FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Detekce pohybu z EEG dat

Jakub Kodera

PLZEŇ

2023

FAKULTA APLIKOVANÝCH VĚD
ZÁPADOČESKÉ UNIVERZITY
V PLZNI

KATEDRA INFORMATIKY
A VÝPOČETNÍ TECHNIKY

Diplomová práce

Detekce pohybu z EEG dat

Bc. Jakub Kodera

Vedoucí práce
Doc. Ing. Roman Mouček, Ph.D.

PLZEŇ **2023**

Citace v seznamu literatury:
KODERA, Jakub. *Detekce pohybu z EEG dat*. Plzeň, 2023. Diplomová práce. Západočeská univerzita v Plzni, Fakulta aplikovaných věd, Katedra informatiky a výpočetní techniky. Vedoucí práce Doc. Ing. Roman Mouček, Ph.D.

ZÁPADOČESKÁ UNIVERZITA V PLZNI
Fakulta aplikovaných věd
Akademický rok: 2022/2023

ZADÁNÍ DIPLOMOVÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Jakub KODERA**
Osobní číslo:	**A21N0029P**
Studijní program:	**N3902 Inženýrská informatika**
Studijní obor:	**Medicínská informatika**
Téma práce:	**Detekce pohybu z EEG dat**
Zadávající katedra:	**Katedra informatiky a výpočetní techniky**

Zásady pro vypracování

1. Prostudujte možnosti detekce pohybu z EEG dat a principy souvisejících rehabilitačních postupů.
2. Seznamte se se scénářem měření EEG dat za účelem detekce pohybu realizovaným na KIV a vzniklou datovou kolekcí.
3. Navrhněte a implementujte vhodnou metodu rozšíření datové kolekce z bodu 2. Proveďte také replikaci experimentu v laboratoři s alespoň třemi dalšími účastníky.
4. Navrhněte a implementujte vhodnou(é) metodu(y) pro detekci pohybu z EEG dat.
5. Porovnejte výsledky detekce pohybu z EEG dat provedené nad původní a rozšířenou datovou kolekcí.
6. Zhodnoťte dosažené výsledky.

Rozsah diplomové práce: **dopuč. 50 s. původního textu**
Rozsah grafických prací: **dle potřeby**
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

dodá vedoucí diplomové práce

Vedoucí diplomové práce: **Doc. Ing. Roman Mouček, Ph.D.**
Katedra informatiky a výpočetní techniky

Datum zadání diplomové práce: **9. září 2022**
Termín odevzdání diplomové práce: **18. května 2023**

L.S.

Doc. Ing. Miloš Železný, Ph.D.
děkan

Doc. Ing. Přemysl Brada, MSc., Ph.D.
vedoucí katedry

V Plzni dne 11. října 2022

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně a výhradně s použitím citovaných pramenů, literatury a dalších odborných zdrojů. Tato práce nebyla využita k získání jiného nebo stejného akademického titulu.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona v platném znění, zejména skutečnost, že Západočeská univerzita v Plzni má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle §60 odst. 1 autorského zákona.

V Plzni dne 29. března 2023

. . . . . .
Jakub Kodera

V textu jsou použity názvy produktů, technologií, služeb, aplikací, společností apod., které mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

Abstrakt

Představa pohybu je jedna z možností jakou může člověk komunikovat pomocí rozhraní mozek počítač. Cílem této práce je prozkoumat existující možnosti a používané metody v oblasti detekce pohybu z naměřeného EEG signálu. Jelikož získávání EEG dat je časově náročná aktivita, jsou v práci prozkoumány možnosti rozšíření existující datové sady bez nutnosti provádění dalšího měření. Práce porovnává detekci pohybu pěti klasifikátory (LDA, SVM, MLP, LSTM a CNN) a zkoumá také využití různých příznakových vektorů. V práci je provedena implementace a porovnání rozšíření datové sady pomocí augmentačních metod NI, cVAE a cWGAN-GP. Nejlepšího klasifikačního výsledku bylo dosaženo klasifikátorem CNN s klasifikační přesností 76.00±0.80%.

Abstract

Motor imagery is one of the ways in which a person can communicate through a brain-computer interface. The aim of this work is to explore existing methods in the field of motor imagery detection from measured EEG signals. Since acquisition of EEG data is a time-consuming activity, options for expanding the existing dataset without the need for additional measurements is also explored. This work compares motor imagery detection using five classifiers (LDA, SVM, MLP, LSTM a CNN) and also examines the use of different feature vectors. Additionally, the study implements and compares the expansion of the dataset using the augmentation methods NI, cVAE a cWGAN-GP. The best classification result was achieved with the CNN classifier, achieving a classification accuracy of 76.00±0.80%.

Klíčová slova

EEG • představa pohybu • augmentace dat • klasifikace • strojové učení

Poděkování

Tímto bych chtěl poděkovat Doc. Ing. Romanovi Moučkovi, Ph.D. za vedení a cenné rady v průběhu tvorby této práce. Dále bych rád poděkoval všem účastníkům laboratorního měření.

Obsah

Úvod 1

Detekce představy pohybu (Motor Imagery, MI) je proces, který umožňuje identifikovat záměry pohybu na základě samotné mozkové aktivity, a to bez potřeby vnějšího podnětu. Bylo prokázáno, že MI je spojena s aktivitou v oblasti motorického kortexu, která je podobná té, která vzniká při skutečném fyzickém pohybu. Rozhraní mezi mozkem a počítačem (Brain-computer interface, BCI) slouží jako prostředek komunikace mezi mozkem a externím zařízením.

Aplikace BCI v rehabilitačním tréninku může být velmi užitečná pro pacienty s paralýzou nervosvalového systému, kteří mají zachovanou normální kognitivní schopnost. Tato technologie jim umožňuje interakci s vnějším světem pomocí interpretace jejich pohybových záměrů na základě mozkové aktivity. Hlavním cílem BCI je tedy rozpoznat a interpretovat zamýšlený pohyb na základě mozkových signálů.

Využití BCI v rehabilitačním tréninku může zahrnovat různé metody, jako je snímání EEG signálů, funkční magnetická rezonance (fMRI) nebo invazivní metody jako je elektrokortikografie (ECoG) či implantace elektrod přímo do mozku. Tyto metody umožňují zachytit mozkovou aktivitu spojenou s pohybovými představami a následně ji interpretovat a přenést do ovládání externích zařízení, například robotických končetin nebo počítačových programů.

Přes rozsáhlý výzkum v oblasti BCI a jejích aplikací v rehabilitaci je třeba poznamenat, že tyto technologie jsou stále ve vývoji a jejich komerční využití je zatím omezené. Nicméně pokroky v oblasti BCI mají velký potenciál přinést nové možnosti a zlepšení života pacientům s poruchou pohybu a paralýzou, a to prostřednictvím obnovy jejich schopnosti komunikovat a interagovat s okolním světem.

Cílem této práce je prozkoumat různé metody detekce pohybu z naměřené datové sady a zkoumat možnosti jejího dalšího rozšíření. Výsledný klasifikátor, natrénovaný na základě těchto metod, by poté mohl být využit ve výzkumu BCI systémů.

Práce je rozčleněna následovně: kapitola 2 pojednává o teoretických vlastnostech EEG signálu v oblasti představy pohybu a možnosti její využití pro rehabilitaci. Kapitola 3 provádí rešerši literatury problematiky detekce pohybu z EEG dat a rozbor prací, na které tato práce navazuje. Kapitola 4 se zabývá možnou reprezentací příznakových vektorů. Kapitola 5 probírá teoretické vlastnosti klasifikátorů

použitelných pro detekci pohybu. Kapitola 6 se zabývá metodami rozšíření datové kolekce. Kapitola 7 popisuje implementaci jednotlivých metod a nakonec kapitola 8 poskytuje přehled a diskusi dosažených výsledků.

EEG 2

Elektroencefalografie (EEG) je diagnostická vyšetřovací metoda používaná k záznamu elektrické aktivity mozku. Měření mozkové aktivity bývá v nejčastějších případech děláno neinvazivně. Jedná se tedy o bezbolestnou metodu, i když dlouhotrvající měření může být pro daného pacienta poněkud nekomfortní. Existují i metody invazivní, kdy jsou měřící elektrody přiloženy přímo na povrch mozku, ovšem tyto metody jsou používány nejčastěji pouze v případech operací [HM01].

Neinvazivní metoda je prováděna rovnoměrným umístěním elektrod na pokožku hlavy. Nejčastější umístění elektrod je podle systému *10-20*, kdy vzdálenost mezi jednotlivými elektrodami je buď 10% nebo 20% celkové vzdálenosti lebky zepředu dozadu nebo zprava doleva. Dle rozmístění jsou v tomto systému elektrody označovány následujícími písmeny: A (ear lobe; ušní lalůček), F (frontal), C (central), T (temporal), P (parietal) a O (occipital). Dále jsou elektrody označeny číslem. Lichým číslem pro elektrody umístěné nad levou mozkovou hemisférou a sudým číslem pro elektrody nad pravou hemisférou. Množství zapojených elektrod odpovídá počtu záznamových kanálů [HM01; Ach+16; NNS16].

Snímaný EEG signál je amplituda elektrodou naměřeného napětí, které v mozku vzniká aktivitou excitačních a inhibičních postsynaptických potenciálů neuronů. Zapojení elektrod může být unipolární nebo bipolární. V případě bipolárního zapojení je snímaná amplituda měřena jako rozdíl potenciálů mezi dvěma aktivními elektrodami. U unipolárního zapojení je snímané napětí detekováno mezi aktivní a referenční elektrodou. Při povrchovém neinvazivním měření mají na výslednou amplitudu vliv především povrchové struktury mozku. Vliv podkorových struktur je na zaznamenaný signál typicky minimální. Elektrický potenciál generovaný jednotlivými neurony je příliš malý na to, aby byl přímo zachycen pomocí EEG. EEG aktivita tedy ve skutečnosti odráží sumaci synchronní aktivity tisíců až milionů neuronů. Naměřená amplituda z povrchu skalpu se typicky pohybuje v řádu desítek μV (mikro voltů) [HM01; Luc05].

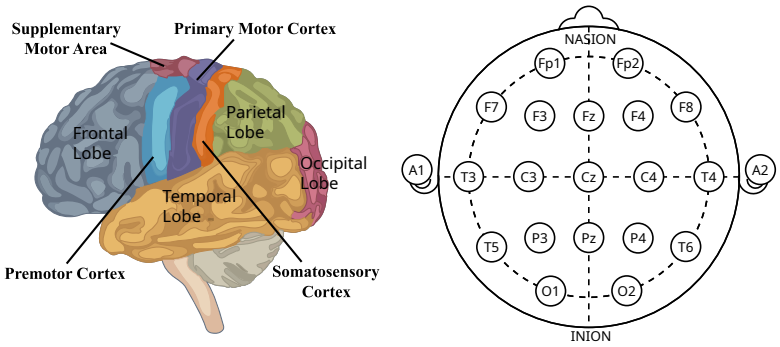
Díky velké rychlosti šíření elektrické aktivity má EEG signál výborné časové rozlišení. Lze zachytit i události, které se vyskytují v milisekundových časových úsecích [Roy+19]. Mezi další výhody EEG, oproti ostatním modalitám zaměřených na

mozkovou aktivitu, patří například *nízká pořizovací cena* a lepší *přenositelnost* [Luc05; Pad+19]. Na druhou stranu ovšem EEG trpí nízkým prostorovým rozlišením. To je způsobeno faktem, že naměřené hodnoty jsou zkreslené mezilehlými tkáněmi a kostí, které působí podobně jako rezistory a kondenzátory v elektrickém obvodu. V důsledku toho jsou kanály EEG signálu poměrně hodně prostorově korelované [Roy+19]. Dalšími problémovými vlastnostmi EEG signálu jsou nestacionarita signálu, velmi vysoké riziko *zašumění* z vnějšího okolí a tím tedy nízký poměr signálu k šumu (SNR), náchylnost na vznik artefaktů například mrkáním nebo svalovou aktivitou [Pad+19]. V neposlední řadě omezuje užitečnost aplikací EEG také vysoká **variabilita** mezi jednotlivými subjekty [Roy+19].

EEG signál se typicky popisuje jako rytmická aktivita a přechodové jevy. Rytmická činnost je rozdělována do frekvenčních pásem. Nejčastěji používaná frekvenční pásma jsou [NA23; NNS16]:

- `Delta` – 0.5-4Hz. Tyto vlny jsou nejčastěji přítomné v hlubokém spánku a jsou nejvíce výrazné v předních a centrálních oblastech hlavy.
- `Theta` – 4-7Hz. Rytmus, který se vyskytuje při ospalosti a také v raných fázích spánku. Nejvýraznější je ve fronto-centrálních oblastech hlavy a pomalu migruje dozadu a nahrazuje alfa rytmus kvůli časné ospalosti. Zvýšené emoční stavy mohou také zvýšit frontální rytmický theta rytmus.
- `Alfa` – 8-12Hz. Zadní dominantní alfa rytmus je charakteristicky přítomen v normálních záznamech EEG při bdělém stavu v týlní oblasti hlavy. Je to určující znak normálního rytmu pozadí záznamu EEG pro dospělé. Rychlé varianty pozadí alfa rytmu jsou vidět v normální populaci. Amplituda alfa rytmu se u různých jedinců liší, ovšem liší se i v různých časech u téhož jedince. Nejvíce se tyto vlny vyskytují při zavřených očích a při duševní relaxaci a je charakteristicky utlumena otevíráním očí a duševním úsilím.

  Mu (μ) rytmus je další typ alfa rytmu, který se vyskytuje v centrálních oblastech hlavy. Tento rytmus charakteristicky mizí s *motorickou* aktivitou kontralaterálních končetin nebo myšlenkou na zahájení motorické aktivity.
- `Beta` – 13-30Hz. Beta rytmus je nejčastěji pozorovaný rytmus u normálních dospělých a dětí. Nejvýraznější je ve frontální a centrální oblasti hlavy a zeslabuje, jak postupuje dozadu. Amplituda aktivity beta je obvykle 10 až 20 mikrovoltů, která se zřídka zvýší nad 30 mikrovoltů. Beta aktivita je úzce spojena s motorickým chováním a při aktivních pohybech je obecně utlumena.
- `Gamma` – 30-80Hz. Gamma rytmus se vyskytuje při bdělém stavu a typicky je připisován smyslovému vnímání integrující různé oblasti.



Obrázek 2.1: Ukázka umístění primárního motorického kortexu vůči klasickému EEG systému zapojení 10-20. Zdroj diagramu mozku: `https://www.chegg.com/learn/topic/diagram-of-premotor-cortex`. Zdroj systému 10-20: `https://en.wikipedia.org/wiki/10-20_system_(EEG)`

2.1 Představa pohybu

Představa pohybu (Motor Imagery, MI) je mentální proces, kterým jedinec nacvičuje nebo simuluje danou činnost bez toho aniž by ji ve skutečnosti vykonal. Jedná se o dynamický stav, při kterém dochází k vědomé aktivaci mozkových oblastí, které jsou zapojeny také při přípravě pohybu a jeho provedení. Pokud se provádí představa pohybu opakovaně, bývá již označována jako mentální trénink. Za cíl má nácvik či zlepšení motorických aktivit [AG17].

Obecně se předpokládá, že se při MI zapojují podobné nervové dráhy a mechanismy jako při skutečném fyzickém pohybu, ovšem s tím rozdílem, že při představě pohybu je aktivace mozkových oblastí nižší. Když si představujeme provádění motorické aktivity, mozek aktivuje oblasti související s motorikou, včetně *primární motorické kůry* a *premotorické kůry* [AG17; Mul07; Krü+20]. Tyto oblasti jsou zodpovědné za plánování, koordinaci a provádění pohybů. Motorická kůra se nachází v centrální oblasti mozku, což při standardním elektrodovém systému 10-20, odpovídá především elektrodám ***C3***, ***Cz*** a ***C4*** viz obrázek 2.1 na str. 9.

S představou pohybu, a s pohybem obecně, jsou v EEG signálu spojovány dva fenomény. Prvním je výskyt tzv. senzomotorického rytmu (SMR) a druhým je výskyt ERD/ERS (Event-related desynchronization)/(Event-related synchronization).

2.1.1 SMR

Jednou ze základních vlastností neuronových sítí je schopnost neuronů pracovat *synchronně* a generovat oscilační aktivitu. Jak již bylo zmíněno, jedna skupina mozkových oscilací má u člověka frekvenční pásmo 8-13Hz. Tyto oscilace vznikají v senzomotorických oblastech. Těmto aktivitám se říká *Mu* rytmus nebo také somato-

senzorický rytmus (SMR) [Pfu+06]. Obecně je dobře známo, že při plánování a samotném provádění pohybu dochází k desynchronizaci tohoto rytmu. Jeho opak, synchronizace rytmu, nastává v období po dokončení pohybu a při relaxaci (klidu). Desynchronizací se rozumí pokles výkonu (druhých mocnin amplitud frekvenčního spektra rytmu), naopak při synchronizaci dochází zpětně k nárůstu výkonu. Protože jsou změny nezávislé na aktivitě v normálních výstupních kanálech periferních nervů a svalů, je možné zvýšení a snížení tohoto rytmu použít jako základ pro BCI [Pfu+06; Vař18], viz následující sekce 2.2.

SMR je tedy základní rytmická aktivita senzomotorické kůry, kdežto ERD/ERS popisuje změny výkonu SMR během MI. Analýzou struktur ERD/ERS lze tedy provést detekci MI z EEG signálu.

2.1.1.1 ERD/ERS

Snížení/zvýšení výkonu související s událostí, a specifické pro frekvenční pásmo, se označuje jako desynchronizace související s událostí (Event-related desynchronization, ERD) a synchronizace (Event-related synchronization, ERS). [Nak+14]. Desynchronizace začíná zhruba **2 sekundy** před tím než je proveden samotný pohyb [PL99]. ERS a ERD mohou být indukovány pomocí MI, motorickou aktivitou a stimulací smyslů. Běžně používané příklady pohybů pro MI EEG je pohyb levé ruky, pohyb pravé ruky, pohyb prstů, pohyb chodidel a pohyb jazyka. Protože se ukázalo, že tyto události způsobují významné a diskriminační změny v EEG signálech [Pad+19].

Jednou ze základních vlastností měření ERD/ERS je, že výkon EEG signálu v pásmech alfa a beta, je procentuálně relativní k výkonu stejného EEG signálu zaznamenaného během referenčního období, typicky používaného jako několik sekund před výskytem samotné události. Protože změny v EEG signálu související s událostí potřebují nějaký čas na vývoj a následné zotavení, měl by interval mezi dvěma po sobě jdoucími událostmi být *alespoň* několik sekund [PL99].

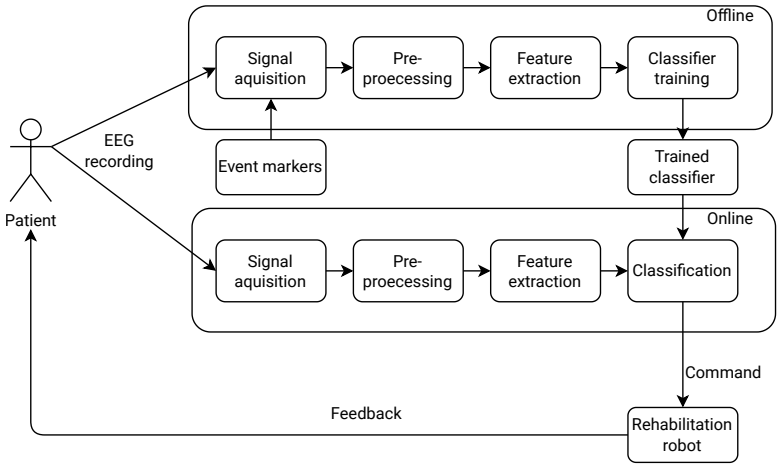
Metoda výpočtu ERD zahrnuje následující kroky [PL99]:

1. Filtrování pásmovou propustí (s mezními frekvencemi pro dané pásmo) všech signálů souvisejících s danou událostí

2. Umocnění všech vzorků vyfiltrovaného signálu na druhou, pro získání výkonu signálu

3. Průměrování výkonu signálu přes všechna měření související s danou událostí

4. Průměrování vzorků v průběhu času pro vyhlazení a snížení variability



Obrázek 2.2: Vizualizace offline a online procesu BCI. Při offline fázi dochází k měření EEG signálu za účelem natrénování klasifikátoru. Při online fázi BCI dochází ke klasifikaci aktuálně sbíraných dat pomocí klasifikátoru, naučeného v offline fázi, a následnému řízení externího zařízení na základě výsledku klasifikace

Finální výpočet procentuálních hodnot ERD/ERS je proveden následující rovnicí [PL99]:

$$ERD\% = \frac{A - R}{R} \cdot 100 \tag{2.1}$$

kde A je průměrný výkon zkoumaného frekvenčního pásma v rámci jedné události a R je průměrný výkon zkoumaného frekvenčního pásma v rámci dané události vypočítaný pouze z referenčního období.

2.2 BCI

Rozhraní mozek-počítač (Brain-computer interface, BCI) představuje přímou komunikační cestu mezi elektrickou aktivitou mozku a nějakým externím zařízením. To znamená, že člověk je teoreticky schopen řídit a komunikovat s vnějším světem pouze pomocí svojí mozkové aktivity.

Přestože se v této diplomové práci nebude přímo pracovat s finální částí BCI, tedy řízením externího zařízení, považuji tuto sekci jako poměrně důležitou. A to z důvodu, že se práce zabývá prvními a to poměrně důležitými kroky pro realizaci BCI a to sběrem EEG signálu, jeho předzpracování a následné klasifikace. Za předpokladu dobrých výsledků klasifikace, a tedy detekce pohybu, je možné v navazujících pracích provádět samotnou komunikaci s externím zařízením. Konkrétně v případě této studie by roli externího zařízení hrál rehabilitační robot, který by měl pacientům zjednodušit a zpříjemnit proces rehabilitace.

Obecně by se dalo BCI rozdělit na dvě fáze: offline a online, viz obrázek 2.2 na str. 11. V offline fázi dochází k měření EEG signálu, který je ukládán na nějaké datové úložiště. Zároveň se signálem jsou také sbírány značky daných událostí (event markers) což mohou být například značky indikující pohyb pravé ruky nebo pohyb levé ruky atd. Výsledné naměřené signály jsou pak předzpracovány a použity jako vstupní data pro trénování klasifikátoru. V online fázi dochází samozřejmě také k měření EEG signálu, ovšem s tím rozdílem, že signál není nutně ukládán na nějaké trvalé úložiště, ale přímo použit jako vstupní data pro klasifikaci natrénovaným klasifikátorem. Výsledek klasifikátoru je pak možné použít pro zaslání nějakého řídícího příkazu rehabilitačnímu robotovi[1]. Robot tak automaticky provede nějakou požadovanou akci a tím dává pacientovi zpětnou vazbu pouze na základě jeho mozkové aktivity.

BCI založené na EEG zaznamenávají stále větší a větší zájem, jakožto jedna z možností poskytující přímou komunikační cestu mezi lidským mozkem a externím zařízením. Jelikož je možné EEG zaznamenávat neinvazivně, představuje tento přístup bezpečnou a snadnou cestu pro tvorbu aplikací, potencionálně použitelnou téměř všemi lidmi, včetně vážně amputovaných a ochrnutých pacientů [TLS17].

BCI založené na EEG lze dále rozdělit do dvou typů: evokované a spontánní. V evokovaných systémech je zapotřebí, pro vyvolaní specifické aktivity mozku, nějaká vnější stimulace. Vyvolané elektrické potenciály se nazývají evokované potenciály (EP). Vnější stimulace je zaměřena na nějaký smyslový orgán, takže je pak možné je dále dělit na vizuální, sluchové atd. Vyvolaná aktivita je pro daný stimul poměrně specifická a BCI systémy se ji pak snaží automaticky detekovat. Na druhou stranu u spontánních BCI systémů není vnější stimulace nutná, jelikož se kontrolní akce provádějí na základě aktivity vznikající v důsledku mentální činnosti uživatele [Pad+19].

Evokované systémy obvykle vyžadují od uživatele nižší náročnost co se týče mentálního tréninku a v důsledku toho jsou zvládnutelné větším počtem uživatelů oproti spontánním systémům. Na druhou stranu ovšem vyžadují aby se uživatel neustále soustředil na externí stimul. Tato konstantní koncentrace může být pro uživatele poměrně vyčerpávající [Pad+19].

Příkladem spontánních BCI systémů jsou systémy založené právě na MI. Automatická detekce představy pohybu svých končetin samozřejmě budí široký zájem v různých oblastech především neurorehabilitace, neuroprotetika ale i v hraní her [Pad+19]. Jelikož lze výše popsané jevy ERD/ERS považovat za intuitivní činnosti, lze na jejich základě stavět BCI. Předpokládá se ovšem, že generování ERD/ERS se poměrně liší v závislosti na jednotlivci a pro většinu začínajících uživatelů jako *obtížná* [Nak+14; Pad+19]. Kromě toho obecně chybí pochopení vztahu mezi dobrým

[1] V obecném BCI se samozřejmě může jednat o libovolné externí zařízení, nemusí to nutně být rehabilitační robot.

a špatným výkonem BCI systému. A to především protože jsou velmi závislé na tom, aby se uživatelé úspěšně naučili vědomě generovat požadované signály [Pad+19].

Přestože systémy založené na MI jsou čím dál tím více populární stále trpí na to, že zpracování dat MI je poměrně náročné a přístupy na klasifikaci jsou složité. Většina systémů trpí nepřesnou klasifikací díky nestabilitě měřeného EEG signálu. Další nepříznivou roli může také hrát fakt, že například u pacientů kteří prodělali mozkovou mrtvici mohou naměřená data vypadat výrazně jinak oproti zdravým subjektům [Pad+19].

Jeden zajímavý problém v oblasti BCI je takzvaná BCI negramotnost. K negramotnosti dochází, když uživatel není schopen ovládat BCI a to protože není schopen produkovat kvalitní mozkové signály, které jsou k ovládání potřebné. Tento problém nemusí nutně znamenat, že je něco špatně s uživatelem. Uživatel může být plně zdravý a jeho nervová spojení mohou produkovat potřebný a správný signál, ovšem aktivita nemusí být detekovatelná pomocí povrchového EEG [Vař18; Pad+19].

2.3 Rehabilitace

EEG signál má v oblasti medicíny mnoho využití. Mezi klasické příklady použití patří především identifikace epilepsie, detekce mozkových nádorů, mozkové mrtvice, spánkových poruch nebo monitorace pacienta při kómatu [HM01]. Ovšem jak z minulé sekce vyplynulo, je měření EEG signálu možné použít i pro účely BCI.

V posledních letech se mnoho zemí potýká se stárnutím společnosti. S nárůstem počtu starších lidí se zvyšuje počet pacientů s mrtvicí a motorickou paralýzou. Jako jeden z účinných neurorehabilitačních prostředků pro pacienty po cévní mozkové příhodě byla navržena BCI, protože mohou uzavírat narušenou senzomotorickou nervovou cestu kompenzací somatosenzorické zpětné vazby na jejich motorický pokus. Zkušenosti s neurorehabilitací pomocí BCI by měly být důležité pro podporu neuroplasticity mozku při zotavování z motorické paralýzy [Nak+14]. Pacient tak nejen trénuje propojení nervové aktivity s pohybem, ale zároveň dostává od robota přímou zpětnou vazbu, což může vést k urychlenému zotavení.

Rehabilitací se tedy rozumí proces učení, kdy se pacient snaží znovu získat staré dovednosti a naučit se nové na základě praxe. Aktivním cvičením se znovu vytváří tok aferentních (tzn. dostředivých, neboli vedoucích od periferních nervů do centrální nervové soustavy) informací [Laz+18; Mul07]. V praktických případech se ovšem nejedná pouze o provádění fyzického cvičení, ale také dodatečné mentální cvičení, např. MI hraje důležitou roli pro zlepšení motorického výkonu [Krü+20].

Bylo prováděno několik studií, používajících MI, které prokázaly zlepšení v rychlosti, síle a přesnosti provádění motorického úkonu. Ve fázi učení byla nalezena podobná asymptotická křivka učení jak pro MI, tak pro fyzické vykonávání pohybů. Což tedy dále podporuje fakt, že MI není pouze epifenomenální (vedlejší a

nepřímé), ale hraje důležitou roli v kortilální plasticitě související s prováděním pohybů [Krü+20; Laz+18].

Pro použití BCI k rehabilitačním účelům existují následující tři strategie [AG17]:

1. Operativní podmiňování, kdy je naučen jeden generalizovaný fixní klasifikační model (inter-subjekt model). Neboli existují již nějaká měření různých pacientů, na kterých byl model naučen a který je pak online použit pro rehabilitační cvičení konkrétního subjektu.

2. Strategie s použitím strojového učení pro vytvoření subjekt specifického modelu (intra-subjekt modelu). Nejdřív proběhne kalibrační fáze, kdy jsou pro daného pacienta prováděna měření bez toho, aniž by dostával nějakou zpětnou vazbu od externího zařízení. Na základě těchto naměřených dat pak proběhne offline trénování nějakého klasifikačního modelu. Natrénovaný model pouze na datech daného pacienta je pak použit v dalších rehabilitačních cvičeních pro klasifikaci a řízení rehabilitačního zařízení.

3. Adaptivní strategie. Tato strategie je velmi podobná té předchozí. Ovšem s tím rozdílem, že při dalších cvičeních dochází postupně k přetrénování vytvořeného modelu na nově naměřených datech.

Související práce 3

V první sekci této kapitoly jsou popsány podobné experimenty vybraných prací provedené na jiných institucích. Druhá sekce 3.2 je věnována popisu prací, které probíhali na Katedře informatiky a výpočetní techniky Fakulty aplikovaných věd na Západočeské univerzitě v Plzni. Konkrétně se jedná o práce Pavla Mochury [Moc21] a Josefa Saleha [Sal22], na kterých tato práce staví a a zároveň na ně navazuje. Tato sekce bude také zabíhat do určitých implementačních detailů obou prací.

V této kapitole budou také zmíněné různé metody a techniky používané v oblasti zpracování, augmentace a klasifikace EEG signálu aniž by nutně došlo k vysvětlení jakým způsobem daná metoda funguje. Techniky vybrané pro implementaci budou popsány detailněji v následujících kapitolách 4 až 6 a způsob jejich implementace v kapitole 7.

3.1 Rešerše literatury

3.1.1 Generování a rozpoznávání signálu MI-EEG 4 tříd pomocí cVAE-GAN

V roce 2021 provedli Yang et al. [Yan+21] studii, jejíž cílem byla detekce čtyř úkonů MI. Konkrétně se jednalo o detekci pohybu levou rukou, pravou rukou, jazykem a obou chodidel. Svoje metodiky ověřovali na dvou datových sadách. První dataset byl jejich soukromý, ve kterém proběhlo měření tří subjektů. Druhý použitý dataset byl veřejný dataset Competition IV Datasets 2a o devíti subjektech. Jejich experimentální měření probíhalo tak, že subjekty byly postaveny před obrazovkou počítače, na které se zobrazovaly úkoly, které mají provádět. Každý úkol trval 8 sekund a mezi po sobě jdoucími úkoly byla stejně dlouhá pauza.

Analýzou dat zjistili, že MI úkoly způsobují ERD a ERS nad pravou a levou hemisférou motorické kůry a to především na elektrodách ***C4*** a ***C3***. Elektroda ***Cz*** byla ovlivněna především MI chodidel a jazyka. Ve výsledku tedy použili záznamy právě z těchto elektrod. Pro augmentaci použili 2 sekundy dlouhé segmenty naměřeného

EEG signálu. Signál byl v rámci předzpracování dat převeden pomocí Short time Fourier transform (STFT) do časově frekvenční oblasti.

Jako augmentační metodu použili kombinaci Conditional Variational Autoencoder (cVAE) a General Adveserial Network (GAN). Jejich architektura cVAE-GAN byla tedy složena ze 3 částí: 1. enkodér, který mapoval vstupní příznaky na latentní reprezentaci (blíže popsáno v kapitole 6.2.1). 2. generátor, který z latentní reprezentace generoval falešné EEG signály. 3. rozpoznávací síť, která rozlišovala mezi skutečnými signály a vygenerovanými signály a zároveň měřila pravděpodobnost příslušnosti dat k třídám MI. 70% reálných vzorků pak bylo použito jako trénovací množina a zbylých 30% sestávající z kombinace reálných a vygenerovaných vzorků byla použita jako testovací množina. Pro všechny subjekty byla křížově validační metrika `accuracy` o několik **jednotek procent**[1] **lepší** za použití vygenerovaných dat jejich augmentační metodou, oproti klasifikaci bez použití vygenerovaných dat.

3.1.2 Single-trial klasifikace MI EEG použitím CNN

Tang et al. [TLS17] ve své práci používali hlubokou konvoluční neuronovou síť (CNN) pro klasifikaci jednotlivých pokusů (single-trial) MI levou a pravou rukou. Jejich dataset sestával z měření dvou zdravých subjektů. Data byla měřená pomocí 28 elektrod s vzorkovací frekvencí 1000Hz. Každý ze subjektů byl posazen před obrazovku. Každý MI pokus začal akustickým signálem a zobrazením kříže na obrazovce. Sekundu na to se na obrazovce objevila náhodně šipka vlevo nebo vpravo indikující jaký typ MI má subjekt provádět. Subjekt pak prováděl MI po dobu pěti sekund. Poté byla krátká pauza náhodně mezi dvěma až pěti sekundami před začátkem dalšího pokusu. Každý ze subjektů takto provedl celkem 460 pokusů, 230 pro každou z MI tříd.

Naměřená data byla filtrována filtrem s pásmovou propustí s mezními frekvencemi 8-30Hz. Následně provedli analýzu dat pomocí vizualizace ERD/ERS, vypočteného pomocí rovnice 2.1, pro každou třídu. Na základě vizualizace pak byl vybrán 3 sekundový úsek (po jedné sekundě od zobrazení šipky) signálu, který byl segmentován po 50ms oknech. Takto předzpracovaný dataset byl rozdělen na 80% trénovacích a 20% testovacích dat.

Architektura jejich konvoluční sítě sestávala z pěti vrstev. První vstupní vrstva, dvě skryté konvoluční vrstvy, které provádí extrakci příznaků, plně propojená skrytá vrstva a výstupní vrstva. Kromě vyhodnocení MI klasifikace na navrženém klasifikátoru také provedli porovnání s ostatními běžně používanými klasifikátory. Konkrétně bylo provedeno porovnání s SVM, CSP + SVM a AR + SVM. Pro každý klasifikátor používali 10-násobnou křížovou validaci. CNN prokázala lepší výsledky než

[1] Ve studii se nenachází tabulka s konkrétními výsledky ale pouze grafová reprezentace. Nelze tedy přesně říci o jaké konkrétní zlepšení došlo.

klasifikátor SVM s různým způsobem extrakce příznaků. CNN dosáhla průměrné klasifikační přesnosti (`accuracy`) 86.41±0.77%, zatímco nejlepší výsledek pomocí klasifikátoru SVM byl dosažen s příznakovou extrakcí pomocí CSP 82.61±6.15%.

3.1.3 Augmentace EEG datových sad pomocí GAN

Abdelfattah et al. [AAW18] experimentovali s rekurentní architekturou GAN augmentační metody, pro rozšíření datasetu pro klasifikaci MI. Používali volně dostupný PhysioNet dataset, který obsahuje měření 109 zdravých subjektů. Dataset byl použit pro detekci následujících tříd MI úkolů: zavření očí, otevření očí, zatnutí levé dlaně v pěst a zatnutí pravé dlaně v pěst. Naměřená data používala 64 elektrod a byla vzorkovaná frekvencí 160Hz. Ve fázi předzpracování provedli filtraci pásmovou propustí s mezními frekvencemi 0.5-42Hz a pro odstranění artefaktů signálu provedli analýzu metodou Independent component analysis (ICA). Výsledný pročištěný dataset pak obsahoval 240 datových vzorků pro každý subjekt za každou třídu MI, tzn. celkový obsahoval 104 640 ($109 \cdot 240 \cdot 4$) datových vzorků.

