, J.; Jernigan, T. L.: The Basics of Brain Development. ročník 20, č. 4: s. 327–348, ISSN 1573-6660, doi:10.1007/s11065-010-9148-4. Dostupné z: https://doi.org/10.1007/s11065-010-9148-4

[20] Brain Basics: The Life and Death of a Neuron | National Institute of Neurological Disorders and Stroke. Dostupné z: https://www.ninds.nih.gov/health-information/patient-caregiver-education/brain-basics-life-and-death-neuron

[21] Subha, D. P.; Joseph, P. K.; Acharya U, R.; aj.: EEG Signal Analysis: A Survey. ročník 34, č. 2: s. 195–212, ISSN 1573-689X, doi:10.1007/s10916-008-9231-z. Dostupné z: https://doi.org/10.1007/s10916-008-9231-z

[22] Das, S.; Tripathy, D.; Raheja, J. L.: An Insight to the Human Brain and EEG. In *Real-Time BCI System Design to Control Arduino Based Speed Controllable Robot Using EEG*, editace S. Das; D. Tripathy; J. L. Raheja, SpringerBriefs in Applied Sciences and Technology, Springer, ISBN 9789811330988, s. 13–24, doi:10.1007/978-981-13-3098-8_2. Dostupné z: https://doi.org/10.1007/978-981-13-3098-8_2

[23] Pentari, A.; Tzagkarakis, G.; Marias, K.; aj.: Graph denoising of impulsive EEG signals and the effect of their graph representation. *Biomedical Signal Processing and Control*, ročník 78, Září 2022: str. 103886, ISSN 1746-8094, doi:10.1016/j.bspc.2022.103886. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1746809422003974

[24] Ovládněte své mozkové vlny | mindball.eu. Dostupné z: https://www.mozkoherna.cz/blog/ovladnete-sve-mozkove-vlny/

[25] electroencephalography | Definition, Procedure, & Uses | Britannica. Dostupné z: https://www.britannica.com/science/electroencephalography

[26] Husain, A. M.: WS5.5. Electroencephalographic Monitoring. *Clinical Neurophysiology*, ročník 132, č. 8, Srpen 2021: str. e59, ISSN 1388-2457, doi:10.1016/j.clinph.2021.02.097. Dostupné z: https://www.sciencedirect.com/science/article/pii/S1388245721001723

[27] Craik, A.; He, Y.; Contreras-Vidal, J. L.: Deep learning for electroencephalogram (EEG) classification tasks: a review. *Journal of Neural Engineering*, ročník 16, č. 3, Červen 2019: str. 031001, ISSN 1741-2560, 1741-2552, doi:10.1088/1741-2552/ab0ab5. Dostupné z: https://iopscience.iop.org/article/10.1088/1741-2552/ab0ab5

[28] Amin, H. U.; Mumtaz, W.; Subhani, A. R.; aj.: Classification of EEG Signals Based on Pattern Recognition Approach. *Frontiers in Computational Neuroscience*, ročník 11, 2017, ISSN 1662-5188. Dostupné z: https://www.frontiersin.org/articles/10.3389/fncom.2017.00103

[29] Hariharan, M.; Vijean, V.; Sindhu, R.; aj.: Classification of mental tasks using stockwell transform. *Computers & Electrical Engineering*, ročník 40, č. 5, Červenec 2014: s. 1741–1749, ISSN 0045-7906, doi:10.1016/j.compeleceng.2014.01.010. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0045790614000251

[30] Wong, T.-T.: Performance evaluation of classification algorithms by k-fold and leave-one-out cross validation. *Pattern Recognition*, ročník 48, č. 9, Září 2015: s. 2839–2846, ISSN 00313203, doi:10.1016/j.patcog.2015.03.009. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S0031320315000989

[31] PRASADA1207: Datová jezera - Azure Architecture Center. Dostupné z: https://docs.microsoft.com/cs-cz/azure/architecture/data-guide/scenarios/data-lake

[32] Halevy, A.; Korn, F.; Noy, N. F.; aj.: Goods: Organizing Google's Datasets. In *Proceedings of the 2016 International Conference on Management of Data*, San Francisco California USA: ACM, Červen 2016, ISBN 978-1-4503-3531-7, s. 795–806, doi:10.1145/2882903.2903730. Dostupné z: https://dl.acm.org/doi/10.1145/2882903.2903730