Generátor architektury GAN používal 2 rekurentní skryté vrstvy o 256 neuronech a 2 plně propojené vrstvy o 128 neuronech. Diskriminátor navržené sítě obsahoval 3 skryté plně propojené vrstvy, kde první dvě měly 256 neuronů a poslední 128. Účinek navržené augmentační metody byl testován na klasifikační přesnosti tří klasifikátorů: MLP, SVM a Random forest tree (RFT).

Pro každou klasifikaci byla provedena 5-násobná křížová validace. Nejprve prováděli experimenty, kde byly klasifikátory učeny pouze na 100% reálných datech. MLP poskytoval nejlepší výsledky o 18% lepší než SVM a o 14% lepší než RFT. Následně trénovali klasifikátory pouze s 25% reálnými daty, kde MLP utrpělo největší propad v přesnosti oproti ostatním klasifikátorům. Tento výsledek byl porovnán s výsledky, kdy bylo opět bráno pouze 25% reálného datasetu a zbylých 75% dat byl doplněn vygenerováním nových vzorků pomocí navržené augmentační metody, viz tabulka 3.1 na str. 18. Stejným způsobem pak zkoušeli podobný experiment s tím, že bylo použito 50% reálných dat. Výsledky byly proporcionálně obdobné výsledkům s využitím 25% dat. U všech klasifikátorů byla klasifikační přesnost vyšší. Tyto výsledky ukázaly citlivost modelů založených na hlubokém učení na velikosti trénovací datové sady. Klasifikační přesnost všech klasifikátorů byla lepší při použití augmentované datové sady.

Kromě porovnávání zlepšení výsledků přesnosti klasifikátorů, také porovnávali navrženou architekturu s ostatními augmentačními metodami. Konkrétně provedli experiment přesnosti rekonstrukce datové sady pomocí jejich augmentační metody, pomocí AE a pomocí VAE. Rekonstrukční přesnost (rekonstrukce je blíže popsána v kapitole 6.2.1) AE byla 54.9±6.5, přesnost VAE byla 69.9±3.7 a rekurentní GAN 89.8±3.5. Rekurentní GAN měla tedy o 34.8% lepší rekonstrukční přesnost dat oproti

Klasifikátor	25% datasetu	25% reálných + 75% vygenerovaných
MLP	46.2±6.3	82.3±4.6
SVM	54.4±5.4	68.5±5.2
RFT	62.1±4.7	75.2±6.1

Tabulka 3.1: Porovnání výsledků Abdelfattah et al. v přesnosti klasifikace při použití pouze 25% dat a při použití 25% data doplněných 75% vygenerovaných dat

AE a o 19.9% lepší než VAE. Tyto výsledky naznačují, že GAN podchycuje statistické charakteristiky EEG signálu lépe než porovnávané metody.

3.1.4 Augmnetace a klasifikace MI signálů použitím hybridních neuronových sítí

Zhang et al. [Zha+20] ve své studii experimentovali s různými augmentačními metodami za účelem zlepšení klasifikace CNN pro detekci MI. Pro vyhodnocení klasifikátorů a augmentačních metod používali 2 datasety.

První použitý dataset je volně dostupný BCI competition IV dataset 1. Z tohoto datasetu použili data od čtyř naměřených subjektů (konkrétně subjekty b, d, e a g). Dataset byl měřen vzorkovací frekvencí 100Hz a každý subjekt prováděl 200 pokusů MI, celkem tedy data sestávala z 800 pokusů, přičemž byly použity pouze elektrody ***C3***, ***Cz*** a ***C4***. Data byla měřena na zdravých subjektech, každý ze subjektů prováděl dvě ze tří tříd MI: pohyb levou rukou, pohyb pravou rukou a nebo pohyb chodidlem (subjekt si mohl vybrat jakým, případně oběma chodidly). Experiment probíhal následovně. Subjekt dostal akustický podnět indikující třídu MI, kterou měl vykonat. Subjekt vykonával MI po doby různé délky od 1,5-8s. Konec pokusu byl označen slovem stop na obrazovce, načež následovala pauza opět mezi 1,5-8 sekundami.

Druhý použitý dataset byl také veřejně dostupný, konkrétně se jedná o BCI competition IV dataset 2b. Dataset obsahuje data měřená od devíti zdravých subjektů. Každý subjekt prováděl celkem pět měření, přičemž v prvních dvou měřeních prováděl vždy každý subjekt 120 pokusů MI a v posledních třech měřeních vždy 160 pokusů. Celkem se tedy jedná o 720 pokusů od každého z devíti subjektů. Třídy MI v tomto datasetu jsou dvě: pohyb levou a pohyb pravou rukou. Data byla měřena vzorkovací frekvencí 250Hz. Experiment probíhal následovně: před prvním měřením si subjekt vybral pohyb, který se mu představoval nejlépe například mačkání míče nebo tahání za brzdu. Následně každý pokus začínal tak, že na obrazovce byl zobrazen kříž a těsně před začátkem MI byl zahrán krátký akustický signál. Následně byla na obrazovce zobrazena šipka směrem vlevo nebo vpravo indikující třídu MI, kterou měl subjekt vykonávat po dobu čtyř sekund. Po každém pokusu

následovala náhodně trvající pauza o délce alespoň jedné sekundy. Z tohoto datasetu byly také použity pouze elektrody ***C3***, ***Cz*** a ***C4***.

V rámci předzpracování byla data filtrována Butterworthovým filtrem s pásmovou propustí s mezními frekvencemi 8-30Hz. Každý z pokusů měřených signálů byl převeden pomocí STFT do časově frekvenční oblasti. Byl použit signál o délce čtyř dekund (v době trvání MI), tzn. pro první dataset 400 vzorků a pro druhý 1000 vzorků. Časově frekvenční oblasti z každého kanálu byly na sebe vertikálně naskládané a jako poslední krok byly převedeny na spektrogramové obrázky o rozměrech 64x64.

V této studii bylo implementováno a porovnáváno pět augmentačních metod. Geometrické transformace (GT), přičemž konkrétně se jednalo o rotaci obrázku o 180° okolo osy x, následný posun obrázků doleva, doprava, nahoru nebo dolů a zbylé pixely byly doplněny náhodným šumem a nakonec augmentace barevného prostoru obrázků. Druhá augmentační metoda použitá ve studii byla přidání náhodného šumu (NI) z normálního rozdělení. Dalšími metodami byly AE a VAE. Poslední metodou byla jejich navržená architektura hluboké konvoluční GAN. Generátor této sítě používal čtyři dekonvolučí vrstvy pro vygenerování falešného obrázku. Obdobně diskriminátor této sítě používal čtyři konvoluční vrstvy pro klasifikaci zda se jedná o reálný nebo falešný vstupní obrázek.

Pro vyhodnocení jednotlivých augmentačních metod byla použita metrika FID. Navržená hluboká konvoluční GAN dosáhla nejlepšího výsledku pro oba datasety. Konkrétně byly vypočteny hodnoty FID: 126.4, 98.2 pro první a druhý dataset pro konvoluční GAN, druhý nejlepší výsledek dosáhla metoda NI s hodnotami 159.1 a 188.5, VAE 277.1, 203.4, AE 323.5, 273.6 a GT 487.7, 501.8. Kromě vyhodnocení technikou FID, byly metody také použity pro augmentaci použitých datových sad pro klasifikaci klasifikátorem CNN. Při klasifikaci byla použita 10-násobná křížová validace, kdy byla v každé iteraci konkrétní datová sada rozdělena na 90% trénovacích dat a 10% testovacích dat. Uvedu zde pouze nejlepší dosažené výsledky jednotlivých metod pro druhý dataset. Pro kompletní přehled veškerých výsledků viz strana 13 a 14 ve studii [Zha+20]. Klasifikační přesnost (`accuracy`), bez rozšíření datové sady augmentační metodou, byla 80.6±3.2%. Nejlepších výsledků bylo dosaženo při použití poměru reálných a vygenerovaných dat 1:3. GT-CNN: 73.2±2.1%, NI-CNN: 86.2±3.1%, AE-CNN: 83.2±3.1%, VAE-CNN: 87.6±2.3% a GAN-CNN: 93.2±2.8%. Použitím všech metod, kromě GT, došlo tedy ke zlepšení klasifikační přesnosti, s tím že použitím konvoluční GAN došlo k nejlepšímu zlepšení a to v průměru o 12,6%.

3.1.5 Komplexní rešerše literatury

Samozřejmě existuje celá řada prací zabývajících se touto problematikou. V předchozích sekcích byly pouze podrobněji přiblíženy některé z nich, které se snažily

řešit podobný nebo dokonce stejný problém. Mimo jiné ovšem vzniklo také několik prací zabývajících se právě rešerší dostupné literatury v oblasti zpracování EEG signálu, tedy nejen problematikou MI ale i jiných. Tyto práce poskytují systematický přehled nejvíce používaných technik a metod pro zpracování, klasifikaci a vyhodnocení EEG signálu. Poskytují tedy statistiky a data, na základě kterých je možné provést informované rozhodnutí o tom, jaké techniky nebo metody je možné pro řešení daného problému využít. Konkrétně se jedná o práci vytvořenou Roy et al. [Roy+19], zabývající se technikami hlubokého učení pro analýzu EEG signálu, práce od Lashgari et al. [LLM20], která poskytuje informace ohledně augmentačních technik používaných pro hluboké učení v oblasti zpracování EEG signálu, a práce od Saegh et al. [ADA21], která se zabývá technikami hlubokého učení přímo pro klasifikaci MI.

3.1.5.1 EEG analýza založená na hlubokém učení: systematický přehled

Roy et al. [Roy+19] provedli systematický přehled studií provedených od roku 2010 do roku 2018. Přičemž prohledávali různé zdroje za použití kombinace různých (18) klíčových slov. Celkově po filtraci všech nalezených studií provedli analýzu ze 154 studií. Z těchto vybraných studií jsou v tomto přehledu podány informace o různých technikách a oblastí týkající se obecného zpracování EEG signálu.

Data. Polovina prozkoumaných studií používala data naměřená z méně než 13 subjektů, přičemž mediánový počet datových vzorků použitých jako vstupní data klasifikátoru byl 14000.

Augmentace dat. Ze 154 studií pouze 3 použily augmentaci dat jako možnost prozkoumání zlepšení přesnosti klasifikace klasifikátorů založených na hlubokém učení. Dalších 30 studií explicitně použilo augmentaci dat, ovšem jen nějaké z nich zkoumaly dopad augmentace na klasifikační přesnost.

Předzpracování dat. 72% studií použilo alespoň nějakou formu předzpracování dat, jako například podvzorkování nebo filtrování pásmovou propustí. Pouze 23% studií se pokoušelo o nějakou formu odstranění artefaktů z EEG signálu, zbylé studie artefakty neodstraňovaly nebo se o tom nezmiňují. Odstraňování artefaktů může být velmi zdlouhavý proces, proto ostatní studie spoléhaly na to, že si zašuměnými signály klasifikátory poradí. Extrakce příznaků je také jedna z náročnějších technik v oblasti zpracování EEG signálu. 49% studií žádnou extrakci příznaků neprovádělo, tzn. používaly přímo naměřenou časovou řadu jako reprezentaci signálu pro vstup klasifikátorů. 38% studií použilo příznaky odvozené z frekvenčního spektra signálu.

Klasifikace. Nejvíce studií (40%) použilo pro klasifikaci architekturu konvoluční neuronové sítě (CNN), 13% rekurentní neuronovou síť (RNN), dalších 13% autoencoder (AE) a 7% použilo kombinaci CNN a RNN. Počet vrstev jednotlivých sítí byl poměrně různý, s tím že většina studií použila od dvou do šesti vrstev. Nejčastěji použitý optimalizátor pro učení klasifikátoru byl ve 30% *Adam*. Většina studií použila pro vyhodnocení klasifikátoru klasické metriky `accuracy`, `recall`, `precision`, `F1-score` a `ROC AUC`. Poněkud překvapivě poměrně velké množství studií (42%) neprovádělo žádnou formu křížové validace, zatímco nejpopulárnější technika u studií, které ji použily, byla k-násobná křížová validace. Pouze 26% studií používala *intra-subjekt* klasifikaci, 62% se zaměřilo na *inter-subjekt* klasifikaci a 8% provedlo oba přístupy.

Reprodukovatelnost. Reprodukovatelnost výsledků experimentů je jedním základních kamenů pro posun vědy obecně. Ovšem bohužel pouze 53% studií použilo data, které jsou veřejně dostupná. Dále se pouze 13% studií rozhodlo veřejně poskytnout zdrojové kódy experimentů. Ve výsledku je tedy reprodukovatelnost studií poměrně náročná. Pouze 8% studií lze jednoduše zreplikovat, kdežto až 40% je naprosto nemožné zopakovat.

3.1.5.2 Augmentace EEG dat pro klasifikaci založené na hlubokém učení

Lashgari et al. [LLM20] se ve svém systematickém přehledu zaměřili především na studie a práce zabývající se datovou augmentací a hlubokým učením s využitím datových sad EEG signálů. Po vyfiltrování studií, které nesplňovaly jejich kritéria, jim zbylo 53 studií psaných od roku 2015 do roku 2019. 21% z těchto studií se zabývalo analýzou EEG signálu pro práci s MI.

Data. Většina studií použila pro předzpracování naměřených dat filtr typu pásmové propusti, což jim umožnilo zaměřit se pouze na frekvenční spektrum specifické pro danou oblast v rámci zpracování EEG. Většina studií (85%) se také ze signálu pomocí filtrů snažila manuálně odstranit šum. Reprezentace vstupních dat, jakožto příznakové vektory do neuronových sítí, bylo možno rozdělit do tří kategorií. První kategorií byl surový EEG signál (časová doména) použitá 36% zkoumaných studií. Druhá kategorie, nejčastěji použitá (49%), byly vypočítané vlastnosti z nezpracovaných signálů, typicky se jedná o převod do frekvenční domény nebo převod na časově-frekvenční spektrum. Třetí kategorií (15%) pak byly spektrogramy převedené na obrázky.

Augmentační metody. Nejčastěji (24%) použitou metodou augmentace byla metoda sliding window (česky volně přeloženo jako metoda klouzavého okna), při

které dochází k rozpadnutí signálu na několik menších částí. Velikost části signálu je definována počtem vzorků v jednom okně a dále také počtem vzorků o kolik se má okno posunout. Tato metoda může být implementována dvěma různými způsoby a to buď bez překrývání, kdy počet vzorků posunutí je o jeden větší než velikost okna, nebo s překryvem, kdy počet vzorků posunutí je menší než počet vzorků okna. Druhou nejpopulárnější (21%) metodou bylo generování syntetických dat pomocí GAN. Třetí metodou (17%) byla metoda Noise injection (NI). Dalšími metodami byly například: Fourierova transformace, nebo převzorkování méně zastoupených tříd nebo naopak podvzorkování nejvíce zastoupených tříd.

Dopad augmentace na výsledky klasifikace. Pouze 29 z 53 studií poskytlo výsledky klasifikátorů před a po augmentaci datové sady. U těchto studií tedy mohli Lashgari et al. vypočítat procentuální zlepšení přesnosti klasifikace. Celkové průměrné zlepšení všech metod bylo v průměru o s 0.29±0.08%, přičemž nejlepší zlepšení bylo metodou NI průměrně o 0.36%, druhá nejlepší metoda byla sliding window o 0.33%, třetí nejlepší GAN o 0.28%. Bylo provedeno také porovnání zlepšení klasifikační přesnosti na základě jednotlivých oblastí zpracování EEG signálu a pro MI bylo zlepšení ze 4 studií v průměru o 0.14%.

Klasifikace. Co se klasifikátorů týče, tak nejčastěji (62%) použitý klasifikátor byla CNN. Druhým nejvíce populárním (16%) klasifikátorem byla kombinace LSTM a CNN tzv. hybridní model. V 8% studií byl použit MLP a v 6% LSTM.

3.1.5.3 Přehled klasifikací založených na hlubokém učení pro MI EEG

Saegh et al. [ADA21] ve své práci probírali pouze studie, které se přímo zabývají zpracováním EEG signálu v oblasti MI. Konkrétně zpracovali techniky ze 40 studií psaných mezi roky 2015 a 2020.

Data. Celkem v prozkoumaných 40 studií bylo použito pouze 15 unikátních datových sad, z toho bylo 7 veřejně dostupných a 8 byly soukromé datasety dané studie. Jednotlivé datové sady se od sebe samozřejmě liší celou řadou metadat jako je počet použitých měřících elektrod, vzorkovací frekvence, počet subjektů nebo úkony vykonávané jednotlivými subjekty. Největší část studií (45.6%) byla zaměřena pouze na detekci dvou MI tříd, pohyb levou a pohyb pravou rukou. Druhá nejčastější (31.6%) klasifikační úloha se zabývala detekcí pohybů levé ruky, pravé ruky, jazykem a pohybem chodidel.

Předzpracování dat. Pouze 5 ze 40 studií se zabývalo nějakou formou odstranění artefaktů. Tyto studie pro odstranění artefaktů použily metody ICA, prahování nebo

Automatic artifact removal toolbox v Matlabu. Všechny studie samozřejmě použily nějakou formu filtrování pro zpracování specifického frekvenčního pásma. Všechny studie použily jednu ze tří kategorií pro reprezentaci vstupních dat klasifikátorům: časová řada (tzn. pouze předzpracovaný naměřený EEG signál), vypočítané příznaky (nejčastěji pomocí waveletové transformace nebo STFT nebo CSP) a obrázek.

Klasifikace. Jednoznačně nejpoužívanějším klasifikátorem byla v 73% studií CNN. Dalších 14% použilo formu hybridní architektury typicky kombinace CNN a LSTM. Nejčastěji (66%) použitou aktivační funkcí klasifikátorů byla ReLU, nejčastěji (47%) použitým optimalizátorem učení byl algoritmus Adam. Studie také používaly různou formu regularizace při učení, jmenovitě datovou augmentaci, dropout a nebo batch normalization. Drtivá většina studií (82%) použila pro experimentování programovací jazyk Python, zbylých 18% provedlo zpracování v Matlabu. Nejpopulárnějšími frameworky pro hluboké učení byly: TensorFlow (35%), PyTorch (26%) a Keras (22%).

3.2 Související práce na KIV

Na katedře informatiky a výpočetní techniky na Západočeské univerzitě v Plzni již proběhlo pár experimentů zaměřených na detekci MI z EEG signálu. Konkrétně se jedná o diplomovou práci Pavla Mochury [Moc21] a bakalářskou práci Josefa Yassina Saleha [Sal22], na které tato práce volně navazuje. Navržený scénář experimentů, které obě práce provedli, bude blíže popsán v kapitole 7.2.1 na str. 60.

3.2.1 Detekce pohybu končetin z EEG signálu při cvičení na rehabilitačním robotovi

Obě práce přistupují ke zpracování a klasifikaci dat odlišným způsobem. Mochura [Moc21] se v práci zabýval konstrukcí různých příznakových vektorů, které pak sloužily jako vstup do vícevrstvého perceptronu (MLP). První, a také nejúspěšnější co se klasifikační přesnosti týče, příznakový vektor byl vytvořen výpočtem ERD a ERS, viz rovnice 2.1 na str. 11, pro každé měření. Vzhledem k tomu, že ERD a ERS se počítá jako průměrný pokles nebo nárůst výkonu signálu v daném období (epoše) pro danou MI třídu, tak to znamená, že pro každé měření vznikl pouze jeden datový vzorek pro každou MI třídu. Další použité příznakové vektory byly vytvořeny vypočtením různých statistických hodnot z vypočítaného ERD a ERS, jako průměr, rozptyl, minimum a maximum, detailněji viz kapitola 10 v [Moc21]. Mochura vytvářel inter-subjekt model, přičemž nejlepšího průměrného výsledku přesnosti klasifikace 90.05%, bylo dosaženo po podvzorkování naměřeného signálu na 128Hz a příznakovým vektorem (označeným jako prvním) tvořeným vypočítaným ERD nasledovaným vypočteným ERS.

Při předzpracování byla zvolena epocha začínající 3.5s před a 0.5s po synchronizační značce pro danou MI třídu. V případě, že se jednalo o data měřené pro pohyb levou rukou, byla použita elektroda ***C4***, pokud se jednalo o data z měření pravou rukou tak byla použita pouze elektroda ***C3***[2]. V rámci práce byla tedy použita pouze jedna třída MI a to pohyb jakoukoliv končetinou, což ve výsledku znamená, že docházelo k binární klasifikaci pohyb vs klid. ERD bylo počítáno z alfa pásma naměřeného signálu, tzn. signál byl nejprve filtrován filtrem s pásmovou propustí s mezními frekvencemi 8-12Hz, kdežto ERS bylo počítáno v nižším beta pásmu (14-22Hz).

3.2.2 Návrh detektoru pohybu naměřených EEG dat

Na druhou stranu Saleh [Sal22] se zaměřil na detekci SMR, kdy je na jednotlivé epochy signálu aplikován filtr typu pásmové propusti s mezními frekvencemi 8-13Hz. Na takto vyfiltrované epochy je pak buď ještě aplikovaná metoda CSP, nebo jsou vyfiltrované epochy přímo vstupem do klasifikátorů SVM a LDA. Tzn. příznakové vektory tvořily přímo naměřené epochy jednotlivých pokusů (tzv. single-trial) nebo příznakové vektory vypočítané pomocí CSP z daných epoch. Pro všechny pokusy i třídy byly použity elektrody ***C3***, ***C4*** a ***Cz***. Saleh tvořil intra-subjekt modely, tedy personalizovaný model pro každý subjekt. Na rozdíl od Mochury také Saleh klasifikoval obě třídy MI, tedy prováděl vícetřídní klasifikaci, kde jednou třídou byly epochy levého pohybu, druhou třídou epochy pravého pohybu a třetí třídou epochy klidového stavu.

Výsledkům prezentovaných v práci popravdě příliš nerozumím. Jsou zde prezentovány spojnicovým[3] grafem výsledky `accuracy` a `precision` jednotlivých klasifikátorů pro jednotlivé subjekty. Respektive takto to platí pro výsledky dosažených využitím pouze Mochurovo subjektů, kde osa x obou grafů skutečně obsahuje 14 hodnot (Mochura prováděl měření na 14 subjektech). V grafech pro klasifikátory použité pro měření subjektů provedené Salehem ovšem osa x v grafu `accuracy` obsahuje 7 hodnot a graf `precision` 26 hodnot. Očekával bych, že se zde bude nacházet 10 hodnot, vzhledem k tomu že Saleh prováděl měření na 10 subjektech. Tabulka obsahující konkrétní hodnoty dosažených výsledků také obsahuje pouze 7 výsledků.

[2] Je možné si povšimnout, že pro levý pohyb byla použita elektroda umístěná nad pravou mozkovou hemisférou a naopak pro pohyb pravou rukou byla použita elektroda umístěna nad levou hemisférou. Což dává určitý smysl vzhledem k tomu, že se obecně předpokládá, že levá strana motorického kortexu primárně ovlivňuje pohyby pravé strany těla a naopak.

[3] Lepším grafem pro vizualizaci a porovnání výsledků by byl sloupcový graf.

3.2.3 Kombinace obou datových sad

Přestože obě práce používají odlišné zpracování EEG signálů, způsob měření dat z obou prací se zdá být totožný viz kapitola 8 v [Moc21] a kapitola 2.4 v [Sal22]. Nabízí se tedy možnost obě datové sady zkombinovat a tím tak získat větší a robustnější datovou sadu pro klasifikaci. Saleh již koneckonců svoji navrženou klasifikaci aplikoval jak na svoje naměřená data tak na data naměřená Mochurou. Každopádně Saleh tvoří intra-subjekt klasifikační modely, takže ke kombinaci datových sad vlastně nedochází.

Oba scénáře experimentu jsou sice totožné, každopádně jsou zde i nepatrné drobné rozdíly. Jedním z rozdílů je, že pro data naměřená Mochurou byla zvolena vzorkovací frekvence 500Hz, kdežto data Saleha byla měřena použitím vzorkovací frekvence 1000Hz. Řešení tohoto problému je poměrně jednoduché. Ve fázi předzpracování signálu bude potřeba buď novější data podvzorkovat nebo naopak na starší data použít nějakou interpolační metodu pro získání signálů stejné délky. Dalším rozdílem je, že při sběru starších dat byl signál měřen na 18 kanálech, zatímco u novějších dat byl měřen pouze na 9 kanálech. V obou případech ovšem při klasifikaci byly použity pouze kanály ***C3***, ***C4*** a ***Cz***, které se nacházejí přímo nad motorickými centry mozku. Dá se tedy předpokládat, že pro klasifikaci pohybu ze signálu budou nejrelevantnější právě tyto kanály, viz kapitola 2.1 a předchozí rešerše literatury. Rozdílný počet měřených kanálů je pro naši potřebu irelevantní. Posledním rozdílem je fakt, že při sběru nových dat měl v některých případech pacient na končetině vibrátor s haptikou. Dle [Sal22] by každopádně tento element neměl mít na výsledná naměřená data vliv. Vynecháním dat s haptikou se výsledná klasifikace příliš nemění, každopádně je možné, že je tento předpoklad mylný.

3.2.4 Analýza implementací předchozích experimentů

3.2.4.1 Detekce pohybu končetin z EEG signálu při cvičení na rehabilitačním robotovi

Mochura ve své práci vytvořil 2 Python skripty. V prvním skriptu dochází k načítání a předzpracování dat a následnému vytvoření jednotlivých příznakových vektorů. Páté, šesté a sedmé (takto označené v kapitole 10 v [Moc21]) příznakové vektory jsou tvořené výpočtem ERD a ERS a pouze pro konkrétní frekvenční pásma. Ovšem výpočty jsou prováděny na signálu, který byl nejprve filtrován pásmovou propustí s mezními frekvencemi nastavenými na 8-12Hz. Pro každé příznakové vektory jsou vytvořené 2 CSV soubory, jeden pro trénovací množinu a jeden pro testovací množinu. Co je poněkud zvláštní, je fakt, že data která tvoří trénovací a testovací množinu jsou přímo specifikována souborovými cestami. Tzn. nedochází k žádnému náhodnému rozdělení celkové datové množiny na množinu testovací a trénovací. Zároveň je poměr trénovacích a testovacích dat přesně 50%, což je v rámci strojo-

vého učení poměrně neobvyklé. Při předzpracování dat také nedošlo k odstranění artefaktů z naměřeného signálu.

Druhý vytvořený skript provádí klasifikaci pomocí klasifikátoru MLP. Klasifikace probíhá poněkud zvláštním způsobem. Dochází ve smyčce po dobu 100 opakování k načítání dat pouze prvních příznakových vektorů, které byly už předem rozděleny na trénovací a testovací sadu, viz předchozí odstavec. Dále dojde k inicializaci modelu klasifikátoru, který je následně učen na trénovacích datech po dobu 100 epoch. Výsledky klasifikátoru jsou pak ověřeny na testovací množině. Opakování tohoto procesu stokrát ve smyčce příliš nerozumím, jelikož jediné co se v každé iteraci změní je inicializace úvodních vah MLP, trénovací a testovací množina je pevně fixní. Nedochází tedy ke křížové validaci a zároveň nedochází ke kontrole cenové funkce, takže s největší pravděpodobností dojde k přeučení klasifikátoru.

Pro zajímavost jsem zkoušel porovnat originální výsledky s výsledky získanými po automatickém odstranění artefaktů a provedení 10-násobné křížové validace pro první příznakové vektory viz tabulka 3.2 na str. 26. Jak je možné si všimnout, tak po provedení křížové validace je průměrná klasifikační přesnost výrazně nižší, než při provedení opakované klasifikace nad stejnou trénovací a testovací sadou. Zároveň je poměrně vysoká variabilita mezi jednotlivými výsledky (vysoká směrodatná odchylka) takže trénování MLP je nestabilní. Hlavním důvodem bude pravděpodobně malý počet vstupních dat, jelikož pro každé měření bylo vypočteno ERD a ERS (průměry) z epoch jednotlivých tříd. Celkem na 14 subjektech bylo provedeno 40 měření (14 subjektů, každý měřen alespoň jednou pro MI levou a MI pravou rukou, některé subjekty byly měřeny dvakrát). Ve výsledku má tedy vstupní dataset pouze 80 vzorků (řádků vstupní matice) z toho 40 pro třídu reprezentující pohyb (jakýkoliv, tedy levou nebo pravou rukou) a 40 představujících klid.

Implementace	Accuracy
originální	90.05±2.67
odstranění artefaktů + křížová validace	67.00±10.54

Tabulka 3.2: Porovnání originálních výsledků Mochurovy implementace s upravenou implementací používající automatické odstranění artefaktů a 10-násobné křížové validaci

3.2.4.2 Návrh detektoru pohybu naměřených EEG dat

Saleh ve své práci vytvořil jeden Python skript, ve kterém tedy dochází k načítání, předzpracování a klasifikace jím i Mochurou naměřených dat. Implementace obsahuje poměrně velké množství duplicitního a zakomentovaného kódu, takže se v něm poměrně špatně orientuje. Při rozdělování datové množiny na testovací a

trénovací část bylo používáno rozdělení na 75% testovací a 25% trénovací množinu. V klasifikačních úlohách se typicky používá rozdělení dat přesně naopak. Křížová validace je implementovaná pouze pro klasifikátor SVM.

Dalším problémem, na který jsem narazil, byl fakt, že pro vytváření intra-subjekt modelů se z nějakého důvodu nenačítala data ze všech složek. Neboli data některých subjektů nebyla použita vůbec, přestože složky nějaké soubory obsahovaly. Ovšem je pravda, že pro pacienta s pořadovým číslem 11[4] existovaly pouze hlavičkové `.vhdr` soubory, ale už se zde nenachází příslušné `.eeg` soubory obsahující naměřená data. Pro pacienta s pořadovým číslem 3 zase chyběla data měřená s haptikou. Zároveň u měření pro levou ruku bez haptiky byly naměřeny pouze 4 pohybové fáze. Pravděpodobně tedy u tohoto pacienta došlo k nějaké chybě při měření. Je tedy možné, že se dané složky nenačítaly z toho důvodu, že by pro intra-subjekt model bylo pro daného pacienta malé množství dat.

Pro porovnání jsem zkoušel použít Salehovu implementaci také pro vytvoření inter-subjekt modelu, výsledky porovnané s průměrem z intra-subjekt modelů je možné nahlédnout v tabulce 3.3 na str. 27. Výsledky byly dosaženy použitím pouze datové sady naměřené Salehem.

	Klasifikátor			
Typ tvořených modelů	SVM	CSP SVM	LDA	CSP LDA
Průměr intra-subjekt modelů	51.69±8.12	64.52±11.08	31.08±5.15	61.51±12.61
Inter-subjekt model	75.51	58.08	68.46	58.61

Tabulka 3.3: Porovnání výsledků přesnosti Salehovy implementace klasifikátorů pro různý typ subjekt modelů, pouze na datech měřených Salehem

Jelikož jsem v této práci chtěl udělat hlavně klasifikaci inter-subjekt modelu ze všech naměřených dat, musel jsem implementovat vlastní načítání souborů (obě datové sady mají trochu jiný formát názvu viz kapitola 7.2.4.1). Výsledná matice při použití implementace vlastního načítání pouze Salehových dat měla zhruba poloviční počet řádek oproti matici získanou načítáním v Salehově programu. Dle [Sal22] bylo měřeno 10 pacientů a načítáno bylo 10 unikátních čísel v názvu datových souborů, a tak se implementace načítání se zdá být korektní. Při dalším zkoumání jsem zjistil, že byla chyba v načítání dat ve skriptu od Saleha viz příloha B na str. 109.

Po odstranění chyby načítání duplicitních dat jsem replikoval experiment porovnání intra-subjekt a inter-subjekt modelů, prezentovaný v tabulce 3.3 na str. 27, a dostal jsem nové výsledky viz tabulka 3.4 na str. 28.

[4] Dle [Sal22] bylo měřeno pouze 10 pacientů, takže je možné tyto soubory k datové sadě vůbec nepatří.

	Klasifikátor			
Typ tvořených modelů	SVM	CSP SVM	LDA	CSP LDA
Průměr intra-subjekt modelů	55.03±10.34	64.47±11.77	34.23±10.93	60.20±14.40
Inter-subjekt model	49.29	57.93	39.80	58.46

Tabulka 3.4: Porovnání výsledků přesnosti Salehovy implementace klasifikátorů pro různý typ subjekt modelů po odstranění chyb duplicitního načítání dat, pouze na datech měřených Salehem

3.3 Diskuze

Z rešerše literatury je vcelku jasné, že existuje několik způsobů a technik používaných pro zpracování EEG signálů a jeho následné využití pro detekci MI. Ve fázi předzpracování typicky dochází k filtrování signálu v určitém frekvenčním pásmu, přičemž v oblasti MI se používá filtr s pásmovou propustí pro *alfa* nebo *beta* pásmo. Další fází předzpracování je také výběr použitých kanálů (elektrod), z rešerše celkem jasně vyplývá, že relevantní jsou zejména kanály jsou ***C3***, ***Cz*** a ***C4***. Naměřené signály se dále tzv. epochují, neboli dochází k výběru těch částí signálů, u kterých došlo k nějaké události (např. subjekt dostal pokyn k pohybu rukou). Doba trvání jedné epochy bývá různá, ve studiích popsaných v rešerši se typicky pohybovala v rozmezí 2-4 sekund. Jedním z kroků by také mělo být odstranění artefaktů signálů, ovšem tento krok může být poměrně komplikovaný a proto ho asi většina studií nedělala.

Posledním krokem předzpracování je výběr příznaků. Většina probraných studií se konstrukcí příznakových vektorů nezabývala a použila jako vstup pro klasifikátory přímo naměřenou časovou řadu jednotlivých epoch. Ostatní studie provedly extrakci příznaků výpočtem spektrálních vlastností signálu, nebo převedením spektrogramu na obrázek. Mochura jako vstupní příznaky použil přímo vypočítané ERD a ERS, neboli průměrný pokles nebo nárůst výkonu jednotlivých epoch. Průměrování se ovšem v ostatních studiích příliš neprovádělo a spíše byl použit přístup tzv. single trials, neboli použití všech jednotlivých pokusů (epoch) jako vstup do klasifikátoru.

Metod pro augmentaci datové sady existuje také celá řada. Nejpopulárnějšími metodami se zdají být GAN (21%), sliding windows (24%), NI (17%) a AE (13%). Na základě výsledků studií ale není dopad augmentace na klasifikační přesnost příliš velký, viz kapitola 3.1.5.2 na str. 22.

Jednoznačně nejvíce populárním klasifikátorem je použití CNN (73%), dále bývá populární použití RNN nebo kombinace obou přístupů (14%). Klasifikátory MLP, LDA a SVM, implementované Mochurou a Salehem, mohou sloužit jako základní klasifikátory pro porovnání výsledků klasifikace komplikovanějších architektur. Po-

pulárním modelem je aktuálně také transformer, a to především v oblast zpracování přirozeného jazyka. V oblasti zpracování EEG signálů zatím tato technika příliš není, ovšem již byly provedeny některé studie zaměřené na použití transformeru, například studie [Tan+23].

Reprezentace EEG signálu

4

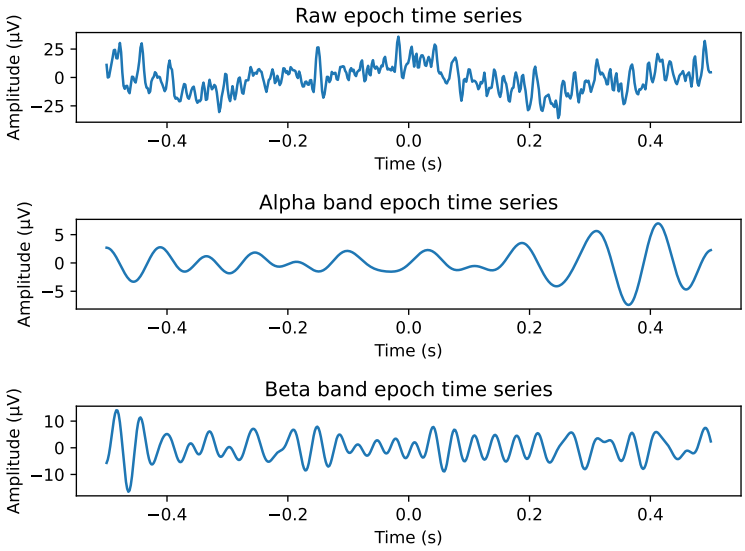
V této kapitole budou probrány různé datové reprezentace, které mohou být použité jakožto vstupní příznakové vektory pro trénování jednotlivých klasifikátorů. Jak z rešerše literatury vyplývá jsou nejpopulárnější vstupní příznaky reprezentovány buď časovou řadou, spektrálními charakteristikami signálu, nebo převedením spektrogramu na obrázek. V následujících částech budou popsány 3 vybrané vstupní reprezentace dat, které budou v práci implementovány.

4.1 Časová řada

Časová řada je s popularitou CNN jedna z nejpoužívanějších reprezentací, jelikož hlavní motivací pro použití CNN je automatické učení příznaků přímo klasifikátorem [LLM20]. Reprezentace časovou řadou je zároveň také nejjednodušší na implementaci, protože není nutné dělat žádnou extrakci příznaků vzhledem k tomu, že při měření EEG signálu sbíráme přímo časovou řadu.

EEG signál jako takový je samozřejmě analogový, tedy spojitý v čase, při měření je ovšem vzorkováním digitalizován analogově-digitálním převodníkem. Jak bylo popsáno v úvodní kapitole 2, dochází ke snímání amplitudy sumací aktivity neuronů. Snímání amplitudy je prováděno s určitou vzorkovací frekvencí f_s(Hz). Pokud provádíme měření po dobu T(s) dostáváme vzorkováním posloupnost amplitud počtu $N = f_s T$. Časová řada tedy není nic jiného než vektor hodnot (amplitud) x. Pro $x[n], n \in 0, \ldots, N$ pak hovoříme o amplitudě pro vzorek n, nebo také amplituda naměřená v čase $\frac{n}{f_s}$. Samozřejmě při měření EEG typicky dochází ke snímání signálu více než jednou elektrodou, dochází tedy ke snímání časové řady pro každou elektrodu. Ve výsledku tedy jedno měření můžeme reprezentovat reálnou maticí $\mathbb{R}^{CxN}$, kde C je počet použitých kanálů (elektrod).

Reálně se ovšem nepoužívá přímo vzorkovaný signál, ale nejprve dojde k jeho předzpracování, jako např. výběr elektrod, filtrování nebo odstranění artefaktů. Pro detekci MI je pro trénování klasifikátorů nutné společně s měřeným signálem mít i nějakou synchronizační značku. Pro analýzu nás pak budou zajímat pouze vybrané



Obrázek 4.1: Příklad vizualizace epochované časové řady v 1 sekundovém okolí okolo synchronizační značky pro různá frekvenční pásma. Čas 0 je čas výskytu synchronizační značky. Pásmo alfa představuje frekvenční pásmo 8-12Hz, pásmo beta 13-30Hz

úseky signálu v určitém časovém okolí okolo synchronizačních značek, těmto úsekům se říká *epochy*. Při měření typicky dochází k opakování několika MI, tzn. ze signálu získáme několik epoch nebo také pokusů (trials) a pak se hovoří o tzv. single-trials klasifikaci. Po získání všech epoch máme ve výsledku vlastně matici (v tomto případě tensor) $\mathbb{R}^{m \times c \times t}$, kde m je počet získaných epoch všech tříd MI, c je počet vybraných relevantních kanálů a t je počet amplitudových vzorků jednotlivých epoch (velikost vektoru), závislý na zvoleném časovém okolí. Příklad vizualizace časové řady jedné epochy viz obrázek 4.1 na str. 32.

4.2 Frekvenční spektrum

Kromě analýzy signálu v časové oblasti, je také možné signál analyzovat ve frekvenční oblasti. Jak bylo popsáno v kapitole 2.1 MI je spojováno s desynchronizací (ERD) a následnou synchronizací (ERS) frekvenčních pásem alfa a beta. Pro detekci pohybu z EEG signálu by se tedy mohl nabízet přístup pomocí analýzy jeho frekvenčního spektra. Hypotéza je tedy taková, že pro epochy spojené s nějakou třídou MI by měl být výkon alfa frekvencí výrazně *nižší* než v případě epoch ve kterých k MI nedošlo.

Pro převedení signálu z časové oblasti do oblasti frekvenční se používá matematická transformace nazývaná Fourierova transformace (FT) [Puk+17]. Jelikož

pracujeme se signálem, který byl diskretizovaný vzorkováním, jedná se konkrétně o transformaci nazývanou Diskrétní Fourierova transformace (DFT). Pro diskrétní signál s N vzorky lze DFT matematicky vyjádřit pomocí rovnice 4.1 [Mau15]

$$X[k] = \sum_{n=0}^{N-1} x[n] e^{-i2\pi nk/N} \quad k = 0, \ldots, N-1. \tag{4.1}$$

kde $X[k]$ je k-tý koeficient (komplexní číslo) DFT, $x[n]$ je n-tý vzorek vstupního signálu a $e^{-i2\pi nk/N}$ je n-tý vzorek komplexní bázové funkce (sinusoidy) s počtem k cyklů přes N vzorků. Je tedy provedena korelace vstupního signálu s bázovými funkcemi, kterou lze považovat za jakousi míru podobnosti mezi vstupním signálem a bázovou funkcí o dané frekvenci. Jaká frekvence koresponduje k-tému koeficientu je pak možno spočítat rovnicí 4.2

$$f = \frac{k}{N} f_s \tag{4.2}$$

kde f_s je vzorkovací frekvence vstupního signálu. Jelikož časová složitost výpočtu DFT je $O(N^2)$, reálně se pro výpočet DFT používá algoritmus Fast Fourier transform (FFT) jehož časová složitost je $O(N \log N)$ [Mau15].

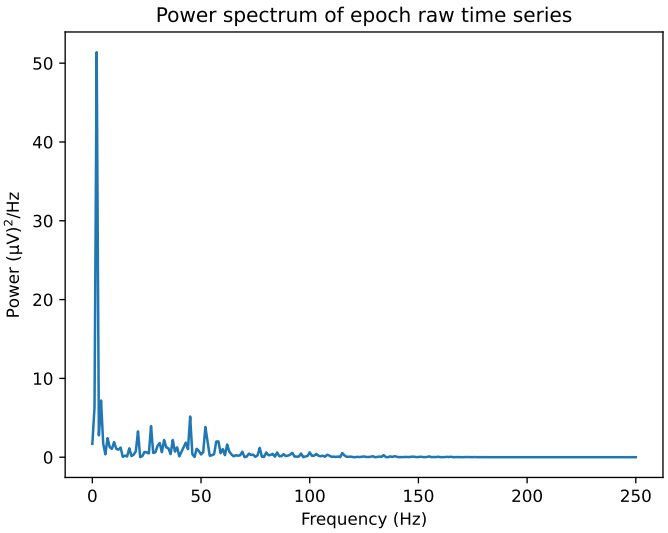
Komplexní čísla lze obecně vyjádřit také v goniometrickém tvaru $z = me^{i\varphi}$, kde m reprezentuje amplitudu koeficientu a φ jeho fázi. Hovoří se pak také o amplitudovém spektru a fázovém spektru. Nejčastěji se ovšem používá tzv. výkonové frekvenční spektrum PSD [Roy+19], kde výkon je definovaný jako druhá mocnina amplitudy. Pro komplexní koeficient ve tvaru $z = a + bi$ je možné amplitudu vypočítat jako $m = |z| = \sqrt{a^2 + b^2}$. Výkon k-tého koeficientu lze tedy vyčíslit rovnicí 4.3.

$$P[k] = X_a[k]^2 + X_b[k]^2 \tag{4.3}$$

Převedením epoch do frekvenčního výkonového spektra pak tedy dostáváme vstupní matici příznaků o rozměrech $\mathbb{R}^{m \times c \times f}$, kde m je počet epoch, c je počet vybraných kanálů a f je počet frekvenčních výkonů dané epochy. Příklad vizualizace výkonového frekvenčního spektra viz obrázek 4.2 na str. 34.

4.3 Časově-frekvenční spektrum

Extrakcí příznaků pouze z časové oblasti nebereme v potaz příznaky frekvenčního spektra. Obdobně extrakcí příznaků pouze z frekvenční oblasti ztrácíme informaci z časové oblasti. Z tohoto důvodu jsou někdy tyto charakteristiky považovány za slabou pro extrakci významných příznaků [ADA21]. V našem případě to pak znamená, že v časové oblasti známe zhruba časovou oblast při které došlo k MI na základě



Obrázek 4.2: Frekvenční spektrum výkonu nefiltrované epochy z obrázku. Epocha byla vzorkována vzorkovací frekvencí 500Hz, spektrum je zobrazeno do Nyquistovy frekvence4.1

synchronizační značky[1], ovšem neznáme frekvenční charakteristiku. Na druhou stranu při analýze pomocí frekvenčního spektra jsme teoreticky schopni detekovat pokles frekvenčního výkonu při MI, ovšem už nejsme schopni říci, kdy k tomuto poklesu došlo.

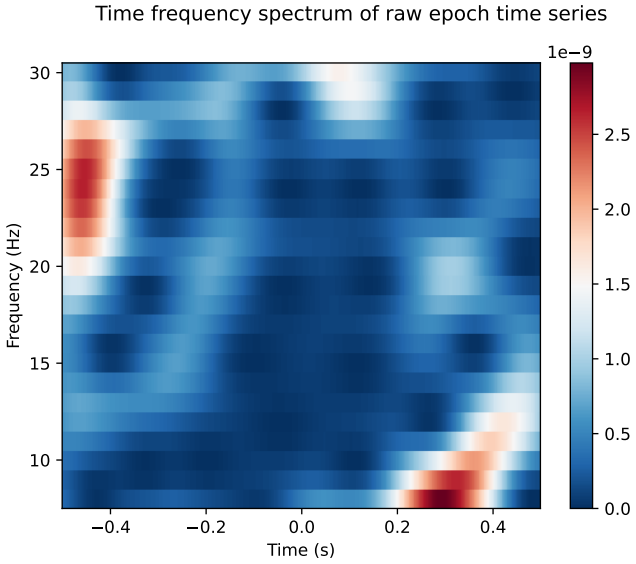
Existují metody, které převádějí vstupní signál do tzv. časově frekvenční oblasti a tím tak spojují informace jak z časové tak z frekvenční oblasti. Populární metody jsou především krátkodobá Fourierova transformace (STFT), waveletová transformace a Hilbertův filtr. Všechny tyto způsoby fungují na podobném principu a poskytují podobné výsledky. Princip výpočtu je podobný jako pro výpočet frekvenčního spektra. Hlavním rozdílem je, že frekvenční spektrum není počítáno pro celý signál, ale pouze pro jeho určitý výsek tzv. okno. Posunem okna po signálu jsme schopni spočítat frekvenční spektrum v určitém čase a tím tak sledovat změny frekvenčního spektra se změnami v čase [Coh19].

Výsledkem výpočtu je tzv. *spektrogram*, který lze získat výpočtem např. pomocí spojité waveletovy transformace v diskrétním čase (DT-CWT) pomocí rovnice 4.4.

$$C_{(a,b)} = \sum_{n=0}^{N-1} x[n]\psi^*\left(\frac{n-b}{a}\right) \tag{4.4}$$

kde $x[n]$ je diskrétní vstupní signál, ψ^* je komplexně sdružená funkce analyzující

[1] Synchronizační značka ovšem vzniká pouze při měření za účelem sběru trénovacích dat. Při online rehabilitaci by pak ke generování těchto značek nedocházelo.



Obrázek 4.3: Časově frekvenční spektrum výkonu nefiltrované epochy z obrázku 4.1, zobrazené pouze pro pásmo Alfa a Beta (8-30Hz). Čas 0 reprezentuje čas výskytu synchronizační značky

waveletovy funkce někdy také označovaná jako mateřský wavelet. a je škálující parametr waveletu, který ovlivňuje počet jeho cyklů a tedy jeho frekvenci což tedy umožňuje wavelet tzv. „rozšiřovat“ a „zužovat“. Parametr b umožňuje wavelet posouvat v čase, neboli umožňuje provádět analýzu frekvenčního spektra v různých časových lokacích signálu. $C_{(a,b)}$ pak reprezentuje koeficienty pro škálu a a posun b.

Co je poměrně důležité, je uvědomit si, že při převedení signálu do časově frekvenční oblasti provádíme jakýsi kompromis mezi rozlišením v obou spektrech. Čím nižší je hodnota parametru a tím vyšší je počet cyklů a tím tedy frekvence waveletu, což vede k vyššímu frekvenčnímu rozlišení. Naopak pro vyšší hodnoty parametru a vede k vyššímu rozlišení v časové oblasti [Coh19]. Zvýšením rozlišení ve frekvenční oblasti dojde ke snížení rozlišení v časové oblasti a naopak. Existuje také celá řada wavelet funkcí, které je pro výpočet waveletové transformace možné použít.

Výsledný spektrogram je tedy vlastně matice, kde hodnota matice v bodě $[f, t]$ reprezentuje spektrální výkon frekvence f v čase t. Převedením všech epoch do časově frekvenční oblasti bychom pak pracovali s maticí příznaků $\mathbb{R}^{m \times c \times f \times t}$. Příklad vizualizace výkonově časově frekvenčního spektra viz obrázek 4.3 na str. 35.

V některých studiích byla také provedena klasifikace nad příznaky vytvořenými převodem vypočítané časově frekvenčního spektrogramu na rastrový obrázek. V tomto případě by pak bylo nutné provést mapování výkonu frekvenčního spektra do barevného prostoru např. RGB obrázek reprezentovaný maticí použít jako vstupní příznakové vektory pro učení klasifikátoru.

Klasifikace 5

Cílem klasifikace je najít rozhodovací hranici (nadplochu, obecně libovolného tvaru), která nejlépe rozdělí prostor příznakových vektorů na podprostory, jeden pro každou třídu. Klasifikátor pak klasifikuje všechny příznakové vektory na jedné straně rozhodovací hranice do třídy daného podprostoru. Vstupním příznakům, u kterých je předem známá třída příslušnosti, se říká trénovací množina.

Trénovací množinu v případě učení s učitelem tvoří následující dvojice: příznakové vektory tzn. matice $\mathbf{X}^{(m \times n)}$, kde m je počet příznakových vektorů a n je počet příznaků jednotlivých vektorů, a vektor příslušnosti příznakových vektorů k třídám $\mathbf{y}^m$. Kde příznakovému vektoru $\mathbf{x}^{(i)}$ přísluší třída $\mathbf{y}^{(i)}$ (každému z m příznakových vektorů byla přidělena jedna z K tříd) [Bis06]. V kontextu této práce pak jeden datový vzorek $\mathbf{x}^{(i)}$ reprezentuje jedna naměřená epocha (určená různými příznaky, viz předchozí kapitola) a $\mathbf{y}^{(i)} \in \{1, \ldots K\}$ jednu z K tříd, kde třídy reprezentují jednotlivé typy prováděných MI, případně klidový stav. Běžně jsou jednotlivé třídy transformovány jako tzv. one hot vektory, kde velikost vektoru je K. Hodnoty one hot vektoru jsou nulové všude, kromě hodnoty indexu reprezentující danou třídu, kde je hodnota rovna 1.

Učením klasifikátoru pak rozumíme proces, při kterém jsou hledány parametry (váhy, nastavení) klasifikátoru tak, aby výsledná rozhodovací hranice byla nejlepší. Výsledkem klasifikace vstupního příznakového vektoru jsou pravděpodobnosti příslušnosti vektoru k jednotlivým třídám. Přičemž typicky je pak příznakovému vektoru přiřazena třída s nejvyšší pravděpodobností (ta samozřejmě nemusí být správná a pak hovoříme o tzv. nesprávné klasifikaci). Na základě klasifikace jsme pak schopni určit, zda v nově naměřené epoše došlo k MI či nikoliv a případně rovnou k jakému typu MI.

V oblasti zpracování EEG signálu je velký problém s dimenzionalitou vstupních příznakových vektorů. Množství dat potřebných ke správné klasifikaci různých tříd roste exponenciálně s dimenzionalitou. Pokud je počet trénovacích dat ve srovnání s velikostí příznakových vektorů malý, bude klasifikátor pravděpodobně poskytovat špatné výsledky. Doporučuje se použít alespoň pětkrát až desetkrát větší počet vstupních vektorů na třídu, než je jejich dimenzionalita. Bohužel tento požadavek

je problematický, protože obvykle je počet vstupních dat malý a dimenzionalita vysoká [Vař18].

V následujících sekcích budou popsány vybrané klasifikátory. Pro jednoduchost budou jednotlivé příklady předpokládat binární klasifikaci, tzn. počet tříd $K = 2$.

5.1 Lineární diskriminační analýza

Linear discriminant analysis (LDA) je populární lineární klasifikátor (tzn. rozhodovací hranice má lineární tvar). Typicky společně s SVM a logistickou regresí používaný jako základní klasifikátor, jehož výsledky je možné používat pro porovnávání komplexnějších modelů [Pad+19]. Jedním z předpokladů pro použití LDA je, že vstupní data jednotlivých tříd pocházejí z normálního rozdělení. V ideálním případě by samozřejmě také měly být třídy lineárně separabilní.

Cílem LDA je dosáhnout největší separace mezi jednotlivými třídami maximalizací rozptylu mezi třídami a zároveň minimalizací rozptylu uvnitř třídy, čímž dojde ke snížení rozptylu jednotlivých tříd. Mezitřídní kovarianční matice je dána vztahem

$$\mathbf{S_B} = (\mathbf{m_1} - \mathbf{m_0})(\mathbf{m_1} - \mathbf{m_0})^T \tag{5.1}$$

kde $\mathbf{m_1}$ je vektor středních hodnot vstupních dat, který přísluší třídě 1 a $\mathbf{m_0}$ vektor středních hodnot třídy 0. Kovarianční matice v rámci třídy

$$\mathbf{S_W} = \sum_{i:y_i=1} (\mathbf{x_i} - \mathbf{m_1})(\mathbf{x_i} - \mathbf{m_1})^T + \sum_{i:y_i=0} (\mathbf{x_i} - \mathbf{m_0})(\mathbf{x_i} - \mathbf{m_0})^T \tag{5.2}$$

Proces učení pak probíhá optimalizací cenové funkce

$$J(\mathbf{w}) = \frac{\mathbf{w}^T \mathbf{S_B} \mathbf{w}}{\mathbf{w}^T \mathbf{S_W} \mathbf{w}} \tag{5.3}$$

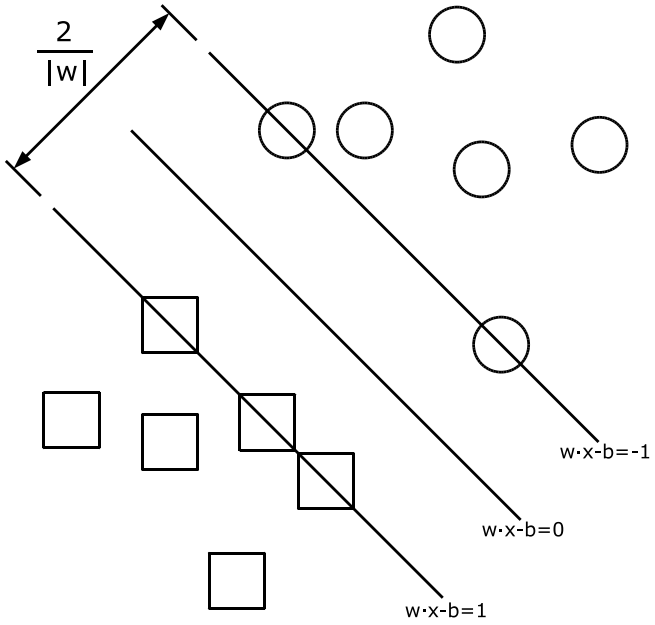
kde $\mathbf{w}$ je vektor vah (nastavení) klasifikátoru.

Nastavením prahu τ pak můžeme příznakové vektory klasifikovat do třídy 1 pokud lineární kombinace vektoru vah a příznakového vektoru je větší nebo roven τ, v opačném případě je příznakový vektor klasifikován třídou 0 [Bis06]. Matematicky zapsáno jako

$$\begin{cases} \hat{y} = 1, & \mathbf{w}^T \mathbf{x} \geq \tau \\ \hat{y} = 0, & \mathbf{w}^T \mathbf{x} < \tau \end{cases} \tag{5.4}$$

5.2 SVM

Support vector machine (SVM) je podobně jako LDA také lineární klasifikátor, který je v oblasti strojového učení velmi populární [Pad+19]. Neexistuje jediná nadrovina,



Obrázek 5.1: Ukázka nalezení optimální lineární separace pomocí SVM. Zdroj: `https://commons.wikimedia.org/wiki/File:SVM_margins.svg`

která by rozdělovala prostor příznakových vektorů na podprostory jednotlivých tříd. Cílem SVM je nalézt takovou nadrovinu, která maximalizuje tzv. okraje, neboli vzdálenost od nejbližších projekcí příznakových vektorů, viz obrázek 5.1 na str. 39.

Rozhodovací hranice lineárního klasifikátoru má obecně předpis

$$h(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + b \tag{5.5}$$

kde $\mathbf{w}$ je vektor vah klasifikátoru, $\mathbf{x}$ klasifikovaný příznakový vektor a b (někdy také w_0) je bias, neboli parametr, který umožňuje libovolný posun nadroviny. Pro body p a q ležící na rozhodovací hranici pak platí vztah

$$\mathbf{w}(p - q) = 0 \tag{5.6}$$

neboli platí, že normála rozhodovací hranice je vektor $\mathbf{w}$. Vektor $\mathbf{w}$ jsme schopni naškálovat tak, že hodnota $h(\mathbf{x_0})$ rozhodovací hranice v nejbližším bodě třídy 0 je rovna 1 a hodnota $h(\mathbf{x_1})$ v nejbližším bodě třídy 1 je rovna -1. Dalším odvozením využitím analytické geometrie se dá ukázat, že šířka okraje je rovna $\frac{2}{\|\mathbf{w}\|}$. Cílem klasifikátoru je tedy tuto šířku maximalizovat. Proces učení SVM tedy probíhá mi-

nimalizací cenové funkce[1] [Bis06]

$$J(\mathbf{w}) = \frac{1}{2}\|\mathbf{w}\|^2 \tag{5.7}$$

Klasifikaci příznakových vektorů pak provádíme na základě

$$\begin{cases} \hat{y} = 0, & \mathbf{w}^T\mathbf{x} + b \geq 1 \\ \hat{y} = 1, & \mathbf{w}^T\mathbf{x} + b \leq -1 \end{cases} \tag{5.8}$$

Obecně lze SVM použít i pro nelineární separaci použitím nelineární transformace vstupních příznaků bázovou funkcí (tzv. kernel, jádro). Velmi často používaným jádrem je Gaussova funkce. Myšlenka je taková, že pokud nejsou vstupní příznakové vektory v příznakovém prostoru lineárně separabilní, aplikace kernelu provede projekci příznaků do prostoru vyšší dimenze, ve které už příznaky lineárně separabilní jsou.

Kompletní popis techniky SVM, která také dovoluje nesprávnou klasifikaci viz kapitola 7.1 v literatuře [Bis06].

5.3 Vícevrstvý perceptron

Multilayer perceptron (MLP) je klasifikátor modelovaný jako vícevrstvá neuronová síť. Jedná se tedy o klasifikátor složený z perceptronů (neuronů) organizovaných do vrstev (2 a více). Neuronové sítě a jejich různé architektury jsou dnes velice populárním klasifikátorem obecně v oblasti strojového učení, ale jak bylo naznačeno v kapitole 3 jsou používané i v oblasti zpracování EEG signálu [Pad+19]. Obecně nevýhodou neuronových sítí například oproti SVM je fakt, že neuronové sítě potřebují pro natrénování mnohem větší trénovací množinu.

Jeden neuron (perceptron) neuronové sítě je modelován podle skutečného neuronu centrální nervové soustavy. Vstupem neuronu jsou jednotlivé příznaky vstupního příznakového vektoru tedy $\mathbf{x^{(i)}} = (\mathbf{x_1}, \ldots, \mathbf{x_n})$, každý vstupní příznak je váhován příslušnou váhou pro daný příznak, tzn. každý vstupní příznak je přenásoben odpovídající váhou z vektoru $\mathbf{w} = (w_1, \ldots, w_n)$, k výsledku násobení je ještě přičten práh w_0, někdy také značen jako bias b. Výsledky násobení jsou následně agregovány (nejčastěji sumou) a tento výsledek je pak vstupem tzv. nelineární *aktivační funkce*. Aktivace neuronu je výstupem aktivační funkce. Matematicky lze zapsat jako

$$z = f\left(\sum_{j=1}^{n} w_j x_j + w_0\right) \tag{5.9}$$

[1] Toto je za předpokladu, že v oblasti okraje neleží žádný příznakový vektor, tzn. takový klasifikátor nepřipouští nesprávnou klasifikaci.

přičemž se používají různé typy aktivačních funkcí. Nejčastěji používanou funkcí je funkce sigmoid

$$f(x) = \frac{1}{1 + e^{(-x)}} \tag{5.10}$$

další často používanou funkcí je ReLU, která je částečně lineární

$$f(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{5.11}$$

ale používají se i jiné. Perceptron používající aktivační funkci sigmoid je pak identický s klasifikátorem logistické regrese a vytváří tedy lineární rozhodovací hranici.

U vícevrstvého perceptronu je pak kombinováno více neuronů do vrstev, viz obrázek 5.2 na str. 42, jejichž spojením pak vzniká síť. Každý neuron tvoří lineární rozhodovací hranici (nadplochu) v n-dimenzionálním prostoru příznaků. Kombinací těchto nadploch pak vznikají složitější útvary, takže je MLP schopen provádět klasifikaci i tříd, které nejsou lineárně separabilní.

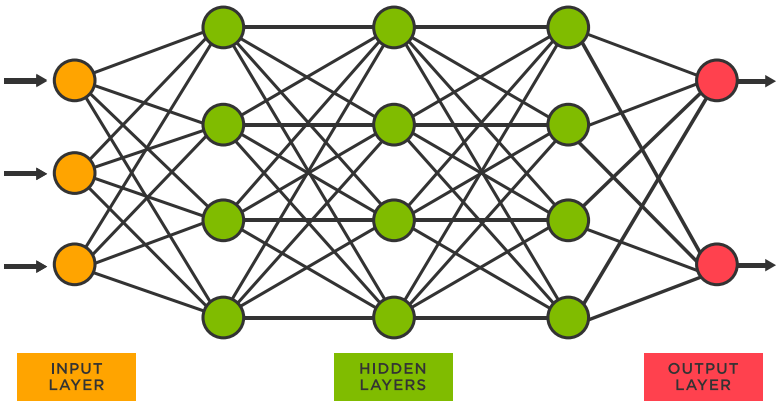
MLP je vždy tvořen jednou vstupní vrstvou, libovolným počtem tzv. skrytých vrstev a jednou výstupní vrstvou. Neurony jsou ve vrstvách organizovány tak, že vstupem neuronu L_i vrstvy jsou aktivace všech neuronů L_{i-1} vrstvy. Rozdílem je samozřejmě vstupní vrstva, kde vstupem neuronů jsou hodnoty jednotlivých příznaků vstupního příznakového vektoru.

Počet neuronů vstupní vrstvy tedy odpovídá počtu příznaků příznakových vektorů, počet neuronů skrytých vrstev je možné volit libovolně a je to tedy společně s počtem skrytých vrstev a výběrem aktivační funkce jedním z hyperparametrů, volitelných při konstrukci konkrétního modelu. Počet neuronů výstupní vrstvy odpovídá počtu klasifikačních tříd (K) vstupních příznakových vektorů. Vstupní příznakový vektor je klasifikován do třídy, která odpovídá výstupnímu neuronu s největší aktivací. Často se také jako aktivační funkce výstupní vrstvy používá funkce softmax

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{5.12}$$

která normalizuje aktivace výstupních neuronů tak, že platí $\sum_{i=1}^{K} z_i = 1$. Aktivaci neuronů výstupní vrstvy pak můžeme interpretovat jako pravděpodobnost příslušnosti příznakového vektoru do dané třídy. Přičemž samozřejmě typicky pak vektor klasifikujeme do třídy největší s pravděpodobností.

Učení klasifikátoru probíhá ve dvou krocích, tzv. dopředné a zpětné šíření (feed forward a backpropagation). Při dopředném šíření je na základě vstupního příznakového vektoru postupně vypočítána aktivace všech neuronů na základě aktuálního nastavení vah $\mathbf{w}$. Při zpětném šíření pak gradientním sestupem chybové funkce klasifikátoru probíhá výpočet příspěvku chyby neuronů v jednotlivých vrstvách.



Obrázek 5.2: Ukázka vícevrstvého perceptronu. Zdroj: `https://www.tibco.com/reference-center/what-is-a-neural-network`

Nejčastěji používaná chybová funkce se nazývá cross entropy, která pro binární klasifikaci nabývá tvaru

$$J(\mathbf{w}) = -\frac{1}{m}\sum_{i=1}^{m}\sum_{k=1}^{K} y_k^{(i)} \log(z_k(\mathbf{x}^{(i)}, \mathbf{w})) + (1 - y_k^{(i)}) \log(1 - z_k(\mathbf{x}^{(i)}, \mathbf{w})) \quad (5.13)$$

kde $y_k^{(i)}$ je k-tá složka one-hot vektoru třídy pro i-tý vstupní příznakový vektor a $z_k(x^{(i)}, \mathbf{w})$ je aktivace k-tého výstupního neuronu v závislosti na i-tém vstupním příznakovém vektoru $\mathbf{x}^{(i)}$ a aktuálním nastavení vah klasifikátoru $\mathbf{w}$.

Pro kompletní popis propagace chyby viz kapitola 5.3 literatury [Bis06].

5.4 Neuronová síť s dlouhodobou a krátkodobou pamětí

Long short-term memory (LSTM) je architektura rekurentních neuronových sítí. Rekurentní neuronové sítě se od běžných neuronových sítí, popsaných v předchozí sekci, liší tím, že na rozdíl od MLP může být výstup neuronu vrstvy L_i vstupem neuronu stejné vrstvy nebo vrstvy předchozí. Takové sítě mnohem lépe modelují chování mozkových sítí, jelikož jsou schopné si udržet jakýsi kontext jednotlivých příznaků. Jelikož v této práci bude prováděna detekce MI, mohl by být tento aparát vhodný pro klasifikaci, jelikož by se síť měla být schopná naučit časové závislosti a využít tak sekvenční povahy naměřeného signálu.

Klasické rekurentní neuronové sítě ovšem ve fázi učení trpí problémy tzv. mizejícího a explodujícího gradientu. Pokud gradient nabývá při zpětném šíření hodnot nižších než 1, dochází k násobení stále nižších a nižších hodnot konvergujících k 0. Naopak pokud je hodnota gradientu vyšší než 1 dochází k násobení exponenciálně

výších čísel konvergujících k nekonečnu. Díky těmto problémům jsou schopny rekurentní sítě uchovávat pouze velmi malý kontext tzv. krátkodobou paměť (short-term memory) [Ola15].

Architektura sítě LSTM tyto problémy řeší zavedením tzv. dlouhodobé paměti. Neurony LSTM sítí mají o něco komplikovanější strukturu, než jaká byla popsána v předchozí sekci 5.3, a nazývají se buňky (cells), viz obrázek 5.3 na str. 44. Buňky uchovávají dlouhodobou paměť pomocí tzv. *cell state*, na obrázku 5.3 proměnná $\mathbf{c_t}$. Krátkodobá paměť je podobně jako u rekurentních neuronových sítí uložena v tzv. *hidden state*, na obrázku 5.3 proměnná $\mathbf{h_t}$. LSTM buňka je rozdělena do 3 částí (bran, gates): forget gate, input gate a output gate [Ola15].

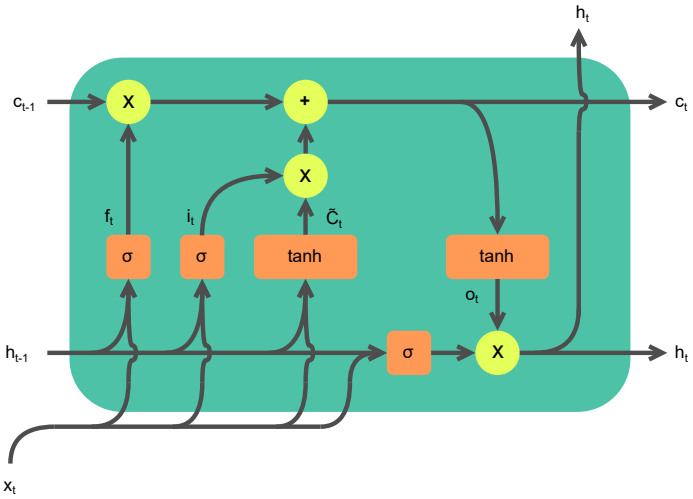
- `Forget gate` ($\mathbf{f_t}$) – brána zapomínání. Tato brána rozhoduje, kolik informace z předchozích buněk bude uchováno. Výstup předchozí buňky a vstup aktuální buňky jsou vstupem do aktivační funkce sigmoid, jejímž výsledkem je hodnota v rozsahu 0-1. Pokud hodnota $\mathbf{f_t}$ nabývá 0, pak dochází k zapomenutí veškeré informace z předchozích buněk ($\mathbf{c_{t-1}}$). Naopak pokud hodnota nabývá hodnoty 1, je zachována veškerá relevantní informace [Ola15].

- `Input gate` ($\mathbf{i_t}$) – vstupní brána. Tato brána určuje, jak relevantní je aktuální vstup $\mathbf{x_t}$ na základě krátkodobé paměti předchozích buněk $\mathbf{h_{t-1}}$. Následně jsou vypočteny nové kandidátní hodnoty $\widetilde{\mathbf{C}}_\mathbf{t}$ pro dlouhodobou paměť. Tyto hodnoty jsou na základě relevantnosti aktuálního vstupu přidané a uložené jako aktuální dlouhodobá paměť buňky [Ola15].

- `Output gate` ($\mathbf{o_t}$) – výstupní brána. Tato brána provede pronásobení dlouhodobé paměti aktivovanou funkcí tanh a aktuálního vstupu aktivovaného funkcí sigmoid, pro výpočet krátkodobé paměti buňky. Výsledná dlouhodobá a krátkodobá paměť slouží jako vstup do následující buňky ve stejné vrstvě. Zároveň je krátkodobá paměť použita jako výstup buňky sloužící jako vstup následující vrstvy [Ola15].

5.5 Konvoluční neuronová síť

Convolutional neural network (CNN) je architekturou hlubokých umělých neuronových sítí, které se od klasických plně propojených sítí (MLP) liší tím, že na začátku sítě jsou ještě tzv. konvoluční vrstvy. CNN zaznamenaly největší využití v oblasti strojového vidění, neboli zpracování obrázků. Ovšem jak bylo popsáno v kapitole 3, začala být tato technika také populární mimojiné v oblasti zpracování EEG signálu. Bylo již provedeno několik studií, ve kterých bylo zjištěno, že hluboké učení překonává výsledky state-of-the-art technik jako například použití extrakce příznaků pomocí CSP a klasifikátoru SVM [Pad+19].



Obrázek 5.3: Příklad jedné LSTM buňky. Aktuální vstup x_t, krátkodobá paměť předchozí buňky h_{t-1}, dlouhodobá paměť předchozí buňky c_{t-1}, forget gate f_t, input gate i_t, kandidátní hodnoty na dlouhodobou paměť $\widetilde{C}_t$, output gate o_t, dlouhodobá paměť buňky c_t a krátkodobá paměť buňky h_t. Zdroj: `https://commons.wikimedia.org/wiki/File:LSTM_cell.svg`

Hlavní výhodou použití CNN pro klasifikaci EEG dat je automatická extrakce příznaků. Tzn. nezpracovaná naměřená data je možné použít jako vstupní příznakové vektory, a tím tak odpadá nutnost vlastní tvorby příznakových vektorů, která může být někdy poměrně komplikovaná a časově náročná. Obecně CNN fungují dobře při použití velkých datových sad. To je ovšem v oblasti zpracování EEG problematické. Jak již bylo popsáno v úvodu této kapitoly, naproti tomu například technika SVM je schopna poskytovat poměrně dobré výsledky i v případě, že počet příznakových vektorů vstupní datové sady je malý. Zároveň sítě obsahují poměrně velké množství hyperparametrů, které se při učení musí naučit, což může vést k delší době trénování sítě oproti ostatním technikám. Interpretace naučených příznaků pomocí konvolučních vrstev může těžko pochopitelná v kontextu původního vstupního signálu [Pad+19; Roy+19].

Typická struktura CNN se skládá z následujících typů skrytých vrstev: konvoluční vrstva, pooling vrstva a plně propojená vrstva. Sekvenčním propojením těchto vrstev pak vzniká celá architektura CNN [ON15]. Při následujícím popisu vrstev bude předpokládat 3D vstupní data, která jsou definována svojí výškou, šířkou a hloubkou. Typickým příkladem takových vstupních dat je obrázek s definovanou výškou a šířkou a hloubkou definovanou jako složky RBG [ON15]. V kontextu práce by se tedy jednalo o vstupní data reprezentovaná časově frekvenčním spektrogramem, kde výška spektrogramu jsou jednotlivé frekvence, šířka by pak byla reprezentovaná časem spektrogramu a hloubka počtem měřených elektrod.

Konvoluční vrstva je hlavním typem vrstvy, která celou architekturu definuje. Tato vrstva je zaměřená na nalezení optimálních parametrů tzv. masek (kernelů, filtrů). Masky jsou reprezentovány maticemi, které mají typicky malé prostorové rozměry. Na základě nastavení vah masek dochází ve vstupních datech k nalezení lokálních příznaků. V konvoluční vrstvě pak dochází ke konvoluci mezi vstupními daty a jednotlivými maskami. Konvoluční vrstva tedy provádí konvoluci mezi jednotlivými filtry přes prostorové dimenze vstupních dat, čímž dojde k vytvoření tzv. 2D aktivačních příznakových map [ON15].

Matematicky je možno konvoluční vrstvu vyjádřit jako

$$y[m,n] = f\left(\sum_{j=0}^{J-1}\sum_{i=0}^{I-1} x[m+i,n+j]w[i,j] + b\right) \tag{5.14}$$

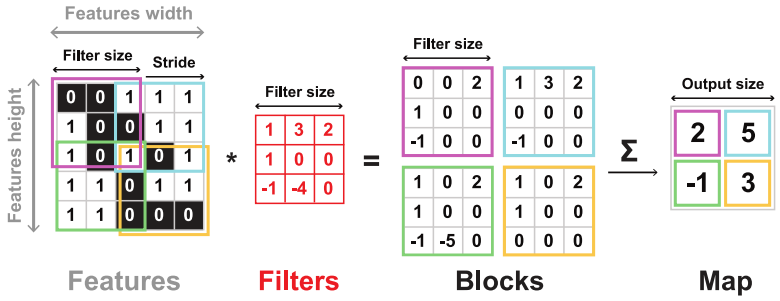
kde $\mathbf{x}$ je vstupní dvoudimenzionální příznakový vektor, $\mathbf{y}$ je výstupní příznaková mapa konvoluční vrstvy o výsledných rozměrech $M \times N$, $\mathbf{w}$ jsou váhy konvoluční masky o rozměrech $J \times I$, b je bias a f je aktivační funkce vrstvy. Vizuální příklad konvoluce dvou rozměrných vstupních dat viz obrázek 5.4 na str. 46. Samozřejmě operace konvoluce není definována pouze pro dvourozměrná data, konec konců rovnice waveletovy transformace 4.4 je vlastně konvoluce jednorozměrného vstupního signálu s různými wavelety.

Počet výstupních příznakových map odpovídá počtu zvolených filtrů, což je jedním z hyperparametrů volitelným při konstrukci modelu. Dalším nastavitelným hyperparametrem je krok, s jakým je filtr v dané dimenzi posunut. Například nastavením kroku na 1 bychom dostali silně překrývající aktivace. Na druhou stranu větší krok pak snižuje míru překrývání a tím tak redukuje rozlišení příznaků v dané dimenzi. Zároveň také čím větší krok, tím nižší je prostorový rozměr výsledné aktivační příznakové mapy. Při konvoluci je samozřejmě nutné, aby filtr plně překrýval vstupní data, což může být problém například při zachycení příznaků na kraji vstupních dat. Tento problém se řeší pomocí tzv. zero-padding, kdy kraje vstupních dat jsou rozšířené nulami (tedy jsou k vstupním datům přidány sloupce a řádky, jejichž hodnoty jsou nulové) tak, aby bylo možné filtr aplikovat například i na poslední sloupec vstupních dat. Na základě volby těchto parametrů pak dochází k produkci aktivačních příznakových map, jejichž prostorový rozměr je definován jako

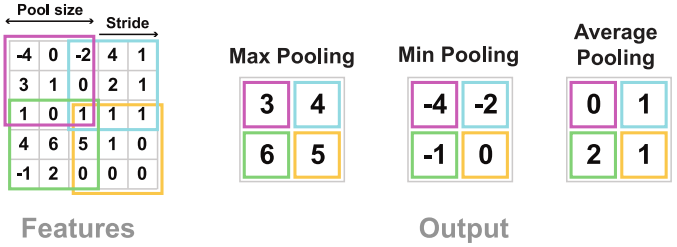
$$\frac{W - F + 2P}{S + 1} \tag{5.15}$$

kde W je velikost prostorové dimenze vstupních dat (tedy výška nebo šířka), F je odpovídající velikost prostorové dimenze filtru, P je velikost zero-paddingu v dané dimenzi a S je velikost kroku [ON15].

Pooling vrstva funguje velice podobným způsobem jako konvoluční vrstva. Záměrem pooling vrstvy je provedení podvzorkování tedy redukci dimenzionality a



Obrázek 5.4: Příklad operace konvoluce ve 2D. Zdroj: `https://epynn.net/Convolution.html`



Obrázek 5.5: Příklad pooling operace ve 2D. Zdroj: `https://epynn.net/Pooling.html`

tím tak redukci počtu parametrů a výpočetní komplexity modelu. Vstupem pooling vrstvy jsou aktivační příznakové mapy. Vrstva provádí stejnou operaci jako konvoluční vrstva, ovšem s tím rozdílem, že jednotlivé masky nemají váhy (váhy jsou nastaveny na 1) a místo operace sumy je používaná nějaká forma agregace. Typicky používané agregace jsou: průměr, maximum nebo minimum [ON15], výsledky jednotlivých operací viz obrázek 5.5 na str. 46.

Augmentace 6

Jak bylo zmíněno v předchozích kapitolách, klasifikace pomocí hlubokého učení v oblasti zpracování EEG signálů nabývá na stále větší popularitě. Ovšem podmínkou pro získání očekávaných výsledků je mít rozsáhlou trénovací množinu, která by měla zajistit větší robustnost a schopnost generalizace klasifikátorů založených na hlubokém učení [He+21; Igl+23].

Získávání rozsáhlé datové sady pomocí měření signálu je časově náročná záležitost. Ve výsledku tento problém vede k tomu, že při aplikaci BCI systému dochází k tzv. overfitting (přeučení, tzn. dochází k tvorbě příliš komplexní rozhodovací hranice) a tak slabé generalizaci naučeného klasifikátoru. Slibným řešením tohoto problému je regularizace, která může zlepšit schopnost generalizace a robustnost klasifikátorů. Existují tři způsoby regularizace: tzv. L2 regularizace, při které dochází k přidání penalizace za příliš komplikovanou rozhodovací hranici přímo do cenové funkce klasifikátoru. Další způsob je použití regularizace přímo v navrženém modelu pomocí např. techniky dropout nebo batch normalization (blíže popsáno v kapitole 7.4.2.3). Poslední slibnou metodou je použití augmentace datové sady (DA) [He+21].

Ve srovnání s prvními dvěma přístupy řeší DA problém přeučení použitím komplexnější sady dat, aby se minimalizovala vzdálenost mezi trénovací a testovací datovou sadou. To je užitečné zejména pro signály EEG, kde omezení malých datových sad výrazně ovlivňuje výkon klasifikátorů. DA je technika, která rozšiřuje trénovací sadu vstupních dat o nově vytvořené umělé vzorky. Existují dva přístupy pro vytváření nových datových vzorků. Prvním přístupem je provádění často jednoduchých změn (manipulací) naměřených příznakových vektorů, čímž dojde k augmentaci dat přímo. Druhým způsobem je použití generativních modelů, jejichž cílem je naučit se distribuci vstupních příznakových vektorů. Vytvoření nových vzorků pak probíhá výběrem z naučené distribuce [He+21]. Jedním problémem DA je, že u určitých datových sad může člověk poměrně jednoduše rozhodnout, zda nový augmentovaný datový vzorek stále připomíná původní třídu (např. u obrázku vizuální inspekcí), v oblasti zpracování EEG signálu to ovšem tak přímočaré být nemusí [LLM20].

Metody DA lze použít pro rozšíření celkové vstupní datové sady, nebo je možné

metody použít pouze pro vygenerování nějaké určité třídy vstupních příznakových vektorů. V některých oblastech může dojít k naměření mnohem více příznakových vektorů jedné třídy na rozdíl od jiné třídy například z důvodu, že měřená třída může být vzácná (například výskyt nějaké vzácné nemoci, pro které neexistuje dostatek dat) [LLM20]. Pomocí DA metod je pak možné nepoměr naměřených dat mezi jednotlivými třídami vybalancovat.

6.1 Manipulace příznaků

Metody založené na manipulaci příznaků trénovacích dat lze rozdělit na dva přístupy. První přístup modifikuje vstupní příznakové vektory aplikací nějaké geometrické transformace, například translace, rotace, zrcadlení atd. viz dále. Druhý přístup je přidání šumu do stávajících trénovacích dat [LLM20]. Rozšíření datové sady tedy funguje tak, že je vybrán trénovací příznakový vektor a aplikací nějaké transformace je z tohoto vektoru vytvořen nový augmentovaný příznakový vektor, který je přidán do trénovací množiny. Vizuální interpretace některých dále popsaných metod viz obrázek 6.1 na str. 49.

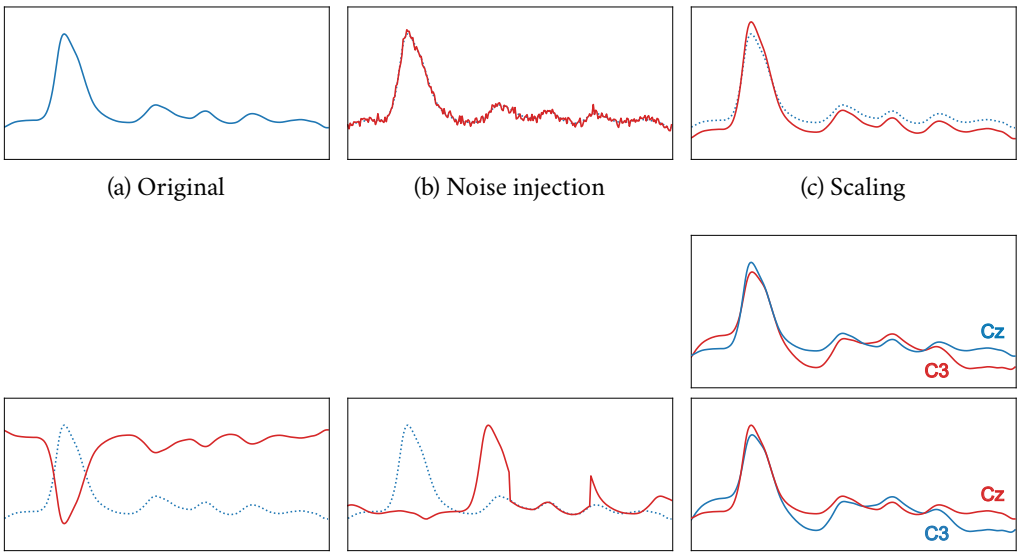
Výhodou těchto algoritmů je především jejich jednoduchost a poměrně malý počet konfigurovatelných hyperparametrů, takže jejich nastavení a implementace je proveditelná za mnohem kratší dobu než v případě generativních modelů. Oproti generativním modelům také potřebují mnohem menší množství dat, jelikož nedochází k žádnému učení. Na druhou stranu oproti generativním modelům jsou poměrně omezené, jelikož dochází pouze k modifikaci již existující datové sady, takže variabilita nově vzniklých vzorků může být poměrně malá. Kvalita generovaných vzorků je většinou také nižší než v případě použití generativních modelů, zároveň ne každou metodu lze použít pro jakoukoliv doménu, jelikož mohou vznikat neplatné vzorky [Igl+23].

6.1.1 Transformace

V kontextu jednotlivých transformací je nutné si uvědomit, že ne všechny metody jsou vhodné pro zpracování EEG signálu, protože mohou narušit časovou doménu signálu, a tím tak vytvářet nevalidní datové vzorky [LLM20].

6.1.1.1 Posuvné okno

Metoda sliding window (posuvné okno), nebo také time slicing (časové řezy) již byla popsána v kapitole 3.1.5.2 na str. 21. Rozšíření datové sady touto metodou dochází tedy tím, že originální vstupní signál je rozložen na kratší časové úseky v závislosti na velikosti řezu (doba trvání okna) [LLM20; Igl+23]. Například použijeme-li délky 50ms na dvou sekundový signál (epochu), dostaneme z jednoho naměřeného



(a) Original (b) Noise injection (c) Scaling

(d) Flipping (e) Permutation (f) Channel permutation

Obrázek 6.1: Ukázka augmentace dat využitím metod založených na manipulaci příznaků v časové doméně signálu. Modrou barvou je vizualizován originální signál, červenou pak nově vzniklý augmentovaný signál. Zdroj: `https://arxiv.org/abs/2004.08780`

datového vzorku 40 nových datových vzorků, které budou použity jako vstupní příznakové vektory klasifikátorů. Jak již bylo v kapitole zmiňováno, jednotlivá okna se mohou překrývat, takže např. použitím překryvu o velikosti 25ms by z jedné epochy vzniklo 80 nových úseků.

Touto metodou může dojít k vytváření neplatných vzorků, jelikož jednotlivé řezy nemusí obsahovat důležité vlastnosti obsažené v kompletní epoše. Je tedy otázkou zda jsou jednotlivé řezy dostatečně reprezentativní pro danou třídu datových vzorků [Igl+23]. Výsledky metody jsou samozřejmě také velice závislé na volbě velikosti okna a velikosti překryvu [LLM20].

6.1.1.2 Škálování

Metoda škálování spočívá ve změně velikosti jednotlivých příznaků vstupních příznakových vektorů. Hlavním cílem této metody je zachovat tvar příznakového vektoru a pouze změnit (naškálovat) jeho hodnoty. Tzn. například pro signál reprezentovaný časovou řadou dojde pouze ke změně naměřených amplitud, ale bude stále zachovaný tvar naměřeného signálu [Igl+23].

$$\hat{\mathbf{x}}(\alpha) = \{\alpha x_0, \ldots, \alpha x_{N-1}\} \tag{6.1}$$

Existují různé implementace škálování. Nejjednodušší metoda je přenásobení všech příznaků příznakového vektoru stejnou konstantní hodnotou viz rovnice 6.1, kde $x_0, \ldots, x_{N-1}$ jsou příznaky příznakového vektoru délky N, α je škálovací konstanta a $\hat{\mathbf{x}}(\alpha)$ nově vzniklý příznakový vektor. Ovšem existují i další techniky, podrobněji viz kapitola 6.1.3 v [Igl+23].

6.1.1.3 Geometrické transformace

Následující transformace se používají především v oblasti zpracování obrázků. Jednotlivé metody lze však implementovat i pro signály, ovšem je otázkou, zda by vzniklé augmentované vzorky mohly vzniknout fyzikálním měřením, a tudíž zda opravdu reprezentují danou třídu.

Zrcadlení. Zrcadlení je jednoduchá metoda, při které nový augmentovaný vzorek vznikne překlopením datového vzorku okolo vertikální nebo horizontální osy [He+21].

Rotace. Rotace je metoda, kdy je vstupní příznakový vektor přenásoben rotační maticí s předem definovaným úhlem [Igl+23; He+21].

Ořezávání. Ořezávání je metoda, při které nový augmentovaný vzorek vznikne výběrem náhodné části signálu určité délky [He+21].

6.1.1.4 Permutace

Permutace je metoda, při které jsou nadefinovány časové výřezy signálu (podobně jako v případě metody sliding window bez překrývání). Pořadí těchto výřezů je pak náhodně promícháno, čímž vzniká nový augmentovaný vzorek. Problémem této metody je, že nezachovává časové závislosti signálu, takže je velice nepravděpodobné, že by zpřeházením pořadí časových výřezů vznikl nový signál, který by mohl být validní [Igl+23].

6.1.1.5 Kanálová permutace

Kanálová permutace je metoda, při které dojde k vytvoření nových příznakových vektorů prohozením příslušného kanálu jednotlivých vektorů. V oblasti zpracování obrázků by se jednalo o prohození jednotlivých RBG kanálů daného obrázku [Igl+23]. V kontextu této práce by šlo o prohození kanálů naměřených signálů v rámci jednotlivých epoch např. prohození kanálu ***C3*** s kanálem ***Cz***.

Je otázkou, zda je tento postup validní při zpracování EEG signálu. Sice nedojde k narušení časové reprezentace dat, ovšem dojde k poškození prostorové informace.

Proto aby mohla tato metoda být validní, musel by platit předpoklad, že měřené signály jsou nezávislé na samotném kanálu [Igl+23].

6.1.2 Přidání šumu

Noise injection (NI) je jednoduchá, ale velice populární metoda, která vytváří nové augmentované datové vzorky přidáním šumu k jednotlivým příznakům. Jak bylo zmíněno v kapitole 2, jednou z vlastností měřeného EEG signálu je malý poměr signálu k šumu (SNR), tzn. měřený signál už je poměrně silně zašuměný. Tato augmentační metoda této vlastnosti využívá [Igl+23].

Existují různé druhy používaných šumů, např. Poisson, sůl a pepř, ovšem nejpoužívanější pro augmentaci je Gaussův šum [Igl+23; LLM20; He+21]. Metoda funguje tak, že ke každému příznaku je přičten jeden vzorek vygenerovaný výběrem normálního rozdělení s volitelnou střední hodnotou a rozptylem. Matematicky lze metodu popsat jako

$$\hat{\mathbf{x}}(\epsilon) = \{x_0 + \epsilon_0, \ldots, x_{N-1} + \epsilon_{N-1}\} \quad \epsilon_i \sim \mathcal{N}(\mu, \sigma^2) \tag{6.2}$$

kde $\hat{\mathbf{x}}(\epsilon)$ je augmentovaný příznakový vektor, $x_0, \ldots, x_{N-1}$ příznaky vektoru ze kterého je prováděna augmentace a $\epsilon_0, \ldots, \epsilon_{N-1}$ náhodný výběr z normálního rozdělení.

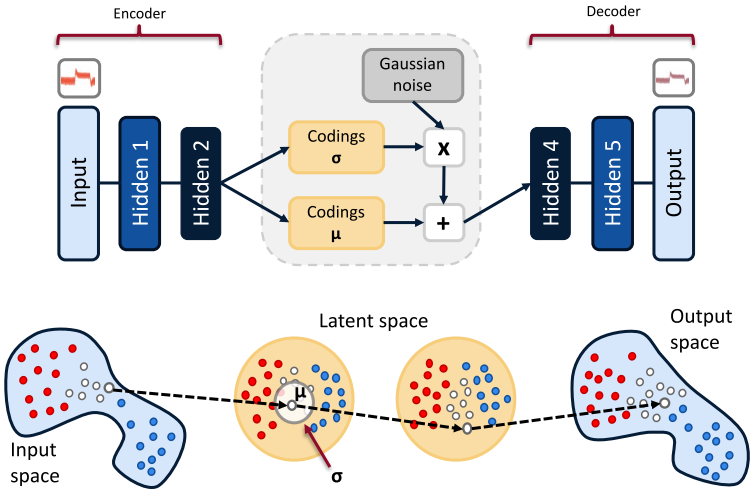
6.2 Generativní modely

Generativní modely jsou třída algoritmů strojového učení, které jsou schopny produkovat nové datové vzorky na základě naučených vlastností získaných z trénovací datové sady. Modely jsou založené na hlubokém učení k získání jednotlivých vzorů a vlastností trénovacích dat pro vytvoření nových syntetických dat. Nejpopulárnější oblastí použití těchto modelů je strojové vidění, kde se modely používají pro generování nových obrázků, každopádně metody jsou používané v různých oblastech.

Cílem generativních modelů tedy není predikce příslušnosti příznakového vektoru k nějaké třídě (klasifikace), ale predikce nových příznakových vektorů náležící podobnému rozdělení jako originální vstupní trénovací sada.

6.2.1 Variační autoenkodér

Variational autoencoder (VAE) je model založený na architektuře Autoencoder (AE). AE je neuronová síť skládající se ze dvou komponent: tzv. enkodér (E) a dekodér (D). Architektura jednotlivých částí neuronové sítě může být libovolná, může se jednat o jednoduchý MLP nebo i klidně CNN. Enkodér má na starosti redukci dimenzionality (zakódování) vstupního příznakového vektoru do tzv. latentního prostoru, zatímco dekodér se snaží z této latentní reprezentace vstupní příznakový vektor



Obrázek 6.2: Vizualizace architektury VAE s vizualizací zakódovaní vstupního příznakového vektoru do latentního prostoru a jeho následného dekódování. Zdroj: `https://www.tvhahn.com/posts/building-vae/`

rekonstruovat. Zakódování v latentním prostoru by tak mělo obsahovat nejvýznamnější vlastnosti vstupního příznakového vektoru, ze kterých je dekodér schopen tento vstup s co nejmenší rekonstrukční chybou zpětně rekonstruovat [Igl+23].

Cenová funkce je v tomto případě poměrně jednoduchá, jelikož rekonstrukční chybu lze definovat například jako rozdíl druhých mocnin vstupu a jeho rekonstrukce

$$J(\theta, \phi) = \frac{1}{N} \sum_{i=0}^{N-1} (\mathbf{x}^{(i)} - D_\theta(E_\phi(\mathbf{x}^{(i)})))^2 \tag{6.3}$$

kde $\mathbf{x}^{(i)}$ jsou trénovací příznakové vektory a $D_\theta(E_\phi(\mathbf{x}^{(i)}))$ jejich rekonstrukce modelem AE.

Tím, že při zakódování dojde ke ztrátě informace, tak rekonstrukce nemůže být perfektní, a tudíž by se dalo nově vzniklý zrekonstruovaný vektor považovat jako nový syntetický vzorek. Ovšem AE je schopen vstupní příznakové vektory mapovat pouze na body v latentním prostoru, a tudíž je schopen generovat pouze omezené množství nových syntetických dat.

VAE se tento problém snaží vyřešit tím že, na místo mapování vstupních příznakových vektorů na vektory v latentním prostoru, provádí mapování na parametry předem definovaného rozdělení v latentním prostoru. VAE přidává omezení na toto latentní rozdělení a snaží se, aby se co nejvíce blížil standardnímu normálnímu rozdělení. Následně je z tohoto latentního rozdělení proveden výběr, ze kterého je pak dekódován nový syntetizovaný výstup [Igl+23]. Architekturu VAE a mapování ze vstupního příznakového prostoru do latentního prostoru je možné vidět na obrázku 6.2 na str. 52.

V modelu VAE jsou vstupní příznakové vektory mapovány na pravděpodobnostní oblasti s příslušným průměrem (μ) a odchylkou (σ). Touto reprezentací průměr distribuce definuje střed oblasti, ze které se budou generovat syntetické datové vzorky a směrodatná odchylka definuje variabilitu vygenerovaného výstupu, tedy jejich diverzitu [Igl+23].

Při učení se tedy VAE snaží minimalizovat dvě cenové funkce. Rekonstrukční chybu, stejně jako architektura AE. Druhou cenovou funkcí je regularizace, která se snaží, aby mapované latentní rozdělení bylo v co nejmenší vzdálenosti od standardizovaného normálního rozdělení. Vzdálenost používaná pro měření této chyby se nazývá Kullback-Leibler divergence (KL-divergence). Při vygenerování nového datového vzorku pomocí VAE se tedy současně měří jak chyba rekonstrukce tohoto výstupu tak chyba výběru z normálního rozdělení. Při výpočtu celkové ztráty se pak obě tyto chyby sečtou [Igl+23].

Cenovou funkci VAE můžeme vyjádřit jako

$$J(\theta, \phi) = \mathbb{E}_{Q_\phi(\mathbf{z}|\mathbf{X})}[\log(P_\theta(\mathbf{X}|\mathbf{z}))] + D_{\mathrm{KL}}(Q_\phi(\mathbf{z}|\mathbf{X})||P(\mathbf{z})) \tag{6.4}$$

kde θ jsou váhy dekodéru, ϕ váhy enkodéru, $\mathbf{z}$ je latentní zakódování získané výběrem z latentní distribuce. $Q_\phi(\mathbf{z}|\mathbf{X})$ je pravděpodobnostní rozdělení $\mathbf{z}$ na základě vstupních dat $\mathbf{X}$ (enkodér), naopak $P_\theta(\mathbf{X}|\mathbf{z})$ je pravděpodobnostní rozdělení $\mathbf{X}$ na základě latentního vektoru $\mathbf{z}$ (dekodér).

Jak již bylo zmíněno, chceme, aby latentní distribuce byla co nejblíže standardnímu normálnímu rozdělení, tzn. je předpoklad že platí $P(\mathbf{z}) = \mathcal{N}(0, \mathbf{I})$. Na základě tohoto předpokladu by se pak tomuto rozdělení mělo blížit i rozdělení $Q_\phi(\mathbf{z}|\mathbf{X})$. Za předpokladu, že se jedná o normální rozdělení s parametry $\mu(\mathbf{X})$ a $\sigma(\mathbf{X})$, můžeme vzdálenost mezi těmito rozděleními $Q_\phi(\mathbf{z}|\mathbf{X})$ a $P(\mathbf{z})$ definovat pomocí KL-divergence, kterou lze odvodit jako

$$D_{\mathrm{KL}}[\mathcal{N}(\mu(\mathbf{X}), \sigma(\mathbf{X}))||\mathcal{N}(0, \mathbf{I})] = -\frac{1}{2}\sum_{l=0}^{L-1}(1 + \log(\sigma_l^2) - \mu_l^2 - \sigma_l^2) \tag{6.5}$$

kde L je dimenzionalita latentního prostoru, μ vektor středních hodnot naučeného rozdělení a σ vektor standardních odchylek.

Latentní proměnné vektoru $\mathbf{z}$ pak lze získat výběrem jako

$$\mathbf{z} = \mu + \sigma \odot \epsilon \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \tag{6.6}$$

kde $\mathbf{z}$ je latentní vektor, μ vektor středních hodnot naučeného rozdělení, σ vektor standardních odchylek a ϵ náhodný výběr ze standardního normálního rozdělení.

Pro vygenerování nového syntetického vzorku pak stačí provést výběr ze standardního normálního rozdělení a provést dekódování pomocí naučeného dekodéru.

6.2.2 Generativní adversariální síť

General Adveserial Network (GAN) je generativní model založený na konkurenci mezi dvěma neuronovými sítěmi. Cílem architektury je replikovat distribuci vstupních dat za účelem tvorby nových syntetických vzorků z dané distribuce. Architektura GAN se tedy skládá ze dvou modelů: tzv. generátoru (G) a diskriminátoru (D). Generátor je generativní model, a tudíž má na starosti generování nových vzorků z distribuce dat, zatímco diskriminátor je diskriminační model a jeho úkolem je odlišit skutečné vstupní data od dat vygenerovaných generátorem. V porovnání s ostatními DA metodami je architektura GAN schopna produkovat rozmanitější vzorky [Igl+23].

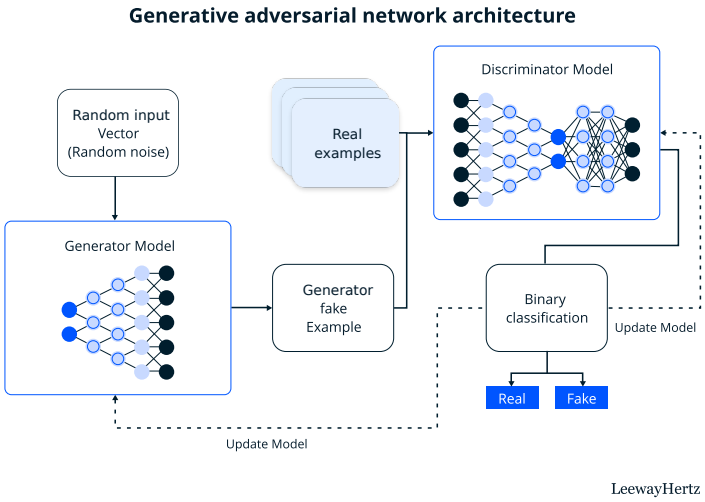
Pro generování nových dat, která jsou nerozeznatelná od distribuce vstupních dat, dochází k interakci obou modelů. Generátor se snaží generovat taková data, která diskriminátor bude klasifikovat jako reálná, zatímco diskriminátor se snaží odlišit vstupní a generovaná data. Pokud je tedy diskriminátor schopen rozlišit distribuci vstupních dat od distribuce generovaných dat, dodává generátoru negativní zpětnou vazbu. Naopak pokud není schopen distribuce rozlišit, dochází k pozitivní zpětné vazbě generátoru. Tímto způsobem se generátor snaží naučit oklamat diskriminátor. Na druhou stranu dochází k pozitivnímu odměnění diskriminátoru pokud provede diskriminaci správně [Igl+23].

Takto soutěživé učení nutí obě sítě ke společnému vývoji. Pokud diskriminátor není schopen správně rozlišit mezi reálnými a generovanými daty, nedojde ke zlepšení generativní schopnosti generátoru, jelikož pokaždé oklame diskriminátor, a to i přes to, že kvalita syntetizovaných vzorků může být slabá. Na druhou stranu pokud diskriminátor pokaždé rozezná mezi reálným a generovaným vzorkem, není generátor schopen diskriminátor oklamat, takže se také nebude vyvíjet [Igl+23]. Standardní architekturu GAN je možné vidět na obrázku 6.3 na str. 55. Architektura jednotlivých modelů, může být stejně jako v případě architektur VAE libovolná.

Matematicky je takovéto konkurenční chování založeno na teorii her, ve které dva hráči soutěží ve hře s tzv. nulovým součtem. Diskriminátor odhaduje aposteriorní pravděpodobnost $p(y|\mathbf{x})$, kde y je označení (reálný nebo falešný) daného vzorku $\mathbf{x}$. Generátor generuje syntetické vzorky z latentního vektoru $\mathbf{z}$. Soutěž mezi generátorem a diskriminátorem je tedy možné definovat jako minimax hru, kde se diskriminátor snaží maximalizovat svoji přesnost při rozlišování mezi falešnou a reálnou distribucí a generátor se tuto přesnost snaží minimalizovat. Matematicky můžeme tento proces vyjádřit jako

$$\min_{G_\phi} \max_{D_\theta} J(D_\theta, G_\phi) = \mathbb{E}_{\mathbf{x}\sim P(\mathbf{X})}[\log(D_\theta(\mathbf{x}))] + \mathbb{E}_{\mathbf{z}\sim P(\mathbf{z})}[\log(1 - D_\theta(G_\phi(\mathbf{z})))] \quad (6.7)$$

kde $\mathbf{z}$ je latentní vektor generovaný výběrem typicky z rovnoměrného nebo nor-



Obrázek 6.3: Vizualizace architektury GAN. Model generátoru z náhodného šumu generuje falešné příznakové vektory. Model diskriminátoru se snaží klasifikovat reálná naměřená data a falešná vygenerovaná data Zdroj: `https://www.leewayhertz.com/a-guide-on-generative-ai-models-for-image-synthesis/`

málního rozdělení $P(\mathbf{z})$, $P(\mathbf{X})$ je distribuce reálných vstupních dat, θ jsou váhy diskriminátoru, ϕ váhy generátoru, $D_\theta(\mathbf{x})$ diskriminátorem provedená klasifikace reálného příznakového vektoru a $D_\theta(G_\phi(\mathbf{z}))$ diskriminátorem provedená klasifikace příznakového vektoru vygenerovaného generátorem z latentního vektoru $\mathbf{z}$ [Igl+23; LLM20].

Bylo ukázáno, že GAN architektura může konvergovat k jedinečnému řešení. Toto řešení je v teorii her známo jako tzv. Nashova rovnováha, která je charakterizována skutečností, že žádná ze sítí již není schopna redukovat hodnotu své cenové funkce. Takového optimálního výsledku je ovšem ve skutečnosti velmi obtížné dosáhnout, jelikož chování GAN je velice nestabilní. Nashova rovnováha ve skutečnosti není dosažena skoro nikdy kvůli neustálé konkurenci mezi oběma sítěmi [Igl+23; He+21].

Architektura GAN je jedna z nejpopulárnějších generativních metod, a to především v oblasti strojového vidění. Na druhou stranu se také jedná o nejsložitější z popsaných modelů. Vzhledem ke způsobu jejich trénování je extrémně obtížné architekturu natrénovat a dosáhnout dobrých výsledků. GAN trpí hlavně problémy nestability, tzv. mode collapse (generátor generuje pouze omezenou nebo opakující se množinu vzorků) a složité vyhodnocení konvergence. Díky těmto problémům nemusí být výsledky GAN úplně spolehlivé [Igl+23].

Implementace 7

Implementace detekce pohybu z EEG dat proběhla v programovacím jazyce Python s využitím především následujících knihoven:

- `NumPy` – (Numerical Python) poskytuje podporu pro efektivní manipulaci s multidimenzionálními poli a výpočty vědeckých operací. Je jednou z nejdůležitějších a nejpoužívanějších knihoven pro vědecké výpočty v Pythonu. Velká většina ostatních knihoven pracuje s objekty této knihovny. V rámci práce je knihovna používaná především pro uložení a manipulaci předzpracovaných dat.
- `MNE` – knihovna zaměřena na analýzu neurofyziologických dat. Poskytuje celou řadu funkcí a objektů pro načítání, zpracování, vizualizaci a analýzu těchto dat (tedy i EEG signálů). Knihovna je v práci používaná pro načítání a předzpracování naměřených dat.
- `scikit-learn` – knihovna poskytující algoritmy strojového učení. Zahrnuje nástroje pro klasifikaci, regresi, shlukování, předzpracování atd. V kontextu práce je knihovna používaná především pro klasifikaci klasifikátory `SVM` a `LDA`, výpočet klasifikačních metrik a předzpracování dat pro klasifikaci.
- `Matplotlib` – knihovna poskytující funkcionalitu pro kreslení a různou vizualizaci dat.
- `Keras/TensorFlow` – `Keras` je knihovna pro hluboké učení a tvorbu neuronových sítí. Poskytuje API s vysokou úrovní abstrakce pro jednoduché sestavení a trénování různých architektur modelů neuronových sítí. Na druhou stranu knihovna `TensorFlow` umožňuje nízkoúrovňové operace jednotlivých modelů. `TensorFlow` poskytuje mimo jiné i možnost použití různých akcelerátorů např. GPU pro urychlení a optimalizaci jednotlivých výpočtů. `TensorFlow` slouží jako backend knihovny `Keras`, tzn. knihovna `Keras` interně používá funkcionalitu knihovny `TensorFlow`. Kombinace těchto knihoven poskytuje uživateli jednoduché rozhraní pomocí knihovny `Keras` pro

tvorbu rychlých prototypů jednotlivých modelů a zároveň poskytuje uživateli přístup k pokročilým možnostem knihovny `TensorFlow`.

7.1 Architektura programu

Implementovaný program je strukturován do několika tříd a modulů, viz diagram 7.1 na str. 59. Modulem se v tomto případě rozumí bezestavový Python skript, který pouze poskytuje ostatním třídám nebo modulům implementaci nějakých funkcí nebo konstant. V programu je zavedena konvence, kdy Python soubor začínající malým písmenem reprezentuje modul, kdežto soubor začínající velkým písmenem třídu.

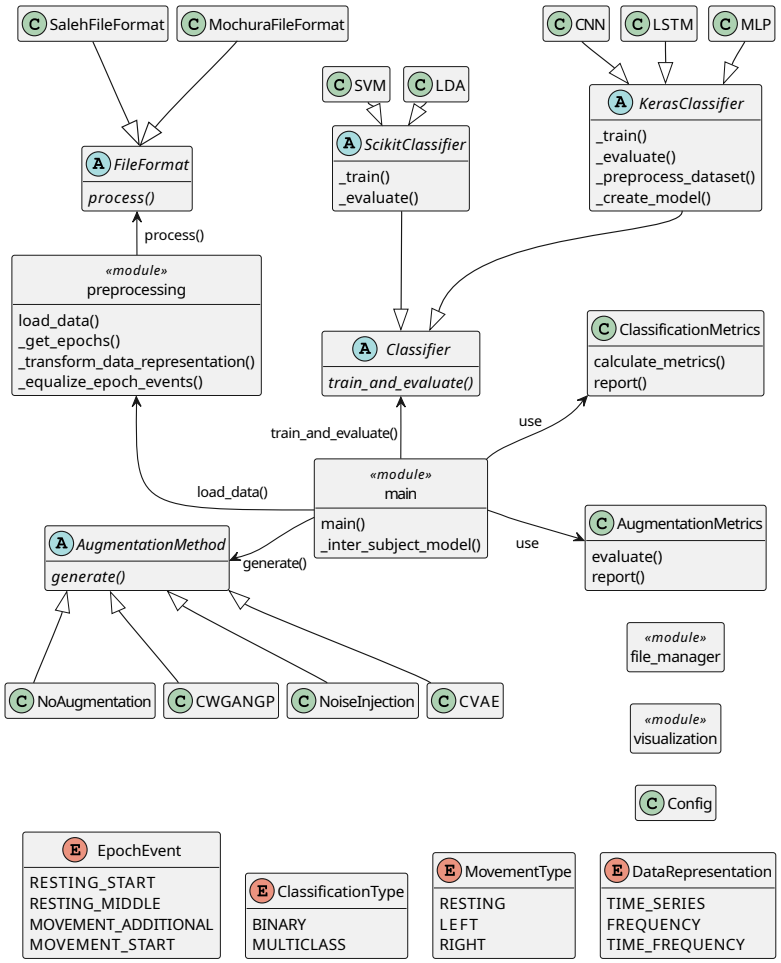
V této sekci budou stručně popsány moduly a třídy, jejichž implementace není příliš zajímavá a pro samotnou detekci pohybu nehrají důležitou roli. Ostatní komponenty budou popsány nebo zmíněny v následujících sekcích.

- `main` – modul `main` je vstupním bodem programu. Volá hlavní funkce a metody ostatních komponent, čímž řídí průběh programu.
- `Config` – singleton třída `Config` reprezentuje globální nastavení programu. Třída obstarává načítání konfiguračního souboru a poskytuje ostatním částem programu hodnoty nastavené uživatelem a interní konfiguraci. Možnosti konfigurace programu jsou popsány v příloze A.2 na str. 106.
- `visualization` – modul poskytuje funkce pro generování různých grafů a vizualizací pro ladění a zobrazování výsledků.
- `file_manager` – modul obstarávající načítání a ukládání souborů. Jedná se hlavně o naměřená vstupní data, serializace a deserializace modelů, vizualizace a grafy.

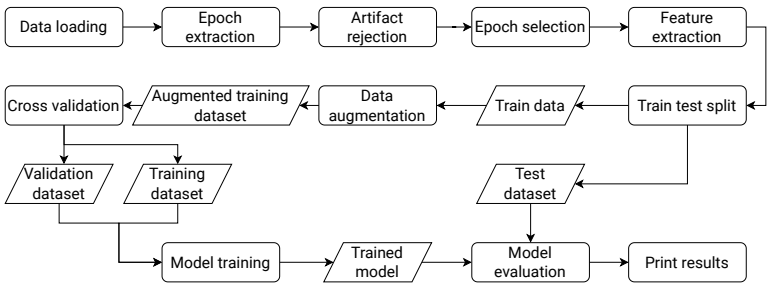
Průběh programu je zobrazen diagramem 7.2 na str. 59. Funkcionalitu programu lze rozdělit do tří hlavních částí: zpracování naměřených dat (načítání, předzpracování, extrakce příznaků a rozdělení na trénovací a testovací datovou sadu), augmentace trénovací sady a trénování a vyhodnocení úspěšnosti klasifikátorů. Jednotlivé části budou podrobněji popsány v následujících sekcích.

7.2 Vstupní data

V této sekci je nejprve popsán scénář navržený pro experiment za účelem detekce MI z EEG dat. Následně dojde k popisu samotného měření na základě navrženého scénáře. Ve zbylých podsekcích je popsáno použité zpracování naměřených dat.



Obrázek 7.1: UML diagram implementovaného programu. Obsahuje pouze výčet důležitých metod, ve skutečnosti jednotlivé komponenty obsahují metod případně funkcí a atributů více. A účelem zachování přehlednosti diagramu jsou zde zobrazeny pouze důležité vazby mezi jednotlivými komponentami, např. modul pro vizualizaci nebo jednotlivé výčtové typy používá většina komponent



Obrázek 7.2: Flowchart diagram, znázorňující průběh programu jedné augmentační metody a jednoho klasifikátoru

7.2.1 Scénář

Scénář experimentu měření EEG dat využitím rehabilitačního robota byl navržen Pavlem Mochurou [Moc21]. Přestože scénář nebyl navržený v této práci, byl použit pro další měření a zároveň stávající data vzniklá měřením na základě tohoto scénáře byla použita pro detekci pohybu. Pro kompletnost a kontext této práce krátce scénář popíšu. Detailnější popis scénáře a průběhu měření viz kapitola 7 a 8 v [Moc21].

Scénář byl navržen následovně: subjekt bude sedět na židli a levou nebo pravou rukou se drží ramene rehabilitačního robota. Samotné experimentální měření pak probíhá střídáním klidové a pohybové fáze po dobu trvání experimentu (10 minut).

- **Klidová fáze** – v této fázi je subjektu znemožněno hýbat ramenem robota a tudíž subjekt nevykonává žádný pohyb. Začátek fáze je subjektu indikovaný rozsvícením červené LED diody. Doba trvání této fáze je nastavena na 10 sekund.

- **Pohybová fáze** – v této fázi subjekt provádí pohyb, tím že se snaží hýbat ramenem rehabilitačního robota po předem definované trajektorii. Začátek této fáze je indikovaný zhasnutím příslušné LED diody. Trvání této fáze bylo nastaveno na 20 sekund.

7.2.2 Měření dat

V rámci této práce bylo provedeno dodatečné měření na pěti subjektech - mužích ve věku od 23 do 25 let, s průměrným věkem 24 let. Měření probíhalo ze dvou důvodů. Prvním důvodem je fakt, že jsem si v rámci práce chtěl vyzkoušet, co takové měření EEG daného experimentu obnáší. Druhým důvodem je další rozšíření celkové datové sady, a to skutečnými naměřenými daty, čímž by měla být zajištěna další robustnost inter-subjekt klasifikátorů.

Měření na základě navrženého scénáře probíhalo v neuroinformatické laboratoři na pracovišti KIV ZČU. Měření probíhalo za přítomnosti zdravotní sestry proškolené v měření EEG signálů. Před měřením byl každému ze subjektů vysvětlen scénář, aby subjekt přesně věděl, co se od něj očekává.

Samotné měření jednotlivých subjektů probíhalo následujícím způsobem. Subjekt byl posazen na židli před rehabilitačního robota. Zdravotní sestřička subjektu nasadila na hlavu EEG čepici se standardním elektrodovým rozmístěním 10-20, viz obrázek 2.1 na str. 9, a elektrody typu Ag/AgCl. Zároveň byla subjektu na ucho připevněna zemnící elektroda a čelo referenční elektroda. Pro měření byly použity elektrody: ***Fz***, ***Cz***, ***Pz***, ***F3***, ***F4***, ***P3***, ***P4***, ***C3*** a ***C4***. Po nasazení čepice sestřička pod každou elektrodu aplikovala vodivý gel, dokud impedance každé z elektrod nebyla nižší než 2kΩ.

Poté, co byl subjekt připraven k vykonávání experimentu, byl spuštěn experiment podle nadefinovaného scénáře. Subjekt v pohybové fázi hýbal ramenem robota po kruhové trajektorii. Trajektorie byla subjektu zobrazena na obrazovce, která se nachází nad pohyblivým ramenem robota. Kromě samotné trajektorie byla na obrazovce zobrazena aktuální pozice subjektu a z ní byly vykresleny dvě šipky. První šipka ukazovala subjektu směr, jakým má ramenem robota pohybovat, aby zůstal na definované trajektorii. Druhá šipka znázorňovala aktuálně vykonávaný směr pohybu. V případě, že se subjekt pohyboval mimo trajektorii, robot automaticky zablokoval pohyblivé rameno, čímž subjekt nutil provádět pohyb správně. Každý ze subjektů vykonával experiment (10 minut) jednou levou rukou a jednou pravou rukou.

Pro zaznamenávání EEG signálu byl použit zesilovač V-AMP 16 firmy BrainProducts. K tomuto zesilovači byla připojena nasazená čepice s elektrodami. Zesilovač byl připojen k počítači, na kterém byl spuštěn zaznamenávající software BrainVision Recorder verze 1.21.0102, taktéž od firmy BrainProducts. V tomto softwaru byla nastavena metadata pro daný scénář, a software zaznamenával signál, vzorkovaný frekvencí 1000Hz, do souborů na disk počítače.

Společně s EEG signálem docházelo k záznamu tzv. stimulů (markerů nebo eventů) jednotlivých událostí (stimuly jsou tedy jednoduché synchronizační značky daných událostí), generovaných rehabilitačním robotem, který byl také připojen k zaznamenávajícímu počítači. Celkem byly zaznamenávány čtyři události.

- **Začátek klidové fáze** – reprezentován značkou označenou hodnotou 1. Je generována při rozsvícení LED diody.
- **Polovina klidové fáze** – označena hodnotou 2. Generována přesně uprostřed klidové fáze, tedy po uběhnutí 5 sekund od rozsvícení diody.
- **První pohyb pohybové fáze** – událost označena hodnotou 5. Vygenerovaná po tom, co subjekt na rameno robota vyvine nadprahovou sílu po skončení klidové fáze.
- **Další pohyby pohybové fáze** – událost označena hodnotou 4. Je generována poté, co subjekt vynaloží na rameno robota nadprahovou sílu kdykoliv v rámci pohybové fáze po vygenerování prvního pohybu označeného značkou 5.

7.2.3 Výsledný dataset

Navržený scénář byl replikován třikrát. Originálně Pavlem Mochurou v rámci jeho diplomové práce [Moc21], následně v rámci bakalářské práce Josefem Salehem [Sal22] a nakonec v rámci této práce. [Moc21] provedl měření na 14 subjektech, z čehož 9 bylo žen ve věku 19 až 23 let, tedy průměrným věkem 19,6 let. Dalších 5

subjektů byli muži ve věku 19 až 22 let s průměrným věkem 20,8 let [Moc21]. Saleh scénář experimentu replikoval na dalších 10 subjektech, přičemž 4 z nich byly ženy a 6 muži. Věkovou kategorii subjektů Saleh neuvádí [Sal22].

Výsledný dataset tedy sestává z měření 29 subjektů. Data byla anonymizovaná a jsou volně dostupná ke stažení na URL adrese `https://zenodo.org/record/7893847`. Data jsou rozdělena do složek podle data prováděného měření. Výsledkem každého experimentu jsou tři soubory. Hlavičkový textový soubor s příponou `.vhdr` obsahuje metadata měření a reference na zbylé dva soubory. Textový soubor s příponou `.vmrk` obsahuje synchronizační značky jednotlivých událostí, tzn. především hodnotu značky a index vzorku signálu, při kterém k události došlo. Binární soubor s příponou `.eeg` obsahuje naměřený EEG signál z příslušných elektrod.

7.2.4 Zpracování naměřených dat

7.2.4.1 Načítání dat

Formáty dat. Mochura a Saleh oba používali různý formát názvů souborů. Originálně byl Mochurův formát následovný

<pohlaví><věk><iniciály><indikátor_pohybu><experiment>

kde <pohlaví> byla jedna z hodnot m, z (muž, žena), <věk> číselná hodnota reprezentující věk subjektu, <iniciály> první písmeno jména a příjmení subjektu, <indikátor_pohybu> jedna z hodnot rh, lh (right hand - pohyb pravou rukou, left hand - pohyb levou rukou) a <experiment> číselný indikátor reprezentující experiment daného subjektu pro daný pohyb (subjekt mohl pro daný pohyb provádět více měření).

V rámci anonymizace subjektů byly soubory tohoto formátu přejmenované do následujícího formátu

<ID><pohlaví><datum><indikátor_pohybu><experiment>

z formátu tedy byl odstraněn věk a iniciály subjektu. <ID> je unikátní identifikátor subjektu v rámci daného data měření, <datum> je datum měření subjektu ve formátu ddmmyyyy. Data naměřená v rámci této práce používají tento formát.

Salehův formát názvu souboru je následující

HR_<datum>_<ID>_<haptika>_<strana_pohybu>

kde <datum> je datum měření také ve formátu ddmmyyyy, <ID> je pořadové číslo měřeného subjektu unikátní v rámci všech subjektů měřených Salehem, <haptika> jedna z hodnot bez_vibratoru_s_haptikou nebo s_vibratory_s_haptikou indikující zda při měření měl subjekt na ruce haptiku (viz sekce 2.4 v [Sal22]). <strana_pohybu> jedna z hodnot leva, prava indikující, jakou končetinou subjekt hýbal.

Implementace načítání. V programu je načítání dat součástí předzpracování, a probíhá tedy jako první část funkce `load_data()` modulu `preprocessing`. Nejprve se program pokouší načítat již předzpracovaná data (má význam pouze v opakovaném spouštění programu), pokud soubory s předzpracovanými daty neexistují, probíhá jejich načítání, předzpracování a extrakce příznaků z dat naměřených signálů.

Samotné načítání dat probíhá ve funkci `group_input_files_per_person()` modulu `file_manager`. Pro reprezentaci jednoho souboru měření byla vytvořena abstraktní třída `FileFormat` od které dědí třídy `MochuraFileFormat` a `SalehFileFormat`. Dochází k postupnému procházení složky `data` a každý soubor nacházející se v této složce je zpracován statickou metodou `process()` každým formátem. Metoda zkontroluje zda formát názvu souboru odpovídá regulárnímu výrazu pro daný formát (Mochury nebo Saleha) a pokud ano, je vytvořena konkrétní instance (tzn. instance třídy `MochuraFileFormat` nebo `SalehFileFormat`). Nad instancí je volána metoda pro načtení dat `read_raw()`, která načítá data funkcí `read_raw_brainvision()` knihovny `MNE`. Tato metoda vrací `Raw` objekt, obsahující data a metadata (např. i synchronizační značky) daného měření.

Seznam všech načtených souborů je následně rozdělen tak, že načtené soubory odpovídající jednomu subjektu jsou spojeny do jednoho vlastního seznamu. Metoda tedy ve výsledku vrací seznam seznamů (`files_per_person`), kde např. na indexu 0 (`files_per_person[0]`) je uložen seznam všech souborů prvního subjektu. Seznam `files_per_person` tedy obsahuje 29 (bylo měřeno 29 subjektů) seznamů s instancemi `FileFormat`.

Jelikož Mochura při měření používal vzorkovací frekvenci 500Hz a Saleh a já vzorkovací frekvenci 1000Hz, je v souborech nalezena nejmenší vzorkovací frekvence (tedy 500Hz), která je v další fázi předzpracování použita pro podvzorkování naměřených dat.

7.2.4.2 Předzpracování dat

Po načtení dat je postupně provedeno předzpracování dat pro každý subjekt. Samotné předzpracování probíhá trochu odlišným způsobem pro různý druh klasifikace viz dále. Program umožňuje uživateli volit buď klasifikaci binární nebo vícetřídní. Tento stav je reprezentován výčtovým typem `ClassificationType`. V případě zvolení binární (`BINARY`) klasifikace, bude program pracovat pouze se dvěma třídami, a to třídou **Pohyb** a třídou **Klid**. Klasifikátor tedy bude výsledné příznakové vektory klasifikovat pouze do těchto dvou tříd, tzn. cílem je detekovat příznakové vektory reprezentující pohyb od příznakových vektorů reprezentují klid.

Při výběru vícetřídní `MULTICLASS` dochází ke klasifikaci to tří tříd: **Pohyb levou rukou**, **Pohyb pravou rukou** a **Klid**.

Předzpracování surových načtených dat probíhá ve funkci `_get_epochs()` modulu `preprocesisng`. Kód metody je možné vidět ve výpisu 7.1 na str. 64.

Zdrojový kód 7.1: Ukázka předzpracování načtených surových dat signálu

```
1  def _get_epochs(files, movement_event, sample_frequency):
2      raws = [file.raw for file in files]
3
4      raw = mne.concatenate_raws(raws)
5
6      events, _ = mne.events_from_annotations(raw)
7      epochs = mne.Epochs(raw,
8                          tmin=config.tmin, tmax=config.tmax,
9                          events=events,
10                         picks=config.channels,
11                         baseline=(None, config.tmin + 0.5))
12
13     epochs.crop(include_tmax=False)
14     epochs.resample(sample_frequency)
15     epochs.filter(config.l_freq, config.h_freq)
16     epochs.drop_bad(reject={'eeg': _REJECTION_THRESHOLD})
17
18     _equalize_epoch_events(epochs, movement_event)
19
20     return epochs, epochs.events[:, 2]
```

Epochování. Pro detekci pohybu ze signálu nás samozřejmě nebude zajímat celý naměřený 10 minutový experiment, ale pouze části signálů okolo synchronizačních značek. Výsekům takového signálu se říká *epocha*. Metoda tedy dostane jako jeden z parametrů soubory subjektu pro daný pohyb (soubory měření levou nebo pravou rukou v případě, že se jedná o vícetřídní klasifikaci, nebo všechny soubory pokud se jedná o binární klasifikaci). Načtené `Raw` objekty jsou na řádku č.4 spojeny v jeden objekt. Řádka č.6 z objektu vybere všechny vygenerované události (`events`).

Na řádcích č.7-11 je provedena konstrukce jednotlivých *epoch* (objekt `Epochs`) z `Raw` objektu. Pro vytváření epoch je nutné volit dobu trvání před (`tmin`) a po (`tmax`) synchronizační značce. Doba trvání před synchronizační událostí byla zvolena na 3,5 sekundy a doba po značce 0,5, tedy stejné hodnoty jako volil Mochura ve své práci [Moc21]. Důvod tohoto volení je jednoduchý, synchronizační značka pro pohyb je generována až poté, co subjekt vyvine na rameno rehabilitačního robota nějakou sílu a zároveň jak bylo zmíněno v kapitole 2, dochází k představě pohybu zhruba 2 sekundy před jeho vykonáním. To znamená, že k představě samotného pohybu tedy muselo dojít někdy před vygenerováním této značky. Dále

zde dochází k volbě používaných elektrod (kanálů). V práci byly na základě rešerše literatury vybrány kanály ***C3***, ***C4*** a ***Cz***. Posledním krokem je tzv. *korekce baseline*. Tato operace spočítá průměrnou amplitudu části epochy (zde volena jako první půl sekunda epochy tedy část signálu od -3,5 do -3 sekundy před synchronizační značkou). Vypočtená průměrná amplituda baseline je odečtena od každé amplitudy signálu epochy.

Podvzorkování. Řádky č.13-14 provádí podvzorkování signálu epochy na specifikovanou vzorkovací frekvenci (tedy 500Hz). Tzn. tímto podvzorkováním vznikají epochy o 2000 vzorcích (jedna epocha trvá 4 sekundy s vzorkovací frekvencí 500Hz tedy $4 \cdot 500$ vzorků signálu).

Filtrování. Řádek č.15 provádí filtrování epochy filtrem typu pásmové propusti s konečnou impulzní odezvou a mezními frekvencemi 8-30Hz, tedy *alfa* a *beta* pásmo EEG signálu (v těchto pásmech dochází k synchronizaci a desynchronizaci, jak bylo popsáno v kapitole 2).

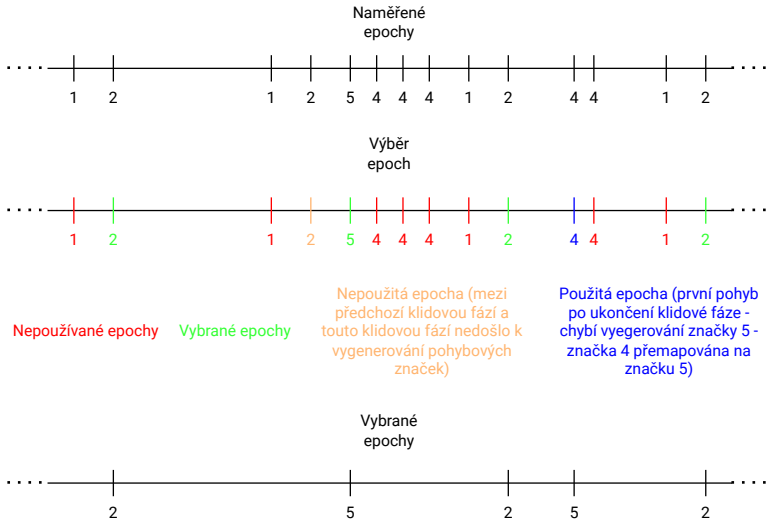
Odstranění artefaktů. Jak bylo zmíněno v kapitole 2, je měření EEG signálů náchylné na výskyt artefaktů, čímž dojde ke ztrátě užitečné informace. Jelikož amplituda měřeného signálu je typicky v řádu desítek mikrovoltů, je pro odstranění artefaktů zvolena jednoduchá strategie tzv. peak-to-peak. Odstranění artefaktů probíhá na řádku č.16 Pro každý kanál epochy je nalezená maximální a minimální amplituda. Pokud platí podmínka

$$\max - \min > \text{threshold}$$

tzn. jeli překročena prahová hodnota (zvolena jako 100 mikrovoltů), dojde k odstranění dané epochy z `Epochs` objektu. Tímto způsobem je například pro binární klasifikaci odstraněno celkem 26,46% epoch reprezentujících třídu klidu nebo pohybu.

Výběr epoch. Dá se předpokládat, že největší úmysl za účelem vykonání pohybu bude následovat po skončení klidové fáze, neboli v prvním pohybu, tedy značce 5. Zároveň lze předpokládat, že pacient bude po skončení pohybové fáze v klidnějším rozpoložení uprostřed klidové fáze. Epochy reprezentující klidovou fázi jsou tedy brány okolo stimulu se značkou 2. Výběr těchto epoch je řešen voláním funkce `_equalize_epoch_events()` na řádku č.18.

Značka signalizující začátek pohybu byla ovšem zavedena až v pozdější fázi diplomové práce Mochury. Datová sada tedy obsahuje soubory, ve kterých se tento stimul vůbec nevyskytuje. Výběr epoch tedy probíhá následovně. Zachovány jsou

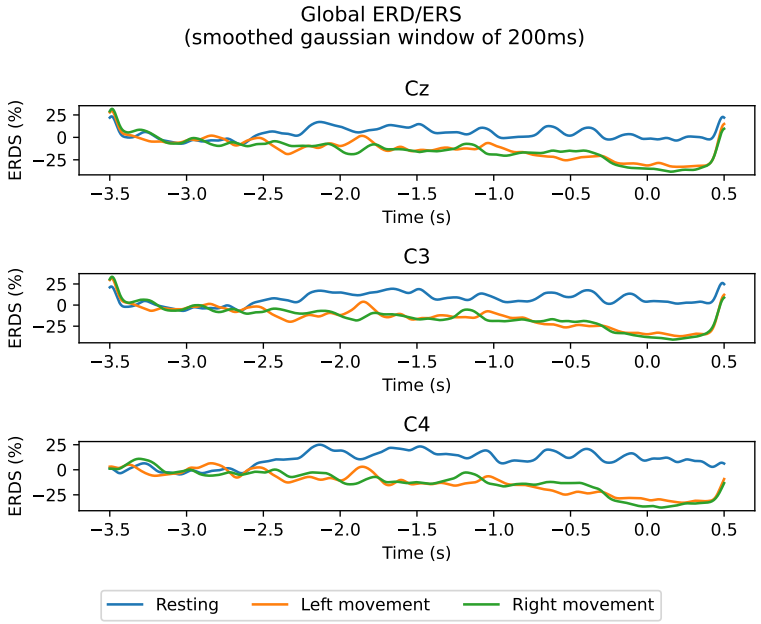


Obrázek 7.3: Ilustrace implementovaného algoritmu pro výběr naměřených epoch pro klasifikaci

pouze takové klidové stimuly (stimul 2), které splňují podmínku, že se mezi aktuálním klidovým stimulem a předchozím zachovaným klidovým stimulem vyskytuje stimul reprezentující pohybovou fázi (stimul 5)[1]. Stimuly představující další pohyb (stimul 4) byly zachovány pouze v případě, že bezprostředně následuje stimul středu klidové fáze. Tzn. pokud se v datech nevyskytuje značka 5, ale pouze značky 4, je použita vždy pouze ta epocha okolo prvního stimulu se značkou 4, která následuje značku 2. Existuje-li v datech například následující sekvence stimulů: 12125444124412 budou pro klasifikaci použity pouze epochy 25242, viz obrázek 7.3 na str. 66. Pokud dojde k nějakému výběru epoch označených stimulem 4, jsou přemapovány na stimul značky 5. Tímto způsobem je zajištěno, že dojde k vybalancovanému výběru epoch reprezentujících klid a epoch reprezentujících pohyb.

V případě provádění vícetřídní klasifikace je pro první pohybovou fázi robotem vždy vygenerovaná značka 5, nehledě na to, jakou horní končetinou subjekt pohybuje. Při zpracování souborů měřených pro pohyb levou rukou dojde k přemapování značky pohybové fáze z hodnoty 5 na hodnotu 6. Výsledné indikátory třídy jsou tedy pak následující: hodnota 2 pro klidové epochy. V případě binární klasifikace hodnota 5 reprezentuje třídu jakéhokoliv pohybu (ať už levou nebo pravou končetinou). V případě vícetřídní klasifikace pak hodnota 5 reprezentuje třídu epoch pohybu pravou rukou a hodnota 6 reprezentuje třídu epoch pohybu levou horní rukou.

[1] V datech se občas vyskytuje několik klidových fází po sobě, aniž by se mezi nimi vyskytoval jakýkoliv stimul reprezentující pohyb. Nejspíše způsobeno chybou při měření.



Obrázek 7.4: Vizualizace vypočítané procentuální hodnoty ERD/ERS z epoch pro každou klasifikační třídu. Vizualizace je vyhlazena Gaussovským filtrem délky 200ms. Počet epoch třídy klidu: 916, třídy pohybu pravou rukou: 860 a třídy pohybu levou rukou: 942

Pro zachování vybalancovaných tříd dochází ještě v případě zvolení vícetřídní klasifikace k náhodnému zahození poloviny klidových epoch z měření experimentu levou rukou a poloviny klidových epoch z měření experimentu pravou rukou.

ERD/ERS. V práci probíhá klasifikace tzv. single trials, neboli na rozdíl od práce Mochury budou vstupem klasifikátorů příznakové vektory jednotlivých vybraných epoch. Ovšem pro vizualizaci bylo z předzpracovaných, vybraných epoch vypočítáno procentuální ERD/ERS podle výpočtu zakončeného rovnicí 2.1 na str. 11. Vypočítané procentuální ERD/ERS bylo vyhlazeno Gaussovským filtrem délky 200ms, viz obrázek 7.4 na str. 67.

Pro vizualizaci obrázku byly použity epochy všech naměřených subjektů. Jak je možné si všimnout, dochází tedy průměrně přes všechny epochy subjektů k poměrně značnému procentuálnímu poklesu výkonu ERD u obou tříd pohybů. Od zhruba 1,5 vteřiny před synchronizační značkou dochází v průměru u obou pohybů ve všech kanálech k poklesu zhruba o 20%. Na druhou stranu v případě klidových epoch dochází naopak k nárůstu (ERS) o 20%. Samozřejmě je nutné si uvědomit, že se jedná o výpočet z epoch všech 29 subjektů. Při použití epoch pouze konkrétního subjektu není u všech subjektů tento jev tak jednoznačný.

7.2.4.3 Extrakce příznaků

Po předzpracování epoch daného subjektu dochází k extrakci příznaků na uživatelem požadovanou reprezentaci dat (výčtový typ `DataRepresentation`). Program nabízí uživateli vybrat ze tří datových reprezentací, které byly diskutovány v kapitole 4. Extrakce příznakových vektorů předzpracovaných vybraných epoch provádí funkce `_transform_data_representation()` modulu `preprocessing`.

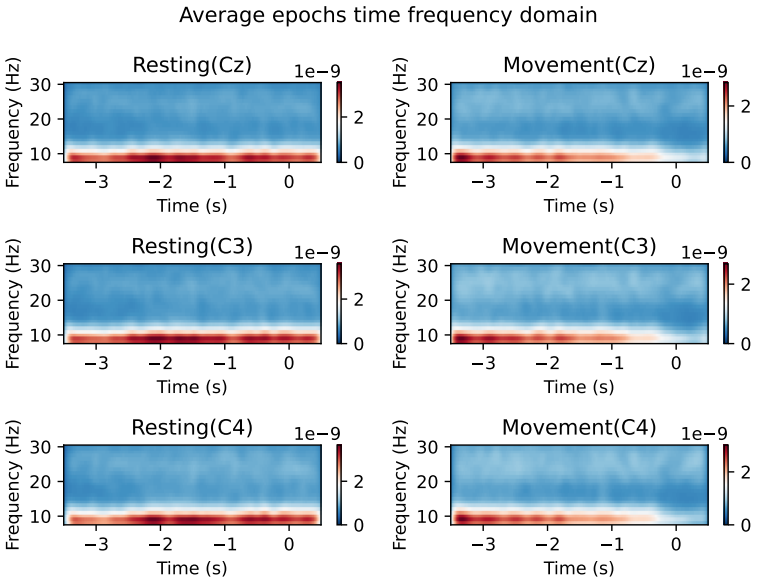
Časová řada. Měřený signál již představuje časovou řadu, takže není potřeba provádět žádnou transformaci epoch. Pro další použití (augmentaci a klasifikaci) není pracováno přímo s objektem `Epochs`, ale je z tohoto objektu získané vícedimenzionální `NumPy` pole (objekt `ndarray`) voláním metody `get_data()` instance objektu `Epochs`. V případě časové řady má pole 3 dimenze: ***n_epochs***, ***n_channels*** a ***n_times***. Počet epoch (***n_epochs***) se samozřejmě liší pro každý subjekt. Počet kanálů (***n_channels***) je 3 a počet časových vzorků (***n_times***) je 2000.

Frekvenční spektrum. Převedení časové řady epoch na frekvenční spektrum je provedeno voláním metody `compute_psd()` instance objektu `Epochs`. Tato metoda provádí převod na výkonově frekvenční spektrum pomocí metody *multitaper*. Metoda probíhá trochu jinak než jak bylo popsáno v kapitole 4.2. Hlavním rozdílem je, že vstupní signál je před výpočtem DFT násoben funkcemi tzv. tapers, které zvýrazňují různé části signálu (v závislosti na tvaru dané taper funkce). Na přenásobený signál je pak aplikovaná operace DFT, čímž vzniká frekvenční spektrum signálu pronásobeného s danou funkcí. Těchto taper funkcí se používá několik, takže výsledné frekvenční spektrum vzniká průměrováním jednotlivých spočtených frekvenčních spekter. Vizualizace tohoto procesu je možná k nahlédnutí na URL adrese `https://shorturl.at/hL045`.

Vypočítané frekvenční spektrum je v práci omezeno pouze na frekvence od 8-30Hz, tedy vyfiltrované frekvenční pásmo epoch. Výsledné `NumPy` pole má tedy rozměry ***n_epochs***, ***n_channels*** a ***n_freqs***.

Časově frekvenční spektrum. Transformace časové řady na časově frekvenční spektrum je provedena pomocí metody `mne.time_frequency.tfr_morlet()`. Metoda funguje způsobem popsaným v kapitole 4.3. Jako waveletovy funkce jsou použité tzv. *Morletovy* wavelety (sinusoida násobená Gaussovskou funkcí). Za účelem redukce dimenzionality a redukce paměťových nároků je provedena decimace časové dimenze faktorem 4. Výsledné pole má tedy rozměry ***n_epochs***, ***n_channels***, ***n_freqs*** (23) a ***n_times*** (500).

Z vizualizace průměrného časového frekvenční spektra, viz obrázek 7.5 na str. 69, je u třídy pohybu poměrně jasně vidět pokles výkonu v alfa pásmu.



Obrázek 7.5: Vizualizace průměrného spektrogramu (časově frekvenční oblasti) vypočteného z epoch pro třídu klidu a třídu pohybu (jakéhokoliv, tedy binární klasifikace)

7.2.5 Rozdělení datové sady

Výsledkem metody načítání a předzpracování dat jsou dvě `NumPy` vícedimenzionální pole. První pole obsahuje data příznakových vektorů jednotlivých subjektů. Tzn. například pro reprezentaci dat časovou řadou má pole rozměry ***n_subjects***, ***n_epochs***, ***n_channels*** a ***n_times***. Druhé pole reprezentuje indikátory příslušnosti tříd jednotlivých epoch a má rozměry ***n_subjects***, ***n_epochs***. Takto předzpracovaná data by se pak dala použít pro konstrukce intra-subjekt modelů, jelikož například výběrem prvního subjektu z pole dat `data[0]` získáme načtená data příslušící pouze jednomu subjektu.

V této práci jsem se ovšem rozhodl zaměřit na tvorbu pouze jednoho modelu, tzn. inter-subjekt modelu, což odpovídá první zmiňované rehabilitační strategii v kapitole 2.3. Je tedy provedeno spojení dat jednotlivých subjektů v jedno pole voláním metody `np.concatenate()`. Tím tedy vzniká pole o rozměrech ***n_epochs***, ***n_channels*** a ***n_times*** (v případě časové řady) a vektor tříd o rozměru ***n_epochs***.

Pro *binární* klasifikaci je počet předzpracovaných epoch (***n_epochs***) roven 3615, z čehož 1808 epoch přísluší třídě ***klid*** (indikátor třídy 2) a 1807 epoch třídě ***pohyb*** (indikátor třídy 5). Pro *vícetřídní* klasifikaci je počet předzpracovaných epoch roven 2718, z čehož 916 náleží třídě ***klid*** (indikátor třídy 2), 942 třídě ***pravý pohyb*** (indikátor třídy 5) a 860 třídě ***levý pohyb*** (indikátor třídy 6).

Celkový dataset je před operací augmentace a následné klasifikace rozdělen

na trénovací data a testovací data, viz diagram 7.2 na str. 59. Rozdělení na trénovací a testovací dataset je řešeno voláním funkce `train_test_split()` knihovny `scikit-learn`. Před rozdělením jsou data náhodně promíchána. Data jsou rozdělena v poměru 80/20, tzn. 80% dat je použitých jako trénovací sada a 20% jako testovací sada.

7.3 Augmentace

Augmentaci dat obstarává abstraktní třída `AugmentationMethod`, která poskytuje abstraktní metodu `generate()`. Metoda vygeneruje příslušnou implementací augmentovanou trénovací datovou sadu o specifické velikosti. Implementace této metody je pak provedena v konkrétních instancích reprezentující danou augmentační metodu. V práci byly implementovány následující augmentační metody: přidání šumu (Noise injection) (třída `NI`), podmíněný variační autoenkodér (třída `CVAE`) a podmíněný GAN s Wasserstein cenovou funkcí a penalizací gradientu (třída `CWGANGP`), viz dále.

Uživatel programu má možnost specifikovat hodnotu parametru `generated_data_multiplier`. Tento parametr řídí počet augmentovaných trénovacích dat. Jedná se o násobek tzn. počet augmentovaných dat v trénovací množině je definován následovně

$$c(\hat{\mathbf{X}}_{train}) = c(\mathbf{X}_{train}) \cdot \text{generated\_data\_multiplier} \tag{7.1}$$

kde $c(\hat{\mathbf{X}}_{train})$ je počet augmentovaných trénovacích dat a $c(\mathbf{X}_{train})$ je počet trénovacích dat. Například v případě binární klasifikace a násobkem nastaveným na hodnotu 1, je počet trénovacích dat roven $3615 \cdot 0.8 = 2892$, to znamená, že bude augmentační metodou vygenerováno 2892 nových syntetických trénovacích dat (pro každou třídu tedy 1446 příznakových vektorů). Výsledná trénovací sada bude tedy obsahovat 5784 příznakových vektorů.

Pro jednoduché porovnání jednotlivých metod umožňuje program uživateli specifikovat více augmentačních metod a klasifikátorů. Zároveň byla implementována třída `NoAugmentation`, jejíž implementace metody `generate()`, pouze vrací předanou trénovací sadu. Třída vznikla pouze za účelem jednodušší generické implementace a je použita v případě, že uživatel chce provést porovnání výsledků klasifikace i bez augmentace (tedy pouze s použitím naměřených dat). Program tedy probíhá tak, že je provedena augmentace trénovací sady pomocí aktuální augmentační metody a následně jsou nad touto trénovací sadou natrénované všechny klasifikátory. Po vyhodnocení všech klasifikátorů následuje augmentace originální trénovací sady další augmentační metodou atd.

7.3.1 Přidání šumu

Augmentace metodou NI je poměrně jednoduchá. Náhodný šum je vygenerován ze standardního normálního rozdělení voláním `NumPy` metody `np.random.randn()`. Šum je tedy `ndarray` o stejných rozměrech jako trénovací data. Následně je proveden náhodný výběr[2] trénovacích dat, která budou použita k augmentaci. Augmentovaná data pak vznikají na základě výpočtu

$$\hat{\mathbf{X}} = \mathbf{X} + \mathbf{X} \odot \mathbf{Z} \tag{7.2}$$

kde $\hat{\mathbf{X}}$ jsou augmentovaná data, $\mathbf{X}$ vybraná trénovací data a $\mathbf{Z}$ vygenerovaný šum.

7.3.2 Podmíněný variační autoenkodér

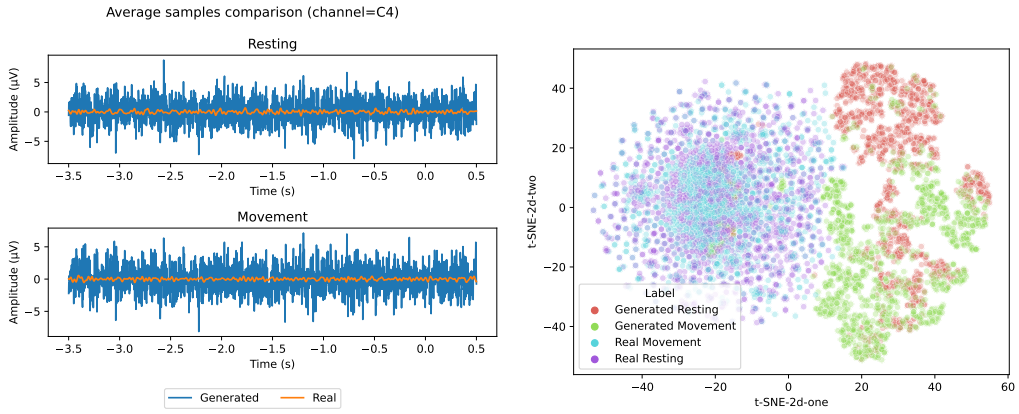
Model cVAE byl sestaven pomocí knihovny `Keras`. Architektura enkodéru i dekodéru je jednoduchý MLP s jednou skrytou vrstvou o 256 neuronech. Enkodér a dekodér jsou tedy zrcadlově symetrické architektury. Obě skryté vrstvy používají ReLU aktivační funkci. Výstupní vrstva dekodéru používá jako aktivační funkci sigmoidu. Latentní dimenze vektorů (vektor středních hodnot a vektor rozptylů rozdělení, viz popis metody v kapitole 6.2.1) byla zvolena na hodnotu 2. Pro učení byl zvolen algoritmus optimalizátoru *Adam* s parametrem rychlosti učení 0.0001. Počet iterací učení byl nastaven na 100. Hodnoty jednotlivých hyperparametrů byly zvoleny empiricky. Byla použita cenová funkce viz rovnice 6.4 a 6.5.

Problém s popisem metody, jak byla popsána v kapitole 6.2.1, ovšem je, že cenová funkce nijak nepracuje s indikátorem třídy vstupního příznakového vektoru. Při implementaci tímto způsobem nemáme při generování nových syntetických příznakových vektorů kontrolu nad tím, z jaké třídy bude vektor vygenerován. Jedním z řešení tohoto problému je možnost rozdělit datovou množinu na podmnožiny, ve kterých budou pouze data příslušné třídy, a pro každou podmnožinu natrénovat vlastní model. Druhou možností je zakódovat indikátor třídy přímo do vstupního příznakového vektoru, čímž provádíme tzv. podmínění.

V implementaci byla zvolena druhá varianta, a to způsobem, že enkodér i dekodér mají dvě vstupní vrstvy. Jedna vrstva pro příznakové vektory a druhá vrstva pro one hot vektory příznaků tříd. Tyto vrstvy jsou následně spojeny v jednu, čímž dojde k rozšíření příznakového vektoru o další příznaky reprezentující třídu daného vektoru.

Vstupní trénovací data jsou před trénováním předzpracována. Nejprve jsou zvektorizována pomocí třídy `Vectorizer` z knihovny `MNE`. Neboli například u dat reprezentovaných časovou řadou má jeden datový vzorek dimenze 3×2000 (***n_channels*×*n_times***). Vektorizací vznikne zploštělý vektor o velikosti 6000. Zároveň,

[2] S opakováním, jelikož pokud je parametr násobku větší než 1, je nutné některé příznakové vektory augmentovat vícekrát.



Obrázek 7.6: Ukázka porovnání vygenerovaných a reálných dat reprezentovaných časovou řadou. Na levém obrázku je porovnání průměrných epoch pro binární třídy a kanál C4. V pravé části je projekce jednotlivých reálných a vygenerovaných epoch do 2D prostoru metodou t-SNE. Vizualizace jasně ukazuje odlišný charakter vygenerovaných dat oproti datům naměřeným

jelikož poslední vrstva dekodéru používá sigmoid aktivační funkci, tak jsou vstupní data škálovaná na rozsah 0-1. Indikátory tříd příznakových vektorů jsou převedeny na one hot vektory, tzn. při binární klasifikaci je indikátor třídy klidu (2) převeden na vektor [1, 0] a indikátor třídy pohybu (5) je převeden na vektor [0, 1].

Data jsou vygenerována použitím natrénovaného dekodéru. Vygeneruje se šum ze standardního normálního rozdělení, který je společně s one hot vektorem třídy vstupem dekodéru. Dekodér vygeneruje nový příznakový vektor, který je inverzní transformací předzpracování převeden do originálních dimenzí.

7.3.3 Podmíněná generativní adversariální síť s Wasserstein cenovou funkcí a penalizací gradientu

Implementace augmentační metody GAN se projevila jako velmi problematická. V průběhu implementace byly zkoušeny různé architektury sítí s různými parametry. Ovšem žádná z nich neposkytovala generování kvalitních reprezentativních dat pro žádnou z datových reprezentací. I výsledná implementovaná architektura generuje data, která nelze považovat za věrohodná viz obrázek 7.6 na str. 72. Ze všech vyzkoušených architektur ovšem zvolená architektura poskytuje nejvěrohodnější vygenerovaná data.

V práci byla použita architektura GAN používající Wasserstein cenovou funkci a penalizaci gradientu. Tato architektura by oproti klasické architektuře GAN, která používá Jensen-Shanon nebo Kullback-Leibler divergenci jako svoji cenovou funkci,

měla řešit problém nestability trénování sítí [He+21; Gul+17]. V práci tedy není používána cenová funkce popsána rovnicí 6.7 na str. 54, ale cena

$$J(\theta, \phi) = \underbrace{\mathbb{E}[D_\theta(G_\phi(\mathbf{z}))] - \mathbb{E}[D_\theta(\mathbf{x})]}_{\text{Wasserstein cena}} + \underbrace{\lambda \mathbb{E}[(\|\nabla_{\hat{\mathbf{x}}} D(\hat{\mathbf{x}})\|_2 - 1)^2]}_{\text{Penalizace gradientu}} \tag{7.3}$$

kde $\mathbf{z}$ je latentní vektor generovaný ze standardního normálního rozdělení, $\mathbf{x}$ je příznakový vstupní vektor, $\hat{\mathbf{x}}$ rovnoměrná interpolace mezi $\mathbf{x}$ a $D_\theta(G_\phi(\mathbf{z}))$ (vygenerovaný příznakový vektor ze šumu $\mathbf{z}$). λ je váha penalizace gradientu, D_θ diskriminátor s váhami θ a G_ϕ generátor s váhami ϕ.

Generátor je MLP architektura se třemi skrytými vrstvami o 128, 256 a 512 neuronech, každá z vrstev používá Leaky ReLU aktivační funkci s parametrem alfa 0,2. Výstupní vrstva používá aktivační funkci tanh. Generátor používá optimalizátor *Adam* s parametrem rychlosti učení 2e-6, parametrem beta_1 s hodnotou 0 a beta_-2 s hodnotou 0,9. Diskriminátor je modelován také jako MLP se třemi skrytými vrstvami o 512, 256 a 128 neuronech, s aktivační funkcí Leaky ReLU s parametrem alfa 0,2. Výstupní vrstva je tvořena jedním neuronem (binární klasifikace) a používá aktivační funkci sigmoid. Diskriminátor používá optimalizátor učení *RMSProp* s parametrem učení 5e-6.

Podobně jako metoda cVAE, popsaná v předchozí sekci, je tato metoda také podmíněna indikátorem třídy. Podmínění je prováděno jinak, tzn. není ke vstupnímu příznakovému vektoru připojen one hot vektor třídy. Indikátor třídy je zakódován tzv. embeddingem do prostoru stejné dimenzionality jako vstupní příznakový vektor. Takto zakódovaná třída je pak pronásobena se vstupním příznakovým vektorem, čímž dojde k zakódovaní indikátoru třídy do tohoto vektoru.

Dimenzionalita latentního vektoru (šumu) byla nastavena na 100. Váha penalizace gradientu byla nastavena na 10 [Gul+17]. Jelikož výstupní vrstva generátoru používá aktivační funkci tanh, jsou před trénováním příznakové vektory naškálované na rozsah -1, 1. Trénování celkového modelu je komplikovanější než u klasických modelů vytvořených knihovnou `Keras`, kdy stačí nad vytvořeným modelem zavolat metodu `fit()` s příslušnými parametry. Je nutné aby docházelo ke střídání učení diskriminátoru s učením generátoru. Na základě [Gul+17] je prováděno 5 kroků učení diskriminátoru na 1 krok učení generátoru. Počet iterací učení byl empiricky zvolen na 525.

Nová data jsou generována tím, že se vygeneruje šum ze standardního normálního rozdělení. Tento šum společně s indikátorem třídy je poté použit jako vstup pro natrénovaný generátor. Generátor vygeneruje nový příznakový vektor v rozsahu -1, 1, který je inverzní transformací naškálován do originálního rozsahu vstupních dat.

7.3.4 Metriky

Ve studiích zabývajících se augmentací dat neexistuje žádná konkrétní shoda pro použití vyhodnocovacích metrik augmentovaných dat. Většina metrik je zaměřena především na oblast strojového vidění, protože v této oblasti se tyto metody používají nejčastěji [Igl+23]. Hlavním porovnáním je samozřejmě zlepšení účinnosti klasifikačních modelů, viz kapitola 8.

Výpočet metrik je implementovaný ve třídě `AugmentationMetrics` metodou `evaluate()`. Metoda je volána vždy po provedení augmentace. Před výpočtem každé z následujících popisovaných metrik jsou naměřená i vygenerovaná data normalizována (transformace dat na průměr 0 a standardní odchylku 1). Výpis vypočítaných metrik je prováděn voláním metody `report()`.

7.3.4.1 Vizualizace

Jednou z možností vyhodnocení vygenerovaných dat je samozřejmě jejich vizualizace. V programu je implementované porovnání reálných a vygenerovaných dat viz obrázek 7.6 na str. 72. Obrázek vizualizuje porovnání vygenerovaných dat reprezentovaných časovou řadou pomocí metody cWGAN-GP. Program obdobně generuje vizualizace i pro ostatní augmentační metody a datové reprezentace.

7.3.4.2 Fréchetova vzdálenost incepce

Fréchet inception distance (FID) je nejpoužívanější metrika používaná k vyhodnocení kvality a rozmanitosti vygenerovaných dat. Metrika počítá statistickou vzdálenost mezi pravděpodobnostním rozdělením reálných dat a pravděpodobnostním rozdělením generovaných dat. Metrika je počítána jako

$$\mathrm{FID} = \left\|\mu_r - \mu_g\right\|^2 + \mathrm{tr}(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}) \tag{7.4}$$

kde μ_r je vektor středních hodnot reálných příznakových vektorů, μ_g střední hodnoty vygenerovaných dat, Σ_r kovarianční matice reálných dat, Σ_g kovarianční matice vygenerovaných dat. Operace tr je výpočet stopy matice (součet prvků na hlavní diagonále).

7.3.4.3 Poměr signálu a šumu

Signal-to-noise ratio (SNR) je metrika pro porovnání zašumění vygenerovaných dat. Metrika je v programu počítána jako

$$\begin{aligned} P_{signal} &= \mathbb{E}[\hat{\mathbf{X}}^2] && (7.5)\\ \mathbf{noise} &= \hat{\mathbf{X}} - \mathbf{E}[\hat{\mathbf{X}}] && (7.6)\\ P_{noise} &= \mathbb{E}[\mathbf{noise}^2] && (7.7)\\ SNR &= 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) && (7.8) \end{aligned}$$

kde $\mathbb{E}[\hat{\mathbf{X}}^2]$ je průměrný výkon vygenerovaných dat, **noise** je odhad šumu ve vygenerovaných datech a $\mathbb{E}[\mathbf{noise}^2]$ průměrný výkon šumu.

7.3.4.4 Odmocnina střední kvadratické odchylky

Root mean square error (RMSE) je metrika používaná pro porovnání rozdílu mezi dvěma datovými sadami. Je počítána jako

$$RMSE = \sqrt{\frac{\sum_{i=0}^{M-1} \mathbf{x}^{(\mathbf{i})} - \hat{\mathbf{x}}^{(\mathbf{i})}}{M}} \quad (7.9)$$

kde $\mathbf{x}^{(\mathbf{i})}$ je i-tý příznakový vektor reálných dat, $\hat{\mathbf{x}}^{(\mathbf{i})}$ je i-tý příznakový vektor vygenerovaných dat.

7.3.4.5 Průměrná křížová korelace

Cross-correlation (CC) je měřítko vyjadřující podobnost mezi dvěma řadami. Čím vyšší korelace, tím více jsou si řady podobné. V práci je počítaná průměrná normalizovaná korelace jednotlivých tříd mezi reálnými daty a vygenerovanými daty. Korelaci mezi reálným příznakovým vektorem a vygenerovaným příznakovým vektorem je možné vyjádřit jako

$$\phi[k] = \sum_{n=0}^{N-1} \mathbf{x}[k+n]\hat{\mathbf{x}}[n] \quad (7.10)$$

kde k je posunutí signálu, jelikož oba příznakové vektory mají stejnou délku N je posunutí nastaveno na 0 (tedy bez posunutí). Korelace je vypočtena mezi příznakovými vektory náležící dané klasifikační třídě a následně je vypočítaná průměrná korelace dané třídy. Výsledná metrika je vypočítaná jako průměrná korelace průměrných korelací tříd.

7.4 Klasifikace

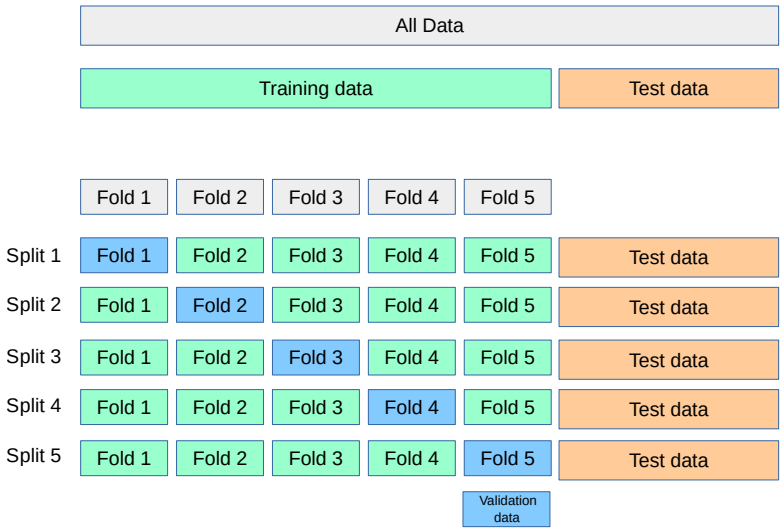
Klasifikace je v programu zprostředkována abstraktní třídou `Classifier` voláním abstraktní metody `train_and_evaluate()`. V práci je implementováno 5 klasifikátorů. Dva klasifikátory (SVM a LDA) pomocí knihovny `scikit-learn` a tři klasifikátory (MLP, LSTM a CNN) pomocí knihovny `Keras`. Pro jednotlivé knihovny byla vytvořena abstraktní třída, tedy `ScikitClassifier` a `KerasClassifier`, které obě dědí od třídy `Classifier`. Tyto dvě abstraktní třídy implementují metodu `train_and_evaluate()`. Dále byly připraveny stejnojmenné třídy pro každý zmíněný klasifikátor, které již představují konkrétní implementaci. Třídy klasifikátorů slouží pouze pro implementaci předzpracování dat a definici konkrétního modelu, samotné trénování a vyhodnocení klasifikace probíhá v předem zmíněných rodičovských třídách viz dále.

Při trénování jednotlivých klasifikátorů je použita technika k-násobné křížové validace, která poskytuje lepší náhled na schopnost generalizace trénovaného modelu. Tato technika je v práci použita následujícím způsobem:

1. Proveď rozdělení trénovacího datasetu na k částí.
2. $k - 1$ částí bude použito jako trénovací příznakové vektory pro natrénování klasifikátoru a 1 část bude použita jako validační dataset (není tím myšleno testovací dataset, viz diagram 7.2 na str. 59)
3. Proveď vytvoření instance nového modelu s náhodně nastavenými vahami
4. Proveď natrénování modelu na $k - 1$ částech trénovacího datasetu a prováděj kontrolu přeučení na validačním datasetu (viz dále v kapitole 7.4.2)
5. Proveď vyhodnocení klasifikačních metrik natrénovaného klasifikátoru na testovacím datasetu
6. Celý proces je k-krát opakován

Celý proces pro $k = 5$ je vizualizován obrázkem 7.7 na str. 77. Výsledkem jsou průměrné klasifikační metriky, viz kapitola 7.4, z k natrénovaných modelů. Natrénovaný model, jehož klasifikační přesnost byla ze všech k modelů nejvyšší, je serializován a uložen na disk do složky `trained_models`.

Klasifikační třídy příznakových vektorů již byly popsány v kapitole 7.2.4.2. Před trénováním klasifikátorů jsou indikátory příslušnosti tříd transformovány na one hot vektory. Zároveň jsou před trénováním modelů příznakové vektory trénovacích a testovacích dat normalizovány (transformace na průměr 0 a standardní odchylku 1). Objekt provádějící normalizaci dat je serializován a ukládán společně s natrénovaným modelem. Natrénované modely je tedy možné načítat při opakovaném



Obrázek 7.7: Ukázka procesu trénování použitím k-násobné křížové validace, pro $k = 5$. Zdroj: `https://scikit-learn.org/stable/modules/cross_validation.html`

spouštění programu, nebo dokonce z jiného programu, kde může být využit pro klasifikaci.

7.4.1 Scikit-learn

Knihovna `scikit-learn` nepodporuje používání validačního datasetu, ovšem jinak trénování probíhá stejným způsobem popsaným v předchozí sekci. Abstraktní třída `ScikitClassifier` poskytuje abstraktní metodu `_create_model()`, kterou potomci implementují pro vytvoření konkrétní instance klasifikátoru při křížové validaci.

V práci je také využíváno knihovny `scikit-learn-intelex`, což je rozšíření knihovny `scikit-learn`, které poskytuje možnost akcelerace klasifikačních algoritmů. Například algoritmus klasifikátoru SVM v knihovně `scikit-learn` probíhá pouze na jednom vlákně CPU. Použitím `intelex` rozšíření jsou využita všechna vlákna CPU. Pro porovnání: 10-násobná křížová validace klasifikátoru SVM pro binární klasifikaci dat reprezentovaných časovou řadou bez augmentace, trvala bez použití rozšíření 00:22:57, a s rozšířením 00:00:45 (hh:mm:ss). Nevýhodou tohoto rozšíření ovšem je, že využitím N vláken také dochází k N násobnému využívání paměti. Pro data reprezentovaná časově frekvenční oblastí dochází i s decimací časové oblasti k nedostatku paměti. Pro tuto datovou reprezentaci se tedy rozšíření nepoužívá, což se také podepisuje na době učení.

7.4.1.1 Lineární diskriminační analýza

Třída `LDA` pouze implementuje metodu `_create_model()`, která vrací instanci třídy `LinearDiscriminantAnalysis` z knihovny `scikit-learn`.

7.4.1.2 Support vector machine

Třída `SVM` pouze implementuje metodu `_create_model()`, která vrací instanci třídy `SVC` z knihovny `scikit-learn`. Klasifikátor používá základní nastavení, tzn. hodnotu parametru C nastavenou na 1 a používá Gaussovský kernel. Toto základní nastavení se ukázalo jako nejlepší v práci Saleha [Sal22].

7.4.2 Keras

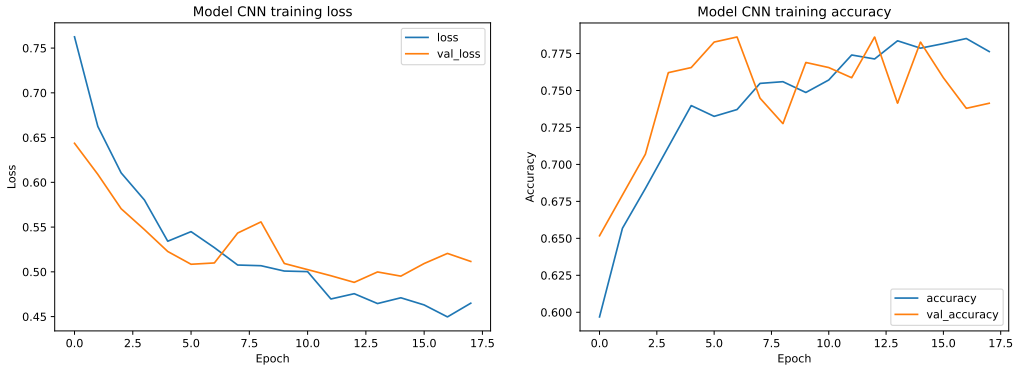
Při trénování `Keras` modelů je validační dataset používán pro kontrolu učení klasifikátoru. Počet iterací trénování modelu je nastaven na 100, ovšem pomocí techniky tzv. early stopping (předčasné zastavení), kdy je monitorována hodnota validační cenové funkce, dojde k zastavení trénování u všech modelů dříve. Implementace trénování modelu v jedné křížově validační iteraci je ukázána ve výpisu kódu 7.2. V každé iteraci tedy dojde k vytvoření nové instance konkrétního modelu (MLP, LSTM nebo CNN, viz následující sekce) s náhodně inicializovanými vahami, voláním metody `_create_model()`. Voláním metody `fit()` dojde k jeho natrénování.

Zdrojový kód 7.2: Ukázka trénování Keras modelu v jedné křížově validační iteraci

```
1 def _train_k_fold(x_train, y_train, x_validation,
    y_validation, num_classes, fold):
2   model = self._create_model(x_train.shape[1:], num_classes)
3   early_stopping = EarlyStopping(monitor='val_loss', patience
    =5)
4   history = model.fit(x_train, y_train, validation_data=(
    x_validation, y_validation), callbacks=[early_stopping],
    shuffle=True, epochs=self._epochs)
```

Jedna iterace trénování modelu tedy probíhá následovně. Je provedený dopředný průchod trénovacích dat modelem, a dojde k výpočtu trénovací cenové funkce. Algoritmem zpětného průchodu dojde k výpočtu gradientů a aktualizování vah klasifikátoru. V posledním kroku iterace dojde k dopřednému průchodu validačních dat a výpočtu validační cenové funkce.

Pokud se hodnota validační cenové funkce po určitý počet iterací (určeno parametrem patience, viz řádka č.3 ukázky kódu 7.2, v práci nastaveno na 5) nezlepší, dojde k předčasnému ukončení trénování. V případě, že by k předčasnému ukončení nedošlo, nastala by situace kdy by došlo k přetrénování modelu na trénovacích datech, čímž by došlo k výraznému poklesu schopnosti generalizace. Knihovna



(a) Cenová funkce

(b) Accuracy

Obrázek 7.8: Ukázka průběhu cenové funkce a přesnosti při trénování klasifikátoru CNN na časové řadě bez provedení augmentace

`Keras` poskytuje historii vývoje cenové funkce a případně dalších klasifikačních metrik v průběhu trénování modelu. Je tedy možné průběh trénování vizualizovat, viz obrázek 7.8 na str. 79.

Knihovna `TensorFlow` podporuje trénování modelů i pomocí akcelerátorů například využitím GPU, čímž dochází ke značnému zrychlení celého trénovacího procesu. Bohužel správa GPU paměti knihovnou `TensorFlow` není ideální. V základním nastavení knihovna alokuje skoro veškerou dostupnou paměť GPU, a to pro *každý* model. Hlavním problémem ovšem je, že potom co dojde k natrénování modelu, nedojde k uvolnění alokované paměti. Tzn. například při použití křížové validace docházelo k ukončení programu, jelikož postupně byla alokována veškerá dostupná paměť.

Knihovna neposkytuje žádné API[3], kterým by umožnila uživateli knihovny manuální dealokaci paměti. Jediné řešení tohoto problému je, na základě diskuze `https://github.com/tensorflow/tensorflow/issues/36465`, spouštění trénování modelů v novém procesu, jelikož po skončení daného procesu dojde k uvolnění paměti operačním systémem. Toto řešení bylo v práci implementováno a zdá se být funkční. Problémem tohoto řešení ovšem je, že dojde k natrénování modelu v jiném procesu, než jaký uživatel originálně spustil. Jelikož je potřeba provést vyhodnocení trénování v hlavním procesu, bylo nutné implementovat komunikaci mezi originálním procesem a procesem vytvořeným za účelem trénování. Komunikace byla řešena serializací a deserializací modelu, tzn. po dokončení trénování je model dočasně serializován na disk v trénovacím procesu a následně je zpětnou deserializací načten do paměti hlavního procesu.

[3]Alespoň v aktuální používané verzi 2.10.0

7.4.2.1 Vícevrstvý perceptron

Podobně jako v práci Mochury [Moc21], je model MLP tvořen třemi skrytými vrstvami. První vrstva je tvořena 512 neurony, druhá 256 a třetí 128. Všechny skryté vrstvy používají aktivační funkci sigmoid. Poslední vrstva je tvořena na základě počtu klasifikačních tříd, tzn. pro binární klasifikaci jsou používány 2 neurony, pro vícetřídní klasifikaci 3 neurony. Výstupní vrstva používá aktivační funkci softmax, viz rovnice 5.12.

Je používaná cenová funkce cross entropy, viz rovnice 5.13, a optimalizátor Adam s parametrem učení s hodnotou 0,001.

7.4.2.2 Neuronová síť s dlouhodobou a krátkodobou pamětí

Architektura sítě LSTM je tvořena třemi skrytými vrstvami. Všechny tři vrstvy používají 256 buněk a je používaná regularizační technika dropout (náhodné vynechání aktivací určitého množství, zde zvoleno jako 20%, aktivací neuronů).

Výstupní vrstva, cenová funkce a optimalizátor učení jsou nastaveny stejně jako v případě architektury MLP.

7.4.2.3 Konvoluční neuronová síť

V práci je používaná CNN architektura tzv. *EEGnet* [Law+18]. Tato architektura je rozdělena do dvou bloků. První blok je tvořen následujícími vrstvami. První skrytá konvoluční vrstva používá 8 filtrů o rozměrech (1, 64) pro extrakci frekvenčních příznaků z časové oblasti. Následuje vrstva provádějící batch normalizaci (Učení klasifikátorů je typicky prováděno po dávkách - batch. Neboli v jedné iteraci učení je trénovací sada rozdělena na dávky určité velikosti, v práci je používaná dávka o 32 příznakových vektorech. Klasifikátor je pak postupně učen na jednotlivých dávkách. Batch normalizace provádí normalizaci v rámci jedné dávky.). Následuje druhá skrytá konvoluční vrstva s filtry rozměrů (C, 1), kde C je počet kanálů (v této práci tedy 3). Tato vrstva provádí extrakci prostorových příznaků. Následuje opět vrstva batch normalizace, následně vrstva aktivační funkce ReLU, po které je provedena pooling vrstva provádějící operaci průměrování s filtrem o rozměru (1, 4). Poslední částí prvního bloku je vrstva provádějící operaci dropout 50% aktivací.

Druhý blok sítě je tvořen následujícími vrstvami. Třetí konvoluční vrstva používající 16 filtrů s rozměry (1, 16). Další vrstvou je batch normalizace, následovaná vrstvou aktivační funkce ReLU, následně pooling vrstva provádějící operaci průměrování s filtrem o rozměru (1, 4), pak dropout vrstva (opět 50% aktivací) a poslední vrstvou je vrstva plného propojení.

Výstupní vrstva, cenová funkce a optimalizátor učení jsou nastaveny stejně jako v případě architektury MLP.

7.4.3 Metriky

Vyhodnocení natrénovaného klasifikátoru probíhá predikcí příslušnosti klasifikační třídy testovacích dat. Na základě predikované třídy a skutečné třídy jednotlivých příznakových vektorů testovacích dat, jsou spočítané různé klasifikační metriky, především založené na hodnotách chybové matice (confusion matrix).

Třída `ClassificationMetrics` poskytuje metody `calculate_metrics()` a `report()` pro výpočet a výpis výsledků klasifikačních metrik.

7.4.3.1 Accuracy

Metrika `accuracy` (přesnost) vyjadřuje poměr správně provedených predikcí k celkovému počtu provedených predikcí.

$$\text{Accuracy} = \frac{\text{Počet správných predikcí}}{\text{Počet predikcí}} \tag{7.11}$$

Pro binární klasifikaci můžeme `accuracy` vyjádřit z chybové matice jako

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{7.12}$$

Označíme-li jednu třídu jako pozitivní a druhou jako negativní, pak TP je počet správně klasifikovaných pozitivních příznakových vektorů (tzn. skutečná třída příznakového vektoru je pozitivní a klasifikátor pro příznakový vektor predikoval pozitivní třídu), TN je počet správně klasifikovaných negativních příznakových vektorů, FP je počet nesprávně klasifikovaných negativních vektorů (tzn. skutečná třída příznakového vektoru je negativní, ale klasifikátor pro příznakový vektor predikoval pozitivní třídu) a FN je počet nesprávně klasifikovaných pozitivních vektorů.

7.4.3.2 Precision

Metrika `precision` (česky také přesnost) je definována jako podíl počtu správně klasifikovaných pozitivních příznakových vektorů k celkovému počtu pozitivně predikovaných příznakových vektorů.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7.13}$$

Precision se zaměřuje na to, jak moc jsou výsledky klasifikace správné v pozitivní třídě. Vyšší hodnota indikuje vysoký počet správně predikovaných pozitivních příznakových vektorů a naopak.

7.4.3.3 Recall

Metrika `recall` vyjadřuje poměr počtu správně klasifikovaných pozitivních příznakových vektorů k celkovému počtu pozitivních příznakových vektorů.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{7.14}$$

7.4.3.4 F1 Score

Metrika `F1 Score` kombinuje metriky `precision` a `recall`. Je počítána jako harmonický průměr mezi těmito metrikami neboli

$$\text{F1 Score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{7.15}$$

Z hodnot chybové matice se tedy metrika dá přímo vyjádřit jako

$$\text{F1 Score} = \frac{2TP}{2TP + FP + FN} \tag{7.16}$$

Dosažené výsledky 8

Veškeré experimenty probíhaly na počítači s následující systémovou specifikací:

- `CPU` – Processor Intel(R) Core(TM) i7-6700 CPU @ 3.40GHz, 3408 Mhz, 4 Core(s), 8 Logical Processor(s)
- `RAM` – 16GB DDR4 3200MHz
- `GPU` – NVIDIA GeForce GTX1660 O6G
- `OS` – Microsoft Windows 10 Pro

Pro vyhodnocení klasifikační schopnosti jednotlivých klasifikátorů byla provedena 10-násobná křížová validace. Pro každou klasifikační metriku byl spočítán průměr a směrodatná odchylka z deseti křížově validačních iterací. Klasifikační výsledky v tabulkách v následující sekci mají formát průměr±směrodatná_odchylka. Nejlepší výsledek dané klasifikační metriky pro daný klasifikátor je zvýrazněn **tučně**. Nejlepší globální výsledek klasifikační metriky pro danou třídu klasifikace a reprezentaci dat je zvýrazněn orámovaně.

Kromě klasifikačních metrik byla také měřena doba učení a doba klasifikace. Doba učení byla měřena jako celková doba trvání 10-násobné křížové validace a doba klasifikace byla měřena jako doba trvání predikce třídy jednoho příznakového vektoru. Pro klasifikátor SVM bylo využito knihovny `scikit-learn-intelex` viz kapitola 7.4.1, kromě datové reprezentace časově-frekvenční oblastí z důvodu nedostatku paměti[1]. Všechny klasifikátory implementované knihovnou `Keras/TensorFlow` byly trénovány s využitím GPU.

Při augmentaci byl parametr `generated_data_multiplier` nastaven na hodnotu 1, viz kapitola 7.3. V případě klasifikace využitím augmentované trénovací sady byla trénovací sada tvořena z jedné poloviny reálných naměřených dat a z druhé

[1] Klasifikace klasifikátorem SVM pro data reprezentována časově-frekvenční oblastí probíhala s využitím pouze jednoho jádra procesoru, kdežto u ostatních reprezentací bylo využito všech 8 jader.

poloviny dat vygenerovaných augmentační metodou. V tabulkách porovnání augmentačních metrik jednotlivých metod je nejlepší výsledek zvýrazněn **tučně**. Pro data reprezentována časově frekvenční oblastí nebyla počítána metrika FID z důvodu nedostatku operační paměti[2].

8.1 Přehled výsledků

V této sekci je proveden přehled výsledků augmentačních metrik, klasifikačních metrik a dob trvání trénování a klasifikace jednotlivých klasifikátorů pro všechny reprezentace příznakových vektorů. Popis a výpočet augmentačních metrik zobrazených v tabulkách 8.1, 8.4, 8.7, 8.10, 8.13 a 8.16 byl provedený v kapitole 7.3.4. Popis klasifikačních metrik zobrazených v tabulkách 8.2, 8.5, 8.8, 8.11, 8.14 a 8.17 byl provedený v kapitole 7.4.3.

8.1.1 Časová řada

Tabulka 8.1 obsahuje výsledky metrik pro augmentační metody implementované pro příznakové vektory reprezentované časovou řadou při provádění binární klasifikace. V tabulce 8.2 jsou pak prezentovány výsledky klasifikačních metrik vyjádřených v procentech pro různé kombinace klasifikátorů a augmentačních metod použitých na příznakové vektory reprezentované časovou řadou při provádění binární klasifikace. Vizuální zobrazení výsledků metriky `accuracy` z tabulky 8.2 je znázorněno v grafu 8.1. V tabulce 8.3 jsou uvedeny časy trvání trénování 10-násobné křížové validace a časy klasifikace jednoho příznakového vektoru pro jednotlivé klasifikátory a jejich kombinace s augmentačními metodami. Podobně jsou v tabulkách 8.4, 8.5, grafu 8.2 a tabulce 8.6 prezentovány výsledky pro vícetřídní klasifikaci příznakových vektorů reprezentovaných časovou řadou.

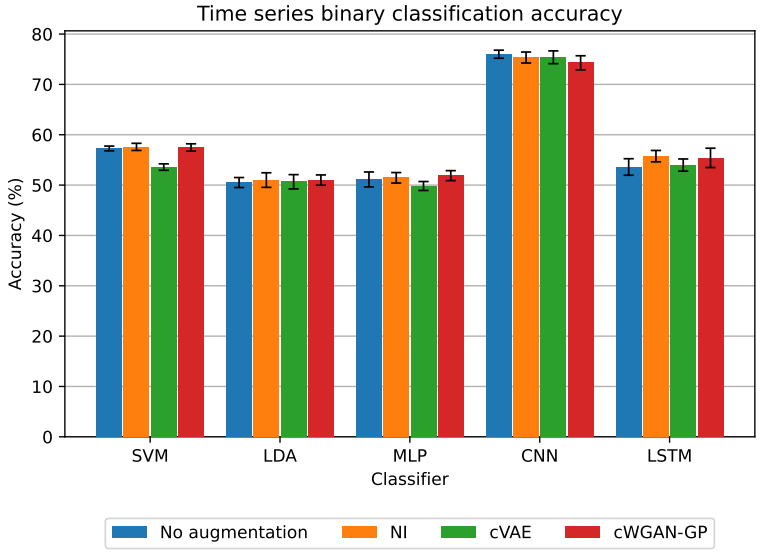
Metoda	FID	SNR	RMSE	CC
NI	**3243.64**	4.207	1.41414	0.194
cVAE	9029.46	**15.772**	**1.41045**	0.489
cWGAN-GP	10900.1	2.953	1.41435	**0.492**

Tabulka 8.1: Výsledné metriky augmentačních metod pro vstupní data reprezentována časovou řadou a provedení binární klasifikace

[2] Pro výpočet FID je počítána kovarianční matice dat, viz rovnice 7.4. Pro data reprezentována časově frekvenční oblastí má matice rozměry 34500 × 34500 a samotná zabírá 8,86 GiB.

Metoda	Accuracy	Precision	Recall	F1 Score
SVM	57.28±0.47	57.28±0.47	57.26±0.47	57.24±0.47
NI SVM	**57.59±0.71**	**57.85±0.71**	**57.65±0.70**	**57.34±0.72**
cVAE SVM	53.58±0.64	53.63±0.66	53.52±0.64	53.22±0.66
cWGAN-GP SVM	57.47±0.73	57.55±0.75	57.42±0.73	57.26±0.73
LDA	50.51±0.99	50.53±0.99	50.52±0.99	50.49±0.98
NI LDA	51.00±1.45	51.02±1.46	51.02±1.45	50.95±1.45
cVAE LDA	50.66±1.42	50.68±1.43	50.68±1.42	50.64±1.43
cWGAN-GP LDA	**51.00±1.02**	**51.02±1.03**	**51.02±1.03**	**50.96±1.02**
MLP	51.12±1.49	51.16±1.55	51.13±1.52	50.55±1.90
NI MLP	51.45±1.04	51.50±1.05	51.47±1.04	51.17±1.18
cVAE MLP	49.82±0.89	49.78±0.92	49.77±0.86	49.24±0.65
cWGAN-GP MLP	**51.88±0.99**	**51.98±1.09**	**51.88±0.97**	**51.38±0.97**
CNN	**76.00±0.80**	**76.73±0.75**	**76.05±0.79**	**75.86±0.90**
NI CNN	75.34±1.09	76.69±0.67	75.41±1.07	75.05±1.30
cVAE CNN	75.39±1.27	75.72±1.31	75.42±1.28	75.33±1.28
cWGAN-GP CNN	74.29±1.40	75.01±1.13	74.30±1.44	74.10±1.56
LSTM	53.60±1.64	53.62±1.65	53.60±1.64	53.55±1.62
NI LSTM	**55.75±1.13**	**55.79±1.12**	**55.77±1.12**	**55.71±1.13**
cVAE LSTM	53.98±1.21	54.00±1.20	53.99±1.21	53.95±1.25
cWGAN-GP LSTM	55.41±1.92	55.45±1.89	55.43±1.90	55.34±1.96

Tabulka 8.2: Klasifikační výsledky pro vstupní data reprezentována časovou řadou a provedení binární klasifikace



Obrázek 8.1: Vizualizace klasifikačních výsledků `accuracy` pro vstupní data reprezentována časovou řadou a provedení binární klasifikace

Metoda	Doba učení(hh:mm:ss)	Doba klasifikace(ms)
SVM	00:00:58	163
NI SVM	00:03:03	16
cVAE SVM	00:02:52	10
cWGAN-GP SVM	00:02:42	11
LDA	00:02:19	1
NI LDA	00:10:05	1
cVAE LDA	00:10:01	1
cWGAN-GP LDA	00:10:08	1
MLP	00:03:52	120
NI MLP	00:04:38	74
cVAE MLP	00:04:18	77
cWGAN-GP MLP	00:04:19	74
CNN	00:08:08	160
NI CNN	00:14:03	139
cVAE CNN	00:16:26	135
cWGAN-GP CNN	00:14:11	136
LSTM	00:08:44	843
NI LSTM	00:10:15	841
cVAE LSTM	00:09:49	834
cWGAN-GP LSTM	00:09:52	868

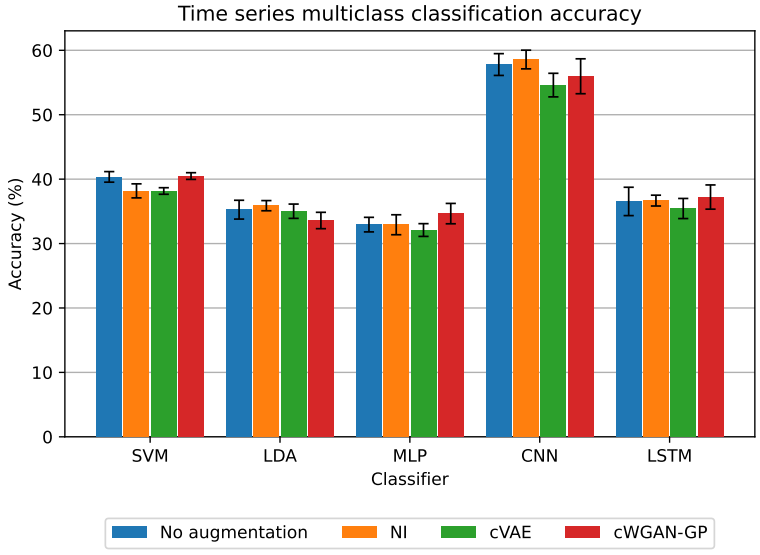
Tabulka 8.3: Porovnání doby učení a doby klasifikace jednotlivých metod pro vstupní data reprezentována časovou řadou a provedení binární klasifikace. Doba učení reprezentuje dobu trvání 10-násobné křížové validace. Doba klasifikace byla měřena na klasifikaci pouze jednoho vstupního vzorku

Metoda	FID	SNR	RMSE	CC
NI	**3303.68**	4.206	1.41461	0.202
cVAE	8869.08	**14.513**	1.41593	0.442
cWGAN-GP	10869.6	3.643	**1.4143**	**0.503**

Tabulka 8.4: Výsledné metriky augmentačních metod pro vstupní data reprezentována časovou řadou a provedení vícetřídní klasifikace

Metoda	Accuracy	Precision	Recall	F1 Score
SVM	40.35±0.82	**40.43±0.82**	**39.82±0.79**	**39.48±0.80**
NI SVM	38.18±1.09	39.56±1.08	38.36±1.10	38.21±1.10
cVAE SVM	38.16±0.51	36.88±0.66	37.04±0.50	36.15±0.53
cWGAN-GP SVM	**40.48±0.52**	39.30±0.95	39.24±0.56	37.97±0.58
LDA	35.26±1.46	35.34±1.45	35.32±1.47	35.20±1.42
NI LDA	**35.88±0.79**	**35.86±0.79**	**35.91±0.72**	**35.81±0.77**
cVAE LDA	35.02±1.11	35.04±1.09	35.06±1.06	34.96±1.08
cWGAN-GP LDA	33.58±1.27	33.68±1.27	33.64±1.33	33.53±1.29
MLP	32.94±1.13	33.03±1.21	33.00±1.21	32.49±1.03
NI MLP	32.92±1.55	33.05±1.55	32.92±1.52	32.85±1.54
cVAE MLP	32.10±0.99	32.01±0.90	31.94±1.07	31.70±1.15
cWGAN-GP MLP	**34.65±1.58**	**34.37±0.91**	**34.30±1.16**	**33.29±1.74**
CNN	57.79±1.69	59.36±2.47	57.60±1.84	57.17±2.08
NI CNN	**58.57±1.45**	**60.32±1.48**	**58.43±1.48**	**57.99±1.72**
cVAE CNN	54.60±1.83	55.39±1.50	54.39±1.66	53.98±2.06
cWGAN-GP CNN	55.97±2.71	58.28±2.28	55.77±2.60	54.87±3.65
LSTM	36.54±2.20	36.59±2.25	36.47±2.22	36.39±2.22
NI LSTM	36.67±0.84	36.76±0.78	36.70±0.79	36.50±0.81
cVAE LSTM	35.44±1.56	35.45±1.52	35.37±1.60	35.31±1.58
cWGAN-GP LSTM	**37.22±1.88**	**37.30±1.96**	**37.12±1.89**	**36.92±1.93**

Tabulka 8.5: Klasifikační výsledky pro vstupní data reprezentována časovou řadou a provedení vícetřídní klasifikace



Obrázek 8.2: Vizualizace klasifikačních výsledků `accuracy` pro vstupní data reprezentována časovou řadou a provedení vícetřídní klasifikace

Metoda	Doba učení(hh:mm:ss)	Doba klasifikace(ms)
SVM	00:00:55	190
NI SVM	00:03:00	24
cVAE SVM	00:02:38	15
cWGAN-GP SVM	00:02:56	20
LDA	00:01:23	2
NI LDA	00:06:30	1
cVAE LDA	00:06:25	1
cWGAN-GP LDA	00:06:25	1
MLP	00:03:34	127
NI MLP	00:04:18	74
cVAE MLP	00:04:08	73
cWGAN-GP MLP	00:03:55	75
CNN	00:08:35	153
NI CNN	00:15:11	143
cVAE CNN	00:15:18	133
cWGAN-GP CNN	00:10:09	135
LSTM	00:09:06	961
NI LSTM	00:09:49	888
cVAE LSTM	00:09:42	841
cWGAN-GP LSTM	00:09:24	873

Tabulka 8.6: Porovnání doby učení a doby klasifikace jednotlivých metod pro vstupní data reprezentována časovou řadou a provedení vícetřídní klasifikace. Doba učení reprezentuje dobu trvání 10-násobné křížové validace. Doba klasifikace byla měřena na klasifikaci pouze jednoho vstupního vzorku

8.1.2 Frekvenční spektrum

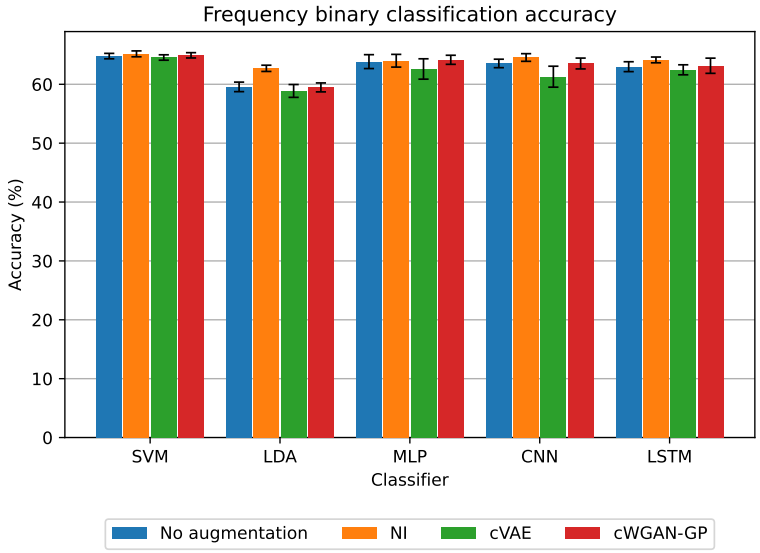
Tabulka 8.7 uvádí výsledky metrik pro augmentační metody použité na příznakové vektory reprezentované frekvenčním spektrem při provádění binární klasifikace. Tabulka 8.8 pak prezentuje výsledky klasifikačních metrik vyjádřených v procentech pro různé kombinace klasifikátorů a augmentačních metod použitých na příznakové vektory reprezentované frekvenčním spektrem při provádění binární klasifikace. Graf 8.3 vizualizuje výsledky metriky `accuracy` z tabulky 8.8. Tabulka 8.9 obsahuje informace o době trvání trénování 10-násobné křížové validace a době klasifikace jednoho příznakového vektoru pro jednotlivé klasifikátory a jejich kombinace s augmentačními metodami. Podobně jsou v tabulkách 8.10, 8.11, grafu 8.4 a tabulce 8.12 prezentovány výsledky pro vícetřídní klasifikaci příznakových vektorů reprezentovaných frekvenčním spektrem.

Metoda	FID	SNR	RMSE	CC
NI	146.959	3.234	1.41416	0.085
cVAE	**88.063**	**27.25**	1.41159	**0.194**
cWGAN-GP	151.276	11.651	**1.41107**	0.173

Tabulka 8.7: Výsledné metriky augmentačních metod pro vstupní data reprezentována jako frekvenční spektrum a provedení binární klasifikace

Metoda	Accuracy	Precision	Recall	F1 Score
SVM	64.79±0.46	64.78±0.46	64.77±0.46	64.76±0.46
NI SVM	**65.17±0.50**	**65.18±0.50**	**65.14±0.51**	**65.13±0.51**
cVAE SVM	64.55±0.46	64.56±0.46	64.56±0.46	64.55±0.46
cWGAN-GP SVM	64.92±0.45	64.93±0.44	64.93±0.44	64.92±0.44
LDA	59.56±0.81	59.71±0.81	59.43±0.82	59.20±0.86
NI LDA	**62.71±0.53**	**63.25±0.59**	**62.54±0.52**	**62.13±0.52**
cVAE LDA	58.87±1.09	58.92±1.10	58.77±1.09	58.63±1.11
cWGAN-GP LDA	59.47±0.76	59.70±0.81	59.33±0.75	59.01±0.75
MLP	63.85±1.18	64.29±1.00	63.89±1.18	63.59±1.38
NI MLP	64.00±1.08	64.42±1.14	64.04±1.11	**63.77±1.14**
cVAE MLP	62.60±1.73	62.76±1.73	62.54±1.74	62.40±1.81
cWGAN-GP MLP	**64.16±0.77**	**64.94±1.11**	**64.17±0.75**	63.71±0.79
CNN	63.54±0.72	63.68±0.72	63.56±0.73	63.47±0.75
NI CNN	**64.55±0.65**	**64.64±0.67**	**64.54±0.67**	**64.47±0.68**
cVAE CNN	61.29±1.78	61.95±1.76	61.29±1.85	60.71±2.12
cWGAN-GP CNN	63.53±0.92	64.00±1.00	63.62±0.92	63.31±0.99
LSTM	62.99±0.84	63.12±0.80	62.95±0.88	62.84±0.94
NI LSTM	**64.14±0.49**	**64.40±0.62**	**64.05±0.48**	**63.88±0.50**
cVAE LSTM	62.46±0.85	62.56±0.91	62.42±0.81	62.34±0.77
cWGAN-GP LSTM	63.15±1.29	63.26±1.26	63.18±1.28	63.10±1.31

Tabulka 8.8: Klasifikační výsledky pro vstupní data reprezentována jako frekvenční spektrum a provedení binární klasifikace



Obrázek 8.3: Vizualizace klasifikačních výsledků `accuracy` pro vstupní data reprezentována jako frekvenční spektrum a provedení binární klasifikace

Metoda	Doba učení(hh:mm:ss)	Doba klasifikace(ms)
SVM	00:00:09	212
NI SVM	00:00:21	1
cVAE SVM	00:00:21	1
cWGAN-GP SVM	00:00:21	1
LDA	00:00:02	1
NI LDA	00:00:03	1
cVAE LDA	00:00:03	1
cWGAN-GP LDA	00:00:03	1
MLP	00:04:03	119
NI MLP	00:04:05	74
cVAE MLP	00:06:07	73
cWGAN-GP MLP	00:04:55	73
CNN	00:08:07	158
NI CNN	00:11:34	137
cVAE CNN	00:18:40	132
cWGAN-GP CNN	00:12:49	132
LSTM	00:09:24	835
NI LSTM	00:10:22	845
cVAE LSTM	00:12:48	839
cWGAN-GP LSTM	00:11:53	834

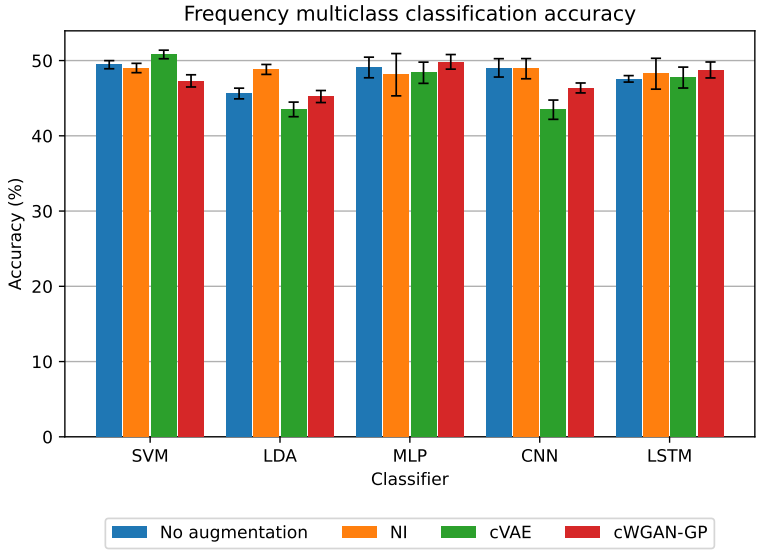
Tabulka 8.9: Porovnání doby učení a doby klasifikace jednotlivých metod pro vstupní data reprezentována jako frekvenční spektrum a provedení binární klasifikace. Doba učení reprezentuje dobu trvání 10-násobné křížové validace. Doba klasifikace byla měřena na klasifikaci pouze jednoho vstupního vzorku

Metoda	FID	SNR	RMSE	CC
NI	148.893	3.248	1.41391	0.145
cVAE	**84.142**	**23.527**	**1.41121**	0.207
cWGAN-GP	153.797	18.413	1.4326	**0.265**

Tabulka 8.10: Výsledné metriky augmentačních metod pro vstupní data reprezentována jako frekvenční spektrum a provedení vícetřídní klasifikace

Metoda	Accuracy	Precision	Recall	F1 Score
SVM	49.45±0.54	49.72±0.62	49.11±0.47	49.02±0.45
NI SVM	49.01±0.62	49.19±0.64	48.61±0.63	48.47±0.66
cVAE SVM	**50.81±0.57**	**51.01±0.54**	**50.79±0.55**	**50.76±0.57**
cWGAN-GP SVM	47.30±0.81	47.71±1.07	46.78±0.80	46.28±0.85
LDA	45.62±0.71	45.78±0.78	45.63±0.70	45.46±0.72
NI LDA	**48.82±0.66**	**48.93±0.62**	**48.73±0.64**	**48.75±0.65**
cVAE LDA	43.51±0.97	43.57±0.97	43.52±0.96	43.39±0.97
cWGAN-GP LDA	45.22±0.80	45.40±0.75	45.21±0.81	45.10±0.79
MLP	49.08±1.37	50.24±1.23	48.93±1.60	48.33±2.13
NI MLP	48.12±2.81	49.56±1.73	48.18±2.50	47.13±3.65
cVAE MLP	48.38±1.41	48.53±1.40	48.23±1.45	47.72±1.81
cWGAN-GP MLP	**49.83±0.97**	**51.25±2.35**	**50.00±1.31**	**49.06±1.21**
CNN	**49.03±1.22**	**49.70±1.14**	48.59±1.33	48.03±1.73
NI CNN	48.92±1.34	49.18±1.35	**48.65±1.28**	**48.40±1.48**
cVAE CNN	43.47±1.28	46.18±1.74	43.05±1.08	40.37±1.38
cWGAN-GP CNN	46.36±0.66	46.99±1.39	46.25±0.92	45.34±1.03
LSTM	47.57±0.43	47.96±0.49	47.47±0.59	47.06±0.73
NI LSTM	48.25±2.05	48.56±1.94	48.09±2.03	47.87±2.03
cVAE LSTM	47.74±1.39	47.78±1.57	47.81±1.48	47.60±1.53
cWGAN-GP LSTM	**48.75±1.06**	**48.88±1.26**	**48.69±1.18**	**48.47±1.17**

Tabulka 8.11: Klasifikační výsledky pro vstupní data reprezentována jako frekvenční spektrum a provedení vícetřídní klasifikace



Obrázek 8.4: Vizualizace klasifikačních výsledků `accuracy` pro vstupní data reprezentována jako frekvenční spektrum a provedení vícetřídní klasifikace

Metoda	Doba učení(hh:mm:ss)	Doba klasifikace(ms)
SVM	00:00:08	157
NI SVM	00:00:22	2
cVAE SVM	00:00:22	2
cWGAN-GP SVM	00:00:22	2
LDA	00:00:02	1
NI LDA	00:00:03	1
cVAE LDA	00:00:03	1
cWGAN-GP LDA	00:00:03	1
MLP	00:03:44	120
NI MLP	00:03:50	74
cVAE MLP	00:06:05	73
cWGAN-GP MLP	00:04:20	71
CNN	00:08:40	154
NI CNN	00:12:20	137
cVAE CNN	00:19:09	132
cWGAN-GP CNN	00:12:00	131
LSTM	00:09:23	957
NI LSTM	00:10:27	841
cVAE LSTM	00:12:05	841
cWGAN-GP LSTM	00:11:04	837

Tabulka 8.12: Porovnání doby učení a doby klasifikace jednotlivých metod pro vstupní data reprezentována jako frekvenční spektrum a provedení vícetřídní klasifikace. Doba učení reprezentuje dobu trvání 10-násobné křížové validace. Doba klasifikace byla měřena na klasifikaci pouze jednoho vstupního vzorku

8.1.3 Časově-frekvenční spektrum

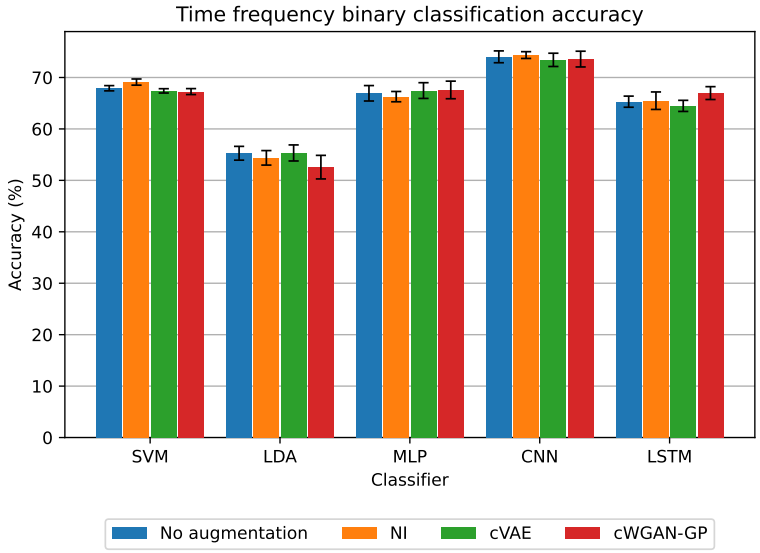
Tabulka 8.13 zobrazuje výsledky metrik pro augmentační metody použité na příznakové vektory reprezentované časově-frekvenční oblastí při provádění binární klasifikace. Tabulka 8.14 pak prezentuje výsledky klasifikačních metrik vyjádřených v procentech pro různé kombinace klasifikátorů a augmentačních metod použitých na příznakové vektory reprezentované časově-frekvenční oblastí při provádění binární klasifikace. Graf 8.5 vizuálně zobrazuje výsledky metriky `accuracy` z tabulky 8.14. V tabulce 8.15 jsou uvedeny časy trvání trénování 10-násobné křížové validace a časy klasifikace jednoho příznakového vektoru pro jednotlivé klasifikátory a jejich kombinace s augmentačními metodami. Obdobně jsou v tabulkách 8.16, 8.17, grafu 8.6 a tabulce 8.18 prezentovány výsledky pro vícetřídní klasifikaci příznakových vektorů reprezentovaných časově-frekvenční oblastí.

Metoda	SNR	RMSE	CC
NI	3.739	1.41512	0.068
cVAE	**23.958**	1.40526	**0.211**
cWGAN-GP	7.895	**1.37666**	0.175

Tabulka 8.13: Výsledné metriky augmentačních metod pro vstupní data reprezentována jako časově frekvenční spektrum a provedení binární klasifikace

Metoda	Accuracy	Precision	Recall	F1 Score
SVM	67.91±0.51	67.92±0.51	67.90±0.51	67.90±0.51
NI SVM	**69.11±0.60**	**69.20±0.59**	**69.14±0.60**	**69.09±0.60**
cVAE SVM	67.40±0.42	67.46±0.42	67.38±0.41	67.35±0.42
cWGAN-GP SVM	67.26±0.58	67.30±0.59	67.24±0.58	67.23±0.58
LDA	55.27±1.34	55.31±1.35	55.29±1.34	55.23±1.34
NI LDA	54.38±1.41	54.42±1.42	54.40±1.41	54.34±1.41
cVAE LDA	**55.33±1.56**	**55.33±1.56**	**55.33±1.56**	**55.31±1.56**
cWGAN-GP LDA	52.57±2.29	52.58±2.29	52.57±2.29	52.57±2.29
MLP	66.93±1.50	67.44±1.44	66.93±1.54	66.68±1.70
NI MLP	66.28±1.00	66.94±1.16	66.33±1.00	66.00±1.06
cVAE MLP	67.46±1.53	67.78±1.36	67.46±1.56	67.30±1.71
cWGAN-GP MLP	**67.58±1.71**	**68.08±1.56**	**67.57±1.75**	**67.33±1.92**
CNN	74.02±1.15	74.50±1.09	74.07±1.15	73.92±1.19
NI CNN	**74.36±0.66**	**74.85±0.72**	**74.40±0.66**	**74.25±0.69**
cVAE CNN	73.43±1.29	73.98±0.81	73.46±1.26	73.28±1.46
cWGAN-GP CNN	73.58±1.53	74.72±1.60	73.64±1.52	73.30±1.65
LSTM	65.30±1.08	65.43±1.11	65.31±1.08	65.24±1.07
NI LSTM	65.49±1.70	65.58±1.71	65.51±1.71	65.45±1.71
cVAE LSTM	64.48±1.08	64.61±1.07	64.50±1.08	64.42±1.11
cWGAN-GP LSTM	**66.97±1.25**	**67.21±1.19**	**66.98±1.25**	**66.86±1.30**

Tabulka 8.14: Klasifikační výsledky pro vstupní data reprezentována jako časově frekvenční spektrum a provedení binární klasifikace



Obrázek 8.5: Vizualizace klasifikačních výsledků `accuracy` pro vstupní data reprezentována jako časově frekvenční spektrum a provedení binární klasifikace

Metoda	Doba učení(hh:mm:ss)	Doba klasifikace(ms)
SVM	09:04:28	278
NI SVM	40:21:26	539
cVAE SVM	22:50:58	337
cWGAN-GP SVM	38:36:10	570
LDA	00:09:43	1
NI LDA	00:25:47	1
cVAE LDA	00:25:38	1
cWGAN-GP LDA	00:25:22	1
MLP	00:06:04	126
NI MLP	00:07:10	85
cVAE MLP	00:06:54	80
cWGAN-GP MLP	00:06:53	83
CNN	00:14:35	138
NI CNN	00:24:29	106
cVAE CNN	00:20:42	106
cWGAN-GP CNN	00:21:54	104
LSTM	00:14:24	1123
NI LSTM	00:16:24	1122
cVAE LSTM	00:13:42	1114
cWGAN-GP LSTM	00:14:00	1270

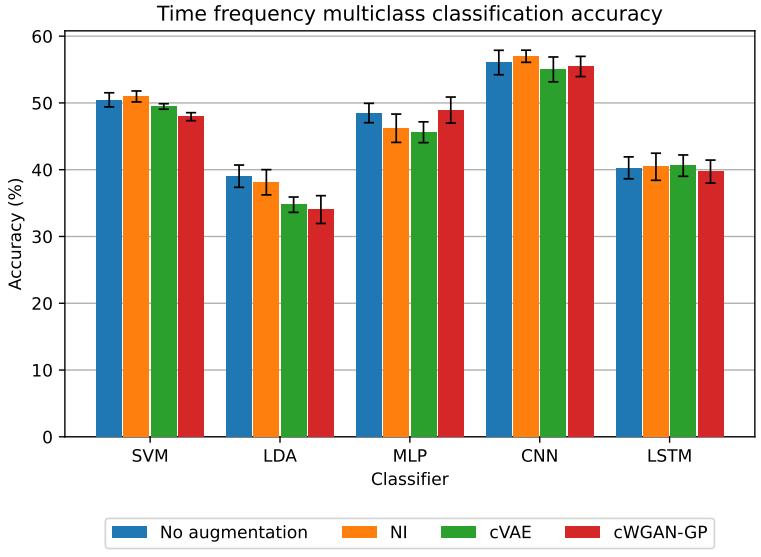
Tabulka 8.15: Porovnání doby učení a doby klasifikace jednotlivých metod pro vstupní data reprezentována jako časově frekvenční spektrum a provedení binární klasifikace. Doba učení reprezentuje dobu trvání 10-násobné křížové validace. Doba klasifikace byla měřena na klasifikaci pouze jednoho vstupního vzorku

Metoda	SNR	RMSE	CC
NI	3.711	1.41177	0.072
cVAE	**24.332**	1.40101	**0.263**
cWGAN-GP	5.704	**1.3565**	0.227

Tabulka 8.16: Výsledné metriky augmentačních metod pro vstupní data reprezentována jako časově frekvenční spektrum a provedení vícetřídní klasifikace

Metoda	Accuracy	Precision	Recall	F1 Score
SVM	50.46±1.06	50.36±1.08	49.95±1.08	49.83±1.16
NI SVM	**50.97±0.82**	**51.22±0.84**	**50.83±0.83**	**50.85±0.83**
cVAE SVM	49.47±0.41	49.04±0.41	48.91±0.39	48.86±0.39
cWGAN-GP SVM	47.94±0.60	48.60±0.68	47.20±0.62	45.82±1.04
LDA	**39.03±1.67**	**38.92±1.65**	**38.93±1.72**	**38.86±1.67**
NI LDA	38.12±1.89	38.20±1.96	38.14±1.95	38.04±1.91
cVAE LDA	34.76±1.15	35.39±1.20	35.00±1.17	34.71±1.16
cWGAN-GP LDA	34.03±2.08	33.86±2.14	33.82±2.11	33.78±2.10
MLP	48.49±1.45	49.75±1.39	48.33±1.46	**47.01±2.67**
NI MLP	46.21±2.12	47.18±2.38	46.06±1.91	44.93±1.89
cVAE MLP	45.61±1.56	47.52±2.93	45.62±1.84	44.44±2.12
cWGAN-GP MLP	**48.93±1.95**	**50.59±2.30**	**48.55±1.85**	46.98±2.94
CNN	56.05±1.83	57.78±1.46	56.18±1.78	56.03±1.88
NI CNN	**56.99±0.91**	**58.17±0.94**	**56.93±0.91**	**56.92±0.88**
cVAE CNN	55.02±1.86	56.53±1.80	55.12±1.92	54.80±2.10
cWGAN-GP CNN	55.46±1.51	56.17±1.48	55.21±1.49	55.07±1.58
LSTM	40.28±1.64	40.46±1.62	40.08±1.58	39.28±2.17
NI LSTM	40.44±2.03	**40.72±2.01**	**40.26±2.04**	**40.10±2.10**
cVAE LSTM	**40.61±1.60**	40.29±1.87	40.21±1.55	39.28±2.07
cWGAN-GP LSTM	39.72±1.72	39.96±1.74	39.53±1.78	39.37±1.83

Tabulka 8.17: Klasifikační výsledky pro vstupní data reprezentována jako časově frekvenční spektrum a provedení vícetřídní klasifikace



Obrázek 8.6: Vizualizace klasifikačních výsledků `accuracy` pro vstupní data reprezentována jako časově frekvenční spektrum a provedení vícetřídní klasifikace

Metoda	Doba učení(hh:mm:ss)	Doba klasifikace(ms)
SVM	06:53:45	259
NI SVM	37:32:17	486
cVAE SVM	19:50:02	369
cWGAN-GP SVM	29:59:59	555
LDA	00:08:22	2
NI LDA	00:14:57	1
cVAE LDA	00:15:05	1
cWGAN-GP LDA	00:32:24	1
MLP	00:05:37	222
NI MLP	00:06:30	79
cVAE MLP	00:06:10	79
cWGAN-GP MLP	00:06:38	80
CNN	00:11:36	164
NI CNN	00:26:08	108
cVAE CNN	00:15:51	109
cWGAN-GP CNN	00:18:47	107
LSTM	00:14:37	1157
NI LSTM	00:15:18	1121
cVAE LSTM	00:13:19	1105
cWGAN-GP LSTM	00:13:40	1103

Tabulka 8.18: Porovnání doby učení a doby klasifikace jednotlivých metod pro vstupní data reprezentována jako časově frekvenční spektrum a provedení vícetřídní klasifikace. Doba učení reprezentuje dobu trvání 10-násobné křížové validace. Doba klasifikace byla měřena na klasifikaci pouze jednoho vstupního vzorku

8.2 Diskuze dosažených výsledků

Byl dosažen nejlepší klasifikační výsledek (76,00±0,80% přesnost) při použití klasifikátoru CNN pro binární klasifikaci dat reprezentovaných časovou řadou, bez použití augmentace. Nejlepší výsledek (65,17±0,50%) binární klasifikace dat reprezentovaných frekvenčním spektrem byl dosažen pomocí klasifikátoru SVM na augmentované sadě metodou NI. Pro data reprezentována časově-frekvenční oblastí byl nejlepší výsledek binární klasifikace (74,36±0,66%) dosažen s využitím klasifikátoru CNN a augmentační metody NI. Výsledky ostatních klasifikačních metrik jsou velmi podobné přesnosti klasifikace (`accuracy`), což naznačuje dobrou funkčnost jednotlivých modelů (s ohledem na vyváženou reprezentaci jednotlivých klasifikačních tříd, byl tento výsledek očekáván). Klasifikátor CNN poskytuje nejlepší výsledky klasifikace ve 4 z 6 kombinací datové reprezentace a klasifikačních tříd. Zároveň klasifikátor CNN poskytuje statisticky významně lepší výsledky než ostatní klasifikátory (na základě McNemarova testu; $p < 0,01$). Výsledky také představují

zlepšení oproti výsledkům dosažených v pracích Mochury a Saleha, viz tabulka 3.2 a 3.4. Vysoké rozdíly mezi jednotlivými klasifikátory jsou pravděpodobně způsobeny vysokou dimenzionalitou jednotlivých příznakový vektorů (především časové řady a časově frekvenční oblasti).

Zajímavé je porovnání výsledků binární a vícetřídní klasifikace, kdy nejlepšího výsledku (58.57±1.45%) vícetřídní klasifikace bylo dosaženo klasifikátorem CNN na datech reprezentovaných časovou řadou a augmentovaných metodou NI. Celkově jsou výsledky vícetřídní klasifikace podstatně horší než výsledky binární klasifikace, většina klasifikátorů při vícetřídní klasifikaci nedosáhla ani přesnosti 50%. Tento fakt naznačuje tomu, že klasifikátory byly schopny dokázat určitou predikční schopnost pro rozlišení mezi klidovým stavem subjektu a pohybovým stavem, ovšem už nejsou schopny rozlišit zda se jednalo o pohyb levou rukou nebo o pohyb pravou rukou.

Dopad augmentačních metod na výsledky klasifikace není v této práci příliš pozitivní. Globálně došlo naopak ke zhoršení výsledků klasifikační přesnosti a to v průměru o -0,13%. Ovšem ve většině případů poskytovala alespoň jedna z augmentačních metod určité zlepšení přesnosti klasifikace oproti klasifikaci bez použití augmentace. Při porovnání jednotlivých augmentačních metod dosáhla v průměru metoda NI zlepšení o 0,29%, metoda cVAE zhoršení o -1,45% a cWGAN-GP zlepšení o 0,76%. Nízké hodnoty zlepšení poměrně odpovídají výsledkům poskytnutým v rešerši augmentační literatury [LLM20], viz kapitola 3.1.5.2. V případě metody cWGAN-GP je věrohodnost výsledku poměrně diskutabilní, jelikož jak bylo popisováno v kapitole 7.3.3 negeneruje metoda realistická data, viz obrázek 7.6. Zajímavý je také dopad metody cVAE, která na základě vizuální inspekce a vyhodnocených augmentačních metrik, poskytuje slušné reprezentativní příznakové vektory, ovšem poskytuje v průměru nejhorší klasifikační zlepšení. Jedním z možných vysvětlení může být, že augmentační metody představují jednu z forem regularizace, a je tedy možné, že klasifikátory produkovaly jednodušší rozhodovací hranice za účelem lepší generalizační schopnosti, která typicky vede k určitému snížení klasifikační přesnosti.

Jedním z důležitých ukazatelů pro použitelnost BCI sytému je také doba trvání klasifikace. Čím delší bude doba klasifikace, tím delší bude odezva BCI systému na požadavek provedení akce (v tomto případě vykonání pohybu). Při příliš dlouhé odezvě nebude pro subjekt komfortní systém používat. Doba učení klasifikátoru samozřejmě hraje také důležitou roli, ovšem učení je možné provést offline, a tudíž není tak důležitá jako doba klasifikace, kterou je nutné dělat v reálném čase.

Je poměrně **důležité** si uvědomit, že přestože bylo dosaženo poměrně slušného klasifikačního výsledku (76,00±0,80%) (obzvláště pro inter-subjekt model při použití klasifikace single-trials), byla data měřena na poměrně malé nerperezentativní populaci (29 *zdravých* subjektů ve věku od 19 do 25 let). Cílovým uživatelem BCI

systému by ovšem byl subjekt, který trpí nějakou chorobou či paralýzou, a je tedy otázkou, zda je fyzikální podstata měřeného EEG signálu stejná jako u zdravých subjektů. Pro získání robustnějších výsledků by tedy bylo nutné naměřit větší množství dat od různých subjektů [Pad+19].

V práci byla většina hyperparametrů klasifikátorů a augmentačních metod nastavena empiricky nebo na základě podobných studií. Vyšší přesnosti klasifikace by teoreticky bylo možné dosáhnout lepším prohledáním prostoru hyperparametrů, knihovny `Scikit-learn` i `Keras` poskytují API pro automatické vyhodnocení klasifikátorů s různým nastavením. Další možností pro budoucí práce by mohlo být prozkoumání redukce dimenzionality příznakových vektorů. Teoreticky se nabízí možnost použití kratšího úseku signálu, jelikož na základě obrázků 7.4 a 7.5 by možná stačilo pro dosažení stejných klasifikačních výsledků použít signál od 2 sekund před synchronizační značkou, místo 3,5 sekund použitých v této práci. Existují i jiné způsoby redukce dimenzionality, například lze využít metodu PCA nebo provést podvzorkování signálu. Určitě by také stálo za to prozkoumat dopad augmentačních metod na výsledky klasifikace s využitím většího počtu vygenerovaných dat. Dále se nabízí implementace intra-subjekt modelů, které by potenciálně pro konkrétní subjekty mohly poskytovat lepší klasifikační přesnosti. Vzhledem k problémům s trénováním klasifikátorů pomocí knihovny `Keras/TensorFlow` využitím GPU, viz kapitola 7.4.2, se také nabízí prozkoumání možnosti použití jiné knihovny např. `PyTorch`.

Závěr 9

Výsledkem této práce je konfigurovatelná konzolová aplikace, která umožňuje uživatelům provádět porovnání detekce pohybu z naměřených EEG dat pomocí několika různých klasifikátorů. Detekce pohybu byla provedena na existující datové kolekci, která vznikla jako součást dvou předchozích prací [Moc21] a [Sal22]. V rámci této práce byla datová sada rozšířena o další data z měření EEG signálu u pěti subjektů. Celkem datová sada obsahuje data z měření 29 zdravých subjektů ve věku od 19 do 25 let. Kromě toho byly v práci použity tři augmentační metody (NI, cVAE a cGAN) pro další rozšíření naměřených dat vygenerováním nových příznakových vektorů.

Detekce pohybu z naměřeného EEG signálu byla provedena pěti klasifikátory (LDA, SVM, MLP, LSTM a CNN) na třech různých reprezentacích vstupních příznakových vektorů (reprezentace časovou řadou, frekvenčním spektrem a časově-frekvenčním spektrem). Klasifikátory byly použity pro sestavení single-trial inter-subjekt modelu. Pro dosažení robustních výsledků bylo trénování klasifikátorů provedeno pomocí 10-násobné křížové-validace. Nejlepší přesnost (76.00±0.80%) byla dosažena použitím klasifikátoru CNN na vstupních příznakových vektorech reprezentovaných časovou řadou bez augmentace datové sady. Dosažený výsledek je srovnatelný s výsledky získanými v literatuře a zároveň přináší určité zlepšení ve srovnání s předchozími pracemi [Moc21] a [Sal22]. Podrobný přehled dosažených výsledků a jejich diskuse jsou uvedeny v kapitole 8.

Zdrojové kódy aplikace a naměřená data jsou volně dostupné ke stažení, viz kapitola A. Experimenty provedené v této práci jsou tak plně reprodukovatelné.

Uživatelská příručka A

Následující postup předpokládá systém s operačním systémem Windows. Na operačním systému Linux bude postup velmi podobný, ovšem je nutné brát na vědomí, že při vývoji projektu byl používán pouze systém Windows. Je tedy možné, že na OS Linux projekt nebude fungovat správně.

Před samotnou instalací a spuštěním projektu je nutná příprava pracovního adresáře (příprava zdrojových kódů a dat).

Zdrojové kódy projektu jsou volně dostupné na GitLab repositáři na URL adrese `https://gitlab.com/Dumby7/eeg-motion-detection`. Pracovní adresář se zdrojovými kódy projektu je tak možné získat následujícím způsobem:

```
C:\Workspace>git clone
    https://gitlab.com/Dumby7/eeg-motion-detection.git

C:\Workspace> cd eeg-motion-detection

C:\Workspace\eeg-motion-detection>
```

Je také možné zdrojové kódy projektu stáhnout přímo zabalené, například ve formátu `.zip`, kliknutím na tlačítko *Download* a výběrem příslušného formátu. Složku se zdrojovými kódy je pak nutné rozbalit na požadované místo a následně z terminálu operačního systému přepnout aktuální pracovní adresář na rozbalenou složku.

Data používaná při experimentech tohoto projektu je možné stáhnout z URL adresy `https://zenodo.org/record/7893847` . Zabalenou složku s daty je nutné rozbalit do pracovního adresáře projektu. Adresářová struktura projektu by pak měla vypadat následovně:

```
C:\Workspace\eeg-motion-detection>dir /b
.gitignore
config.ini
data
LICENSE
README.md
requirements.txt
src
```

```
tests
thesis
```

A.1 Instalace a spuštění

Instalaci závislostí projektu je možné udělat dvěma způsoby. Prvním způsobem je použitím správce pro Python balíčky *Conda*. Toto je doporučený způsob instalace v případě, že je požadováno využití CUDA GPU[1], pro trénování klasifikace Keras/-Tensorflow modelů, viz kapitola 7.4.2 na str. 78. Pokud systém nedisponuje CUDA kompatibilním GPU, nebo není požadováno trénování modelů na GPU, je možné použít druhý způsob instalace přímo pomocí programovacího jazyku Python.

Volitelně je možné nainstalovat software *Graphviz*, volně dostupný ke stažení z URL adresy `https://graphviz.org/download/`. Bez tohoto softwaru není program schopen kreslit grafy architektur implementovaných Keras modelů.

A.1.1 Instalace závislostí pomocí Conda

Nejprve je nutné stáhnout a nainstalovat správce balíčků Conda. Je možné použít distribuci *Anaconda* volně dostupná ke stažení z URL adresy `https://www.anaconda.com/download/` nebo distribuci *Miniconda* dostupná ke stažení z URL adresy `https://docs.conda.io/en/latest/miniconda.html`.

Po instalaci je potřeba vytvořit nové Conda prostředí (za účelem vyhnutí se konfliktu mezi závislostmi), v následujícím příkladu je používané prostředí s názvem *eeg*, možné použít jakýkoliv libovolný alias.

```
C:\Workspace\eeg-motion-detection>conda create -n eeg python=3.10
```

Vytvořené prostředí je nutné aktivovat.

```
C:\Workspace\eeg-motion-detection>conda activate eeg

(eeg) C:\Workspace\eeg-motion-detection>
```

Následně je nutné nainstalovat závislosti pro umožnění spouštění trénování Keras/Tensorflow modelů s využitím CUDA GPU. Tento krok je možné přeskočit, pokud tato vlastnost není vyžadována.

```
(eeg) C:\Workspace\eeg-motion-detection>conda install -n eeg -c
    conda-forge cudatoolkit=11.3 cudnn=8.1.0

(eeg) C:\Workspace\eeg-motion-detection>conda install -n eeg -c
    nvidia cuda-nvcc=11.3
```

[1] `https://www.tensorflow.org/install/pip#hardware_requirements`

Posledním krokem je instalace Python balíčků ze souboru `requirements.txt`.

```
(eeg) C:\Workspace\eeg-motion-detection>pip install -r
    requirements.txt
```

A.1.2 Instalace závislostí pomocí Python

Nejprve je nutné stáhnout a nainstalovat Python verze 3.10[2] dostupné na URL adrese `https://www.python.org/downloads/release/python-31011/`. Po instalaci je vhodné ověřit používání správné verze, například pokud již na systému byla nainstalována nějaká předchozí verze Pythonu. V následujícím postupu je tedy předpokládáno, že alias příkazu `python` používá právě verzi 3.10.11. Ověření je možno provést:

```
C:\Workspace\eeg-motion-detection>python --version
Python 3.10.11
```

Je nutné vytvořit nové Python virtuální prostředí (za účelem vyhnutí se konfliktu mezi závislostmi).

```
C:\Workspace\eeg-motion-detection>python -m venv venv
```

Aktivujte vytvořené virtuální prostředí.

```
C:\Workspace\eeg-motion-detection>venv\Scripts\activate

(venv) C:\Workspace\eeg-motion-detection>
```

Posledním krokem je instalace Python balíčků ze souboru `requirements.txt`.

```
(venv) C:\Workspace\eeg-motion-detection>pip install -r
    requirements.txt
```

A.1.3 Spuštění

Po úspěšné instalaci všech závislostí jedním ze dvou popsaných způsobů, je možné program spustit příkazem:

```
(eeg) C:\Workspace\eeg-motion-detection>python src/main.py
    [-f|--config_file <config_file_path>]
```

Program má jeden nepovinný argument, jímž je cesta ke konfiguračnímu souboru. V případě že není cesta k žádnému konfiguračnímu souboru použita, je použit konfigurační soubor `config.ini`. Možná konfigurace je popsána v následující sekci.

[2]Je možné, že pro tento projekt bude fungovat jakákoliv verze Pythonu 3.10 nebo vyšší, ovšem v rámci vývoje byla používána verze **3.10.11**.

A.2 Konfigurace

Konfigurace projektu je umožněna pomocí textového souboru formátu `INI`. Součástí projektu je soubor `config.ini`, ve kterém jsou nastavené základní hodnoty nastavení. Soubor obsahuje několik sekcí, které obsahují parametry týkající se dané sekce. Veškeré sekce i parametry jsou povinné.

- Augmentation
  - *method_names* – seznam názvů augmentačních metod, viz kapitola 7.3 na str. 70, oddělených čárkou. Možné hodnoty augmentačích metod jsou: `NI`, `cVAE`, `cWGAN-GP` a prázdný řetězec. Pokud není žádná hodnota zadána, bude program spuštěn bez jakékoliv augmentace. Pokud je požadováno porovnání klasifikace bez augmentace s ostatními metodami je nutné v seznamu na libovolné pozici použít prázdný řetězec (nebo jakékoliv bílé znaky) oddělené čárkou. Např. `  , cVAE`.
  - *generated_data_multiplier* – celé nebo desetinné číslo určující násobek reálných trénovacích dat, který bude použit k výpočtu počtu dat vytvořených augmentací, viz rovnice 7.1 na str. 70.
- Classification
  - *model_names* – seznam názvů klasifikátorů, viz kapitola 7.4 na str. 76, oddělených čárkou, které se mají použít pro klasifikaci eeg dat. Možné hodnoty jsou: `SVM`, `LDA`, `MLP`, `CNN`, `LSTM`. Pokud bude hodnota prázdná, nebude prováděna žádná klasifikace a dojde pouze k vyhodnocení augmentačních metod.
  - *use_pre_trained_models* – bool hodnota nastavující zda má program použít již předem natrénované modely. Může nabývat jednou z hodnot: `true`, `false`. Pokud je hodnota nastavena na hodnotu true budou použity všechny modely, i augmentační, které byly natrénovány v nějakém z předchozích běhů programu. U těchto modelů tedy nebude docházet k trénování, pouze k vyhodnocení.
  - *k_folds* – celočíselná hodnota reprezentující počet cross validačních iterací použitých při trénování klasifikátorů. Hodnota musí být alespoň 2.
- Metrics
  - *classification_metrics* – seznam názvů klasifikačních metrik, viz kapitola 7.4.3 na str. 81, oddělených čárkou, které bude program vypisovat do konzole nebo případně kreslit jejich grafy. Možné hodnoty jsou:

`accuracy`, `precision`, `recall`, `f1_score`, `auc` a `confusion_matrix`.

- *augmentation_metrics* – seznam názvů augmentačních metrik, viz kapitola 7.3.4 na str. 74, oddělených čárkou, které bude program vypisovat do konzole. Možné hodnoty jsou: `fid`, `snr`, `rmse` a `cc`.

- GPU
  - *use_gpu* – bool hodnota nastavující zda se má učení a klasifikace Keras/-Tensorflow modelů (augmentační: `cVAE`, `cWGAN-GP` klasifikační: `MLP`, `CNN`, `LSTM` provádět pomocí GPU místo CPU. Pro použití GPU je nutné na systému mít CUDA kompatibilní GPU a provést instalaci podle návadu v kapitole A.1.1 na str. 104. Může nabývat jednou z hodnot: `true`, `false`.

- Other
  - *classification_type* – hodnota indikující jaký typ klasifikace budou klasifikátory používat. Může nabývat jednou z hodnot: `binary`, pro binární klasifikaci ***Pohyb vs Klid***, nebo `multiclass` pro klasifikaci ***Pohyb levou rukou vs Pohyb pravou rukou vs Klid***. Jednotlivé třídy byly popsány v kapitole 7.2.4.2.
  - *data_representation* – hodnota indikující jaká reprezentace EEG dat se má použít pro klasifikaci. Může nabývat jednou z hodnot: `time_series` pro reprezentaci signálu časovou řadou, nebo `frequency` pro transformaci signálu do frekvenčního spektra, nebo `time_frequency` pro transformaci signálu na spektrogramy (časově frekvenční spektrum). Používané reprezentace dat byly popsány v kapitole 7.2.4.3.
  - *save_plots* – bool hodnota nastavující zda má program generovat a ukládat soubory s vizualizacemi. Může nabývat jednou z hodnot: `true`, `false`. Pokud je hodnota nastavena na `true` bude program generovat různé vizualizace do složky `images`.
  - *deterministic* – bool hodnota nastavující zda má program produkovat deterministické výsledky. Může nabývat jednou z hodnot: `true`, `false`. Pokud je hodnota nastavena na hodnotu `true` bude nastaven seed jednotlivým generátorům náhodných čísel a program bude při opakovaném spuštění se stejným nastavením produkovat stejné výsledky. Hodnotu je **nutné** nastavit na true, pro replikaci výsledných experimentů.
  - *save_load_preprocessed_data* – bool hodnota nastavující zda má docházet k načítání a ukládání již předzpracovaných dat. Může nabývat jednou

z hodnot: `true`, `false`. Pokud je hodnota nastavena na `true` tak při prvním spuštění souboru dojde k načtení dat ze složky `data`, načtená data jsou následně předzpracována a transformována do požadované reprezentace, viz kapitola 7.2.4 na str. 62. Výsledná $\boldsymbol{n}$ dimenzionální matice vstupních dat a jejich příznaků třídy jsou uloženy do složky `preprocessed_data`. Pokud je při dalším spuštění tato hodnota stále nastavena na hodnotu true, jsou data pro danou reprezentaci načítána přímo předzpracována a tudíž je přeskočen celý proces předzpracování.

Ukázka chyby duplicitního načítání dat ve skriptu Saleha

B

Nově měřená data mají následující formát názvu souboru:
`HR_<čas_měření>_<unikátní_pořadové_číslo>_<s|bez>_haptika_<strana_pohybu>`
chyba byla v následující části načítání souborů.

Zdrojový kód B.1: Ukázka chyby v načítání souborů ve skriptech [Sal22]

```
numbers = []
for file_names in files:
    values = file.split('_')
    value = values[2] + values[3]
    if value not in numbers:
        numbers.append(value)
    else:
        continue

    ... process files with unique number value
```

Myšlenka kódu byla taková, že pokud se již načítal nějaký soubor s unikátním pořadovým číslem tak už byl daný subjekt zpracován. Ovšem chyba je v tom, že není kontrolovaná pouze unikátní hodnota (hodnota na indexu 2 v poli values), ale zároveň i zda daný soubor odpovídá datům s nebo bez haptiky (hodnota na indexu 3 v poli values). Data pro každého pacienta byla tedy načítána dvakrát. Jednou v pořadí s haptikou→bez haptiky, podruhé v pořadí bez haptiky→s haptikou.

Bibliografie

[AAW18] ABDELFATTAH, Sherif M.; ABDELRAHMAN, Ghodai M.; WANG, Min. Augmenting The Size of EEG datasets Using Generative Adversarial Networks. In: *2018 International Joint Conference on Neural Networks (IJCNN)*. 2018, s. 1–6. Dostupné z DOI: 10.1109/IJCNN.2018.8489727.

[Ach+16] ACHARYA, Jayant N.; HANI, Abeer J.; THIRUMALA, Partha D.; TSUCHIDA, Tammy N. American Clinical Neurophysiology Society Guideline 3: A Proposal for Standard Montages to Be Used in Clinical EEG. *Journal of Clinical Neurophysiology*. 2016, roč. 33, č. 4. Dostupné také z: https://shorturl.at/gimHP.

[AG17] ANG, Kai Keng; GUAN, Cuntai. EEG-Based Strategies to Detect Motor Imagery for Control and Rehabilitation. *IEEE Transactions on Neural Systems and Rehabilitation Engineering*. 2017, roč. 25, č. 4, s. 392–401. Dostupné z DOI: 10.1109/TNSRE.2016.2646763.

[Bis06] BISHOP, Christopher M. *Pattern Recognition and Machine Learning*. New York: Springer, 2006. Dostupné také z: https://shorturl.at/hxz0Z.

[Coh19] COHEN, Michael X. A better way to define and describe Morlet wavelets for time-frequency analysis. *NeuroImage*. 2019, roč. 199, s. 81–86. ISSN 1053-8119. Dostupné z DOI: https://doi.org/10.1016/j.neuroimage.2019.05.048.

[Gul+17] GULRAJANI, Ishaan; AHMED, Faruk; ARJOVSKY, Martin; DUMOULIN, Vincent; COURVILLE, Aaron C. Improved Training of Wasserstein GANs. *CoRR*. 2017, roč. abs/1704.00028. Dostupné z arXiv: 1704.00028.

[He+21] HE, Chao; LIU, Jialu; ZHU, Yuesheng; DU, Wencai. Data Augmentation for Deep Neural Networks Model in EEG Classification Task: A Review. *Frontiers in Human Neuroscience*. 2021, roč. 15. ISSN 1662-5161. Dostupné z DOI: 10.3389/fnhum.2021.765525.

[HM01] HRAZDIRA, Ivo; MORNSTEIN, Vojtěch. *Lékařská biofyzika a přístrojová technika*. 1. vyd. Brno: Neptun, 2001. ISBN 80-902896-1-4.

[Igl+23] IGLESIAS, Guillermo; TALAVERA, Edgar; GONZÁLEZ-PRIETO, Ángel; MOZO, Alberto; GÓMEZ-CANAVAL, Sandra. Data Augmentation techniques in time series domain: a survey and taxonomy. *Neural Computing and Applications*. 2023, roč. 35, č. 14, s. 10123–10145. ISSN 1433-3058. Dostupné z DOI: 10.1007/s00521-023-08459-3.

[Krü+20] KRÜGER, Britta et al. Practice modality of motor sequences impacts the neural signature of motor imagery. *Scientific Reports*. 2020, roč. 10, č. 1, s. 19176. ISSN 2045-2322. Dostupné z DOI: 10.1038/s41598-020-76214-y.

[LLM20] LASHGARI, Elnaz; LIANG, Dehua; MAOZ, Uri. Data augmentation for deep-learning-based electroencephalography. *Journal of Neuroscience Methods*. 2020, roč. 346, s. 108885. ISSN 0165-0270. Dostupné z DOI: https://doi.org/10.1016/j.jneumeth.2020.108885.

[Law+18] LAWHERN, Vernon J et al. EEGNet: a compact convolutional neural network for EEG-based brain–computer interfaces. *Journal of Neural Engineering*. 2018, roč. 15, č. 5, s. 056013. Dostupné z DOI: 10.1088/1741-2552/aace8c.

[Laz+18] LAZAROU, Ioulietta; NIKOLOPOULOS, Spiros; PETRANTONAKIS, Panagiotis C.; KOMPATSIARIS, Ioannis; TSOLAKI, Magda. EEG-Based Brain–Computer Interfaces for Communication and Rehabilitation of People with Motor Impairment: A Novel Approach of the 21st Century. *Frontiers in Human Neuroscience*. 2018, roč. 12. ISSN 1662-5161. Dostupné z DOI: 10.3389/fnhum.2018.00014.

[Luc05] LUCK, S.J. *An Introduction to the Event-related Potential Technique*. MIT Press, 2005. Cognitive neuroscience. ISBN 9780262621960. Dostupné také z: https://books.google.cz/books?id=J%5C_QgAQAAIAAJ.

[Mau15] MAUTNER, Pavel. *Analýza a zpracování signálů*. 2015. Dostupné také z: https://www.kiv.zcu.cz/~mautner/Azs/.

[Moc21] MOCHURA, Pavel. *Detekce pohybu končetin z EEG signálu při cvičení na rehabilitačním robotovi*. Katedra informatiky a výpočetní techniky Západočeské univerzity v Plzni, 2021. Dostupné také z: https://dspace5.zcu.cz/bitstream/11025/45196/1/A19N0072P.pdf. Diplomová práce.

[Mul07] MULDER, Theo. Motor Imagery and Action Observation: Cognitive Tools for Rehabilitation. *Journal of Neural Transmission (Vienna)*. 2007, roč. 114, č. 10, s. 1265–1278. Dostupné z DOI: 10.1007/s00702-007-0763-z.

[Nak+14] NAKAYASHIKI, Kazuhiko; SAEKI, Masashi; TAKATA, Yasushi; HAYASHI, Yasuo; KONDO, Toshio. Modulation of event-related desynchronization during kinematic and kinetic hand movements. *Journal of NeuroEngineering and Rehabilitation*. 2014, roč. 11, s. 90. Dostupné z DOI: 10.1186/1743-0003-11-90.

[NA23] NAYAK, Chakrapani S.; ANILKUMAR, Anu C. EEG Normal Waveforms. *StatPearls*. 2023. Dostupné také z: https://www.ncbi.nlm.nih.gov/books/NBK539805/.

[NNS16] NUNEZ, Michael; NUNEZ, Paul; SRINIVASAN, Ramesh. Electroencephalography (EEG): neurophysics, experimental methods, and signal processing. In: 2016, s. 175–197. ISBN 9781482220971. Dostupné z DOI: 10.13140/RG.2.2.12706.63687.

[ON15] O'SHEA, Keiron; NASH, Ryan. An Introduction to Convolutional Neural Networks. *arXiv preprint arXiv:1511.08458*. 2015. Dostupné také z: https://arxiv.org/abs/1511.08458.

[Ola15] OLAH, Christopher. Understanding LSTM Networks. 2015. Dostupné také z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

[Pad+19] PADFIELD, Nicholas; ZABALZA, Jaime; ZHAO, Huihua; MASERO, Valerio; REN, Jinchang. EEG-Based Brain-Computer Interfaces Using Motor-Imagery: Techniques and Challenges. *Sensors*. 2019, roč. 19, č. 6, s. 1423. Dostupné z DOI: 10.3390/s19061423.

[PL99] PFURTSCHELLER, G.; LOPES DA SILVA, F.H. Event-related EEG/-MEG synchronization and desynchronization: basic principles. *Clinical Neurophysiology*. 1999, roč. 110, č. 11, s. 1842–1857. ISSN 1388-2457. Dostupné z DOI: https://doi.org/10.1016/S1388-2457(99)00141-8.

[Pfu+06] PFURTSCHELLER, Gert; BRUNNER, Clemens; SCHLÖGL, Alois; LOPES DA SILVA, Fernando H. Mu Rhythm (De)Synchronization and EEG Single-Trial Classification of Different Motor Imagery Tasks. *NeuroImage*. 2006, roč. 31, č. 1, s. 153–159. Dostupné z DOI: 10.1016/j.neuroimage.2005.12.003.

[Puk+17] PUKHOVA, Valentina; GORELOVA, Elizaveta; FERRINI, Gabriele; BURNASHEVA, Sakhaya. Time-frequency representation of signals by wavelet transform. 2017, s. 715–718. Dostupné z DOI: 10.1109/EIConRus.2017.7910658.

[Roy+19] ROY, Yannick et al. Deep learning-based electroencephalography analysis: a systematic review. *Journal of Neural Engineering*. 2019, roč. 16, č. 5, s. 051001. Dostupné z DOI: 10.1088/1741-2552/ab260c.

[ADA21] AL-SAEGH, Ali; DAWWD, Shefa A.; ABDUL-JABBAR, Jassim M. Deep learning for motor imagery EEG-based classification: A review. *Biomedical Signal Processing and Control*. 2021, roč. 63, s. 102172. ISSN 1746-8094. Dostupné z DOI: https://doi.org/10.1016/j.bspc.2020.102172.

[Sal22] SALEH, Josef Yassin. *Design of movement detector of measured EEG data*. Katedra informatiky a výpočetní techniky Západočeské univerzity v Plzni, 2022. Dostupné také z: https://dspace5.zcu.cz/bitstream/11025/49544/1/Josef_Yassin_Saleh_BT_2022.pdf. Bakalářská práce.

[Tan+23] TAN, Xiyue; WANG, Dan; CHEN, Jiaming; XU, Meng. Transformer-Based Network with Optimization for Cross-Subject Motor Imagery Identification. *Bioengineering*. 2023, roč. 10, č. 5, s. 609. ISSN 2306-5354. Dostupné z DOI: 10.3390/bioengineering10050609.

[TLS17] TANG, Zhichuan; LI, Chao; SUN, Shouqian. Single-trial EEG classification of motor imagery using deep convolutional neural networks. *Optik*. 2017, roč. 130, s. 11–18. ISSN 0030-4026. Dostupné z DOI: https://doi.org/10.1016/j.ijleo.2016.10.117.

[Vař18] VAŘEKA, Lukáš. *Methods for Signal Classification and their Application to the Design of Brain-Computer Interfaces*. Katedra informatiky a výpočetní techniky Západočeské univerzity v Plzni, 2018. Dostupné také z: https://dspace5.zcu.cz/bitstream/11025/33651/1/PhdThesisVareka2018.pdf. Disertační práce.

[Yan+21] YANG, Jun; YU, Huijuan; SHEN, Tao; SONG, Yaolian; CHEN, Zhuangfei. 4-Class MI-EEG Signal Generation and Recognition with CVAE-GAN. *Applied Sciences*. 2021, roč. 11, č. 4. ISSN 2076-3417. Dostupné z DOI: 10.3390/app11041798.

[Zha+20] ZHANG, Kai et al. Data Augmentation for Motor Imagery Signal Classification Based on a Hybrid Neural Network. *Sensors*. 2020, roč. 20, č. 16. ISSN 1424-8220. Dostupné z DOI: 10.3390/s20164485.

Seznam zkratek

AE Autoencoder 17–19, 21, 28, 51–53

API Application Programming Interface 57, 79, 100

AR AutoRegression 16

BCI Brain-computer interface 5, 10–14, 47, 99, 117

CC Cross-correlation 75

CNN Convolutional neural network ii, 16–19, 21–23, 28, 31, 43, 44, 51, 76, 78, 80, 98, 99, 101

CPU Central processing unit 77, 107

CSP Common spatial pattern 16, 17, 23, 24, 43

CSV Comma-separated values 25

cVAE Conditional Variational Autoencoder ii, 16, 71, 73, 99, 101

DA Data augmentation 47, 48, 54

DFT Diskrétní Fourierova transformace 33, 68

DT-CWT Discrete-time continuous wavelet transform 34

EEG Elektroencefalografie ii, 5, 7–13, 15, 16, 18, 20–23, 28, 29, 31, 32, 37, 40, 43, 44, 47, 48, 50, 51, 57, 58, 60–62, 65, 100, 101, 117

EP Evokovaný potenciál 12

ERD Event-related desynchronization 9–12, 15, 16, 23–26, 28, 32

ERS Event-related synchronization 9–12, 15, 16, 23–26, 28, 32

FFT Fast Fourier transform 33

FID Fréchet inception distance 19, 74, 84

FT Fourierova transformace 32

GAN General Adveserial Network ii, 16–19, 22, 28, 54, 55, 70, 72, 74, 99, 101

GPU Graphics processing unit 57, 79, 83, 100, 107

GT Geometric transformation 19

ICA Independent component analysis 17, 22

LDA Linear discriminant analysis ii, 24, 28, 38, 57, 76, 101

LSTM Long short-term memory ii, 22, 23, 42, 43, 76, 78, 80, 101

MI Motor Imagery 5, 9, 10, 12, 13, 15–24, 26, 28, 31–34, 37, 42, 58

MLP Multilayer perceptron ii, 17, 18, 22, 23, 26, 28, 40–43, 51, 71, 73, 76, 78, 80, 101

NI Noise injection ii, 19, 22, 28, 51, 71, 98, 99, 101

PCA Principal component analysis 100

PSD Power spectral density 33

ReLU Rectified linear unit 23, 41, 71, 73, 80

RFT Random forest tree 17, 18

RMSE Root mean square error 75

RNN Recurrent neural network 21, 28

SMR Senzomotorický rytmus 9, 10, 24

SNR Signal-to-noise ratio 8, 51, 75

STFT Short time Fourier transform 16, 19, 23, 34

SVM Support vector machine ii, 16–18, 24, 27, 28, 38–40, 43, 44, 57, 76, 77, 83, 98, 101

VAE Variational autoencoder 17–19, 51–54

Seznam obrázků

Seznam tabulek

Seznam výpisů