[33] Cafarella, M. J.; Halevy, A.; Wang, D. Z.; aj.: WebTables: exploring the power of tables on the web. *Proceedings of the VLDB Endowment*, ročník 1, č. 1, Srpen 2008: s. 538–549, ISSN 2150-8097, doi:10.14778/1453856.1453916. Dostupné z: https://dl.acm.org/doi/10.14778/1453856.1453916

[34] How Crowdsourcing Works. Dostupné z: https://www.investopedia.com/terms/c/crowdsourcing.asp

[35] Armanious, K.; Jiang, C.; Fischer, M.; aj.: MedGAN: Medical Image Translation using GANs. *Computerized Medical Imaging and Graphics*, ročník 79, Leden 2020: str. 101684, ISSN 08956111, doi:10.1016/j.compmedimag.2019.101684, arXiv:1806.06397 [cs]. Dostupné z: http://arxiv.org/abs/1806.06397

[36] HoloClean: A Machine Learning System for Data Enrichment. Červenec 2022, original-date: 2018-11-09T19:55:30Z. Dostupné z: https://github.com/HoloClean/holoclean

[37] Shalev-Shwartz, S.: *Understanding Machine Learning: From Theory to Algorithms*. Cambridge University Press, první vydání, 2014.

[38] What Is Big Data? Dostupné z: https://www.oracle.com/big-data/what-is-big-data/

[39] Zhang, Y.: *New Advances in Machine Learning*. BoD – Books on Demand, ISBN 978-953-307-034-6, google-Books-ID: XAqhDwAAQBAJ.

[40] Sutton, R. S.; Barto, A. G.: *Reinforcement Learning, second edition: An Introduction*. MIT Press, ISBN 978-0-262-35270-3, google-Books-ID: uWV0DwAAQBAJ.

[41] Mathieu, A.: What Are Real-Time Applications? Dostupné z: https://vantiq.com/what-are-real-time-applications/

[42] Documentation | NestJS - A progressive Node.js framework. Dostupné z: https://docs.nestjs.com

[43] Garrett, J.: *The Elements of User Experience: User-Centered Design for the Web and Beyond*. Prosinec 2010, ISBN 978-0-321-68368-7.

[44] Why Your Computer Keyboard Has a QWERTY Layout. Section: ThoughtCo. Dostupné z: https://www.thoughtco.com/history-of-the-computer-keyboard-1991402

[45] The Dvorak Keyboard |. Dostupné z: https://dvorak-keyboard.com

[46] Pie menu. Květen 2022, page Version ID: 1087102444. Dostupné z: https://en.wikipedia.org/w/index.php?title=Pie_menu&oldid=1087102444

[47] Radial Menu (Concept). Dostupné z: https://www.giantbomb.com/radial-menu/3015-737/

[48] The Design and Implementation of Pie Menus – Dr. Dobb's Journal, Dec. 1991 | Don Hopkins. Prosinec 2009. Dostupné z: https://web.archive.org/web/20091225004939/http://www.donhopkins.com/drupal/node/98

[49] Documentation | NestJS - A progressive Node.js framework. Dostupné z: https://docs.nestjs.com

[50] Getting Started. Dostupné z: https://emotiv.gitbook.io/cortex-api/

[51] Overview. Červenec 2022, original-date: 2019-04-16T08:09:05Z. Dostupné z: https://github.com/Emotiv/cortex-v2-example

[52] EEG Eye State. Dostupné z: https://kaggle.com/markwallbang/eeg-eye-state

[53] The Ultimate Guide to Decision Trees for Machine Learning. Dostupné z: https://www.keboola.com/blog/decision-trees-machine-learning

[54] Liu, D. C.; Nocedal, J.: On the limited memory BFGS method for large scale optimization. ročník 45, č. 1: s. 503–528, ISSN 1436-4646, doi:10.1007/BF01589116. Dostupné z: https://doi.org/10.1007/BF01589116

[55] Dahl, G. E.; Dong Yu; Li Deng; aj.: Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. ročník 20, č. 1: s. 30–42, ISSN 1558-7916, 1558-7924, doi:10.1109/TASL.2011.2134090. Dostupné z: http://ieeexplore.ieee.org/document/5740583/

Příloha **A**

Seznam použitých zkratek

EEG Elektroencefalografie

EEG signál Elektroencefalografický signál

UI User Interface (uživatelské rozhraní)

GUI Grafic User Interface (grafické uživatelské rozhraní)

UX User Experience (uživatelské zkušenost)

SVM Support Vector Machine (metoda podpůrných vektorů)

MLP Multilayer Perceptron (vícevrstvý perceptron)

NB Naive Bayes

k-NN K-Nearest Neighbor

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

XHR XMLHttpRequest

TCP Transmission Control Protocol

API Application Programming Interface (rozhraní pro programování aplikací)

BCI Brain Computer Interface (neurální rozhraní)

TTD Thought Translation Device

VK Virtual Keyboard (virtuální klávesnice)

Hz Hertz - jednotka frekvence

ICA (analýza nezávislých komponent)

CSP Constraint Satisfaction Problem (problém spokojenosti s omezujícími podmínkami)

PCA (analýza hlavních komponent)

Goods Google Data Search

GAN Generative Adversarial Network

MTurk Amazon Mechanical Turk

RL Reinforcement Learning (posílené učení)

MIME Multipurpose Internet Mail Extensions (víceúčelová rozšíření internetové pošty)

JSON JavaScript Object Notation

JSON-RPC JavaScript Object Notation - remote procedure call protocol

HTML Hypertext Markup Language

CSS Cascading Style Sheets (kaskádové styly)

JS JavaScript

WS WebSocket

DI Dependency Injection

RxJs Reactive Extensions Library for JavaScript

USB Universal Serial Bus

ID Identifikace

SDK Software development kit

DOM Document Object Model (objektový model dokumentu)

SSE Server-sent events

W3C World Wide Web Consortium

3D Trojdimenzionální

L-BFCS Limited-memory Broyden–Fletcher–Goldfarb–Shanno algoritmus

RNN Recurrent neural network

LSTM Long short-term memory

URL Uniform Resource Locator

RPG Role-paying game

SID A security identifier

Příloha **B**

Obsah elektronické přílohy

- **Algoritmy strojového učení** https://github.com/terez2/signal-trans
- **Frontendová část aplikace s uživatelským rozhraním** https://github.com/terez2/keyboard
- **Backendová část aplikace** https://github.com/terez2/keyboard-backend
- **Aplikace s připojením na Cortex API** https://github.com/terez2/keyboard-cortex

UNIVERZITA HRADEC KRÁLOVÉ
Fakulta informatiky a managementu
Akademický rok: 2020/2021

Studijní program: Aplikovaná informatika
Forma studia: Kombinovaná
Obor/kombinace: Aplikovaná informatika (ai2-k)

Podklad pro zadání DIPLOMOVÉ práce studenta

Jméno a příjmení:	**Bc. Tereza Kvášová**
Osobní číslo:	**I2000312**
Adresa:	Krapkova 306/6, Hradec Králové – Malšovice, 50009 Hradec Králové 9, Česká republika
Téma práce:	Uživatelské rozhraní pro osoby s nižší hybností a model pro snímání elektrické aktivity mozku
Téma práce anglicky:	User Interface for Disabled People and Model for Sensing Electrical Activity of Brain
Vedoucí práce:	Ing. Pavel Kříž, Ph.D. Katedra informatiky a kvantitativních metod

Zásady pro vypracování:

Cíl: Cílem práce je zhotovit uživatelské rozhraní pro osoby s postižením pohybového aparátu, které bude schopné na základě několika příkazů fungovat jako klávesnice. Součástí práce je průzkum machine learning algoritmů a oblasti zpracování EEG signálů.
Osnova:

1. Úvod
2. Cíl práce
3. Rešerše stávajících řešení
4. EEG Signal processing
5. Machine learning algorithm
6. Návrh UI
7. Návrh nástroje pro vytvoření datasetu
8. Návrh modelu pro zpracování datasetu
9. Realizace
10. Literatura

Externí vedoucí Ing. Vladimír Blažek (Quanti s.r.o)

Seznam doporučené literatury:

SANEI, Saeid a J. A. CHAMBERS. EEG Signal Processing. John Wiley & Sons, 2007. ISBN 9780470025819.
CAMPESATO, Oswald. Angular and Machine Learning Pocket Primer. Mercury Learning & Information, 2020. ISBN 9781683924708.

Podpis studenta: Datum:

Podpis vedoucího práce: Datum: