FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Detekce úseků EKG vhodných pro další zpracování
Student:	Jozef Koleda
Vedoucí:	Ing. Jan Hejda, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

Cílem práce je návrh metody analýzy dlouhodobého EKG signálu za účelem detekce segmentů obsahujících artefakty, které by mohly významně ovlivnit další zpracování.
Realizujte následující kroky:
1. Seznamte se s typickými artefakty v EKG záznamu;
2. Proveďte rešerši metod strojového určení vhodných pro detekci artefaktů;
3. Navrhněte nástroj a formát pro anotaci EKG signálu;
4. Navrhněte metodu strojového učení pro detekci artefaktů a implementujte jí v Pythonu;
5. Statisticky vyhodnoťte přesnost detekce.
Dataset bude poskytnut vedoucím práce a anotaci signálu realizují experti z Fakulty biomedicínského inženýrství.

Elektronicky schválil/a Ing. Magda Friedjungová, Ph.D. dne 25. října 2022 v Praze.

Bakalárska práca

DETEKCE ÚSEKŮ EKG VHODNÝCH PRO DALŠÍ ZPRACOVÁNÍ

Jozef Koleda

Fakulta informačních technologií
Katedra aplikované matematiky
Vedúci: Ing. Jan Hejda, Ph.D.
11. mája 2023

Odkaz na túto prácu: Koleda Jozef. *Detekce úseků EKG vhodných pro další zpracování.* Bakalárska práca. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Chcel by som poďakovať vedúcemu mojej bakalárskej práce, Ing. Jánovi Hejdovi, Ph.D. za pomoc a vedenie pri tvorbe práce. Taktiež by som rád poďakoval svojim rodičom a priateľke za podporu a pomoc počas písania a tvorby práce.

Vyhlásenie

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisů, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programů, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo“), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli způsobem, který nesnižuje hodnotu Díla, avšak pouze k nevýdělečným účelům. Toto oprávnění je časově, teritoriálně i množstevně neomezené.

V Praze dňa 11. mája 2023 .....

Abstrakt

Elektrokardiogram alebo EKG zaznamenáva elektrickú aktivitu srdca a používa sa najmä na medicínske účely. V prípade dlhodobých experimentov môže byť EKG jedným zo sledovaných a vyhodnocovaných fyziologických javov. Počas merania môže nastať množstvo nepresností a chýb, ktoré je potrebné rýchlo detegovať a odstrániť, inak hrozí skreslenie výsledkov. Rozoznávame dokopy 3 úrovne kvality EKG signálu, každá z nich závislá od možnosti detekcie relevantných EKG segmentov. V práci navrhujeme a realizujeme niekoľko modelov na princípe LSTM a konvolúcie, vrátane špeciálnej architektúry využívajúcej Omni-Scale konvolučný blok. Modely fungujú na princípe posuvného okna, kde vyhodnocujeme každé krátke okno samostatne a následne výsledky jednotlivých okien spájame dokopy, aby sme získali výsledok pre dlhý záznam. Celkovo popisujeme tvorbu 4 modelov, ktoré sú následne implementované v rámci Python knižnice ecg-quality voľne dostupnej cez utilitu pip. Naše modely zaostávajú za ľuďmi vykonanou detekciou, ale dokážu v prípade trojtriednej klasifikácie pri signále najvyššej kvality dosiahnuť presnosť 97,7 % pri pokrytí 79,5 %. V prípade, že signál delíme binárne, dosahujú naše modely presnosť lepšieho signálu na úrovni vyše 99 %, čím poskytujú garanciu, že ak označia signál za nezávadný, naozaj ním aj je.

Klíčová slova EKG, kvalita EKG, klasifikácia časovej rady, CNN, LSTM, Omni-Scale, Python, knižnica

Abstract

Electrocardiogram or ECG measures the electrical activity of the heart. It is used mainly for medical uses. During long-term experiments, ECG can be one of the watched and analyzed physiological functions. While measuring, many inaccuracies and errors can happen. It is important to detect and remove these, as they can seriously impact the results of further analysis. We consider three different signal qualities, each of them dependent on the ability to detect certain relevant ECG segments. In the thesis, we use multiple models based on LSTM architecture or on the convolutional architecture, including a special Omni-Scale convolutional block. Our models use a sliding window method, where short windows are individually evaluated and these results are then combined to obtain results for longer recordings. We describe the creation of 4 models in total. These models are made available in a Python library ecg_quality through the pip utility. Our models are still behind human detection in terms of quality. However, a clean ECG signal can be detected with 97,7 % precision while retaining recall of 79,5 %. In the case, that we do a binary split on our three quality classes, our models achieve precision of the higher quality signal at over 99 %, which means they can strongly guarantee that if they label a signal to be clean, it really is clean.

Keywords ECG, ECG quality, time series classification, CNN, LSTM, Omni-Scale, Python, library

Obsah

Úvod

EKG je jedna z najdôležitejších fyziologických funkcií, ktoré sa pri človeku môžu sledovať. Popisuje srdcovú aktivitu a už od svojho objavenia má veľký význam v medicíne, kde je v súčasnosti kľúčovou časťou diagnostiky a každý deň pomáha zachrániť množstvo životov.

EKG sa však zvykne používať aj na iné ako medicínske účely, keď skúmame srdcovú aktivitu pre jej naviazanosť na iné fyziologické funkcie a nie z dôvodu diagnostiky. Počas takéhoto výskumu vzniknú desiatky hodín záznamu, ktorý sa následne vyhodnotí. Predtým je však potrebné všetky dáta skontrolovať. Počas merania EKG môže nastať obrovské množstvo chýb, od náhodných trvajúcich zlomok sekundy až po systematické, ktoré ovplyvnia celé hodiny. Ak by sme ich neodstránili, môžu mať zásadný vplyv na výsledky výskumu. Ľudská sila je drahá, a v prípade odborníkov to platí viacnásobne. Výpočetná sila, pokiaľ je primeraná, je výrazne lacnejšia. Potreba manuálne prejsť a označiť desiatky hodín dát je náročný problém, ktorý vie pripraviť odborníkov o množstvo času a v prípade výskumu ho aj zastaviť alebo extrémne spomaliť, pretože tráviť týždne rutinným prechádzaním signálu je niečo, čo si väčšina výskumných tímov nemôže dovoliť. Existujúce implementované nástroje buď majú značné slabosti, rozlišujú kvalitu iba dlhších úsekov ako celkov, alebo pracujú iba na princípe jednoduchého delenia na zlý a dobrý signál.

V priebehu posledných rokov došlo k výraznému rozvoju umelej inteligencie na princípe hlbokého učenia. Spracovanie signálov, prípadne časových rád nie je výnimkou. Existuje dostatočný teoretický základ na to, aby sa vytvorila knižnica v jazyku Python na detekciu artefaktov v EKG využívajúca modely na princípe hlbokého učenia.

Cieľom tejto práce je popísať základné príznaky a fungovanie EKG a definovať artefakty, ktoré môžu nastať počas dlhodobého merania. Ďalej preskúmame už teraz dostupné metódy na určenie kvality EKG. Následne, na základe teoreticky navrhnutých metód na spracovanie a určenie kvality EKG, ale aj všeobecných metód klasifikácie časových rád, navrhneme metódy na princípe hlbokého učenia. Tieto metódy implementujeme ako súčasť voľne dostupnej Python knižnice. Získané výsledky spracovania vyhodnotíme. Štatisticky určíme ich presnosť a porovnáme medzi sebou jednotlivé metódy klasifikácie. Súčasťou vyhodnotenia bude aj porovnanie týchto metód voči klasifikácii vykonanej človekom.

Kapitola 1
EKG

EKG je skratka pre elektrokardiogram alebo elektrokardiograf. Zvykne sa taktiež niekedy označovať ako ECG, najmä v anglickej literatúre. EKG je zápis elektrickej aktivity. Skúma vznik elektrických impulzov a ich šírenie srdcom. Tradične býval papierový, ale v súčasnosti už existujú prístroje, ktoré EKG zapisujú v digitálnej podobe.

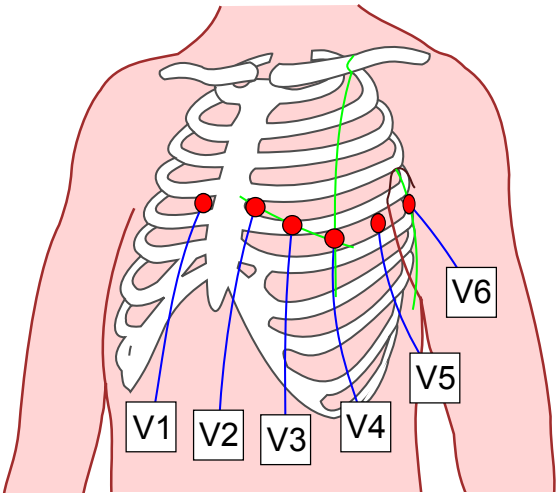
Elektrická aktivita srdca vzniká v jeho vnútri, v takzvanom sinoatriálnom uzle, a jeho normálny rytmus sa označuje ako sínusový rytmus. Zmeny tejto aktivity je možné zobraziť na EKG a môžu znamenať poškodenie srdcového svalu. Preto je potrebné všetky abnormality EKG brať vážne, pretože v prípade, že tieto abnormality nie sú spôsobené chybou merania, môžu indikovať vážny zdravotný problém [1, s. 17].

1.1 Meranie EKG

Elektrická aktivita srdca sa zaznamenáva pomocou elektród, ktoré sú pripojené na končatiny alebo hrudník. Celkovo sa zvykne používať 12 rôznych pohľadov alebo zvodov, čiže rôznych pripojení elektród. Pre úplný prehľad srdcovej aktivity sa používa všetkých 12 zvodov, avšak zvykne sa používať aj menej zvodov, bežne aj iba jeden. Viacero zvodov sa používa z dôvodu potreby mať viacero „pohľadov“ na srdce. Každý zvod poskytuje pohľad z iného uhla, niektoré v horizontálnej a iné vo vertikálnej rovine.

Pri 12-zvodovom EKG sa umiestni jedna elektróda na každú končatinu, čím sa zabezpečí prvých 6 pohľadov, ktoré sa označujú I, II, III, aVR, aVL a aVF. Tieto zvody sledujú srdce vo vertikálnej rovine. Následne sa umiestni 6 elektród na hrudník a zaznamenávajú zvody V_1–V_6, ktoré poskytujú pohľad na srdce v horizontálnej rovine. Ako je možno vidieť na Obrázku 1.1, tieto elektródy je potrebné umiestniť veľmi precízne a aj malá odchýlka môže mať za následok zavádzajúce výsledky EKG. Aj preto nie sú vhodné na dlhodobé meranie EKG, kde toto nie je možné po celý čas zabezpečiť. V tomto je prvých 6 spomenutých zvodov vhodnejších, keďže elektródy, ktoré pre ne zapájame, sú vo svojom umiestnení výrazne flexibilnejšie. Pre horné končatiny je možné elektródy zapojiť kdekoľvek od ramena alebo vonkajšej kľúčnej kosti až k zápästiu. Pri dolnej končatine je zase vhodné umiestniť elektródy kdekoľvek od podbruška až po chodidlo [1, s. 18-19].

Samotné určenie EKG prebieha určením rozdielu nameraných hodnôt elektrického prúdu jednotlivými elektródami. Tento výsledok sa nazýva zvod, ktorých je celkovo 12. Každý zvod má presne definované skupiny elektród, ktorých výsledky porovnáva. Presné určenie skupín použitých elektród pre jednotlivé zvody ukazuje Tabuľka 1.1.



Obr. 1.1 Zapojenie hrudných elektród [2]

Zvod	Porovnanie elektrickej aktivity
I	LA a RA
II	LL a RA
III	LL a LA
aVR	RA a priemer (LA + LL)
aVL	LA a priemer (RA + LL)
aVF	LL a priemer(LA + RA)
V_1	V_1 a priemer (LA + RA + LL)
V_2	V_2 a priemer (LA + RA + LL)
V_3	V_3 a priemer (LA + RA + LL)
V_4	V_4 a priemer (LA + RA + LL)
V_5	V_5 a priemer (LA + RA + LL)
V_6	V_6 a priemer (LA + RA + LL)

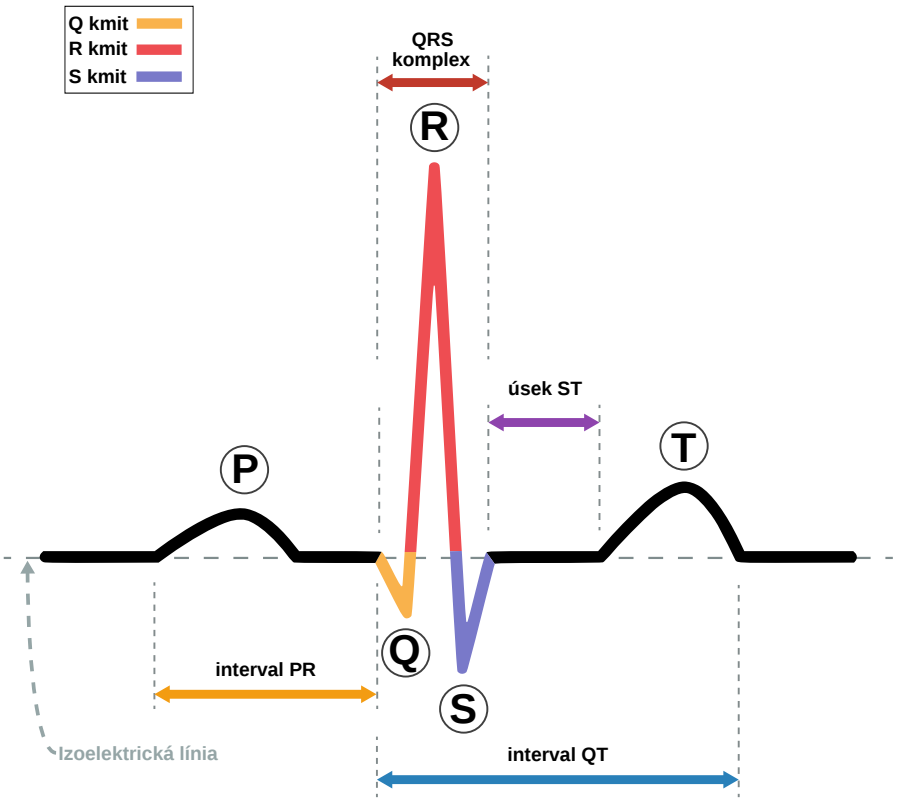
Tabuľka 1.1 Zvody EKG [1, s. 49]

1.2 Časti EKG

Pri určovaní príznakov EKG a ich ďalšom popise delíme signál EKG na časti, ktoré zachytávajú elektrickú aktivitu srdca počas jedného úderu, čiže počas kontrakcie predsiení a následnej kontrakcie komôr. Napriek tomu, že srdce má dokopy 4 časti (2 komory a 2 predsiene), z pohľadu elektrickej aktivácie ho pozorujeme, ako by malo iba 2, keďže obe predsiene a potom obe komory sa sťahujú súčasne [1, s. 41].

Počas jedného úderu ako prvé dochádza ku kontrakcii predsiení. Ich depolarizácia, teda elektrická aktivácia, je spojená s vlnou P. Následne dochádza ku kontrakcii komôr. Ich svalovina je výrazne silnejšia, a preto ich depolarizácia spôsobuje výrazne väčšiu výchylku. Táto sa nazýva QRS komplex. QRS komplex je tvorený celkovo z troch kmitov, Q, R a S. Ak je prvá výchylka QRS komplexu negatívna, nazýva sa kmit Q. Ak je výchylka pozitívna, nazýva sa kmit R, bez ohľadu na to, či jej predchádzal kmit Q. Každá výchylka pod izoelektrickú líniu za kmitom R sa nazýva kmit S bez ohľadu na to, či nastal predchádzajúci kmit Q. Po QRS komplexe nasleduje vlna T, ktorá odráža repolarizáciu komôr [1, s. 43].

Okrem týchto základných segmentov sa zvyknú ešte označovať segmenty z nich odvodené, ako napríklad PR interval, QT interval a ST úsek [1, s. 43, 45]. Ukážka jednotlivých úsekov a častí EKG sa nachádza na Obrázku 1.2. Jemne odlišný, ale stále bežne využívaný je interval R-R, ktorý je ohraničený R kmitmi dvoch susedných úderov [1, s. 45].



Obr. 1.2 EKG jedného úderu a všetky jeho časti [3]

1.3 Artefakty pri meraní EKG

Počas merania EKG dochádza k množstvu artefaktov, ktoré môžu ovplyvniť výsledné namerané hodnoty. Príčiny mnohých z nich sa v medicínskom prostredí počas vyšetrenia pacienta dajú odstrániť, keďže takéto vyšetrenia bežne netrvajú dlho. Avšak v prípade, že meranie musí prebiehať dlhodobo, sa väčšine artefaktov nedá zabrániť.

Abnormality EKG môžu byť spôsobené aj poruchou srdcového svalu alebo iným medicínskym problémom. Vzhľadom na to, že v tomto prípade ide o detekciu artefaktov na nediagnostické účely, budeme sa podrobnejšie zaoberať iba odchýlkami, ktoré sú spôsobené chybami merania, a teda nie samotnou funkciou srdca. Pre ľudí, ktorí trpia chorobou, ktorá ovplyvňuje srdcovú činnosť alebo EKG, môžu nastať abnormality, ktoré nebudú spôsobené skúmanými artefaktami.

To, ako vyzerá čistý EKG signál a ako vyzerá po umelom zašpinení jednotlivými artefaktami, je možno vidieť na Obrázku 1.3. Jedinou výnimkou je záznam z odpojeného zariadenia na meranie EKG. V reálnych dátach sa budú môcť artefakty vyskytovať aj naraz, čo môže sťažiť ich detekciu.

1.3.1 Fluktuácia izoelektrickej línie

Tento artefakt môže byť spôsobený pomerne širokým spektrom príčin, ako je napríklad pohyb subjektu, problém s elektródami alebo dýchanie. Odstránenie tejto fluktuácie je kľúčové pre ďalšiu analýzu EKG. Tento artefakt sa prejavuje šumom s nízkymi frekvenciami, rádovo okolo 0,5 Hz. Rozsah frekvencií šumu môže byť ešte rozšírený v prípade výraznejšieho pohybu tela, cvičenia alebo v prípade stresu. Jedná sa o skutočnosti, ktoré počas dlhodobého merania môžu nastať [4]. Časť tohto šumu sa dá odstrániť pomocou filtrovania. Toto filtrovanie je dostupné ako súčasť knižnice NeuroKit2 [5].

1.3.2 Elektrická interferencia

EKG meria elektrickú aktivitu pomocou elektrického zariadenia. Preto vie byť takéto meranie náchylné na vplyv elektromagnetických polí vytváraných striedavým prúdom. Zapojené elektrické zariadenia v blízkosti môžu spôsobiť, že na EKG je vidieť pravidelnú osciláciu, bežne s frekvenciou okolo 50 až 60 Hz. Elektrická interferencia je jednou z hlavných príčin potlačenia P a T vĺn [1, s. 64] [4]. Táto interferencia sa tiež dá odstrániť pomocou filtrov, ktoré sú dostupné v knižnici NeuroKit2 [5].

1.3.3 Zlý kontakt elektród s pokožkou

Pre presné meranie EKG je potrebné, aby mali elektródy dobrý elektrický kontakt s kožou. Je potrebné, aby bola koža čistá a suchá. Taktiež je vhodné, aby elektródy neboli umiestnené na ochlpení, keďže to bráni nalepeniu a ochlpenie je nevodivé [1, s. 64]. Správne umiestnenie elektród vieme zabezpečiť aj počas dlhotrvajúceho merania, avšak čistú a suchú kožu zaručiť nevieme. Je rozumné predpokladať, že človek sa počas merania môže spotiť a toto môže mať vplyv na kvalitu EKG. Tento artefakt je podobný fluktuácii izoelektrickej línie s tým rozdielom, že tento typ šumu sa vyskytuje v EKG signále s vyššou frekvenciou, asi 1–10 Hz. Najmä počas dlhodobých meraní môže spôsobiť nesprávne detekovanie QRS komplexu [4].

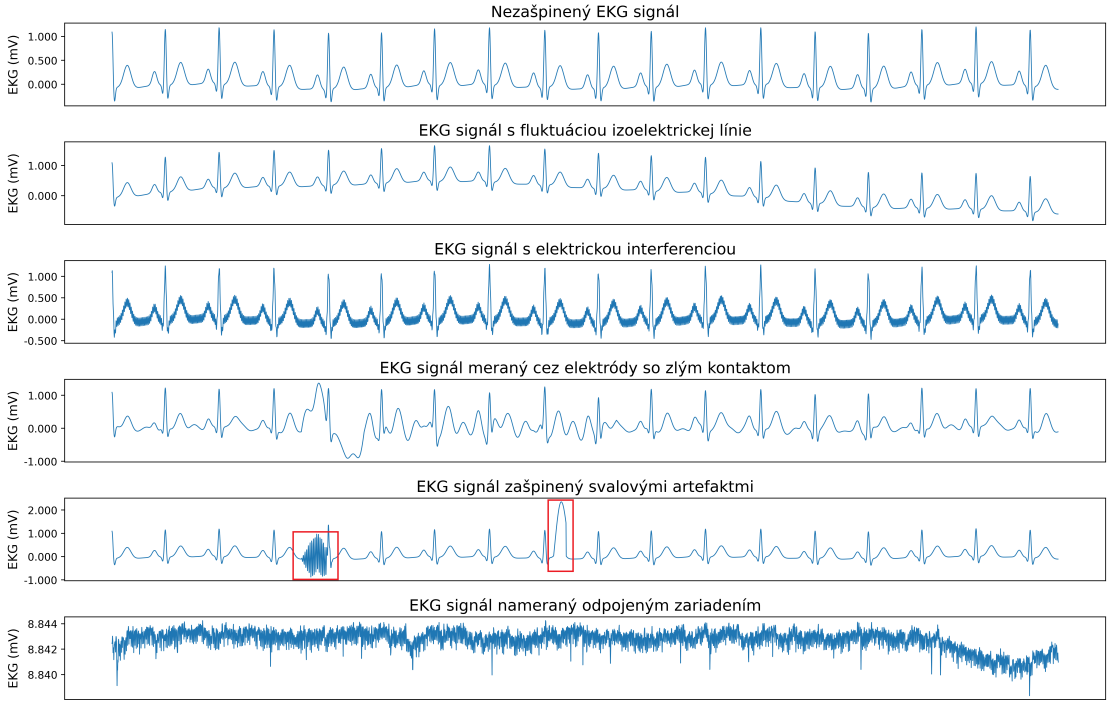
1.3.4 Svalové artefakty

Pohyb každého svalu je spojený s elektrickými zmenami a tieto zmeny je možné zaznamenať aj elektródami počas merania EKG [1, s. 41]. Pri dlhodobom meraní, ktoré má navyše zachytiť aktivitu srdca počas bežných činností, nie je možné, aby sledovaný subjekt obmedzil používanie kostrových svalov. Tento šum neobsahuje vysoké frekvencie ako elektrická interferencia ani nízke

frekvencie ako fluktuácia izoelektrickej línie. Býva distribuovaný cez všetky segmenty EKG, čo robí jeho detekciu veľmi dôležitou [4].

1.3.5 Odpojenie elektród

Ďalší špecifický artefakt pri dlhodobom sledovaní EKG je odpojenie elektród. Takýto problém sa väčšinou pomerne rýchlo napraví, výnimkou je iba ak nastane v období, kedy subjekt nemôže reagovať, napríklad uprostred noci. V prípade odpojenia sa nedá očakávať signál s príznakmi EKG. Napriek tomu je potrebné venovať sa aj tomuto artefaktu, pretože počas merania k nemu môže dôjsť a tento artefakt má potenciál trvať aj hodiny po tom, čo nastane, a teda môže ovplyvniť výsledky počas dlhého úseku merania.



Obr. 1.3 Nezašpinené EKG a následne EKG s popísanými artefaktmi.

1.4 Predspracovanie EKG

Bežnou súčasťou práce s EKG je predspracovanie, ktoré sa snaží zlepšiť kvalitu signálu a odstrániť aspoň niektoré z artefaktov. Budeme využívať predspracovanie implementované v NeuroKit2, konkrétne funkciu ecg_clean. Táto funkcia v základnom nastavení využíva hornopriepustný butterworth filter úrovne 5 s frekvenciou 0,5 Hz a ešte filtruje elektrickú interferenciu s frekvenciou 50 Hz [6].

1.4.1 Hornopriepustný filter

Hornopriepustné filtre fungujú na princípe, že všetky frekvencie vyššie ako istá medza zostanú nedotknuté a frekvencie menšie ako táto medza budú potlačené. Problém, ktorý je spojený s filtrami je takzvaná fázová odozva a s tým spojený fázový posun, teda posun istých frekvencií

v čase po prechode filtrom. Toto môže mať za následok zmenu signálu. Butterworth filter, ktorý budeme používať má takmer pomerne lineárnu fázovú odozvu pre prepustené frekvencie v porovnaní s inými bežnými filtrami. Jeho nevýhoda spočíva v tom, že pri potláčaní frekvencií má pomerne dlhé prechodné pásmo. To vieme kompenzovať použitím vyššej úrovne filtra, čo má však za následok menšiu linearitu fázovej odozvy. Vzhľadom na rozsah tejto témy sa jej nebudeme bližšie venovať a iba použijeme už spomenutú implementáciu z knižnice NeuroKit2 [7] [6].

1.4.2 Filtrovanie elektrickej interferencie

Už sme spomínali, že elektrická interferencia má frekvenciu asi 50 Hz. Žiadne príznaky EKG nemajú takúto alebo vyššiu frekvenciu, a teda sa tiež otvára možnosť použitia frekvenčného filtra. S tým sa však spájajú už spomenuté problémy fázového posunu. Namiesto toho používa NeuroKit2 kĺzavý priemer dĺžky jednej periódy signálu s frekvenciou 50 Hz, čím dosahuje porovnateľný výsledok [8] [9].

Kapitola 2
Detekcie artefaktov

Na detekciu artefaktov, respektíve na popis kvality EKG signálu už existuje pár implementácií. Množstvo postupov je zatiaľ iba navrhnutých bez ľahko dostupnej implementácie v jazyku Python. Existuje množstvo modelov, ktoré pracujú s EKG za účelom diagnostiky alebo detekcie srdcových porúch a taktiež existuje množstvo postupov na prácu so všeobecnými časovými radami. Veľká časť z týchto postupov využíva metódy, ktoré môžu byť použité aj pri skúmanom probléme na detekciu artefaktov.

2.1 Implementované postupy

Na priamu detekciu artefaktov nebolo možné nájsť žiadnu voľne dostupnú knižnicu v jazyku Python. Namiesto toho existujú dve knižnice, ktoré v istej miere určujú kvalitu EKG. Budeme predpokladať, že ak je kvalita EKG nízka, je to zrejme spôsobené artefaktami, takže aj takýto spôsob môžeme použiť na detekciu úsekov, ktoré nie sú vhodné na ďalšie spracovanie.

2.1.1 NeuroKit2

Prvá knižnica, ktorá ponúka funkcie na určenie kvality EKG je NeuroKit2. Uvedená knižnica je určená na spracovanie biologických signálov, teda aj EKG. Táto knižnica disponuje funkciou ecg_quality, ktorá následne ponúka dva rôzne postupy na určenie kvality signálu EKG [6].

2.1.1.1 averageQRS metóda

Táto metóda počíta pre zadaný úsek kontinuálny index kvality, ktorý je v rozmedzí 0 až 1. Robí to výpočtom vzdialenosti jednotlivých QRS komplexov od priemerného QRS komplexu v dátach. Hodnota 1 zodpovedá úderom, ktoré sú najbližšie k tomuto priemeru, zatiaľ čo hodnota 0 zodpovedá úderom, ktoré sú najďalej. Výsledný index 1 však nemusí nutne znamenať kvalitný signál, keďže tento index je iba relatívny voči priemeru. Ak je celý analyzovaný úsek alebo aspoň jeho väčšina poznačený artefaktami, tak aj vysoké skóre môže signalizovať zlý signál a nízke zase dobrý. Ďalšia slabina spočíva v tom, že táto metóda sa nevenuje P a T vlnám, čo môže byť problém, ak sa chceme venovať analýze týchto segmentov [6].

2.1.1.2 zhao2018 metóda

Táto metóda pre signál vypočíta niekoľko kvalitatívnych indexov, pSQI, kSQI, basSQI. Všetky tieto indexy vychádzajú z podrobnej znalosti EKG a jeho frekvenčnej analýzy a každý sa zameriava na iné oblasti možného zašpinenie EKG signálu. Tieto indexy sú po vypočítaní sčítané

s váhami [0.6, 0.2, 0.2]. Hlavná nevýhoda tejto metódy je to, že hodnotí celý záznam EKG a pre celý vracia iba jedno z troch možných hodnotení: Unacceptable, Barely Acceptable alebo Excellent [10] [6]. Pri spracovaní záznamu trvajúceho niekoľko hodín toto nie je dostatočné.

2.1.2 ecg-qc

Knižnica špecificky vyvinutá na určenie kvality EKG signálu v reálnom čase je ecg-qc. Implementuje triedu, ktorá na základe analýzy EKG vypočíta 6 metrík kvality EKG [11], ktoré následne využíva v štvorici modelov. Tieto modely boli trénované na oknách s pevne danou dĺžkou, takže pri takýchto dátach zrejme budú dosahovať najlepšie výsledky. Avšak vzhľadom na princíp klasifikácie nám nič nebráni použiť aj dlhšie okná. Modely z tejto knižnice signál klasifikujú binárne ako buď dobrý alebo zlý. Prehľad jednotlivých modelov, dĺžiek ich okien a prípadnej štandardizácie vstupných dát pri trénovaní ukazuje Tabuľka 2.1.

Táto knižnica bola vytvorená za účelom kontroly EKG pre detekciu epileptického šoku [12], takže pri jej tvorbe bol veľký dôraz na rýchly beh aj na malých zariadeniach. Samotní autori uznávajú, že presnosť detekcie nie je až taká vysoká, ako by bola v prípade použitia metód hlbokého učenia [13] [14].

Typ modelu	EKG časové okno (tréning)	Štandardizácia jednotlivých EKG segmentov (tréning)
Rozhodovací strom	2 sekundy	Nie
Náhodný les	2 sekundy	Nie
Náhodný les	2 sekundy	Áno
XGboost	9 sekúnd	Nie

Tabuľka 2.1 Prehľad modelov v knižnici ecg-qc

2.2 Navrhnuté metódy detekcie artefaktov

Existujú navrhnuté postupy detekcie artefaktov v EKG, ktorých implementácia síce nie je ľahko dostupná, ale stále môžu ponúkať efektívne metódy, ktoré by bolo možné implementovať do knižnice v jazyku Python. Celkový počet navrhnutých metód na detekciu artefaktov v EKG signále však nie je veľký. Napriek tomu existuje množstvo rôznych navrhnutých postupov. Viaceré postupy používajú autokorelačnú funkciu vypočítanú z EKG, z ktorej následne získavajú príznaky a podľa týchto detegujú artefakty pomocou ensamble algortimu RUSBoost [15], metódy podporných vektorov [16] alebo odstránením úsekov mimo určeného percentilu [17]. Taktiež bolo navrhnuté použitie jednoduchej štatistiky a porovnávanie susedných krátkych úsekov EKG na určenie náhlych zmien, ktoré môžu signalizovať prítomnosť artefaktu [18]. Ďalší postup napríklad využíva detekciu motívov časových rád [19].

Omnoho rozšírenejšie je spracovanie EKG za účelom diagnostiky alebo detekcie chorôb, napríklad arytmie. Jedná sa o odlišný problém, ale aj v tomto prípade je potrebné pracovať s charakteristikami EKG a detegovať abnormality od bežného EKG. Nebudeme sa zaoberať metódami, ktoré využívajú detekciu a následné vlastnosti jednotlivých segmentov EKG, keďže na toto sa pri detekcii artefaktov nemôžeme spoľahnúť kvôli ich schopnosti zabrániť prítomnosti všetkých príznakov EKG. Za účelom diagnostiky chorôb sa používa napríklad zhlukovanie [20] alebo metóda podporných vektorov [21], prípadne ich kombinácia [22]. Pred spracovaním sa na dáta môže použiť napríklad aj wavelet transformácia [21] [20]. Všetky tieto prístupy však používajú istú mieru predspracovania a analýzy EKG, ktoré vychádzajú z jeho podrobnej znalosti.

Okrem tohto je stále možné použiť aj všeobecnejšie metódy na detekciu. Klasifikácia časových rád je výrazne rozšírenejšie odvetvie a aj preto je v ňom navrhnutých výrazne viac state-of-the-art

metód na princípe hlbokého učenia, ktoré využívajú výrazne väčšiu variabilitu prístupov. Bežné sú architektúry na princípe MLP, FCN, residuálnych sietí [23] alebo LSTM [24].

Kapitola 3
Hlboké učenie

Spracovanie EKG vrátane detekcie artefaktov ešte donedávna stálo na jeho odbornej znalosti a na následnej analýze, ktorá vyústila do potrebných algoritmov. Toto platí aj pre dostupné nástroje na detekciu artefaktov v EKG signále. Jedna z veľkých výhod neurónových sietí je to, že aj bez pokročilejších znalostí skúmaného problému je možné vytvoriť model, ktorý bude dosahovať dostatočne presné výsledky za zlomok času v porovnaní s človekom. Podrobná znalosť problematiky vie stále pomôcť, ale jej dôležitosť sa postupne zmenšuje.

O neurónových sieťach, ktoré majú 3 a viac vrstiev, môžeme povedať, že fungujú na princípe hlbokého učenia [25]. Prehlbovanie modelov pomáha riešiť komplexné úlohy a dosahovať lepšie výsledky. Neurónové siete sú čím ďalej tým hlbšie. Kým model známy ako AlexNet navrhnutý v roku 2012 mal 7 skrytých vrstiev [26], už o 3 roky neskôr dosahovali modely aj vyše 1000 skrytých vrstiev pri riešení tej istej úlohy [27]. Modely nerastú do šírky, ale výrazne viac do hĺbky.

3.1 Typy úlohy

V strojovom učení rozoznávame množstvo rôznych typov úloh. Pre jednoduchosť však predstavíme iba dva základné, čo bude postačujúce na pochopenie následného postupu tvorby modelov. Každý z týchto typov má svoje špecifiká a odlišnosti, ktoré je pri tvorbe modelov a výbere postupu potrebné poznať. Spracovanie časových rád môžeme považovať za samostatné odvetvie strojového učenia a výrazne sa líši od klasických klasifikačných a regresných úloh. Delenie úloh na iba dva typy je zjednodušenie, ktoré sa nehodí pre použitie cez celé odvetvie strojového učenia. Na náš problém detekcie artefaktov v EKG je však dostačujúce.

3.1.1 Klasifikačná úloha

Pri riešení tohto problému je potrebné priradiť jednu z aspoň dvoch tried každému vstupu. Niektoré modely, ako napríklad rozhodovacie stromy alebo zhlukovanie, určujú triedu bez toho, aby spresnili určitosť. Niektoré modely vrátane neurónových sietí počítajú pravdepodobnosti, z ktorých následne určujeme triedy.

Jeho najjednoduchšia verzia je binárna klasifikácia, pri ktorej iba určujeme, či nejaké tvrdenie platí alebo nie. Jedná sa o najjednoduchší typ klasifikácie.

V prípade, že každému vstupu priradíme maximálne jednu triedu z viac ako dvoch, hovoríme o viactriednej klasifikácii. Klasifikované triedy vtedy nie sú nutne protiklady ako v prípade binárnej klasifikácie, ale jedná sa o niekoľko rôznych prípadov. Ako príklad môžeme uviesť problém klasifikácie typu ovocia na obrázku, kedy na obrázku neočakávame viac druhov ovocia naraz a

našou úlohou je určiť, aké ovocie je možné vidieť. Existuje viacero spôsobov, ako tento problém riešiť, či už je to vytvorenie viacerých binárnych klasifikátorov, alebo použitie modelov, ktoré sú schopné pracovať aj s viacerými výstupmi, napríklad neurónové siete. Bežne získame pravdepodobnosť a vyberáme triedu, ktorá je najpravdepodobnejšia. Výnimkou sú modely, ktoré s pravdepodobnosťami nepracujú, napríklad zhlukovania.

Posledný typ úlohy je viacpríznaková klasifikačná úloha. Cieľom modelu je pre každý vstup vrátiť ľubovoľný počet tried zo všetkých možných. Príklad takejto úlohy môže byť diagnostika z EKG, kde je možné, že model bude detegovať viacero diagnóz na jednom zázname. V prípade, že náš model, respektíve modely, pracujú na princípe vrátenia pravdepodobnosti, označíme, že vstup odpovedá tým triedam, ktoré prekročili stanovenú hranicu pravdepodobnosti. Uvedený postup podobne ako v prípade viactriednej klasifikácie taktiež neplatí pre modely, ktoré s pravdepodobnosťou nepracujú.

Problém detekcie artefaktov v EKG môžeme riešiť ako klasifikačnú úlohú. Implementácia v knižnici ecg_qc pristupuje k detekcii artefaktov ako ku klasifikačnej úlohe, kde sa najskôr rozdelí signál na úseky s konštantnou dĺžkou a tie sú následne binárne klasifikované ako vhodné, teda bez artefaktov a nevhodné, teda s artefaktmi. Druhý možný prístup je využitie viacpríznakovej klasifikácie tým spôsobom, že budeme rozlišovať jednotlivé artefakty a bude nás zaujímať, ktorý artefakt sa kde vyskytuje. Tu môže nastať, že signál je zašpinený aj viacerými artefaktmi naraz. O klasifikačnú úlohu sa bude jednať aj v prípade, že budeme rozlišovať rôzne úrovne zašpinenia signálu. V tom prípade sa bude jednať o viactriednu klasifikáciu, keďže signál môže dosahovať iba jednu kvalitu naraz. [28, s. 95-103].

3.1.2 Regresná úloha

Cieľom regresnej úlohy je čo najpresnejšie odhadnúť nejakú hodnotu. Hlavný rozdiel oproti klasifikačnej úlohe je ten, že množina možných výsledkov je spojitá. Model teda vracia hodnotu z definovaného intervalu. Tento interval môže pokrývať všetky reálne čísla, iba kladné čísla alebo iba obmedzený interval. Medzi regresnú úlohu môžeme napríklad zaradiť aj predpovedanie ďalších údajov v časovej rade. Každá z variácií regresie má svoje špecifiká, najmä v prípade použitia a implementácie neurónových sietí.

Pomocou regresie môžeme taktiež pristúpiť ku problému kvality EKG. Jedna zo spomínaných implementácií ecg_clean v knižnici NeuroKit2 pristupuje k určeniu kvality ako ku regresnej úlohe, kde vracia hodnotu z intervalu 0 až 1, ktorá signalizuje kvalitu signálu, respektíve v tomto prípade vzdialenosť od priemerného signálu. Istá nevýhoda takéhoto prístupu je, že výsledná hodnota nám priamo nepomôže odstrániť zašpinený signál. Na to budeme musieť nastaviť nejakú hranicu, ktorá určí, ktorý signál sa zahodí a ktorý ponechá.

3.1.3 Riešenie s využitím opačného postupu

Klasifikácia a regresia majú každá svoje špecifiká, výhody, nevýhody, ale najmä rozličné prístupy. Najmä v prípade komplikovanejších problémov môže byť výhoda použiť oba tieto prístupy a pri riešení jednej z týchto úloh zadefinovať a vyriešiť úlohu druhého typu. Takýto typ riešenia úlohy si zadefinujeme ako:

▶ **Definícia 3.1.** *Ak pri riešení klasifikačnej, resp. regresnej úlohy využijeme riešenie opačného typu úlohy ako medzivýsledok alebo medzivýsledky, ktoré nie sú pravdepodobnosťou, definujeme, že toto riešenie využíva opačnú úlohu.*

Znamená to teda, že napríklad v prípade klasifikácie vypočítame najskôr kontinuálnu hodnotu v odvodenej regresnej úlohe a následne z tejto hodnoty získame výslednú triedu, respektíve triedy. Nebyť podmienky s pravdepodobnosťou, dala by sa za takúto úlohu považovať aj časť štandardných klasifikačných úloh, ak by sme využili model, ktorý pravdepodobnosť používa. Preto je táto podmienka podstatná, aby definícia nebola príliš obsiahla.

Za riešenie regresnej úlohy, pri ktorom využívame opačnú úlohu, môžeme považovať napríklad riešenie, ktoré najskôr na základe vstupu vykoná klasifikáciu a až tieto medzivýsledky použije na výpočet výslednej kontinuálnej hodnoty. Toto sa môže oplatiť napríklad v prípade, že na vstupe sa nachádza množstvo rôznych dát z rozdielnych tried, ktoré majú veľmi odlišné súvislosti a je jednoduchšie natrénovať regresný model pre každú triedu namiesto toho, aby sme použili jeden model, ktorý sa snaží popísať všetky tieto súvislosti.

Zavádzanie takéhoto typu problému nie je bežné a zaviedli sme si ho iba pre naše účely. Väčšinou sa úloha špecifikuje výsledkom a nie postupom. Ak však veľkú časť získania výsledku napríklad klasifikačnej úlohy tvorí úloha regresná, je potrebné to jasne pomenovať a vysvetliť. Pre naše účely je lepšie zaviesť pomenovanie tohto javu, aby sme sa neskôr tomuto faktu nemuseli venovať.

3.2 Metriky merania presnosti

Počas tvorby modelu je potrebné dávať pozor na chyby. Pravdepodobnosť, že model vždy vráti správny výsledok, je pri klasifikácii extrémne nízka a pri regresii vzhľadom na spojitosť množiny možných výsledkov ešte nižšia. Preto je potrebné mať zavedené metriky, ktoré môžeme použiť na účel určenia kvality jednotlivých modelov. Tieto metriky sa zároveň používajú aj v priebehu tvorby modelov, čo ich dôležitosť ešte zvyšuje.

3.2.1 Metriky klasifikácie

Pri klasifikácii je niekoľko druhov metrík, ktoré sa dajú použiť. Každá z nich má iné vlastnosti a je vhodné ich použiť v rozdielnych prípadoch. Niektoré z metrík zohľadňujú iba predpovedané triedy a porovnávajú ich s reálnymi triedami a na základe toho vracajú výsledok. Iné berú pri výpočte do úvahy aj pravdepodobnosť, podľa ktorej sme sa pre jednotlivé triedy rozhodli a uprednostňujú modely, ktoré sú si istejšie so svojím výberom.

3.2.1.1 Podiel správnych odpovedí

Podiel správnych odpovedí je najjednoduchšia metrika používaná pri klasifikácii. Určuje iba podiel správne klasifikovaných vstupov. Táto metrika je však vysoko náchylná voči nevyrovnanému datasetu. Ak by sme napríklad mali model riešiaci problém binárnej klasifikácie, ktorý rozhoduje, či pre vstup bude platiť tvrdenie alebo nie, kde však iba 5 % dát je klasifikovaných ako pravdivých, tak model, ktorý vždy odpovie, že pre vstup tvrdenie neplatí, by dosiahol podiel správnych odpovedí 95 %. Aj kvôli tejto vážnej slabine nie je vždy úplne vhodné túto metriku použiť. Niekedy je to priam až škodlivé.

3.2.1.2 Presnosť

Pri počítaní presnosti potrebujeme rozoznávať dve podstatné hodnoty, počet tzv. true positives a false positives. Tieto hodnoty sa dajú pri viactriednej klasifikácii počítať aj pre všetky triedy, my sa však budeme zaoberať iba ich definíciou pre binárnu klasifikáciu. Ich rozšírenie na viactriednu klasifikáciu bude spočívať v tom, že ich vypočítame pre každú triedu zvlášť.

▶ **Definícia 3.2.** *Za true positives (TP) triedy c považujeme všetky priradenia tejto triedy vstupu, pre ktorý trieda c platí. Túto hodnotu značíme TP*(c).

▶ **Definícia 3.3.** *Za false positives (FP) triedy c považujeme všetky priradenia tejto triedy vstupu, pre ktorý trieda c neplatí. Túto hodnotu značíme FP*(c).

Definované hodnoty použijeme na výpočet presnosti (precision) klasifikácie triedy c:

$$\text{presnosť}(c) = \frac{\text{TP}(c)}{\text{TP}(c) + \text{FP}(c)}$$

Presnosť je definovaná ako podiel správnych klasifikácií voči celkovému počtu klasifikácií triedy. Hneď na prvý pohľad je jasné, že aj táto metrika má slabú stránku.

Nezáleží na tom, aká je hodnota $\text{TP}(c)$ pre model. Ak je hodnota $\text{FP}(c)$ rovná 0, presnosť bude dosahovať maximum napriek tomu, že model nemusí fungovať dobre a vstupom môže priraďovať triedu c nedostatočne. Aj preto sa presnosť zvykne používať v kombinácii s ďalšími metrikami [28, s. 87–88].

3.2.1.3 Pokrytie

Metrika pokrytie (recall) sa snaží nedokonalosť presnosti kompenzovať. Popisuje podiel správnych priradení tried z celkového počtu platných priradení tried. Pred zavedením výpočtu je však potrebné zaviesť tzv. false negatives.

► **Definícia 3.4.** *Za false negatives (FN) pre triedu c považujem všetky nepriradenia tejto triedy vstupu, pre ktorý táto trieda platí. Túto hodnotu značíme FN*(c).

S využitím tejto metriky môžeme následne vypočítať pokrytie:

$$\text{pokrytie}(c) = \frac{\text{TP}(c)}{\text{TP}(c) + \text{FN}(c)}$$

Pozrime sa na slabé miesta tejto metriky. Ak by náš model vždy klasifikoval platnosť všetkých tried, dosiahol by model pokrytie 100 %. Ľahko však vidieť, že presnosť by v tomto prípade bola veľmi nízka, najmä v prípade, že pracujeme s veľa vstupmi s veľa možnými triedami [28, s. 87].

3.2.1.4 F-skóre

Presnosť a pokrytie je nemožné maximalizovať naraz. Od svojej podstaty má jedna metrika tendenciu klesať, zatiaľ čo druhá stúpa a naopak. Toto je známy jav po anglicky nazývaný precision/recall trade-off [28, s. 89-92]. F-skóre, presnejšie F_β-skóre sa snaží kombinovať presnosť a pokrytie do jednej spoločnej metriky. Jej cieľom je nájsť istý rozumný kompromis medzi týmito metrikami a to najmä v prípade, keď potrebujeme nejakú metriku na doladenie modelu, napríklad nastavenie hraníc pravdepodobnosti na prijatie triedy. F_β-skóre počítame ako harmonický priemer presnosti a pokrytia, kde priradíme pokrytiu istú váhu β, ktorá symbolizuje dôležitosť pokrytia oproti presnosti. Väčšie β znamená vyššiu dôležitosť pokrytia. Vzorec na výpočet je nasledovný:

$$\text{F}_\beta(c) = (1 + \beta^2) \cdot \frac{\text{presnosť}(c) \cdot \text{pokrytie}(c)}{\beta^2 \cdot \text{presnosť}(c) + \text{pokrytie}(c)} = \frac{(1 + \beta^2) \cdot \text{TP}(c)}{(1 + \beta^2) \cdot \text{TP(c)} + \beta^2 \cdot \text{FN}(c) + \text{FP}(c)}$$

Tieto hodnoty sa vždy vypočítajú iba pre jednu konkrétnu triedu. Aby sme určili celkovú presnosť modelu v prípade viactriednej klasifikácie, môžeme výsledky metriky pre jednotlivé triedy napríklad spriemerovať, alebo inak skombinovať, alebo budeme posudzovať každú triedu samostatne [28, s. 88–89] [29].

3.2.1.5 Matica chyby

Jeden zo spôsobov, ako pomerne jednoducho vizualizovať výsledky modelu voči reálnym hodnotám a následne aj vypočítať hodnoty potrebné pre výpočet metrík, je matica chyby. Budeme sa

zaoberať maticou pre klasifikačnú úlohu, kde každému vstupu priradíme práve jednu triedu. Jednoduchou verziou takéhoto problému je binárna klasifikácia, kde určujeme iba či pre vstup niečo platí alebo nie. Takúto maticu máme uvedenú v Tabuľke 3.1. Riadky v tejto matici reprezentujú reálne hodnoty a stĺpce reprezentujú predikované hodnoty. Môžeme pozorovať, že z tejto matice sa dajú odčítať hodnoty potrebné pre výpočet vyššie spomínaných metrík.

		Predikovaná hodnota	
		Pravda	Nepravda
Reálna hodnota	Pravda	18	3
	Nepravda	6	19

Tabuľka 3.1 Matica zámen pre binárnu klasifikáciu

V prípade, že tried je viac, je táto matica komplikovanejšia, ale stále pochopiteľná. Zadefinujem preto, ako bude táto matica vyzerať pre všeobecný počet tried a následne ako bude vyzerať výpočet hodnôt potrebných pre výpočet spomenutých metrík v takomto prípade [28, s. 86-87].

▶ **Definícia 3.5.** *Majme klasifikačný problém, pri ktorom je potrebné priradiť vstupom $v_1, v_2, \ldots, v_n$ práve jeden prvok z množiny tried $C = \{c_1, c_2, \ldots, c_m\}$. Pre tieto účely si definujeme funkciu:*

$$p(v_i, c_j) = \begin{cases} 0 & \text{pre vstup } v_i \text{ neplatí trieda } c_j \\ 1 & \text{pre vstup } v_i \text{ platí trieda } c_j \end{cases}$$

kde v_i je vstupom a c_j je možnou triedou. S podobnými podmienkami si potom zadefinujeme dve funkcie simulujúce model, pre ktoré platí:

$$m(v_i, c_j) = \begin{cases} 0 & \text{model pre vstup } v_i \text{ nepredpovedá platnosť } c_j \\ 1 & \text{model pre vstup } v_i \text{ predpovedá platnosť } c_j \end{cases}$$

Maticu zámen $M \in \mathbb{N}_0^{m,m}$ následne definujeme takto:

$$M_{i,j} = \sum_{k=1}^{n} (p(v_k, c_i) \times m(v_k, c_j))$$

V tejto matici riadky reprezentujú reálne hodnoty a stĺpce predpovedané hodnoty.

Vďaka definícii vieme teraz všeobecne určiť výpočet potrebných hodnôt na výpočet spomenutých metrík. Tieto hodnoty sa dajú odpozorovať aj z matice, ale je možné ich definovať aj bez jej použitia.

▶ **Veta 3.6.** *Majme vstupy $v_1, v_2, \ldots, v_n$ a majme triedu $c_i \in \{c_1, c_2, \ldots, c_m\}$. Potom pre túto triedu platí:*

- $TP(c_i) = \sum_{j=1}^{n} (p(v_j, c_i) \times m(v_j, c_i))$
- $FP(c_i) = \sum_{j=1}^{n} ((1 - p(v_j, c_i)) \times m(v_j, c_i))$
- $FN(c_i) = \sum_{j=1}^{n} (p(v_j, c_i) \times (1 - m(v_j, c_i)))$

Uvedené výpočty sa dajú efektívnejšie vykonať pomocou matice chyby $M \in \mathbb{N}_0^{m,m}$, ktorá reprezentuje výsledky nášho modelu. Znovu budeme ukazovať výsledky pre triedu c_i.

- $TP(c_i) = M_{j,j}$
- $FP(c_i) = \sum_{j=1}^{m} M_{j,i}$ *,kde* $i \neq j$
- $FN(c_i) = \sum_{j=1}^{m} M_{i,j}$ *,kde* $i \neq j$

Vďaka tomuto vieme vypočítať spomínané metriky pri klasifikačnej úlohe, kde každému vstupu priraďujeme práve jednu triedu z ľubovoľného počtu tried. Tento postup by sa v prípadoch iných typov klasifikácie mohol meniť. V prípade, že by mohla byť klasifikovaná ľubovoľná podmnožina tried, by boli tieto výpočty komplikovanejšie. Reprezentáciu matice zámen vidieť v Tabuľke 3.2. Je bežné upraviť maticu tak, že sa hodnoty normalizujú po riadkoch, aby súčet skóre všetkých predikcií v riadku bol 1.

		Predikovaná hodnota		
		Trieda A	Trieda B	Trieda C
Reálna hodnota	Trieda A	18	1	3
	Trieda B	6	19	10
	Trieda C	2	0	23

Tabuľka 3.2 Matica zámen pre problém s troma triedami klasifikácie

3.2.2 Metriky regresie

Pri hodnotení regresných problémov sa nemôžeme spoliehať na konečný počet košov, do ktorých musí padnúť výsledok. Aj tie najlepšie modely môžu mať problém s presnosťou a málokedy budú predpovedať výsledok dokonale presne. Toto musíme zohľadniť aj pri výbere metrík na meranie chyby. Pri popise metrík sa budeme zaoberať hodnotením modelov a problémov, kde je výstup tvorený iba jednou hodnotou. V prípade viacdimenzionálnych hodnôt sa vlastnosti týchto metrík nebudú meniť a ich princíp zostane rovnaký.

3.2.2.1 Priemerná absolútna chyba

Priemerná absolútna chyba, označovaná ako MAE (Mean Absolute Error), popisuje priemer euklidovských vzdialeností reálnych hodnôt od ich predikcií. Ak máme $X, Y \in \mathbb{R}^n$, kde X sú reálne hodnoty a Y sú predikcie také, že $X = \{x_1, x_2, \ldots, x_n\}$ a $Y = \{y_1, y_2, \ldots, y_n\}$, tak MAE vypočítame:

$$MAE(X, Y) = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n}$$

Táto metrika je veľmi základná. Jej výhoda je, že pracuje s hodnotami, ktoré sú ľahko pochopiteľné, keďže vracia výsledky v rovnakých jednotkách, aké vracia model. Teda ak model napríklad odhaduje vzdialenosť v metroch, tak aj MAE vráti chybu v metroch [28, s. 39].

3.2.2.2 Priemerná kvadratická chyba

Priemerná kvadratická chyba, označovaná tiež MSE (Mean Squared Error), chybu meria pomocou priemeru druhých mocnín vzdialeností predikcií od reality. Pre reálne hodnoty a predikcie $X, Y \in \mathbb{R}^n$, také, že $X = \{x_1, x_2, \ldots, x_n\}$ a $Y = \{y_1, y_2, \ldots, y_n\}$ MSE počítame:

$$MSE(X, Y) = \frac{\sum_{i=1}^{n} (x_i - y_i)^2}{n}$$

Výhoda tejto metriky je najmä v tom, že výrazne viac penalizuje veľké rozdiely. Ak sa absolútna chyba zdvojnásobí, tá kvadratická sa stane štvornásobnou. Táto metrika môže byť užitočná, ak akceptujeme, že model sa bude bežne mýliť o malé odchýlky, ale chceme sa vyhnúť veľkým chybám. Ak preferujeme malú častú chybu oproti menej častej, ale výrazne väčšej chybe, je potrebné použiť práve túto metriku alebo metriku s podobnými vlastnosťami.

Jej nevýhoda spočíva v slabej vysvetliteľnosti, keďže chyba je mocnená na druhú. Je ťažké predstaviť si v praxi chybu modelu iba pomocou tohto čísla. Pri trénovaní modelu, ktoré často predstavuje hľadanie optima funkcie, však môže byť použitá.

3.2.2.3 Odmocnina priemernej kvadratickej chyby

Metrika RMSE (Root Mean Squared Error) rieši problém vysvetliteľnosti, keďže vracia výsledky v podobných jednotkách ako MAE. Dosahuje to tým, že výsledok MSE umocní.Pre reálne hodnoty a predikcie $X, Y \in \mathbb{R}^n$, také, že $X = \{x_1, x_2, \ldots, x_n\}$ a $Y = \{y_1, y_2, \ldots, y_n\}$ RMSE počítame:

$$RMSE(X, Y) = \sqrt{\frac{\sum_{i=1}^{n}(x_i - y_i)^2}{n}}$$

RMSE má podobné vlastnosti ako MSE v spôsobe penalizácie väčších chýb. Ak teda chceme porovnať modely a zároveň chceme, aby výsledky týchto metrík boli ľahko vysvetliteľné, práve RMSE nám to môže umožniť [28, s. 37-39].

3.3 Základ neurónových sietí

Zopakovať to, čo funguje, je prirodzený postup, ktorý sa používa pri riešení problémov. Keby to tak nebolo, koleso by sme vymýšľali každý druhý týždeň. Preto, keď sa prví pionieri umelej inteligencie zamýšľali, ako vytvoriť systém schopný vykonávať náročné úlohy a ukazovať známky inteligencie, veľmi neskromne sa pozreli na to najinteligentnejšie, čo vedeli nájsť, na seba. Táto introspektíva vyústila do vzniku neurónových sietí, ktoré sa dnes stávajú čím ďalej tým používanejšími a vďaka nim sú dnes stroje najbližšie k našej inteligencii v ľudskej histórii (aj keď stále extrémne ďaleko).

3.3.1 Základná jednotka neurónových sietí

Pre pochopenie fungovania neurónových sietí je kľúčové pochopiť ich základnú jednotku. Nič to nezmení na tom, že tieto siete aj potom ostanú „black box“ a nebudeme vedieť vysvetliť ich jednotlivé rozhodnutia. Ich lepšie pochopenie ale môže pomôcť riešiť problémy, ktoré počas vytvárania neurónových sietí určite nastanú.

Základnou jednotkou neurónových sietí je neurón, ktorý sa pri samostatnom použití zvykne nazývať aj perceptron. Je síce založený a inšpirovaný na neurónoch v našom mozgu, ale pre naše účely sa jedná iba o jednoduchú parametrickú funkciu.

▶ **Definícia 3.7.** *Buď $x = \{x_1, x_2, \ldots, x_n\} \in \mathbb{R}^n$ vektor vstupov, váhy $w = \{w_1, w_2, \ldots, w_n\} \in \mathbb{R}^n$, kde $n \in \mathbb{N}$, buď bias $b \in \mathbb{R}$ a buď aktivačná funkcia a: $\mathbb{R} \to \mathbb{R}$. Potom definujeme neurón ako funkciu:*

$$p_w(x) = a(x^T \times w + b)$$

Ľahko vidieť, že v prípade, že aktivačná funkcia je lineárna, bude výsledok neurónu iba lineárnou kombináciou vstupného vektora, ku ktorej pričítame konštantu. V takomto prípade nezáleží na tom, koľko rôznych neurónov v akej architektúre použijeme, výsledok bude stále iba lineárnou kombináciou vstupov. Je teda potrebné, aby aktivačná funkcia bola nelineárnou. Rôzne typy aktivačných funkcií predstavíme neskôr [28, s. 259-260].

3.3.2 Učenie neurónovej siete

Už sme ukázali, ako neuróny fungujú, ale nie, ako sa pomocou nich dajú riešiť úlohy. Výsledok očividne veľmi záleží od vstupných váh a bias-u. Záleží aj od aktivačnej funkcie, ale tej sa budeme podrobnejšie venovať neskôr.

Samotný problém učenia vieme zadefinovať ako problém hľadania optima. Už sme si ukázali, že výsledok jedného neurónu je iba funkcia s parametrami a ak ich zapojíme za sebou do vrstiev, nič sa na tom nezmení, iba bude táto funkcia komplikovanejšia s väčším počtom parametrov. Optimalizovať nebudeme výstup našej siete, ale funkciu „vzdialenosti“ od reálnych výsledkov.

Optimalizovať túto funkciu analyticky je sizyfovská práca, preto je potrebné k tomuto problému pristúpiť inak.

V súčasnosti používaný postup sa nazýva spätná propagácia. Model najskôr dostane na vstup trénovacie dáta, s ktorými postupne počíta výstup každého neurónu až k poslednej, výstupnej, vrstve. Potom vypočíta hodnotu chyby siete a následne vypočíta, ako veľmi k nej prispievajú jednotlivé predchádzajúce neuróny. Predchádzajúcim neurónom túto informáciu spätne propaguje a tie tento proces opakujú, až kým sa nedostane ku vstupnej vrstve. Tento spätný prechod počíta gradient, „strmosť“ chyby, pre všetky váhy prepojení a robí to propagovaním gradientu chyby. Toto sa dosahuje postupným počítaním derivácií výstupov neurónov. Po vypočítaní gradientu pre všetky vstupné dáta, je tento gradient pre všetky parametre spriemerovaný a dochádza ku „kroku“, úprave váh podľa veľkosti gradientu. Veľkosť tejto úpravy záleží ešte aj od dôležitého parametra nazývaného rýchlosť učenia alebo learning rate. Taktiež sa bežne nepočíta gradient cez celé dáta, ale tie sa delia na množstvo rovnako veľkých skupín (batch), a zmena váh je vykonaná pre každú takúto skupinu [28, s. 263-264]. Tento proces opakujeme na celých dátach, až kým sa skóre neustáli a nezačne konvergovať ku nejakej hodnote. Jedno spracovanie kompletných dát bez opakovania sa nazýva epocha. Detailnejší popis algoritmu sa nachádza v originálnej štúdii [30]. V súčasnosti sa už bežne používajú pokročilejšie algoritmy, ako napríklad Adam, Adagrad, Nadam a ďalšie. Tie však stále pracujú na veľmi podobnom princípe.

3.3.3 Aktivačné funkcie

Aktivačná funkcia je kľúčová časť fungovania neurónovej siete. Už sme sa vyjadrili, že ak sa jedná o lineárnu funkciu, náš model bude bez ohľadu na jeho veľkosť vedieť modelovať iba lineárne vzťahy medzi vstupom a výstupom. Preto si teraz predstavíme niekoľko rôznych aktivačných funkcií, ktoré sa bežne používajú v neurónoch. Všetky sú nelineárne, čím umožňujú modelom riešiť aj nelineárne problémy.

3.3.3.1 Sigmoida

Táto aktivačná funkcia je relatívne blízko tomu, ako funguje aktivácia neurónov v našom mozgu. Preto sa javí ako logická voľba, keďže existencia neurónových sietí je celá založená na napodobňovaní fungovania mozgu. Sigmoida je definovaná nasledovne:

$$\sigma(x) = \frac{1}{1 + \epsilon^{-x}}$$

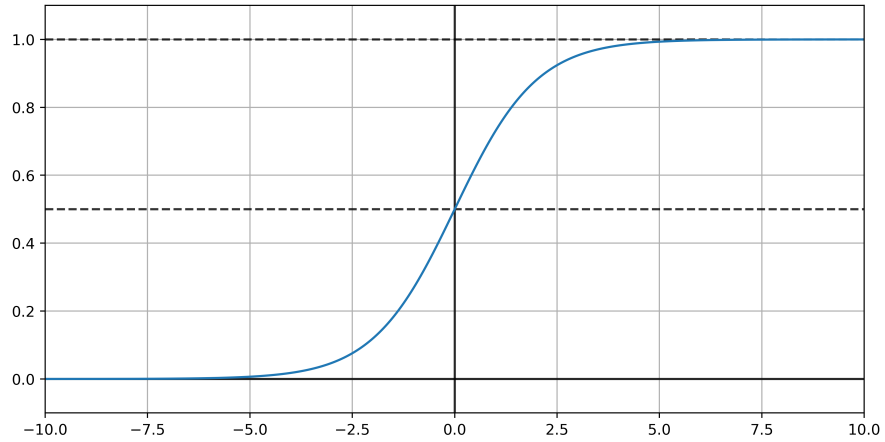
Keď sa pozrieme na tvar sigmoidy na Obrázku 3.1, všimneme si hneď niekoľko zaujímavých vlastností. Ako prvé vidieť, že jej výsledné hodnoty sú obmedzené do intervalu 0 až 1, kde sa nachádzajú aj jej limity v $-\infty$, respektíve v $+\infty$. Táto vlastnosť môže byť vhodná napríklad v prípade klasifikácie, keď chceme výstupný neurón, ktorý by nám vrátil pravdepodobnosť klasifikácie ako triedu.

Nevýhoda sigmoidy spočíva v tom, že limita jej derivácie v $\pm\infty$ je 0. To má za následok, že vypočítaný gradient počas spätnej propagácie vie byť veľmi malý, čím sa trénovanie extrémne spomalí. Tento jav sa zvykne nazývať problém miznúceho gradientu. Je možné ho čiastočne riešiť pomocou špeciálnych distribúcií váh na začiatku trénovania, ale omnoho častejšie sa používa iná aktivačná funkcia [28, s. 277-279].

3.3.3.2 ReLU

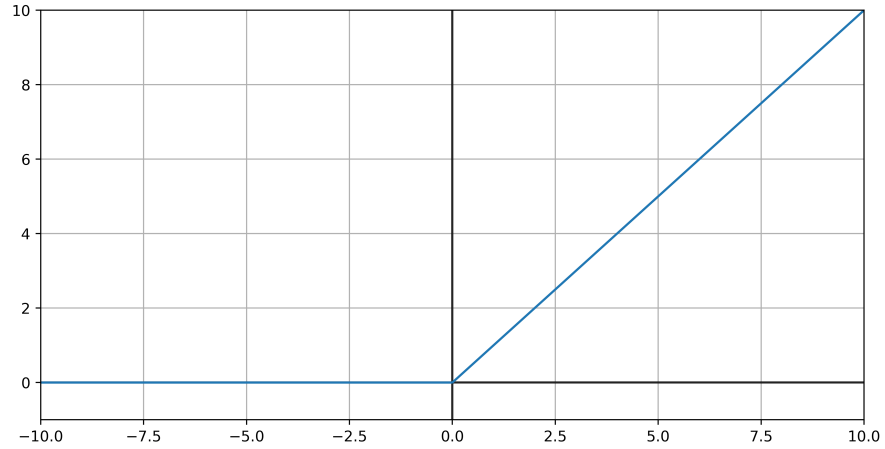
ReLU aktivačná funkcia sa používa častejšie, pretože pri nej nevzniká problém miznúceho gradientu. Je to vďaka tomu, že jej derivácia je pri kladných číslach konštantná. Jej definícia je

$$\mathrm{ReLU}(x) = \begin{cases} 0 & x \leq 0 \\ x & \text{inak} \end{cases}$$



Obr. 3.1 Graf zobrazujúci výsledné hodnoty funkcie sigmoidy

Nedostatkom ReLU aktivačnej funkcie je, že pri záporných hodnotách je hodnota derivácie 0 a pre $x = 0$ dokonca ani neexistuje. Toto má za následok, že počas trénovania neurónu s ReLU aktivačnou funkciou môže nastať „úmrtie“ neurónu, keď vstup do funkcie bude vždy záporný, a teda neurón bude stále vracať hodnotu 0. Keďže derivácia aktivačnej funkcie bude tiež 0, bude pre neurón nulový aj vypočítaný gradient a prestane sa učiť, takže sa z tohto stavu už nezotaví [28, s. 281]. Úmrtie jednotiek neurónov alebo malého množstva neurónov relatívne voči veľkosti siete nemusí spôsobiť problém. Ak však začne byť toto číslo príliš vysoké, je potrebné tento problém riešiť, keďže sa znižuje počet využívaných parametrov, a teda aj kapacita siete.



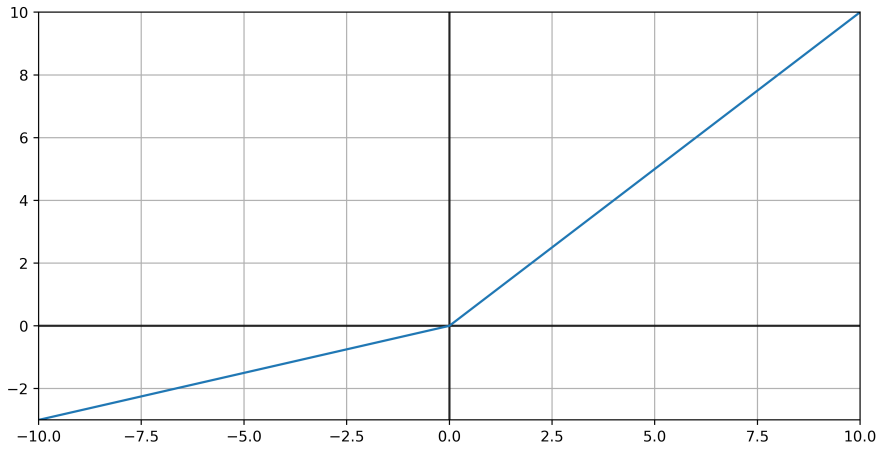
Obr. 3.2 Graf zobrazujúci výsledné hodnoty funkcie ReLU

3.3.3.3 LeakyReLU

Jedna z možností riešenia problému umierajúcich ReLU neurónov je pomocou tzv. LeakyReLU. Táto funkcia pracuje s parametrom α, ktorý je pevne nastavený na konštantnú hodnotu (existujú aj aktivačné funkcie na základe LeakyReLU, kde je α nastavované náhodne alebo je to parameter, ktorý sa tiež optimalizuje) [28, s. 281]. Štúdia porovnávajúca výkon ReLU aktivačných funkcií došla k názoru, že LeakyReLU je takmer vždy lepšou voľbou a dosahuje lepšie výsledky ako obyčajné ReLU [31].

$$\text{LeakyReLU}(x) = \begin{cases} \alpha x & x \leq 0 \\ x & \text{inak} \end{cases}$$

Z definície vidieť, že pre záporné čísla existuje konštantá derivácia, takže aj pre záporné vstupy sa bude neurón pomaly učiť. Tempo bude síce pomalšie, ale neurón nebude mŕtvy. Hodnota α použitá napríklad v knižnici keras je 0,3 [32].



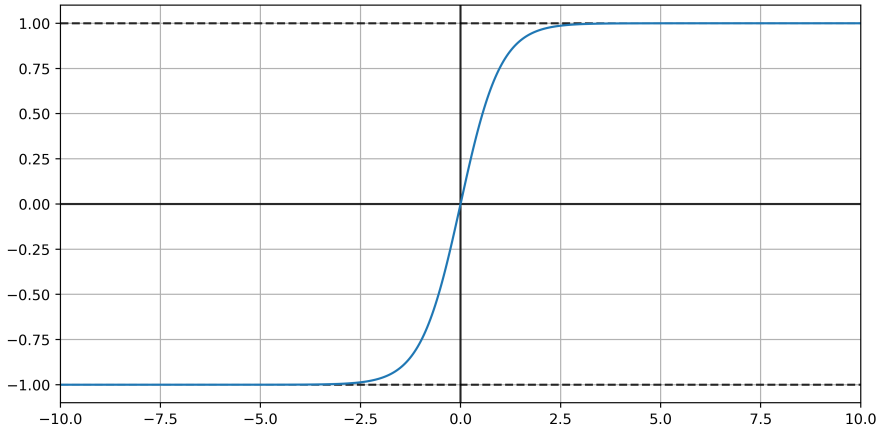
Obr. 3.3 Graf zobrazujúci výsledné hodnoty funkcie LeakyReLU

3.3.3.4 Hyperbolický tangens

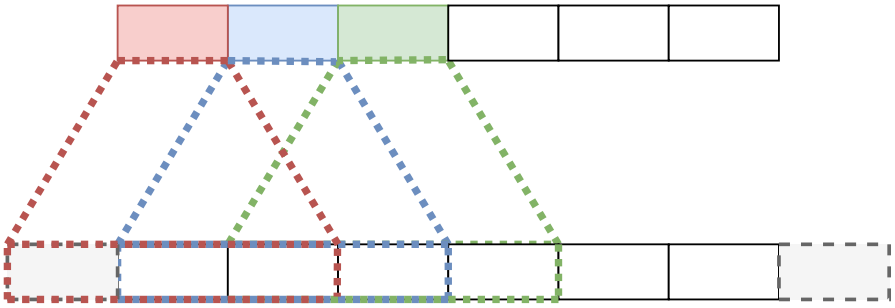
Táto funkcia sa dá definovať pomocou sigmoidy ako

$$\tanh(x) = 2 * \sigma(2x) - 1$$

So sigmoidou má hyperbolický tangens aj veľmi podobné vlastnosti. Hlavný rozdiel je v tom, že jej výsledné hodnoty sú v rozmedzí -1 až 1. Aj v prípade tejto aktivačnej funkcie existuje problém miznúceho gradientu. Nás zaujíma najmä preto, že je používaná v špeciálnej LSTM jednotke [28, s. 264].



Obr. 3.4 Graf zobrazujúci výsledné hodnoty hyperbolického tangensu



Obr. 3.5 Ukážka konvolúcie na 1D dátach s veľkosťou okna 3

3.4 Architektúry vrstiev

Už vieme, ako funguje jednotlivý neurón, ale málokedy nám stačí použiť iba jeden. Neurónové siete v súčasnosti obsahujú tisíce neurónov, ktoré sú usporiadané v rôznych vrstvách s rôznymi funkciami. Počet vrstiev nám určuje, či môžeme hovoriť o hlbokom učení. Teraz sa pozrieme na to, aké vrstvy v neurónových sieťach poznáme a aké býva ich využitie.

3.4.1 Plne prepojená vrstva

Plne prepojená vrstva je najjednoduchšia vrstva v neurónovej sieti. Je tvorená skupinou jednoduchých neurónov, ktoré na vstupe prijímajú všetky výstupy predchádzajúcej vrstvy. Napriek svojej jednoduchosti stále môže byť využívané po špecializovaných vrstvách, kde rozhodne na základe vyťažených znalostí zo špecializovaných vrstiev.

3.4.2 Konvolučná vrstva

Konvolučná vrstva je najdôležitejšia súčasť konvolučných neurónových sietí (CNN), ktoré sú najčastejšie používané na obrázky. Upravené verzie však dosahujú state-of-the-art výsledky aj v problémoch, ktoré sa venujú časovým radám, napríklad text-to-speech alebo strojové generovanie hudby [33].

Hlavná myšlienka konvolúcie je, že neuróny nemajú na vstupe všetky dáta, ale iba malú časť vo svojom „zornom poli“. Toto zorné pole môže čiastočne vychádzať mimo signál. Vtedy sa zvyknú dokladať za takto chýbajúce hodnoty 0, prípadné sa inak vyberajú hodnoty na doplnenie.

Pri konvolúcii rozlišujeme jednotlivé filtre. Každý filter prechádza signálom ako na Obrázku 3.5. Jedná sa však stále o ten istý neurón, ktorý je iba „skopírovaný“. Preto keď pridáme ďalšie neuróny, hovoríme o ďalších filtroch. Ukážka s množstvom neurónov je iba pomôcka, v skutočnosti sa pri jednom filtre jedná iba o jeden neurón, ktorý však postupne dostane na vstup všetky svoje „zorné polia“ a tieto výsledky sú zoradené na výstupe. Toto je potom dôležité pri učení, keď sa pri spätnej propagácii pre každý filter zohľadňujú všetky tieto výstupy. Taktiež je možné využiť viacero filtrov s rôznymi veľkosťami. Každý filter taktiež posiela ďalej celú výstupnú mapu príznakov.

Výhoda konvolúcie je, že dokáže dobre získavať z dát lokalizované príznaky, čo môže byť pri spracovaní signálu a aj detekcii artefaktov veľmi užitočné. Bežne sa umiestňuje viacero vrstiev za sebou, čo umožňuje získať komplexnejšie príznaky [28, s. 359-364].

Určitá nevýhoda je, že potrebuje presne určenú veľkosť konvolučných okien. Príliš veľké zorné pole bude mať za následok, že konvolúcia nebude výrazne lepšia vo vyťažovaní informácií ako hustá vrstva, príliš malé bude znamenať, že nemusí zachytiť príznaky, ktoré sú väčšie. Proces určenia veľkosti okien vie byť časovo náročný a vyžaduje množstvo experimentov a pokusov [34].

3.4.3 Pooling vrstva

Hlavná myšlienka tejto vrstvy je zmenšiť výsledok klasickej konvolučnej vrstvy, čo má pozitívne vplyvy na výpočetnú náročnosť a aj množstvo parametrov, čo znižuje riziko preučenia. Táto vrstva funguje rovnako ako konvolučná vrstva, s tým rozdielom, že namiesto výsledku neurónu vracia iba agregáciu svojich vstupov. Bežná agregácia je buď priemer alebo maximum [28, s. 367–368].

Špeciálna verzia pooling vrstvy sa nazýva globálna pooling vrstva. Jej hlavná myšlienka je odstrániť potrebu hustých vrstiev po konvolúcii na získanie samotného výsledku, čím sa znižuje riziko pretrénovania. Táto vrstva vracia agregáciu pre každú mapu príznakov. Túto agregáciu počíta cez celé mapy [35]. Ako agregácia sa tiež bežne používa maximum alebo priemer.

3.4.4 LSTM

LSTM je označenie pre špeciálnu jednotku určenú na spracovanie sekvenčných dát. Jedná sa o rozšírenie predchádzajúcich rekurzívnych neurónových sietí (RNN), ktoré sa snažili o to isté, ale mali problém naučiť sa dlhodobé vzory v dátach. LSTM jednotka je dizajnovaná práve aby tento problém riešila [36].

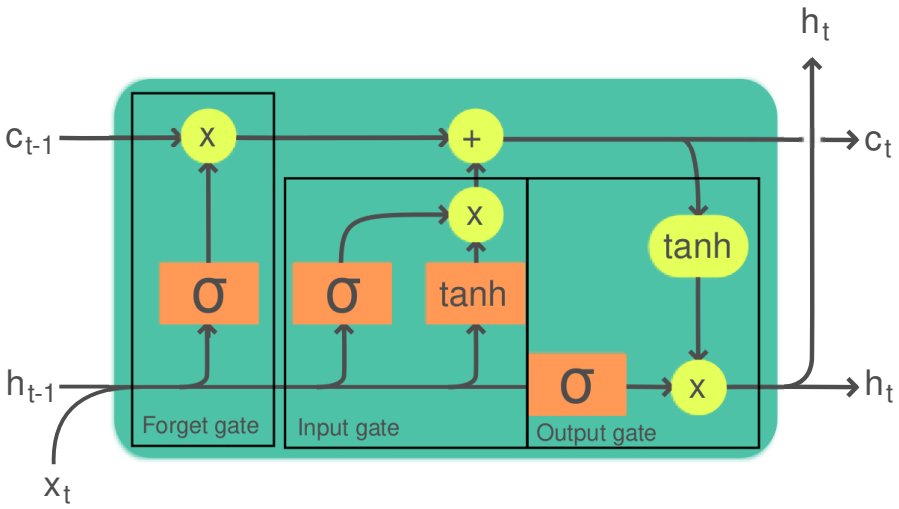
LSTM spracuje dáta sekvenčne po krokoch a pri každom kroku používa aj výsledok toho predchádzajúceho. Počas kroku t prichádzajú na vstup LSTM jednotky dáta x_t (teda v čase t), stav LSTM jednotky na konci predchádzajúceho kroku c_{t-1} a výsledok LSTM jednotky v predchádzajúcom kroku h_{t-1}. LSTM z týchto vypočíta nový stav c_t a nový výsledok h_t. h_t si môžete predstaviť ako nositeľa krátkodobej informácie a c_t ako nositeľa dlhodobej informácie. Nič nám nebráni použiť viac LSTM jednotiek vo vrstve, kde by každá spracovávala dáta nezávisle od ostatných. Častejšie sa však používa jedna jednotka, ktorá vnútorné využíva jednoduché neurónové siete. Ak chceme zvýšiť kapacitu LSTM jednotky, zväčšíme veľkosti vnútorných neurónových sietí. Ako výsledok sa bežne používa výsledok posledného kroku LSTM. V prípade, že chceme zapojiť viacero LSTM vrstiev za seba, vrátime sekvenciu všetkých výsledkov, ktoré následne ďalšia LSTM vrstva sekvenčne spracuje. Sekvenciu všetkých výsledkov môžeme vrátiť, aj keď ďalšia vrstva nie je LSTM typu.

Popíšeme si, ako funguje LSTM jednotka, jej vlastnosti a prípadné slabé stránky. Jej architektúru delíme na tri hlavné časti, takzvané brány. Dlhodobý stav c_{t-1} prechádza cez sieť zľava doprava. Najskôr prejde cez forget gate, kde prichádza o nejaké informácie, následne v input gate získava informácie vybraté vnútri tejto brány prostredníctvom operácie pričítania. Výsledok týchto operácií sa okamžite označí za c_t a pošle sa na výstup. Tento stav je po sčítaní ešte skopírovaný a po prejdení hyperbolickým tangensom je prefiltrovaný výstupom output gate, čím získame h_t.

V bránach sa nachádzajú husté vrstvy. Počet neurónov v nich odpovedá dimenzii výstupu. Tieto vrstvy zväčšujeme, ak chceme zväčšiť kapacitu LSTM. Špeciálna je vrstva v input gate, ktorá využíva hyperbolický tangens ako aktivačnú funkciu. Úlohou tejto vrstvy je analyzovať vstup x_t a predchádzajúci výstup h_{t-1}. Ďalšie tri vrstvy plnia úlohu kontroly. Každá z nich má ako aktivačnú funkciu sigmoidu, teda ich výstup je 0 až 1. Každá z nich teda rozhoduje, ktoré príznaky sa ponechajú a ktoré budú potlačené. Vo forget bráne sa určuje, ktoré z dlhodobých príznakov sa potlačia. V input bráne sa rozhoduje, čo všetko sa pridá k dlhodobému stavu a v output gate sa rozhoduje, ktoré časti pôjdu do výstupu [28, s. 405-407].

Vnútri LSTM sa používajú sigmoida a hyperbolický tangens, ktoré sú náchylné na miznúci gradient. Napriek tomu LSTM v porovnaní s predchádzajúcimi RNN tento problém čiastočne rieši. RNN sú tiež náchylné na problém explodujúceho gradientu. Tento problém je opačný problému miznúceho gradientu, tu sa gradient počas tréningu stále zväčšuje. Vrstvy majú stále väčšie a väčšie úpravy váh a skóre siete začne divergovať. Tréningový proces začne byť numericky nestabilný a môže až zlyhať. Aj tento problém LSTM jednotky do istej miery riešia. Stále však môžu nastať oba z týchto problémov. Riziko, že sa tak stane, stúpa, čím dlhšiu sekvenciu spracovávame

[36].



■ **Obr. 3.6** LSTM jednotka [37]

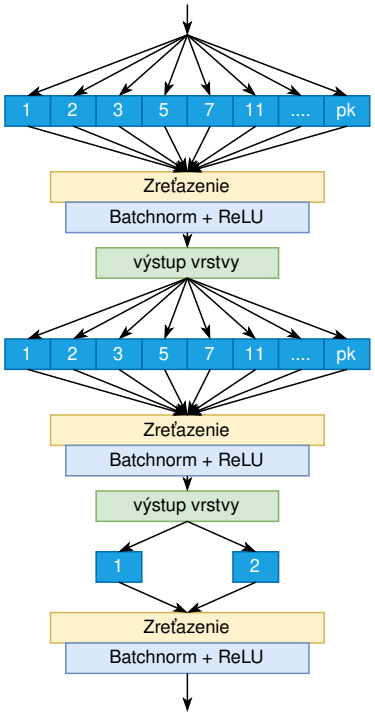
3.4.5 Omni-Scale blok

Jeden z najväčších problémov pri klasifikovaní časových rád je určenie, v akej časovej škále má model pracovať, teda akým dlhým podsekvenciám sa treba venovať a hľadať v nich relevantné príznaky. Pri tvorbe modelu je potrebné túto podsekvenciu identifikovať, čo je náročné, keďže to zahŕňa množstvo rôznych pokusov a trénovanie množstva modelov. Pri výbere dĺžky relevantnej podsekvencie je kľúčová dĺžka konvolučného okna. Väčšinou sa veľkosť tohto okna hľadá ako hyperparameter, avšak v niektorých prípadoch môže byť potrebné použiť filtre s rôznou veľkosťou.

Omni-Scale blok (OS blok) je riešenie navrhnuté na obmedzenie zbytočného hľadania a skúšania parametrov siete. Tento blok navrhuje univerzálne pravidlo, ktoré pokrýva dĺžky všetkých podsekvencií a namiesto hľadania optimálnych veľkostí filtrov sa tieto veľkosti učí počas tréningu.

Architektúra OS bloku vychádza z Goldbachovej hypotézy. Tá znie, že každé párne kladné číslo sa dá vyjadriť ako súčet dvoch prvočísel. Nepodarilo sa ju dokázať, ale bola overené pre hodnoty výrazne vyššie ako budeme kedykoľvek potrebovať [38]. Celkovo sa jeden OS blok skladá z troch konvolučných vrstiev. V prvej aj druhej vrstve umiestnime filtre veľkosti všetkých prvočísel menších ako štvrtina dĺžky dát a filter veľkosti 1. V poslednej vrstve budeme mať iba dva filtre, veľkosti 1 a 2. Toto nám umožní mať pokryté všetky možné dĺžky podsekvencií signálu, nielen párne. Po každej vrstve filtrov sa nachádza tzv. batch normalizácia a aktivačná funkcia ReLU. Diagram popisujúci architektúru jedného OS bloku vidieť na Obrázku 3.7

Podľa tvorcov OS bloku je možné ho použiť viackrát za sebou, aj ako súčasť moderných architektúr na mieste konvolučnej vrstvy. Po poslednom bloku navrhujú použiť globálny pooling, konkrétne verziu s priemerom. Hneď za touto vrstvou nasleduje už iba posledná výstupná vrstva [34] [39].



Obr. 3.7 Architektúra jedného Omni-Scale bloku

3.5 Optimalizácia trénovania

Architektúra sietí je veľmi dôležitá, ale aj sieť so špičkovou architektúrou nebude poskytovať dobré výsledky, ak ju nebudem správne trénovať. Trénovanie každej siete je veľmi špecifické a prináša množstvo problémov. Už sme spomínali problémy, ako sú umierajúce neuróny, miznúci alebo explodujúci gradient. Okrem týchto ale môže nastať množstvo iných problémov, ktoré je počas trénovania potrebné riešiť.

3.5.1 Dropout

Jeden z najbežnejších problémov pri tvorbe modelu je pretrénovanie. To nastáva, keď namiesto toho, aby sa model naučil princípy, ktoré vedú k správnym výsledkom, tak sa naučí trénovací dataset „naspamäť“. Toto sa stáva najmä v prípadoch, kedy má model príliš veľa parametrov.

Existuje viacero spôsobov, ako znižovať riziko, respektíve mieru pretrénovania modelu. Tými sú napríklad L1, respektíve L2 regularizácia. Tieto regularizácie sa snažia zmenšiť váhy v modeli, čo má pozitívny efekt na pretrénovanie. Jeden zo spôsobov na zníženie preučenia, ktorý sa používa napríklad pri rozhodovacích stromoch, je použitie rôznych ensamble metód. Tieto pri neurónových sieťach nie je praktické použiť, keďže ich trénovanie je výrazne časovo a výpočetne náročnejšie.

Spôsob, ktorým sa budeme zaoberať my, je dropout. Jeho princíp spočíva v tom, že počas každého kroku tréningu náhodne „vypneme“ v každej vrstve niekoľko neurónov. Toto spravíme tak, že namiesto ich výsledku pošleme ďalej hodntou 0. Toto má za následok, že model lepšie generalizuje a výrazne ťažšie sa mu dáta učia naspamäť. Podľa pôvodnej štúdie je na husté vrstvy najlepšie aplikovať dropout s mierou 50 %, čo znamená, že v každom kroku vypneme polovicu neurónov [40]. Za konvolučnými sieťami je možné aplikovať dropout s mierou okolo 10 % [41].

Okrem toho je taktiež možné aplikovať dropout aj v LSTM [42].

3.5.2 Batch Normalization

Jeden z dôvodov, prečo je trénovanie neurónových sietí náročné, je fakt, že počas tréningu sa zmenou prechádzajúcich vrstiev mení každému neurónu distribúcia vstupov. Toto má za následok potrebu opatrnejšieho prístupu k trénovaniu, používanie nižšieho learning rate a celkovo k dlhšiemu a náročnejšiemu trénovaniu.

Riešenie tohto problému je batch normalization, ktoré pre každú batch normalizuje vstupné dáta. V prípade viacdimenzionálneho vstupu $x = (x_1, \ldots, x_d)$ normalizujeme každú dimenziu samostatne ako

$$\hat{x}_i = \frac{x_i - \mathrm{E}[x_i]}{\sqrt{\mathrm{var}(x_i)}}$$

kde stredná hodnota a rozptyl sa počítajú pre každú batch samostatne. Následne pri tejto operácii máme dva parametre γ_i a β_i, pomocou ktorých získame výslednú hodnotu

$$y_i = \gamma_i \hat{x}_i + \beta_i$$

.

Tieto parametre trénujeme popri zvyšku modelu. Výhoda ich použitia je, že nám dávajú výrazne väčšiu voľnosť pri transformácii. V krajnom prípade umožňujú dokonca aj to, že batch normalizácia dáta vôbec nezmení.

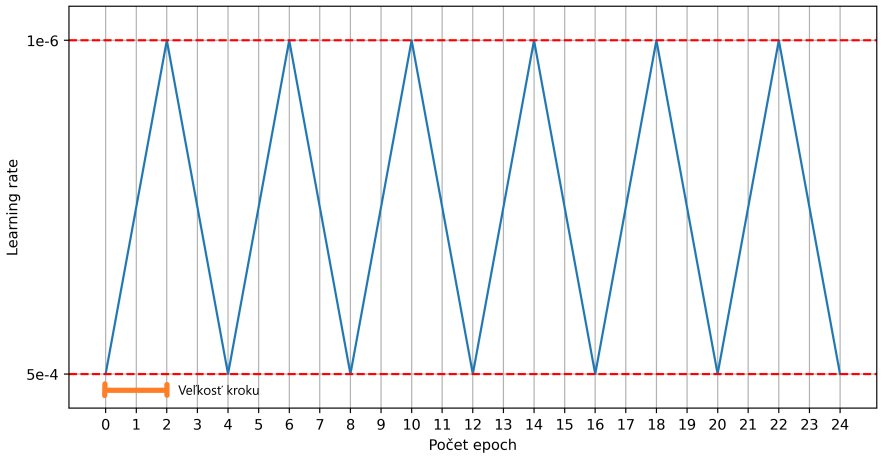
Táto metóda vykazuje jasné pozitíva, ako je zrýchlenie tréningu a bránenie preučeniu. Dokáže niekoľkonásobne zrýchliť tréningy modelov a pomaly sa stáva bežnou súčasťou pokročilých architektúr [43].

3.5.3 Cyklický learning rate

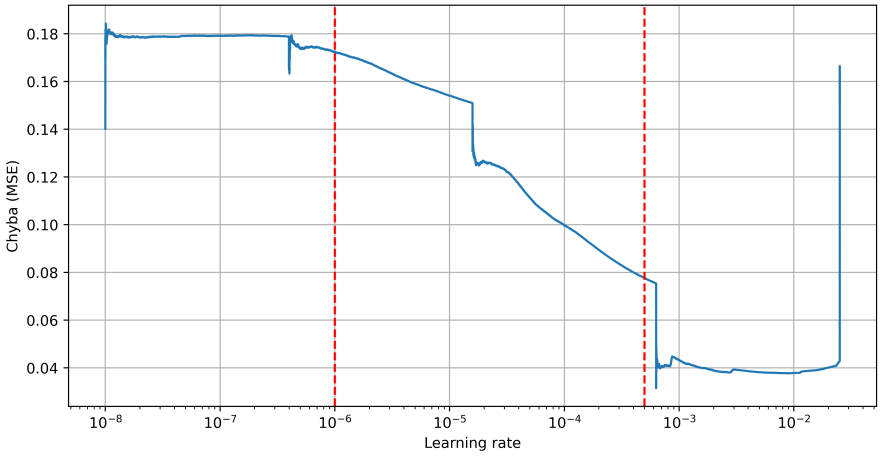
Jeden z najdôležitejších hyperparametrov neurónových sietí je learning rate. Tento parameter je potrebné často hľadať experimentálne a takto treba hľadať aj proces a postup jeho znižovania. Alternatívny postup je však takzvaný cyclický learning rate (CLR), ktorého myšlienka spočíva v tom, že odhadneme rozmedzie „dostatočne dobrého“ learning rate, v ktorom ho budeme cyklicky meniť, teda lineárne zvyšovať, kým prídeme k hornej hranici a potom lineárne znižovať, kým prídeme k spodnej hranici. Tento proces opakujeme. Pri trénovaní určíme, ako dlho má trvať prechod od minima k maximu a naopak. Táto doba sa nazýva jeden krok a určuje sa v počte epoch. Bežne sa určuje na jednotky epoch, napríklad 2 až 8. Príklad vývoja learning rate môžeme vidieť na Obrázku 3.8, kde je rozmedzie 10^{-6} až 5×10^{-4} a veľkosť kroku je nastavená na 2 epochy.

Spôsob, akým určujeme rozmedzie hodnôt learning rate je taký, že náš model trénujeme po dobu niekoľkých epoch a počas nich learning rate lineárne zvyšujeme. Počas tohto si zapisujeme veľkosť nami sledovanej metriky, ktoré vykreslíme do grafu. Príklad takéhoto grafu je uvedený na Obrázku 3.9. Priebeh grafu nie je uhladený, keďže počas jednotlivých epoch môže skóre klesať aj stúpať. Dôležitý je však celkový trend grafu, nie lokálne odchýlky. Ako minimum určíme hodnotu, keď sa metrika začne zlepšovať, v našom prípade napríklad 10^{-6}. Ako maximum zase určíme hodnotu, kde sa model ešte učí, takže metrika ešte nezačala stagnovať alebo výrazne kolísať, v našom prípade napríklad 5×10^{-4}.

Použitie tejto metódy umožňuje rovnakú kvalitu modelu s menším počtom potrebných epoch na trénovanie. Taktiež nevyžaduje žiadnu výpočetnú kapacitu navyše [44].



■ **Obr. 3.8** Zmena learning rate počas trénovania s intervalom 10^{-6} až 5×10^{-4}



■ **Obr. 3.9** CLR test s vyznačenou maximálnou a minimálnou hodnotou identifikovanou na ďalšie trénovanie

3.6 Hľadanie hyperparametrov

Neurónové siete poskytujú obrovskú variabilitu, čo sa týka ich architektúry. Toto nám dáva do rúk obrovské možnosti, ale zároveň aj špecifický problém. Ako vybrať čo najlepšiu architektúru? Tento problém vieme do istej miery vyriešiť tým, že sa inšpirujeme už existujúcimi architektúrami. To nám však vo väčšine prípadov počet možností výrazne nezmení. Stále potrebujeme zistiť obrovské množstvo hyperparametrov, ako je počet vrstiev, ich veľkosť, typ, miera dropout-u a mnoho ostatných.

3.6.1 Extenzívne hľadanie

Jeden z bežných postupov je celý priestor prehľadať s nejakou mierou diskretizácie. Toto môže byť efektívne v prípade, že hľadaný priestor je malý a vytvoriť model trvá krátko. Ak však pracujeme s neurónovými sieťami, ktorých tréning trvá dlho a ktoré môžu mať obrovské množstvo hyperparametrov, je jeho použitie prakticky nemožné [28, s. 73-75].

3.6.2 Náhodné hľadanie

Tento postup namiesto extenzívneho prehľadávania možností náhodne vyberie vopred určený počet nastavení hyperparametrov a pre ne model natrénuje. Počet modelov si vieme určiť podľa dostupných prostriedkov. Stále je však potrebné vyskúšať veľké množstvo modelov. Napriek tomu je však možné túto alternatívu použiť aj v prípade neurónových sietí [28, s. 75-76].

3.6.3 Bayesovská optimalizácia

Algoritmus bayesovskej optimalizácie sa pri vyberaní ďalších hyperparametrov na otestovanie pozerá na predchádzajúce výsledky a podľa nich sa snaží vybrať čo najlepšie, čo robí s využitím Gaussovského procesu. Tento výber je výpočetne náročnejší ako pri predchádzajúcich spôsoboch, ale v prípade, že hodnotenie parametrov trvá dlho, čo v prípade hlbokého učenia platí, tak je tento čas zanedbateľný v porovnaní s tým, čo nám vie ušetriť. Pri použití tohto postupu vieme tiež určiť maximálny počet modelov, ktoré chceme trénovať, čo nám tiež dáva kontrolu nad časom, ktorý zaberie hľadanie hyperparametrov. Presný popis algoritmu a ako funguje, sa nachádza v štúdii [45].

3.6.4 Hyperband

Hyperband algortimus na hľadanie hyperparametrov je rozšírenie náhodného hľadania. Autori tohto algoritmu popisujú hľadanie hyperparametrov ako problém mnohorukého banditu. V tomto probléme musíme obmedzené prostriedky rozdeliť medzi súťažiace možnosti tak, aby sme maximalizovali ich skóre. Tento algoritmus si vytvorí množstvo náhodne vybraných riešení, ktorým následne priraďuje prostriedky vo forme epoch. Obmedzenie doby behu tohto modelu nie je také jednouché ako v predchádzajúcich dvoch prípadoch, ale stále je možné. Presný popis algoritmu aj s analýzou jeho teoretických vlastností sa nachádza v originálnej štúdii [46].

Kapitola 4
Popis dát

Pri tvorbe modelu je potrebné brať do úvahy aj typ dát, ktoré máme k dispozícii. Bez kvalitných dát, ktoré dobre reprezentujú reálny stav, je veľmi náročné vytvoriť model, ktorý dokáže fungovať v praxi. Kvalita a dostupnosť dát ovplyvňuje kvalitu výstupného trénovaného modelu, keďže je náročnejšie učiť model, ak vstupné dáta neobsahujú relevantné vzorky príkladov. V tejto kapitole rozoberieme, aké dáta budú použité na vytváranie modelu a taktiež ako budú delené na trénovaciu, validačnú a testovaciu množinu.

4.1 BUT QDB

Tento dataset bol vytvorený na FEKT VUT v Brne. Pozostáva z 18 dlhodobých meraní jednozvodového EKG na 15 osobách, z toho 9 žien a 6 mužov s vekom v rozmedzí 21 až 83 rokov. Dáta boli zbierané medzi augustom 2018 a októbrom 2019 počas vykonávania bežných činností. Dáta boli zbierané pomocou zariadenia Faros 180, ktoré podľa dokumentácie meria v zvode II [47] s frekvenciou 1000 Hz. Dáta sú merané v µV. Minimálna dĺžka merania je 24 hodín.

Tri signály boli kompletne anotované. Z ostatných signálov boli anotované z každého záznamu dva segmenty dĺžky 20 minút. Okrem toho bolo označených 5 ďalších segmentov zlej kvality. Signál bol z pohľadu kvality rozdelený na 3 triedy:

- Trieda 1 popisuje, že všetky dôležité časti EKG sú identifikovateľné, teda všetky vlny aj kmity.
- Trieda 2 popisuje, že signál je zašpinený a nie je možné spoľahlivo identifikovať všetky segmenty EKG vrátane ich začiatkov a koncov, ale detekcia QRS komplexu je stále možná.
- Trieda 3 popisuje, že ani QRS komplex nie je možné spoľahlivo detegovať a signál je nevhodný na ďalšiu analýzu.

Všetky anotované intervaly boli najskôr anotované tromi nezávislými odborníkmi a následne bolo vytvorené konsenzuálne riešenie. Súčasťou anotovaných úsekov sú aj zmeny dát, teda sa nejedná o homogénne úseky. Presnejší popis dát, použitých metód a možnosť stiahnutia dát je dostupná na stránke PhysioNet [48] [49].

Keď sa pozrieme na vlastnosti datasetu v Tabuľke 4.1, môžeme vidieť najmä to, že bude potrebné zachytávať aj veľmi krátke úseky so zhoršenou kvalitou. To iba potvrdzuje, čo sme rozoberali pri analýze artefaktov EKG a teda, že tieto môžu byť aj extrémne krátkodobé. To, že tieto úseky predstavujú zhoršenie signálu, usudzujeme z toho, že sú kratšie ako je dĺžka úderu. Vzhľadom na to, že kvalita signálu je úzko spojená s možnosťou identifikácie segmentov EKG, nie je možné, aby sme na takto krátkom úseku identifikovali potrebné segmenty EKG. Muselo teda dôjsť ku zhoršeniu kvality. Celková dĺžka anotovanej časti datasetu je takmer 99 a pol hodín.

	Celková dĺžka signálu podľa kvality				Vlastnosti dĺžok segmentov		
Označenie záznamu	Kvalita 1	Kvalita 2	Kvalita 3	Dokopy	Priemer	Medián	Minimum
100001	16:36:15	07:31:30	00:03:40	24:11:26	00:01:54	00:00:15	0,332 s
103002	00:36:11	00:03:47	00:00:00	00:39:59	00:00:21	00:00:03	0,372 s
105001	16:11:21	09:18:33	13:09:10	38:39:05	00:01:42	00:00:13	0,504 s
111001	10:33:40	13:30:03	10:06:57	25:10:42	00:00:33	00:00:04	0,303 s
100002	00:26:04	00:13:55	00:00:00	00:39:59	00:00:10	00:00:03	0,328 s
103001	00:28:58	00:10:53	00:00:08	00:39:59	00:00:13	00:00:03	0,405 s
125001	00:39:10	00:00:46	00:00:03	00:39:59	00:01:04	00:00:03	0,391 s
103003	00:33:56	00:06:03	00:00:00	00:39:59	00:00:13	00:00:02	0,321 s
104001	00:26:31	00:13:28	00:00:00	00:39:59	00:00:23	00:00:02	0,302 s
126001	00:39:25	00:00:34	00:00:00	00:39:59	00:00:41	00:00:02	0,322 s
113001	00:39:34	00:18:43	00:01:41	00:59:59	00:00:16	00:00:04	0,346 s
115001	00:27:20	00:12:39	00:00:00	00:39:59	00:00:16	00:00:04	0,350 s
118001	00:25:23	00:14:36	00:00:00	00:39:59	00:00:19	00:00:04	0,548 s
121001	00:33:34	00:06:24	00:00:00	00:39:59	00:00:14	00:00:03	0,304 s
122001	00:13:34	00:06:25	00:19:59	00:39:59	00:01:35	00:00:08	0,375 s
123001	00:36:06	00:03:44	00:00:09	00:39:59	00:00:21	00:00:03	0,375 s
114001	00:38:43	00:09:16	00:14:00	01:01:59	00:02:49	00:01:01	0,810 s
124001	00:23:01	00:45:30	00:11:27	01:19:59	00:00:08	00:00:04	0,321 s
Spolu	51:08:55	33:08:55	15:07:18	99:23:12			

Tabuľka 4.1 Vlastnosti BUT QDB datasetu, všetky číselné stĺpce okrem prvého zľava sú vo formáte HH:MM:SS, prvý zľava je v sekundách

4.2 FBMI ČVUT dataset

Druhý dataset poskytnutý z FBMI ČVUT bol získaný počas riadeného 7 denného experimentu. EKG bolo merané 6 mužom vo veku 25 až 55 rokov. Žiadna z týchto osôb v čase merania nemala diagnostikovanú chorobu, ktorá by mohla mať vplyv na priebeh EKG. Meranie bolo vykonané zariadením vyvinutým FBMI ČVUT, ktoré bolo validované voči ostatným zariadeniam na meranie EKG. Dáta boli merané vo zvode II s frekvenciou 250 Hz a boli merané v mV. Tento dataset nie je verejne dostupný.

Tento dataset bol anotovaný expertami z FBMI ČVUT po vzore tried z BUT QDB datasetu. Hlavný rozdiel je v tom, že každý úsek je z pohľadu kvality zložený z homogénneho signálu. Tieto úseky majú rôznu dĺžku. Anotácia prebiehala iba výberom úsekov, ktoré spĺňali podmienky jednotlivých tried a následným vyňatím a uložením signálu. Celková dĺžka datasetu je niečo vyše 6 hodín. Zhrnutie datasetu sa nachádza v Tabuľke 4.2. Presný rozpis jednotlivých úsekov podľa kvality sa nachádza v Tabuľkách 4.3, 4.4 a 4.5.

Kvalita	Počet úsekov	Kompletná dĺžka	Najkratší úsek	Najdlhší úsek	Priemerný úsek	Mediánový úsek
1	13	02:15:00	00:00:28	00:36:56	00:10:59	00:10:53
2	9	00:44:13	00:00:14	00:14:35	00:04:25	00:01:58
3	6	02:23:21	00:02:04	00:59:27	00:28:40	00:12:38

Tabuľka 4.2 Prehľad datasetu z FBMI ČVUT

ID záznamu	Dĺžka
fbmi-1-1	00:06:59
fbmi-1-2	00:03:09
fbmi-1-3	00:00:28
fbmi-1-4	00:03:29
fbmi-1-5	00:15:04
fbmi-1-6	00:00:59
fbmi-1-7	00:04:58
fbmi-1-8	00:36:56
fbmi-1-9	00:14:33
fbmi-1-10	00:11:50
fbmi-1-11	00:09:56
fbmi-1-12	00:13:54
fbmi-1-13	00:12:38
Spolu	02:15:00

Tabuľka 4.3 Zoznam označení signálov najlepšej kvality z FBMI ČVUT

ID záznamu	Dĺžka
fbmi-2-1	00:00:14
fbmi-2-2	00:01:09
fbmi-2-3	00:01:30
fbmi-2-4	00:00:20
fbmi-2-5	00:01:58
fbmi-2-6	00:03:39
fbmi-2-7	00:14:35
fbmi-2-8	00:14:29
fbmi-2-9	00:04:18
Spolu	00:42:13

Tabuľka 4.4 Zoznam označení signálov strednej kvality z FBMI ČVUT

ID záznamu	Dĺžka
fbmi-3-1	00:59:16
fbmi-3-2	00:59:27
fbmi-3-3	00:02:04
fbmi-3-4	00:09:55
fbmi-3-5	00:12:38
Spolu	02:23:21

Tabuľka 4.5 Zoznam označení signálov najhoršej kvality z FBMI ČVUT

4.3 Delenie datasetu

Dostupné dva datasety majú rovnaké pravidlá anotovania. Vzhľadom na odlišné meracie zariadenie a vzhľadom na to, že BUT QDB dataset má aj prechody v kvalite signálu, zatiaľ čo FBMI ČVUT dataset nie, bude vhodné tieto datasety pri delení zmiešať. Podstatný rozdiel spočíva v rozdielnych frekvenciách. Uvedený rozdiel vyriešime znížením frekvencie BUT QDB datasetu na 250 Hz, keďže táto frekvencia je dostatočná na prácu s EKG, čo ukazuje aj fakt, že FBMI ČVUT dataset túto frekvenciu použil. Zatiaľ sa sústredíme iba na delenie záznamov, nie na ich predspracovanie, keďže to závisí od postupu, ktorý zvolíme na vytváranie modelu. Vieme, že vzhľadom na rozdielne jednotky, v ktorých boli dáta v dvoch prípadoch merané, prevedieme BUT QDB dataset tak, aby ako jednotky používal mV.

Datasety sa budeme snažiť deliť tak, aby dĺžky signálu podľa kvality tried boli približne vyrovnané. Nepodarí sa nám vytvoriť úplne vyrovnané datasety bez veľkej redukcie dát, čomu sa chceme vyhnúť. Naším cieľom teda je, aby aspoň testovací, validačný a trénovací dataset boli navzájom približne rovnaké. Zároveň sa budeme snažiť dlhodobý signál čo najmenej rozdeľovať medzi rôzne datasety. Vzhľadom na možné rozdiely medzi dvoma použitými datasetmi sa zároveň pokúsime využiť v každej množine dáta z oboch zdrojov.

Problém, ktorý môže nastať, súvisí so spôsobom našej anotácie a faktom, že pri delení zohľadňujeme iba kvalitu signálu. To síce nevadí pri EKG, ktoré je čisté, avšak čím je signál menej kvalitný, tým viac záleží na zdroji tohto zašpinenia. Ukázali sme si niekoľko artefaktov, ktoré môžu mať zásadný vplyv na kvalitu EKG. Ak by sme v trénovacom datasete použili signál najhoršej kvality spôsobený iba pohybovými artefaktmi a žiaden spôsobený zlým kontaktom elektród, môže mať toto za následok preučenie našej siete na špecifický vstup a horšie výsledky v reálnej praxi. Toto je niečo, čomu nevieme zabrániť bez toho, aby sme dáta manuálne prechádzali a anotovali typ zachytených artefaktov. Je však potrebné si byť tejto možnosti vedomý.

Trénovací dataset

Do trénovacieho datasetu chceme použiť čo najväčšiu variabilitu rôznych signálov od rôznych ľudí, aby sme znížili riziko, že model sa preučí na EKG konkrétnej osoby.

Z BUT QDB datasetu sme použili záznamy 10001, 103002, 105001, 111001. Z datasetu FBMI sme použili všetky fbmi-1-x záznamy okrem fbmi-1-2, fbmi-1-6 a fbmi-1-12. Ďalej sme použili všetky fbmi-2-x záznamy a všetky fbmi-3-x záznamy.

Validačný dataset

Použili sme väčšinu datasetu BUT QDB, presne sú to záznamy 100002, 103001, 125001, 103003, 104001, 126001, 113001, 115001, 118001, 121001, 122001, 123001. Z datasetu FBMI ČVUT sme vo validačnom datasete nepoužili žiadne dáta.

Testovací dataset

Tu sme použili záznamy 114001 a 124001 z BUT QDB a fbmi-1-2, fbmi-1-6 a fbmi-1-10 z FBMI ČVUT.

Pri delení sme sa snažili aspoň trochu zachovať pomer dĺžky signálu podľa kvality. Výsledné dĺžky jednotlivých datasetov a zastúpenie kvalít v nich sú uvedené v Tabuľke 4.6. Vo validačnom datasete je menšie zastúpenie signálu kvality 3 ako vo zvyšných dvoch. To môže mať za následok, že výsledný model bude na reálnych dátach dosahovať horšie výsledky ako sa bude zdať počas trénovania.

Testovací dataset je výrazne reprezentatívnejší, takže takýto problém by sme mali vedieť odhaliť a modely objektívne porovnať. Výsledky jednotlivých modelov nebudeme hodnotiť prostredníctvom iba jednej hodnoty, ale prostredníctvom analýzy výsledkov pre jednotlivé triedy.

Typ datasetu	Kvalita 1	Kvalita 2	Kvalita 3	Spolu
Trénovací	45:56:30	31:06:08	16:43:10	93:45:48
Validačný	06:09:40	01:48:16	00:22:02	08:19:58
Testovací	01:17:44	00:54:47	00:25:27	03:37:58

■ **Tabuľka 4.6** Zastúpenie jednotlivých kvalít vnútri trénovacieho, validačného a testovacieho datasetu

Kapitola 5
Tvorba modelov a knižnice

V tejto kapitole sa budeme zaoberať návrhom riešenia, trénovaním modelov a následnou implementáciou knižnice na spracovanie EKG signálu.

5.1 Analýza problému

Cieľom je navrhnúť čo najlepšiu metódu detekcie a označenia artefaktov na signále EKG. Súčasťou metódy nebude identifikácia jednotlivých artefaktov, pretože metóda bude používaná na spracovanie dát po skončení merania. V prípade vytvorenia modelu, ktorý by bežal v reálnom čase môže takáto funkcionalita okamžite upozorniť na chybu merania a jej typ, a tým urýchliť opravu. Vzhľadom na tri triedy našich dát máme niekoľko možností, aký typ úlohy budeme riešiť:

- Delenie na dobrý a zlý signál, kde triedu 1 označíme za dobrý signál, keďže je možné detegovať všetky segmenty EKG a triedu 2 a 3 označíme za zlý signál, keďže sa nejedná o úplne čistý signál.
- Delenie na signál, kde je možné detegovať QRS komplex, teda triedy 1 a 2 a na signál, kde ho nie je možné detegovať, teda trieda 3.
- Ponechanie všetkých troch tried.

Pri ponechaní všetkých troch tried vieme následne na základe výsledkov určiť aj prvé dve delenia. Je rozumné predpokladať, že presnosť bude v tom prípade horšia ako v prípade, že by sme model vytvárali priamo za účelom binárneho delenia. Umožní to však väčšie zameranie ako v prípade iba binárnej klasifikácie.

Keďže ideme pracovať s troma triedami, stretávame sa so špecifickým problémom určovania presnosti nášho modelu. Naše tri triedy nie sú navzájom nezávislé a nachádza sa v nich istá postupnosť. Ak by sme mali zle klasifikovať triedu 3, je pre nás vhodnejšie, ak je klasifikovaná ako trieda 2, nie ako trieda 1. Zároveň je pre nás vhodnejšie, ak náš model radšej označí nejaký dobrý signál za nevhodný ako by mal označiť zlý signál za dobrý. Čím kvalitnejší signál, tým viac nás zaujíma presnosť klasifikácie a pri horších kvalitách signálu zase pokrytie. Toto by sme vedeli riešiť klasickým klasifikačným prístupom s vlastnou metrikou, avšak existuje aj ďalší možný prístup.

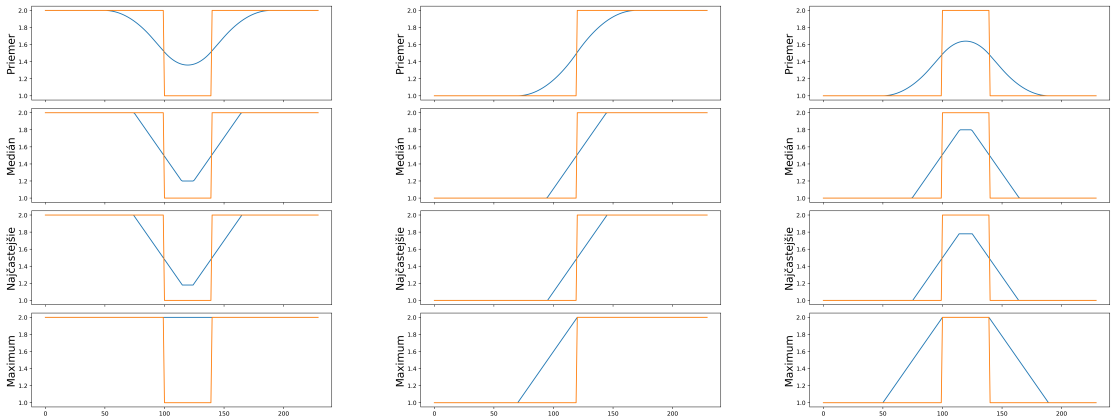
Budeme riešiť klasifikačnú úlohu využívajúcu opačnú úlohu, teda si ju preformulujeme na regresnú úlohu. Ak priradíme triedam 1, 2 a 3 číselné hodnoty 0, 0,5 a 1, môžeme klasifikáciu úseku určiť ako regresnú úlohu, kde náš výsledok musí pochádzať z intervalu 0 až 1, čo vieme zabezpečiť napríklad použitím sigmoidy ako aktivačnej funkcie na výstupnom neuróne. Toto nám zároveň umožní, aby náš model pracoval s istotou podľa toho, ako blízko sa nachádza ku ktorej

hodnote. Ak budeme chcieť získať výsledné triedy, aplikujeme na toto skóre medze, ktorým ho zdiskretizujeme naspäť na naše výsledné triedy. Nastavovaním medzí zároveň vieme preferovať chybu smerom ku klasifikácii horšej kvality signálu.

Ďalšia vec, ktorú nám tento prístup umožní, je spracovávať dlhý signál metódou posuvného okna s prekryvmi. To znamená, že budeme postupne spracovávať signál po konštantne dlhých úsekoch, podobne ako pri konvolúcii. Tieto okná sa môžu aj prekrývať. V tomto prípade zisťujeme skóre ako priemer výsledkov okien na ľubovoľnom mieste prekryvu. Naša úloha sa teda mení z problému spracovávania ľubovoľne dlhých úsekov EKG na určenie skóre v rozmedzí 0 až 1 pre úseky konštantnej dĺžky.

Náš model teda dostane na vstup EKG signál konštantnej dĺžky, pre ktorý vráti skóre od 0 do 1. Chceme, aby toto skóre bolo čo najbližšie k reálnej hodnote, teda pre triedu 0 skóre 0, pre triedu 2 skóre 0,5 a pre triedu 3 skóre 1.

Pre posuvné okno konštantnej kvality vieme, aké skóre by náš model mal vrátiť. V prípade, že v signále nastáva prechod, alebo iba krátkodobý artefakt trvajúci zlomok sekundy, začína byť tento problém náročnejší. Predstavíme si, že pracujeme s ideálnym modelom, ktorý nám na našu regresnú úlohu vždy vráti reálne hodnoty. Teraz potrebujeme zistiť, čo chceme nastaviť ako tieto hodnoty. Naším cieľom je zachytávať aj veľmi krátke artefakty, takže najbežnejšia hodnota ani priemerná alebo mediánová hodnota nie sú vhodné, keďže tieto by ideálny model aj pri maximálnom prekryve nedokázal zachytiť. Metrika, ktorú vyberieme, je maximálna hodnota v okne. Toto bude mať za následok to, že budeme za zlý označovať aj signál okolo artefaktov a nebudeme schopní zachytiť krátkodobé zlepšenie signálu. Bude to mať však pozitívny vplyv na presnosť signálu lepšej kvality. Vplyv jednotlivých metrík na výsledok v prípade ideálneho modelu je zobrazený na Obrázku 5.1



Obr. 5.1 Grafy zobrazujúce výsledné metriky v ideálnom prípade zobrazené na niekoľkých prípadoch zmeny kvality. Originálna kvalita je označená modrou, výsledok ideálneho modelu riadiaceho sa metrikou je oranžovou.

Je vhodné zvoliť malú veľkosť posuvného okna, aby sme mali dostatočné rozlíšenie a krátke artefakty nám nezahadzovali veľké množstvo signálu okolo. Súčasne toto okno musí byť dostatočne dlhé na to, aby sme mali dostatok informácií na rozhodnutie sa. Vzhľadom na to, že naše triedy závisia od segmentov jedného EKG úderu, je potrebné, aby sa v našom okne nachádzal aspoň jeden úder. Rozhodli sme sa, že vyskúšame dĺžky okien 2 a 5 sekúnd. 2 sekundy by mali byť dostatočne dlhé pre väčšinu EKG signálu, avšak v prípade, že by mal byť klasifikovaný signál osoby s tepom napríklad 40 úderov za minútu, pri ktorom by dĺžka jedného úderu bola 1,5 sekundy, bude vhodnejšie použitie okna s dĺžkou 5 sekúnd.

Ďalšia hodnota, ktorú pri posuvnom okne musíme zadefinovať, je prekryv, respektíve posun okna. Náš výsledný model bude podporovať všetky možné posuny, my sa však pri tvorbe modelu obmedzíme na jednosekundový posun. Je vhodné, aby veľkosť posunu bola deliteľom veľkosti

posuvného okna. Ak by to tak nebolo, počet okien, na základe,ä ktorých vykonávame klasifikáciu, sa bude v priebehu signálu meniť. Pri dodržaní tejto podmienky bude odlišný počet okien iba na okrajoch spracovaného signálu.

Výsledný model bude prechádzať signál metódou posuvného okna s nejakým posunom. Signál vnútri každého okna vyhodnotíme pomocou neurónovej siete. Výsledky okien na mieste prekryvov budeme priemerovať. Takýmto spôsobom získame pre takmer celý signál skóre kvality. Výnimkou bude iba koniec časovej rady, ktorý nemusí byť pokrytý. Dĺžka nepokrytého signálu záleží od veľkosti okna a posunu, rádovo to však nebude viac ako jednotky sekúnd. Na skóre signálu potom aplikujeme medze, ktoré nám určia príslušnosť k jednotlivým triedam.

5.2 Spracovanie dát

Aby sme mohli začať tréning, musíme dáta najskôr spracovať. Úseky z trénovacieho a validačného datasetu najskôr vyčistíme pomocou ecg_clean z knižnice NeuroKit2 so základným nastavením. Následne prejdeme všetky anotované úseky a uložíme si signál okna a maximálnu hodnotu, ktorú v tomto okne dosiahla kvalita signálu. Tieto dáta budú použité na trénovanie modelov. Pri ukladaní okien prechádzame signál s posunom 1 sekunda. Takýmto spôsobom vytvoríme dve verzie datasetov, jednu s veľkosťou okna 2 sekundy a druhú s veľkosťou okna 5 sekúnd.

Validačný a testovací dataset si uložíme samostatne ako nevyčistené dáta. Na testovacích dátach budeme na záver zisťovať kvalitu našej klasifikácie. Nevyčistený validačný dataset potrebujeme z dôvodu, že na ňom budeme už na vytrénovaných modeloch nastavovať medze na klasifikáciu. Tieto budeme získavať samostatne pre každý model, keďže tie môžu mať rozdielnu distribúciu výsledných hodnôt. Je podstatné povedať, že v testovacom datasete sa nachádza po spracovaní celkovo 11 záznamov, z toho prvých 8 pochádza z BUT QDB. Je to relevantné z dôvodu, že pre tieto záznamy existujú názory troch odborníkov, ktoré budeme vedieť na záver vyhodnotiť.

Vyčistené a po oknách spracované dáta sa nachádzajú v súboroch formátu TFRecord. Tieto dáta nie sú zamiešané, a teda sú uložené v takom poradí, ako boli pridávané jednotlivé okná a ich výsledky. Nevyčistené dáta určené na beh posuvného okna po signále sú uložené v obyčajnom csv formáte. Všetky súbory vrátane skriptov, ktoré ukazujú, ako načítať oba TFRecord súbory, sú dostupné na kaggle[1].

5.3 Tvorba modelov

Počas tvorby modelov sa budeme venovať rôznym architektúram, ktoré majú počas tréningu vlastné špecifiká. Vysvetlíme postup, akým sme modely vytvárali.

Jednotlivé prístupy sú odlišné, ale niektoré kroky zdieľajú. Všetky trénovania prebiehali prostredníctvom knižnice TensorFlow, verzia 2.12.0. Pri trénovaní sa vždy používal optimalizátor Adam a ako metrika chyby bol použitý MSE. Dáta boli pred každou epochou kompletne zamiešané.

Pri prípadnom hľadaní hyperparametrov sme použili algoritmus Hyperband dostupný v knižnici keras-tuner. Vnútri Hyperband algoritmu sme používali konštantný learning rate s hodnotou 0,0001. Tento learning rate nie je univerzálny. Môže napríklad existovať architektúra, ktorá by dosahovala výrazne lepšie výsledky, ale použitý learning rate je pre ňu príliš vysoký. Nájdeme však modely, ktoré sú schopné sa zlepšiť a tie vieme dotrénovať za kratší čas.

Počas následného dotrénovávania modelov sme pre každý model využili cyklický learning rate, ktorý bol získaný testom na 15 epochách. Pri CLR bola veľkosť kroku 3 epochy. Maximálna dĺžka tréningu bola nastavená na 250 epoch so skorým zastavením, ak sa sledovaná metrika nezlepšila po dobu 25 epoch. Jedinou výnimkou bolo, ak došlo k viditeľnému pretrénovaniu a model sa

[1] `https://www.kaggle.com/datasets/koledjoz/ecg-quality-data-used-for-training`

začínal na validačných dátach výrazne zhoršovať. V tom prípade sme kvôli šetreniu času takýto tréning manuálne zastavili.

Pre výsledné modely sme následne na validačných dátach získali medze pre trojtriednu klasifikáciu. Najskôr sme našli nižšiu medzu v rozmedzí 0 až 0,5 s veľkosťou kroku 0,025 tak, aby sme maximalizovali $F_{0,5}$-skóre pre triedu kvality 1. S touto pevne nastavenou medzou sme potom našli hornú medzu v rozmedzí 0,5 až 1 s krokom veľkosti tiež 0,025, pričom sme maximalizovali $F_{0,5}$-skóre pre triedu kvality 2. Toto robíme tak, že najskôr naším modelom metódou posuvného okna spracujeme signál s posunom 1 sekunda a pre každý záznam spriemerujeme skóre. Následne aplikovaním medzí iba maximalizujeme spomínané metriky.

5.3.1 LSTM

LSTM je špeciálne dizajnované na spracovanie časových rád, preto sa môže javiť ako jasná voľba na náš problém. Pri LSTM budeme trénovať model na dvojsekundových oknách, tento model si označíme ako LSTM2s.

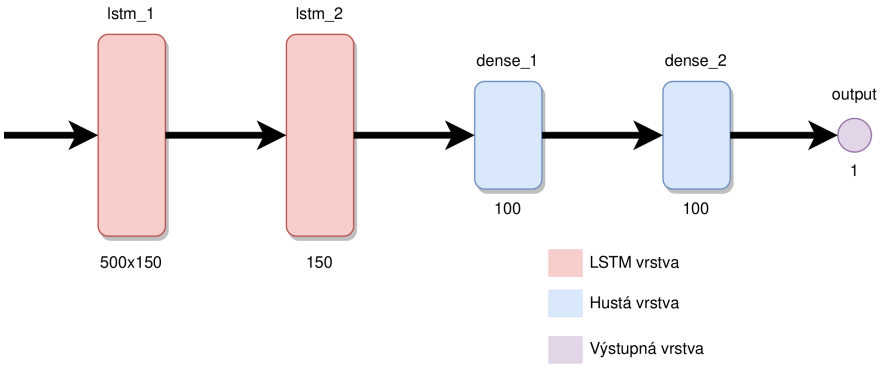
5.3.1.1 Hľadanie hyperparametrov

Hyperparametre sme hľadali pomocou algoritmu Hyperband z knižnice keras-tuner. Nastavenie optimalizácie boli nasledovné:

Hyperparameter	Od	Do	Veľkosť kroku
Počet LSTM vrstiev	1	3	1
Počet neurónov LSTM vrstvy	50	200	50
Počet hustých vrstiev	1	3	1
Počet neurónov v hustej vrstve	50	300	50

Tabuľka 5.1 Priestor hyperparametrov pre LSTM2s

Hyperband sme spustili s maximálnym počtom epoch 50 a s redukčným faktorom 5. Následne sme vybrali 3 modely s najlepším skóre, pre tie sme zistili rozmedzie cyklického learning rate a dotrénovali ich. Výsledný model dosahoval pri konvergencii MSE skóre na testovacích dátach 0,02825. Veľkosť batch bola počas celého tréningu 32. Aktivačné funkcie vnútri LSTM zostali nedotknuté, po hustých vrstvách sme použili LeakyReLU a vo výstupnom neuróne sme použili sigmoidu, čo zaručilo, že naše výsledky sú z rozmedzia 0 až 1. Za každou hustou vrstvou sme pri trénovaní použili dropout s hodnotou 0,5.



Obr. 5.2 Výsledná architektúra LSTM2s

Tento model sme následne uložili a našli pre neho medze vyššie popísaným spôsobom. Výsledné medze dosahovali hodnotu 0,125 a 0,5.

5.3.2 CNN

Ďalšia podstatná architektúra, ktorú sme použili, funguje na základe konvolúcie. Konvolúciu sme trénovali na dvojsekundovom aj päťsekundovom okne. Tieto modely si označíme CNN2s a CNN5s.

5.3.2.1 Hľadanie hyperparametrov

Hľadanie hyperparametrov sme vykonali iba na dvojsekundových oknách a najlepšie architektúry sme následne použili na päťsekundových dátach. Toto má za následok síce to, že vybraná architektúra nemusí byť vhodná pre päťsekundové dáta, ale ušetrí nám to množstvo času pri trénovaní. Priestor hyperparametrov je nasledovný:

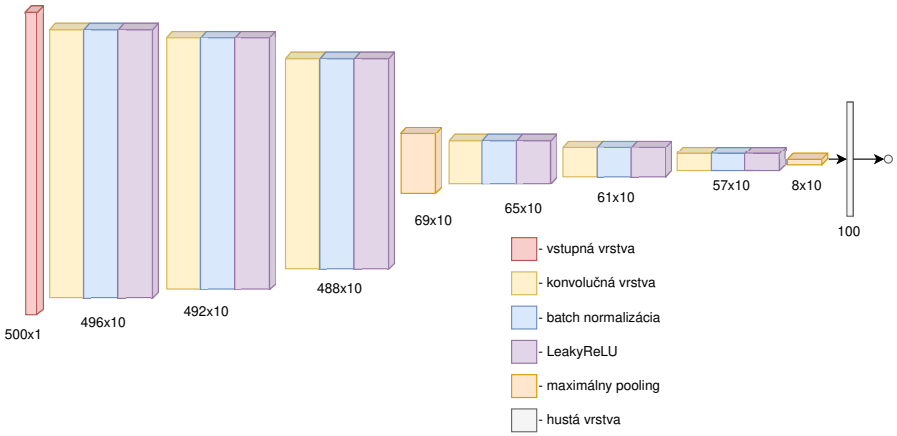
Hyperparameter	Od	Do	Veľkosť kroku
Počet konvolučných blokov	1	3	1
Počet konvolučných vrstiev v bloku	1	3	1
Počet filtrov vo vrstve	10	50	10
Veľkosť filtrov	3	11	2
Veľkosť pooling okna	2	9	1
Počet hustých vrstiev	1	3	1
Veľkosť hustých vrstiev	50	300	50

Tabuľka 5.2 Priestor hyperparametrov pre CNN2s

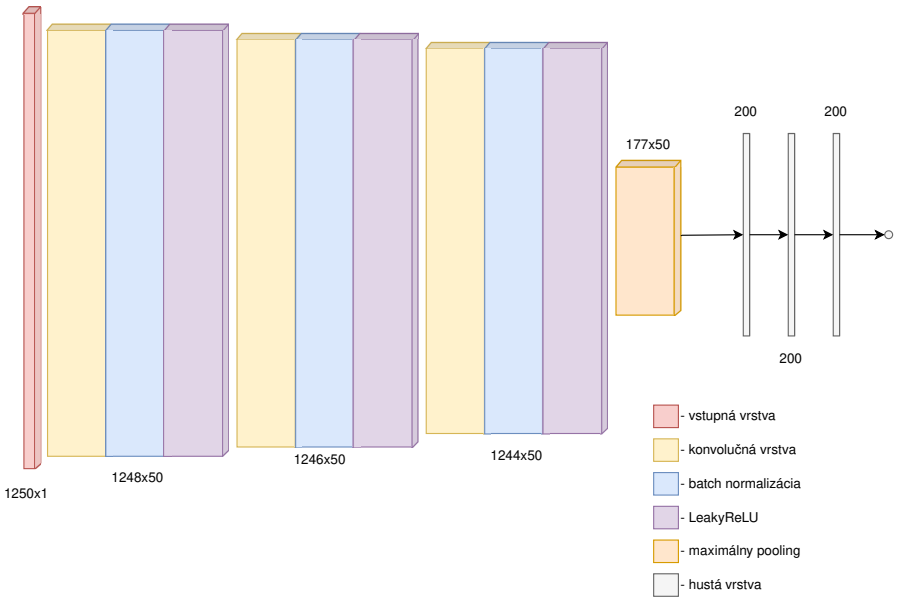
Ďalší hyperparameter, ktorý sme hľadali, je typ poolingu, kde je na výber buď priemer alebo maximum. Hľadaná architektúra sa bude skladať z niekoľkých konvolučných blokov, po ktorých budú nasledovať husté vrstvy. Každý konvolučný blok sa bude skladať z rovnakého množstva konvolučných vrstiev s presne daným počtom filtrov s presne danou veľkosťou. Každý blok bude ukončený pooling vrstvou s určeným typom a presne určenou veľkosťou poolingu. Po každej konvolučnej vrstve sme použili dropout s hodnotou 0,1 a batch normalizáciu. Usporiadanie týchto vrstiev je najskôr konvolúcia s lineárnou aktiváciou, následne batch normalizácia, potom LeakyReLU aktivácia a až potom dropout. Po hustých vrstvách sme použili dropout s hodnotou 0,5. V hustých vrstvách sme použili ako aktivačnú funkciu LeakyReLU. Výnimkou je iba výstupný neurón, kde sme použili sigmoidu, aby sme dosiahli výsledky z rozmedia 0 až 1. Na hľadanie hyperparametrov sme použili Hyperband s maximálnym počtom epoch nastaveným na 100, so skorým zastavením po 25 epochách bez zlepšenia s deliacim faktorom 5.

Následne sme vybrali 10 najlepších nastavení hyperparametrov, na základe ktorých sme natrénovali modely s batch size 128 a 512, všetky s manuálne určeným CLR. Výsledný najlepší model s architektúrou na Obrázku 5.3 dosiahol skóre 0,02669 s batch size veľkosti 512. Ako je možné vidieť, bol vybratý model, ktorý využíva filtre s veľkosťou 5 a v každej konvolučnej vrstve ich používa 10. Veľkosť použitého pooling filtra je 7. Nájdené medze na trojtriednu klasifikáciu boli 0,1 a 0,625.

Pri päťsekundovom okne sme nehľadali hyperparametre pomocou algoritmu Hyperband, ale iba sme rovnako ako pri dvojsekundovom okne iba natrénovali 10 modelov podľa najlepších hyperparametrov s batch size buď 128 alebo 512 a s manuálne určeným CLR. Takto vytrénovaný model zrejme nebude mať takú vysokú presnosť, ako keby sme hľadali architektúru priamo na tento konkrétny problém, ale sme názoru, že presnosť bude dostatočná a na výmenu ušetríme veľké množstvo výpočetnej sily a času. Architektúru výsledného modelu je vidieť na Obrázku 5.4. Tento model využíva pri konvolúcii 50 filtrov s veľkosťou 3. Pri poolingu bol použitý filter s veľkosťou 7. Pri hľadaní medzí sme získali hodnoty 0,2 a 0,525.



Obr. 5.3 Architektúra modelu CNN2s



Obr. 5.4 Architektúra modelu CNN5s

5.3.3 Omni-Scale blok

Pri použití OS bloku sme nepoužívali žiadne hľadanie hyperparametrov. Oproti architektúre sme použili jemné úpravy, ReLU aktivačnú funkciu sme nahradili LeakyReLU. Taktiež sme vyskúšali natrénovať jeden model bez dropout-u a druhý s dropout-om na úrovni 0,1 hneď za aktivačnou funkciou. Okrem toho sme použili štandardnú architektúru, teda OS blok, za ktorým je zapojená globálna pooling vrstva s priemerom a následne výstupná vrstva. Najlepší model na dvojsekundovom okne dosiahol úroveň validačnej chyby 0,02979 pri modele bez dropoutu. Pri päťsekundovom okne sme nedosiahli uspokojivé výsledky, aby sme pristúpili k použitiu modelu a hľadaniu medzí. V oboch prípadoch, teda s dropout-om aj bez neho, sa model veľmi rýchlo začal pretrénovavať a validačná chyba dosiahla najmenšiu hodnotu 0,04987 s dropoutom. Pre každý model sme použili CLR s manuálne určenými hodnotami. Na ďalšie použitie sme pripravovali iba najlepší model na dvojsekundovom okne. Získané medze dosahujú hodnoty 0,1 a 0,7. Tento model budeme označovať ako OS2s.

5.4 Tvorba knižnice

Všetky tieto modely je potrebné poskytnúť užívateľom na jednoduché použitie prostredníctvom knižnice. Táto knižnica by mala jednoducho umožniť určiť kvalitu EKG signálu prostredníctvom jednoduchého rozhrania. Zároveň by mala byť jednoduchá na inštaláciu.

Rozhodli sme sa teda vytvoriť balíček voľne dostupný cez utilitu pip pod menom ecg-quality[2]. V čase písania práce je najaktuálnejšia verzia 0.1.3 a v budúcnosti sa teda poskytnuté informácie budú môcť ešte meniť. Tento balíček bude ako hlavné rozhranie poskytovať triedu, pri ktorej vytvorení užívateľ zadefinuje, akým spôsobom chce presne určovať kvalitu signálu a ďalšie veci s tým spojené. Parametre, ktoré bude môcť užívateľ určiť, sú:

- model : Táto premenná určuje typ modelu, ktorý sa použije. Dostupné budú modely, ktoré sme predstavili skôr, teda CNN2s, CNN5s, LSTM2s a OS2s.
- stride : Táto metóda určuje posun okna pri spracovávaní signálu ako podiel z jeho dĺžky, teda pomocou čísla z intervalu 0 až 1. Pre rovnomerné pokrytie je potrebné, aby veľkosť posunu bola deliteľom veľkosti posuvného okna. Toto zabezpečíme tým, že vyberieme najbližšiu hodnotu z možných deliteľov k tomuto pomeru. Kvôli možnému spomaleniu výkonu bude najmenší možný posun rovný 1 sekunde.
- return_mode : Určuje, čo má vrátiť ako anotáciu signálu. Možnosti sú buď vrátiť nezdiskretizované skóre, trojtriednu klasifikáciu alebo jeden z dvoch druhov binárnej klasifikácie.
- thresholds : Zoznam hodnôt na určenie medzí. Je potrebné, aby ich počet odpovedal vybranému módu klasifikácie.
- return_type : Určuje, či chce užívateľ dostať ako výsledok hodnoty dĺžky signálu, teda hodnotu, kde sa budú jednotlivé úseky opakovať, alebo chce dostať jednu hodnotu pre každý krátky interval s rovnakým skóre. V takomto prípade bude každá vrátená hodnota odpovedať signálu dĺžky posunu okna.
- sampling_rate : Určuje frekvenciu signálu, ktorý bude spracovávať. Keďže modely boli vytrénované iba na dátach s frekvenciou 250 Hz, je toto jediná prijateľná hodnota.
- clean_data : Určuje, či má trieda počas spracovávania dát využívať čistenie signálu pomocou ecg_clean z knižnice NeuroKit2. Modely boli trénované na vyčistených dátach, takže je odporúčané, aby model pracoval iba s vyčistenými dátami. Tento argument však môže byť použitý, ak náš model dostáva na spracovanie už vyčistené dáta.

[2]Zdrojový kód je dostupný na `https://github.com/koledjoz/ecg_quality`.

Všetky tieto argumenty majú svoje východzie hodnoty a knižnica má uložené aj hodnoty medzí pre jednotlivé modely. Nie je teda potrebné špecifikovať žiadne z parametrov, respektíve stačí špecifikovať iba typ modelu. Užívateľ má vďaka tomuto rozhraniu pomerne vysokú voľnosť v tom, ako použije jednotlivé modely. Presnejší popis sa nachádza v dokumentácii konštruktora triedy.

Signál sa následne vyhodnotí zavolaním metódy, ktorá prijíma ako jediný argument signál. Tento spracuje a v prípade, že to bolo žiadané, zdiskretizuje hodnoty a vráti výsledky. V prípade vracania kontinuálnych hodnôt vracia hodnoty z rozmedzia 0 až 1. V prípade trojtriednej klasifikácie vracia hodnoty 1, 2 a 3. Pre binárnu klasifikáciu zase vracia 1 pre lepší signál a 2 pre signál horší. Ukážka použitia našej knižnice:

```
from ecg_quality.ECGQualityChecker import ECGQualityChecker

signal = ...

checker = ECGQualityChecker('cnn2s', stride=0.5,
                                     return_mode='three_value')

results = checker.process_signal(signal)
```

Knižnica je voľne dostupná pod licenciou GPL 3.0 a vyžaduje inštaláciu niekoľkých balíčkov:

- NeuroKit2
- TensorFlow
- NumPy

Tieto balíčky, najmä TensorFlow, ale čiastočne aj NeuroKit2, vyžadujú v základnej verzii veľké množstvo ďalších balíčkov, a teda inštalácia našej knižnice dokáže vyžadovať vyše 500 MB úložného priestoru. Do budúcnosti bude teda vhodné preskúmať existenciu a prípadné využitie obmedzených verzií týchto knižníc, keďže väčšinu ich funkcionalít nepotrebujeme využívať. Z NeuroKit2 využívame iba funkciu ecg_clean, ktorú je možné nahradiť vlastnou funkcionalitou pomocou menších matematicky zameraných knižníc.

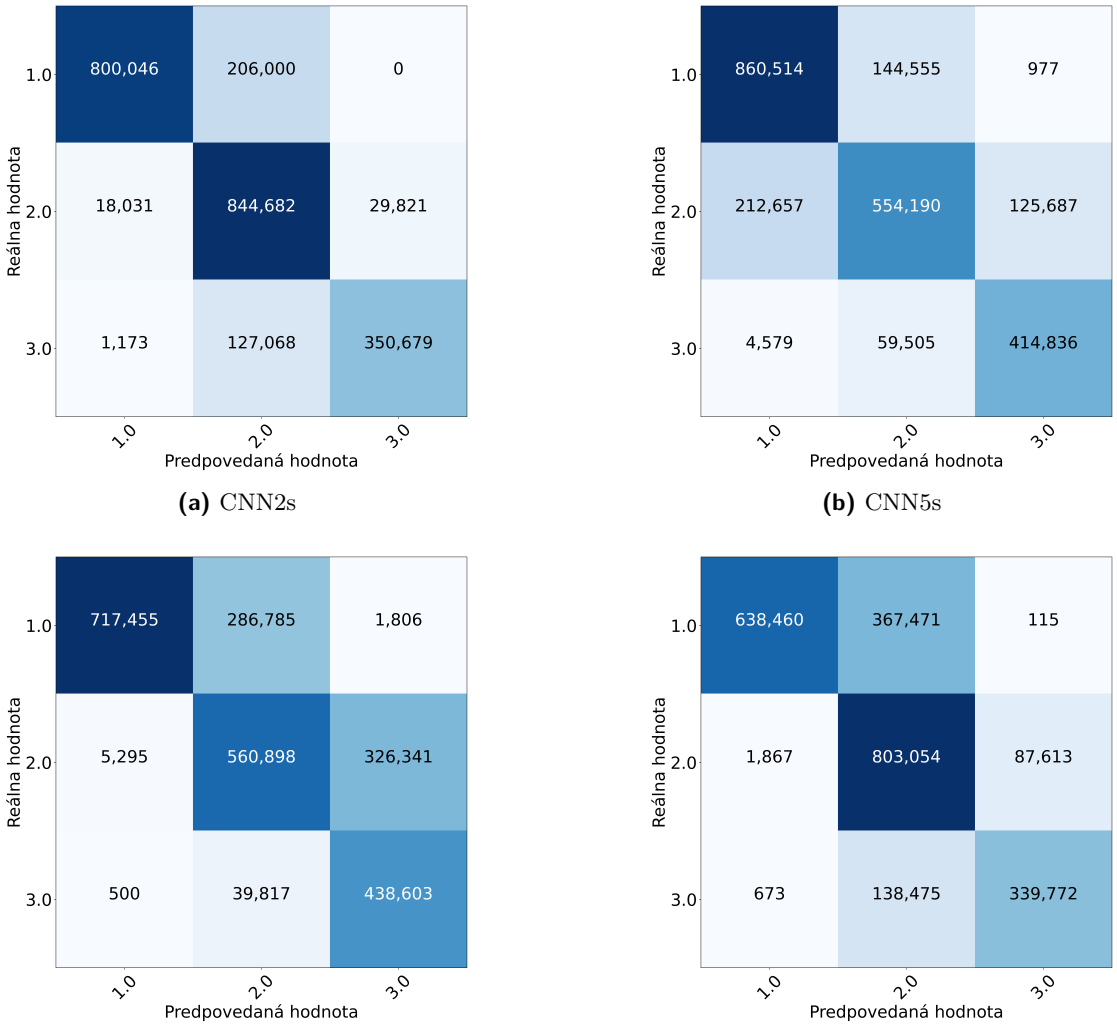
5.5 Vyhodnotenie presnosti modelov

Výsledky našich modelov je potrebné porovnať. Ako sme už vraveli, pri každej triede nás zaujíma iná hodnota, pri lepšej kvalite skôr presnosť a pri horšej kvalite skôr pokrytie. Porovnáme všetky tieto metriky pre všetky naše triedy. Taktiež porovnáme podiel správne klasifikovaného signálu. Ďalšou pre nás podstatnou informáciou je podiel celkového signálu, ktorý bol klasifikovaný ako kvalitnejší v porovnaní s reálnym stavom. Toto nám vie ukázať, nakoľko naše modely spĺňajú našu požiadavku, aby radšej kvalitný signál označovali za signál horšej kvality ako naopak. Takýto signál budeme nazývať preceneným. Zavedieme teda metriku, ktorá iba určí, aký podiel signálu bol precenený.

Všetky modely boli testované s veľkosťou posunu na úrovni 1 sekundy. Najskôr sme získali výsledky trojtriednej klasifikácie a až z tých sme získali jednotlivé binárne klasifikácie. Presný rozpis týchto výsledkov, z ktorých sme získavali jednotlivé metriky modelov, je uvedený na Obrázku 5.5.

5.5.1 Trojtriedna klasifikácia

Trojtriedna klasifikácia je najkomplikovanejšia z úloh, ktoré náš model musí riešiť. Celkovo sme vybrali 4 modely, ktorých výsledky je možné vidieť v Tabuľke 5.3.



(a) CNN2s **(b)** CNN5s

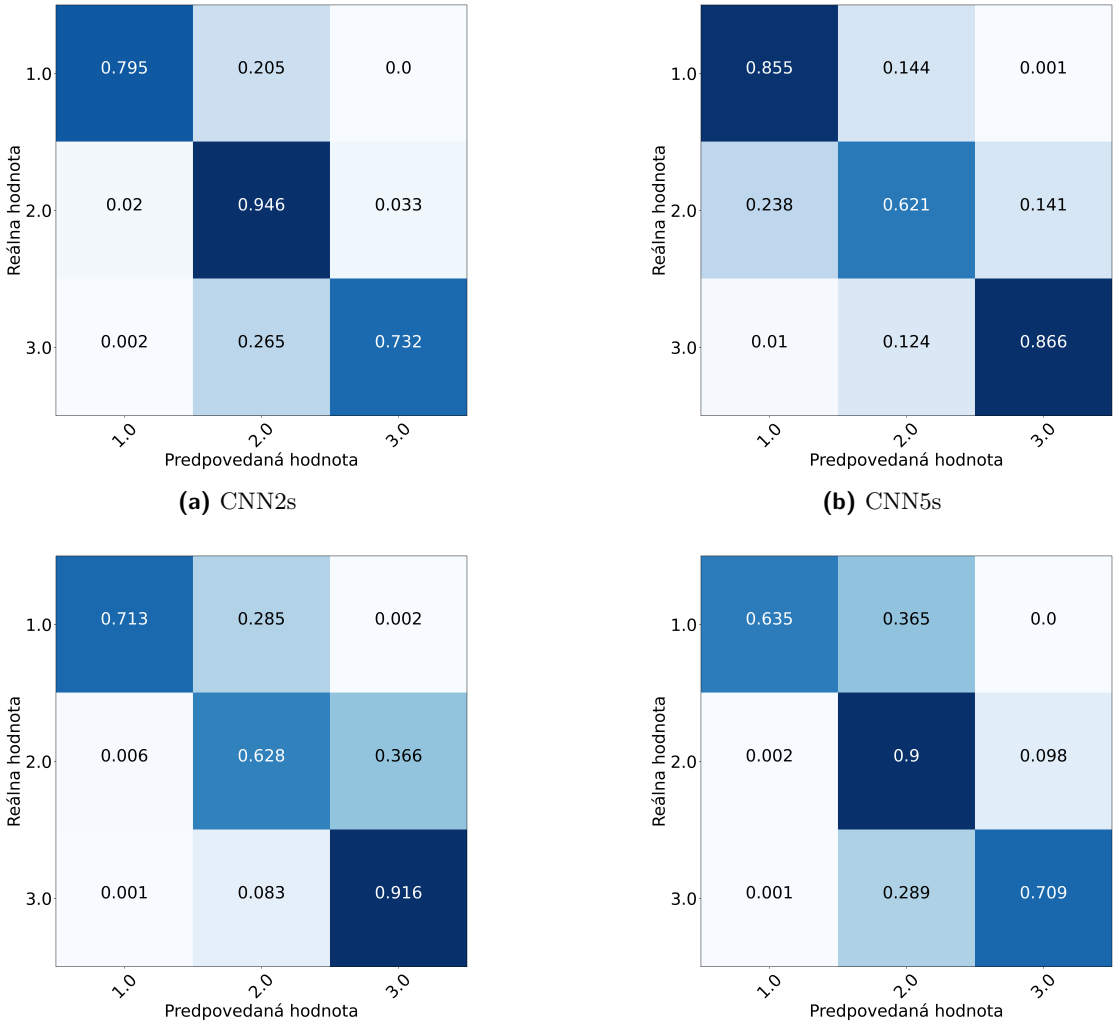
(c) LSTM2s **(d)** OS2s

■ **Obr. 5.5** Matice chyby pri trojtriednej klasifikácii. Čísla odpovedajú jednotlivým vzorkám signálu. 250 vzoriek teda odpovedá jednej sekunde kvôli frekvencii 250 Hz.

	Presnosť			Pokrytie				
Kvalita	1	2	3	1	2	3		
Model							Správne	Precenené
CNN2s	0,977	0,717	0,922	0,795	0,946	0,732	0,839	0,062
CNN5s	0,798	0,731	0,766	0,855	0,620	0,866	0,770	0,116
LSTM2s	0,992	0,632	0,572	0,713	0,628	0,916	0,722	0,019
OS2s	0,996	0,613	0,795	0,635	0,900	0,709	0,749	0,059

■ **Tabuľka 5.3** Prehľad výsledkov modelov pri trojtriednej klasifikácii

Vidíme, že modely majú medzi sebou rozdiely medzi metrikami jednotlivých tried. To znamená, že pri výbere modelu sa môžeme rozhodovať podľa toho, či je pre nás podstatný celkový podiel správne určeného signálu, alebo chceme najmä zahodiť zlý signál aj za cenu nižšieho množstva správne určeného signálu. Všetky modely okrem CNN5s dosahujú pri kvalite 1 vysokú presnosť, čím spĺňajú našu požiadavku. Rozloženie predikovaných hodnôt možno vidieť na Obrázku 5.6.



(a) CNN2s **(b)** CNN5s

(c) LSTM2s **(d)** OS2s

■ **Obr. 5.6** Po riadkoch normalizované matice chyby modelov pre trojtriednu klasifikáciu

5.5.2 Binárna detekcia čistého signálu

Jedno z možných využití, ktoré sme spomínali, je detekcia signálu, ktorý má všetky segmenty detekovateľné, čo je v našom prípade kvalita 1. Rozpis jednotlivých metrík pre tento problém pre každý model je možno vidieť v Tabuľke 5.4.

Vyšší podiel správne určených tried ako v prípade trojtriednej klasifikácie je logický. Celkovo vidíme, že naše modely vedia veľmi dobre detegovať triedy 1, keď všetky s výnimkou CNN5s dosahujú vysokú presnosť pri tejto triede a zároveň vysoké pokrytie pri triede 2/3.

	Presnosť		Pokrytie			
Kvalita	1	2/3	1	2/3		
Model					Správne	Precenené
CNN2s	0,977	0,868	0,795	0,986	0,905	0,008
CNN5s	0,798	0,888	0,855	0,878	0,847	0,091
LSTM2s	0,992	0,826	0,713	0,996	0,876	0,002
OS2s	0,996	0,778	0,635	0,998	0,844	0,001

Tabuľka 5.4 Prehľad výsledkov modelov pri binárnej klasifikácii úplne čistého signálu

5.5.3 Binárna detekcia signálu s QRS komplexom

Posledné možné nastavenie je využitie modelov na detekciu tých úsekov signálu, kde je, respektíve nie je detekovateľný QRS komplex. Presný rozpis výsledkov jednotlivých modelov je možné vidieť v Tabuľke 5.5.

	Presnosť		Pokrytie			
Kvalita	1/2	3	1/2	3		
Model					Správne	Precenené
CNN2s	0,936	0,922	0,985	0,732	0,934	0,054
CNN5s	0,965	0,766	0,933	0,866	0,920	0,027
LSTM2s	0,975	0,572	0,827	0,916	0,876	0,017
OS2s	0,929	0,795	0,954	0,709	0,905	0,059

Tabuľka 5.5 Prehľad výsledkov modelov pri binárnej klasifikácii signálu s QRS komplexami

Všetky modely dosahujú kvalitné výsledky, vrátane CNN5s, ktorý tiež vykazuje vysokú presnosť pri lepšej kvalite signálu. Celkovo naše modely dosahujú vyšší podiel správne klasifikovaného signálu, ale dosahujú menšiu presnosť pri lepšej kvalite signálu v porovnaní s predchádzajúcou binárnou klasifikáciou. Presnosť je však stále dostatočne vysoká.

5.5.4 Presnosť klasifikácie človekom

Vzhľadom na to, že anotácia datasetu BUT QDB prebiehala pomocou názorov trojice odborníkov a následného konsenzu, vieme určiť všetky nami sledované metriky aj pre človeka a porovnať náš model s klasifikáciou vykonanou ľuďmi. Dáta z FBMI ČVUT však túto vlastnosť nemajú, takže toto porovnanie spravíme iba na záznamoch 114001 a 124001 z BUT QDB datasetu, ktoré v spracovaných testovacích dátach odpovedajú záznamom číslo 1 až 8.

V Tabuľke 5.6 vidíme, že naše modely za človekom v prípade trojtriednej klasifikácie zaostávajú, avšak stále vykazujú dobré vlastnosti. Tou hlavnou je to, že preceňujú niekedy dokonca ešte menej ako človek. Pre nás je výhodné, že ak aj model nie je tak kvalitný ako človek, tak to bude mať za následok skôr menšie množstvo signálu, ktorý bude označený za kvalitný a nebude to mať za následok označenie množstva zašpineného signálu za čistý.

Ak sa pozrieme na binárne klasifikácie, rozdiely sa postupne začnú znižovať, ale stále sú relatívne jasné. Ak sa pozrieme na výsledky klasifikácie čistého signálu v Tabuľke 5.7, môžeme vidieť, že naše modely síce pri lepšej kvalite dosahujú vysokú presnosť, ale stále nedostatočné pokrytie v porovnaní s človekom. Avšak v prípade klasifikácie prítomnosti QRS komplexu, kde môžeme vidieť porovnanie v Tabuľke 5.8, naše modely dokážu dosiahnuť aj vysoké pokrytie. Stále môžeme vidieť, že každý model aspoň v jednej sledovanej metrike výraznejšie zaostáva, avšak ukazuje nám to, že naše modely dosahujú porovnateľné výsledky ako ľudia, aj keď stále horšie.

	Presnosť			Pokrytie				
Kvalita	1	2	3	1	2	3		
Spôsob							Správne	Precenené
CNN2s	0,967	0,717	0,922	0,731	0,947	0,732	0,821	0,068
CNN5s	0,741	0,731	0,766	0,810	0,621	0,866	0,744	0,130
LSTM2s	0,988	0,632	0,572	0,624	0,628	0,916	0,691	0,021
OS2s	0,994	0,613	0,795	0,520	0,900	0,709	0,721	0,066
Človek	0,944	0,859	0,866	0,921	0,876	0,882	0,896	0,045

Tabuľka 5.6 Porovnanie modelov s klasifikáciou vykonanou človekom na obmedzenom datasete v prípade trojdielnej klasifikácie

	Presnosť		Pokrytie			
Kvalita	1	2/3	1	2/3		
Spôsob					Správne	Precenené
CNN2s	0,969	0,868	0,731	0,986	0,895	0,009
CNN5s	0,741	0,888	0,810	0,841	0,830	0,101
LSTM2s	0,988	0,826	0,624	0,996	0,862	0,003
OS2s	0,994	0,788	0,520	0,998	0,827	0,001
Človek	0,944	0,940	0,921	0,958	0,942	0,024

Tabuľka 5.7 Porovnanie modelov s klasifikáciou vykonanou človekom na obmedzenom datasete v prípade klasifikácie čistého signálu

	Presnosť		Pokrytie			
Kvalita	1/2	3	1/2	3		
Spôsob					Správne	Precenené
CNN2s	0,927	0,922	0,982	0,732	0,926	0,060
CNN5s	0,960	0,766	0,924	0,866	0,911	0,030
LSTM2s	0,971	0,572	0,802	0,916	0,828	0,019
OS2s	0,919	0,795	0,947	0,709	0,894	0,065
Človek	0,974	0,866	0,970	0,882	0,954	0,021

Tabuľka 5.8 Porovnanie modelov s klasifikáciou vykonanou človekom na obmedzenom datasete v prípade klasifikácie podľa prítomnosti QRS komplexu

5.6 Porovnanie výkonnosti

Vzhľadom na rozdielnu veľkosť modelov a aj základné rozdiely v architektúre je potrebné porovnať rýchlosť jednotlivých modelov. Testujeme rýchlosť spracovania úsekov bez aplikovania medzí. Porovnávame výkon modelov na rôznych dĺžkach. Okná sú modelom spracovávané po jednom, bez využitia GPU akcelerácie. Tieto testy boli vykonávané na zariadení s procesorom Intel Core i5-9300HF 2.40GHz a s 16Gb RAM. Dáta boli iba konštantná časová rada dopredu načítaná v pamäti. Všetky modely pracovali s veľkosťou posunu okna 1 sekunda. Pri získavaní časov bola použitá implementácia ecg-quality, verzia 0.1.3.

Model	1 minúta	5 minút	10 minút
CNN2s	00:02,612	00:12,936	00:25,483
CNN5s	00:02,525	00:13,814	00:29,202
LSTM2	00:07,338	00:30,487	01:01,478
OS2s	00:04,255	00:14,647	00:35,286

Tabuľka 5.9 Čas potrebný na spracovanie signálu podľa jeho dĺžky

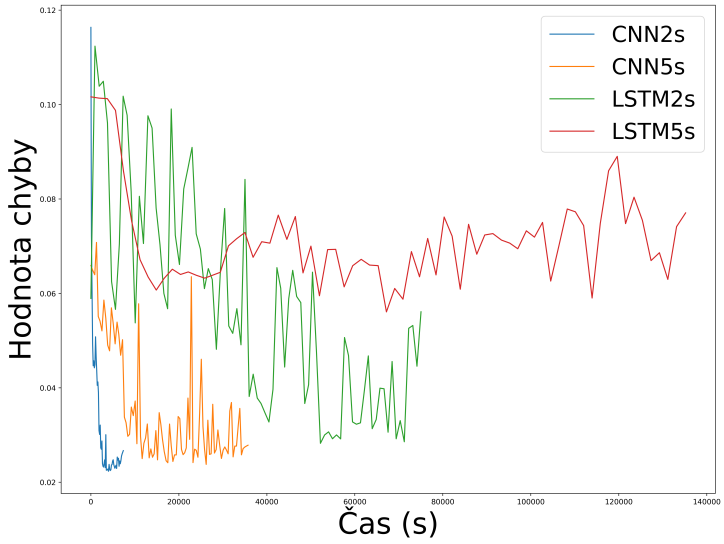
Budeme predpokladať, že konštantná réžia modelov je nulová. V tom prípade by signál dĺžky 24 hodín náš najrýchlejší model spracovával približne 1 hodinu. Tomu najpomalšiemu by to trvalo takmer 2 a pol hodiny. Toto ukazuje výrazné ušetrenie času, kedy aj na relatívne nevýkonnom zariadení dokážeme za relatívne krátky čas spracovať aj dlhý EKG signál. Dá sa predpokladať, že reálne časy by boli dlhšie vzhľadom najmä na pamäťovú réžiu. Taktiež by mohlo byť potrebné spracovávať signál po kratších intervalov, keďže knižnica pracuje s pamäťou veľmi neefektívne. Toto sú však problémy, ktoré sa dajú v budúcnosti pomerne jednoducho vyriešiť bez potreby tvorby nových modelov.

Kapitola 6
Diskusia

Výsledky pri porovnávaní našich modelov ukazujú, že aj keď zaostávajú svojou kvalitou za ľudskými odborníkmi, naše modely stále vedia poskytnúť dostatočnú detekciu úsekov s viacerými dobrými vlastnosťami, ako je napríklad vysoká presnosť pri signále najlepšej kvality. Nami poskytnuté modely majú rozdielne vlastnosti a poskytujú možnosť užívateľovi vybrať, aký typ anotácie vyžaduje, s možnosťou upravovať si výsledky modelov pomocou medzí počas diskretizácie výsledkov, čím vie tieto vlastnosti ešte ďalej upravovať. CNN2s model považujeme s našimi medzami za najrobustnejší a celkovo najspoľahlivejší model. Avšak aj modely LSTM2s a OS2s vedia byť použité v prípade, kedy je presná klasifikácia najlepšieho signálu vyššou prioritou aj za cenu zahodenia väčšieho množstva dobrého signálu. Najhorší výkon podľa nás poskytuje CNN5s, ktorý nedosahuje takú presnosť triedy najlepšej kvality ako modely LSTM2s a OS2s a zároveň neposkytuje takú všeobecnú kvalitu ako CNN2s. Jediné použitie, kde by sme ho odporúčali, je v prípade binárnej klasifikácie signálu detekovateľnosti QRS komplexu.

Naše modely dokážu presne pracovať na úsekoch, kde sa kvalita dlhodobo nemení a jej zmeny nastávajú iba občas. Takým záznamom je napríklad 114001 z BUT QDB, kde naše modely dosahujú ešte vyššiu presnosť ako naše celkové výsledky. Kde sa našim modelom nedarí, je napríklad záznam 124001 z BUT QDB, kde nastáva zmena kvality veľmi často a úseky s konštantnou kvalitou zvyknú trvať veľmi krátko. Toto zníženie je podľa nás spôsobené našou metódou posuvného okna, kedy nevieme dostatočne presne zachytiť tieto prechody a zmeny vzhľadom na nízke rozlíšenie výsledkov nášho modelu. Tento problém je veľmi náročné odstrániť bez toho, aby sme výrazne zmenili fungovanie metódy posuvného okna alebo použili úplne iný postup. Rozlíšenie vieme zvyšovať pomocou znižovania dĺžky okna. Ak však bude okno príliš malé, nebude sa nám dariť rozoznať z neho kvalitu signálu.

Na našich modeloch sme objavili jednu zásadnú slabú stránku. Tá sa prejavuje, ak dostanú na vstup malý šum na úrovni stotín mV a menej. V takom prípade je tento vstup omnoho menší ako čokoľvek, na čo je model dostatočne naučený. Pre všetky takéto vstupy preto vracia ako výsledok číslo veľmi blízko k hodnote 0, čo znamená, že tento signál klasifikuje ako najvyššej kvality. Pri dlhých úsekoch takéhoto šumu zvyknú naše modely dosahovať aj 0 % podiel správne určeného signálu. Jedná sa zrejme o systémovú chybu, ktorá znamená, že v trénovacom datasete takýto šum nebol dostatočne zastúpený, čo je spôsobené tým, že pri delení signálu sme sa pozerali iba na kvalitu signálu a nie na jeho špecifiká. Jedna z možností, ako tento problém riešiť, je rozšíriť trénovací dataset o takýto, kľudne aj umelo vytvorený signál. Druhá z možností je, aby model pred spracovávaním každý vstup štandardizoval. Toto by však mohlo mať za následok zhoršenie klasifikácie iných typov signálov, keďže aj rozdielna výška kmitov, respektíve vĺn vie signalizovať prítomnosť artefaktu a detekcia práve takýchto typov by sa mohla zhoršiť. Poslednou a najrýchlejšie realizovateľnou opravou je iba vytvorenie nejakých jednoduchých kontrol, ktoré takýto veľmi malý šum budú detegovať a označovať ako signál najhoršej kvality automaticky.



Obr. 6.1 Graf porovnávajúci trvanie tréningu CNN a LSTM architektúr

Takáto záplata nie je robustný prístup k tomuto problému, a teda pri prípadnom ďalšom trénovaní modelov je potrebné pristúpiť k jednému z dvoch vyššie spomínaných prístupov alebo sa zamyslieť nad ďalším.

Pri tvorbe modelov sme vyskúšali 3 hlavné typy architektúr, konvolučné, na princípe LSTM a na princípe konvolučného Omni-Scale bloku. Všetky z nich dokázali na dvojsekundovom posuvnom okne dosiahnuť uspokojivé výsledky. Konvolúcia fungovala aj v prípade päťsekundového okna, kde však zrejme ešte existuje priestor na zlepšenie modelu, ak by sa použilo hľadanie hyperparametrov špeciálne pre tento model namiesto iba použitia jemne upravenej architektúry z dvojsekundového okna. Celkovo CNN5s nedosiahol pre nás uspokojivé výsledky. Stále však nemôžeme povedať, že to bolo spôsobené dĺžkou okna.

Pri LSTM a Omni-Scale bloku sa nám nepodarilo vytvoriť model s dlhším oknom. Pri LSTM to bolo spôsobené najmä náročnosťou a dlhým trvaním tréningu, keď sa model postupne učil, ale toto trvalo príliš dlho a hrozilo, že ak aj model začne konvergovať, tak za násobne dlhší čas ako v prípade ostatných architektúr. Dlhý čas trénovania LSTM však nevieme zmeniť bez jeho výraznej zmeny. Alternatívou je použiť alternatívnu rekurzívnu GRU jednotku, ktorá je výpočetne menej náročná, ale nemusí dosahovať až takú vysokú presnosť. Porovnanie času potrebného na trénovanie konvolučných a LSTM architektúr je možné vidieť na Obrázku 6.1. Vidieť, že LSTM trvá trénovanie násobne dlhšie. Toto je podľa nás spôsobené dvoma hlavnými dôvodmi. Prvý je fakt, že jedna epocha pri LSTM trvá dlhšie ako pri konvolúcii, čo je spôsobené tým, že potrebuje spracovať signál sekvenčne, a to čiastočne obmedzuje možnosť použitia GPU paralelizmu. Druhý dôvod je v princípe fungovania LSTM a v jeho sekvenčnom spracovaní. Tomuto modelu nestačí vedieť detegovať horšiu kvalitu signálu, ale musí si tento fakt aj zapamätať po celý zvyšok spracovania signálu. To znamená, že model sa nemusí naučiť iba túto informáciu získať, ale zároveň si ju aj zapamätať až do konca. Toto by sa dalo obmedziť vrátením stavu LSTM v každom kroku, aké sa napríklad využíva pri zapojení viacerých vrstiev za seba. Tieto vysokodimenzionálne údaje by však bolo následne potrebné spracovať, čo by predstavovalo ďalšiu náročnú úlohu.

Pri Omni-Scale bloku bol problém s pretrénovaním modelu pri dlhšom časovom okne s obmedzením čoho nepomohlo ani použitie dropout-u. Sme názoru, že to bolo spôsobené príliš veľkými konvolučnými filtrami, ktoré sa snažili pokryť podsekvencie všetkých dĺžok, pričom nám sa väčšinou stačí pozerať iba na podsekvencie dĺžky maximálne jedného úderu, keďže kvalita je v našom

prípade pevne spojená so segmentami jednotlivých EKG úderov. Táto možná zmena by do budúcnosti mohla ešte vylepšiť použitie Omni-Scale bloku, ktorý sa ukázal ako veľmi vhodný, keďže dosiahol na kratšom okne uspokojivé výsledky napriek tomu, že neprebehlo žiadne hľadanie hyperparametrov, ktoré výpočetne v prípade ostatných modelov zabralo najväčšiu časť tréningu. Veríme, že úpravou maximálnej veľkosti filtrov v tomto bloku by sme vedeli znížiť problém pretrénovania a dosiahnuť ešte lepšie výsledky.

Nami použité architektúry boli veľmi jednoduché, konvolúcia napríklad využívala iba rovnaké typy konvolučných vrstiev, ktoré nekombinovali rôzne veľkosti alebo počet filtrov. LSTM model taktiež iba kombinoval niekoľko rovnakých vrstiev nasledovaných hustými vrstvami. Jediný model, ktorý využíval modernejšie a pokročilejšie prístupy, bol na princípe Omni-Scale bloku. Do budúcnosti je potrebné venovať viac času hľadaniu optimálnejších a komplikovanejších hyperparametrov pre modely. To bude výpočetne náročné, ale môže to zvýšiť ich kvalitu. Myslíme si však, že v tomto smere je obmedzený priestor na zlepšenie bez zmenenia hlavného princípu, teda posuvného okna, respektíve jeho fungovania.

Ak chceme konkurovať v kvalite klasifikácie odborníkom, potrebujeme model, ktorý dokáže pracovať s výrazne vyšším rozlíšením a bude presne zachytávať aj veľmi krátke zmeny kvality bez toho, aby to malo vplyv na príliš veľké okolie. Toto nám nami používaná metóda posuvného okna poskytnúť nevie. Niektoré zmeny kvality trvajú menej ako pol sekundu, čo je výrazne menej ako jeden úder. Je teda potrebné preskúmať aj metódy, ktoré by dokázali pracovať s výrazne väčším rozlíšením. Signál bude stále potrebné spracovávať po krátkych častiach, zmena však musí nastať vo výstupe modelov, ktoré okno spracovávajú. Momentálne naše modely pre každé okno vracajú jednu hodnotu. My by sme potrebovali model, ktorý nám v tomto prípade vráti hodnôt viac. Výstup môže byť napríklad v podobnom štýle ako v prípade detekcie objektov na obrázku, teda prostredníctvom intervalov, ktoré by popisovali, že bol detegovaný artefakt. Ďalším možným prístupom je, aby výstup modelu bol rovnako dlhý ako veľkosť vstupu. Na to môže byť použitá napríklad architektúra 1D-UNet, ktorá sa už využila na detekciu jednotlivých EKG segmentov [50].

Všetky modely sú momentálne dostupné prostredníctvom našej Python knižnice, ktorá je dostupná cez utilitu pip. Táto poskytuje možnosti použitia všetkých nami poskytnutých modelov vo všetkých typoch klasifikácie aj s nastavením vlastných medzí. Spracovanie signálu však vykonáva pomerne neefektívne, keď si ukladá do pamäte v istom momente dáta až 3-násobné oproti dĺžke spracovávaného signálu. Signál taktiež spracováva iterovaním cez pole v Pythone, ktoré je pomalé a vie byť výrazne zrýchlené. Rýchlosť je podľa nás možné zvýšiť ešte viac. Ďalší možný problém do budúcnosti je použitie knižnice NeuroKit2 na čistenie signálu. Vzhľadom na to, že táto knižnica je vysoko špecifická, je výraznejšie riziko, že v budúcich verziách môže dôjsť k zmene použitej vysoko úrovňovej funkcie. Preto môže byť vhodnejšie napísať vlastnú verziu tohto čistenia pomocou nízko úrovňových matematických funkcií. Použité filtre sú napríklad súčasťou knižnice SciPy a pri týchto je výrazne menšia šanca, že dôjde k výraznejším zmenám, ktoré by zmenili ich správanie.

Kapitola 7
Záver

Stroje odjakživa nahrádzajú prácu ľudí. V poslednej dobe začínajú ukazovať skvelé výsledky v oblastiach, v ktorých by sme to nikdy nečakali.

Cieľom tejto práce bolo navrhnúť a vytvoriť metódy na spracovanie a anotovanie EKG signálu podľa kvality a tieto následne implementovať v rámci voľne dostupnej knižnice v jazyku Python. Popísali sme signál EKG, jeho relevantné segmenty, vlastnosti a spôsob merania. Využili sme voľne dostupný dataset BUT QDB určujúci kvalitu EKG podľa možnosti detekcie jednotlivých relevantných segmentov a pridali k nemu záznamy vybrané špeciálne pre účely tvorby našich modelov odborníkmi z FBMI ČVUT. Celkovo sme rozlišovali 3 úrovne kvality EKG. Vytvorili sme 4 modely, ktoré na signále konštantnej dĺžky určujú skóre tohto signálu indikujúce kvalitu. Následne sme prechádzali celý EKG záznam pomocou metódy posuvného okna, kde sa skóre na mieste prekryvu priemeruje. Z výsledného skóre aplikovaním medzí získavame výsledné 3 triedy. Medze sme určovali postupne tak, aby sme maximalizovali $F_{0,5}$-skóre pre triedy lepšej kvality.

Tieto metódy sme implementovali v jazyku Python ako súčasť voľne dostupnej knižnice ecg-quality. Túto knižnicu je možné nainštalovať pomocou utility pip a umožňuje jednoduché určenie kvality EKG signálu. Knižnica poskytuje možnosť rýchlej anotácie kvality EKG aj v prípade dlhodobého merania. Zároveň umožňuje užívateľovi jednoducho využiť nielen trojtriednu klasifikáciu, ale aj jeden z dvoch typov binárnej klasifikácie.

Naše modely sme navzájom porovnali medzi sebou. Model CNN2s dosiahol pri trojtriednej klasifikácii najvyšší podiel správne určeného signálu 83,9 %. Pri EKG signáli najvyššej kvality súčasne dosiahol presnosť 97,7 % a pokrytie 79,5 %. Pri oboch typoch binárnej klasifikácie tieto metriky ešte výraznejšie stúpali.

Na zmenšenej vzorke sme modely porovnali s klasifikáciou signálu vykonanou odborníkom. Naše modely výrazne zaostávali v celkovom podiele správne určeného signálu, ale kompenzovali to dobrými výsledkami v iných metrikách. Pri klasifikovaní signálu za signál najlepšej kvality dosahovali všetky naše modely presnosť porovnateľnú a často aj vyššiu ako odborníci. Naše modely teda poskytujú silnú záruku, že ak je signál označený za nezávadný, naozaj ním aj je.

Rozsiahlejším hľadaním hyperparametrov modelov a použitím komplikovanejších architektúr by sme mohli zlepšiť výsledky našich modelov. Alternatívnou cestou je použitie state-of-the-art architektúr na náš problém. Tieto modely môžu využiť analýzu časových rád bežne využívanú pri spracovaní EKG na získanie ďalších informácií. V prípade trénovania ďalšieho modelu je potrebné rozšíriť dáta o ďalšie vstupy, najmä o signál najhoršej kvality, ako je napríklad šum.

Slabou stránkou našich modelov je nízke rozlíšenie výsledných hodnôt. Toto je spôsobené metódou posuvného okna, ktoré nemôže byť príliš malé, inak by modely nemali na vstupe dostatok informácií. Model pre každé okno vráti iba jednu hodnotu, a preto aj v prípade maximálneho prekryvu nevieme presne určiť, kde sa nachádza zmena kvality. Jednou z možností ako daný problém riešiť je použitie modelov, ktoré by pre vstupné okno nevrátili jednu hodnotu, ale viacero. Ideálne

je použitie modelu, kde sa veľkosť výstupu rovná veľkosti vstupu. Príkladom takej architektúry je 1D-UNet. Okrem toho je potrebné preskúmať aj ostatné možnosti, ktoré by dokázali zvýšiť rozlíšenie výslednej klasifikácie.

Človeka nevieme nahradiť na rovnakej úrovni. Umelá inteligencia zatiaľ nedokáže dosiahnuť rovnaké výsledky. Rýchlosťou ale človeka poráža. Dôraz by však nemal byť na súboj človeka so strojmi, ale na potenciál, ktorý ponúka to, že ľudia sa budú môcť venovať iným, podstatnejším veciam.

Bibliografia

1. HAMPTON, J.; HAMPTON, J. *EKG stručně, jasně, přehledně*. 9. vyd. Prel. LANDA, Leoš. Praha: Grada Publishing a.s., 2022. ISBN 978-80-271-1317-0.
2. WIKIMEDIA COMMONS. *Precordial electrodes necessary for a 12-lead ECG* [online]. 2010. [cit. 2023-04-18]. Dostupné z : `https://commons.wikimedia.org/wiki/File:Precordial_Leads_2.svg`.
3. WIKIMEDIA COMMONS. *Schematic diagram of normal sinus rhythm for a human heart as seen on ECG (with English labels).* [Online]. 2007. [cit. 2023-01-18]. Dostupné z : `https://en.wikipedia.org/wiki/File:SinusRhythmLabels.svg`.
4. PAL, K. et al. Advanced Methods in Biomedical Signal Processing and Analysis. In: Elsevier Science, 2022, s. 88. ISBN 978-0-323-85955-4.
5. MAKOWSKI, D. et al. NeuroKit2: A Python toolbox for neurophysiological signal processing. *Behavior Research Methods*. 2021, roč. 53, č. 4, s. 1689–1696. Dostupné z DOI: `10.3758/s13428-020-01516-y`.
6. MAKOWSKI, D. *Neurophysiological Data Analysis with NeuroKit2* [online]. 2020–2023. [cit. 2023-04-29]. Dostupné z : `https://neuropsychology.github.io/NeuroKit/`.
7. ZUMBAHLEN, Hank. Phase Response in Active Filters Part 2, the Low-Pass and High-Pass Response [online]. 2009 [cit. 2023-04-29]. Dostupné z : `https://www.analog.com/en/analog-dialogue/articles/phase-response-in-active-filters-2.html`.
8. MAKOWSKI, D. *Source code for neurokit2.ecg.ecg_clean* [online]. 2020–2023. [cit. 2023-04-29]. Dostupné z : `https://neuropsychology.github.io/NeuroKit/_modules/neurokit2/ecg/ecg_clean.html#ecg_clean`.
9. MAKOWSKI, D. *Source code for neurokit2.signal.signal_filter* [online]. 2020–2023. [cit. 2023-04-29]. Dostupné z : `https://neuropsychology.github.io/NeuroKit/_modules/neurokit2/signal/signal_filter.html#signal_filter`.
10. ZHAO, Z.; ZHANG, Y. SQI quality evaluation mechanism of single-lead ECG signal based on simple heuristic fusion and fuzzy comprehensive evaluation. *Frontiers in physiology*. 2018, roč. 9, s. 727.
11. AURA HEALTHCARE. *ecg_qc documentation* [online]. 2021. [cit. 2023-04-22]. Dostupné z : `https://aura-healthcare.github.io/ecg_qc/`.
12. AURA HEALTHCARE. *Data science* [online]. [cit. 2023-04-30]. Dostupné z : `https://en.aura.healthcare/analyse-des-donn%C3%A9es`.
13. AURA HEALTHCARE. *ECG_QC METHODOLOGY* [online]. 2021. [cit. 2023-04-30]. Dostupné z : `https://github.com/Aura-healthcare/ecg_qc/tree/main/ecg_qc-methodology`.

14. CHIROUZE, A. et al. *ECG_QC (Quality Classification)* [online]. GitHub, 2021. Dostupné tiež z: https://github.com/Aura-healthcare/ecg_qc.
15. MOEYERSONS, J. et al. ECG artefact detection using ensemble decision trees. In: *2017 Computing in Cardiology (CinC)*. 2017, s. 1–4. Dostupné z DOI: 10.22489/CinC.2017.240-159.
16. MOEYERSONS, J. et al. Supervised SVM Transfer Learning for Modality-Specific Artefact Detection in ECG. *Sensors*. 2021, roč. 21, s. 662. Dostupné z DOI: 10.3390/s21020662.
17. VARON, C. et al. Robust artefact detection in long-term ECG recordings based on autocorrelation function similarity and percentile analysis. In: *2012 Annual International Conference of the IEEE Engineering in Medicine and Biology Society*. 2012, s. 3151–3154. Dostupné z DOI: 10.1109/EMBC.2012.6346633.
18. BRÁS, S. et al. ECG ARTEFACT DETECTION ALGORITHM - An Algorithm to Improve Long-term ECG Analysis. In: *Proceedings of the International Conference on Bio-inspired Systems and Signal Processing - Volume 1: BIOSIGNALS, (BIOSTEC 2012)*. SciTePress, INSTICC, 2012, s. 329–333. ISBN 978-989-8425-89-8. Dostupné z DOI: 10.5220/0003729503290333.
19. SIVARAKS, H.; RATANAMAHATANA, C. A. Robust and Accurate Anomaly Detection in ECG Artifacts Using Time Series Motif Discovery. *Computational and Mathematical Methods in Medicine*. 2015. ISSN 1748-670X. Dostupné z DOI: 10.1155/2015/453214.
20. SAINI, R. et al. Classification of heart diseases from ECG signals using wavelet transform and kNN classifier. In: *International Conference on Computing, Communication & Automation*. 2015, s. 1208–1215. Dostupné z DOI: 10.1109/CCAA.2015.7148561.
21. VENKATESAN, C. et al. ECG Signal Preprocessing and SVM Classifier-Based Abnormality Detection in Remote Healthcare Applications. *IEEE Access*. 2018, roč. 6, s. 9767–9773. Dostupné z DOI: 10.1109/ACCESS.2018.2794346.
22. HOMAEINEZHAD, M. R. et al. ECG arrhythmia recognition via a neuro-SVM–KNN hybrid classifier with virtual QRS image-based geometrical features. *Expert Systems with Applications*. 2012, roč. 39, č. 2, s. 2047–2058. ISSN 0957-4174. Dostupné z DOI: https://doi.org/10.1016/j.eswa.2011.08.025.
23. ISMAIL FAWAZ, H. et al. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*. 2019, roč. 33, č. 4, s. 917–963. ISSN 1573-756X. Dostupné z DOI: 10.1007/s10618-019-00619-1.
24. KARIM, F. et al. LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access*. 2018, roč. 6, s. 1662–1669. Dostupné z DOI: 10.1109/ACCESS.2017.2779939.
25. IBM. What is deep learning? [Online]. [B.r.] [cit. 2023-04-13]. Dostupné z : https://www.ibm.com/topics/deep-learning.
26. KRIZHEVSKY, A. et al. ImageNet Classification with Deep Convolutional Neural Networks. In: PEREIRA, F. et al. (ed.). *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2012, zv. 25. Dostupné tiež z: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.
27. HE, K. et al. Deep Residual Learning for Image Recognition. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. Dostupné tiež z: https://openaccess.thecvf.com/content_cvpr_2016/papers/He_Deep_Residual_Learning_CVPR_2016_paper.pdf.
28. GÉRON, A. *Hands-on machine learning with Scikit-Learn and TensorFlow : concepts, tools, and techniques to build intelligent systems*. Sebastopol, CA: O'Reilly Media, 2017. ISBN 978-1491962299.

29. KRISHNA, V. B. *Using Binary Classification Metrics to Maximize Enterprise AI's Potential* [online]. 2020. [cit. 2023-04-29]. Dostupné z : https://c3.ai/blog/using-binary-classification-metrics-to-maximize-enterprise-ais-potential/.
30. RUMELHART, D. E. et al. Learning Internal Representations by Error Propagation. In: RUMELHART, D. E.; MCCLELLAND, J. L. (ed.). *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Volume 1: Foundations*. MIT Press, 1986, s. 318–362.
31. XU, B. et al. Empirical Evaluation of Rectified Activations in Convolutional Network. *arXiv e-prints*. 2015, arXiv:1505.00853. Dostupné z DOI: 10.48550/arXiv.1505.00853.
32. KERAS. *LeakyReLU* [online]. 2023. [cit. 2023-04-26]. Dostupné z : https://keras.io/api/layers/activation_layers/leaky_relu/.
33. OORD, A. et al. WaveNet: A Generative Model for Raw Audio. *CoRR*. 2016. Dostupné z arXiv: 1609.03499.
34. TANG, W. et al. Rethinking 1D-CNN for Time Series Classification: A Stronger Baseline. *CoRR*. 2022. Dostupné z arXiv: 2002.10061.
35. LIN, M. et al. Network in network. *arXiv preprint arXiv:1312.4400*. 2013. Dostupné tiež z: https://arxiv.org/abs/1312.4400.
36. HOCHREITER, S.; SCHMIDHUBER, J. LSTM can Solve Hard Long Time Lag Problems. In: MOZER, M.C. et al. (ed.). *Advances in Neural Information Processing Systems*. MIT Press, 1996, zv. 9. Dostupné tiež z: https://proceedings.neurips.cc/paper_files/paper/1996/file/a4d2f0d23dcc84ce983ff9157f8b7f88-Paper.pdf.
37. WIKIMEDIA COMMONS. *LSTM Cell* [online]. 2018. [cit. 2023-04-27]. Dostupné z : https://commons.wikimedia.org/wiki/File:LSTM_Cell.svg.
38. RICHSTEIN, J. Verifying the Goldbach conjecture up to 4 10^{14}. *Mathematics of computation*. 2001, roč. 70, č. 236, s. 1745–1749.
39. TANG, W. et al. Omni-Scale CNNs: a simple and effective kernel size configuration for time series classification. *arXiv preprint arXiv:2002.10061*. 2020. Dostupné tiež z: https://arxiv.org/abs/2002.10061v1.
40. HINTON, G. E. et al. Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*. 2012. Dostupné z arXiv: 1207.0580.
41. CAI, S. et al. Effective and Efficient Dropout for Deep Convolutional Neural Networks. *CoRR*. 2019. Dostupné z arXiv: 1904.03392.
42. BILLA, J. Dropout Approaches for LSTM Based Speech Recognition Systems. In: *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2018, s. 5879–5883. Dostupné z DOI: 10.1109/ICASSP.2018.8462544.
43. IOFFE, S.; SZEGEDY, C. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In: BACH, F.; BLEI, D. (ed.). *Proceedings of the 32nd International Conference on Machine Learning*. Lille, France: PMLR, 2015, zv. 37, s. 448–456. Proceedings of Machine Learning Research. Dostupné tiež z: https://proceedings.mlr.press/v37/ioffe15.html.
44. SMITH, L. N. Cyclical Learning Rates for Training Neural Networks. In: *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*. 2017, s. 464–472. Dostupné z DOI: 10.1109/WACV.2017.58.
45. SNOEK, J. et al. Practical bayesian optimization of machine learning algorithms. *Advances in neural information processing systems*. 2012, roč. 25.
46. LI, L. et al. Hyperband: A novel bandit-based approach to hyperparameter optimization. *The Journal of Machine Learning Research*. 2017, roč. 18, č. 1, s. 6765–6816.

47. MEGA ELECTRONICS LTD. 800778 eMotion Faros Series Manual. 2017, s. 17. Dostupné tiež z: https://ecgcloud.co.uk/software/800778-2.3.0%5C%20eMotion%5C%20Faros%5C%20Series%5C%20Manual.pdf.

48. NEMCOVA, A. et al. Brno University of Technology ECG Quality Database (BUT QDB) (version 1.0.0). *Physionet* [online]. 2020 [cit. 2023-01-23]. Dostupné z DOI: 10.13026/kah4-0w24.

49. GOLDBERGER, A. L. et al. PhysioBank, PhysioToolkit, and PhysioNet: Components of a New Research Resource for Complex Physiologic Signals. *Circulation*. 2000, roč. 101, č. 23, s. 215–220. Dostupné z DOI: 10.1161/01.CIR.101.23.e215.

50. CHEN, Z. et al. Post-processing refined ECG delineation based on 1D-UNet. *Biomedical Signal Processing and Control*. 2023, roč. 79, s. 104106. ISSN 1746-8094. Dostupné z DOI: https://doi.org/10.1016/j.bspc.2022.104106.

Obsah priloženého archívu

`thesis` ..... súbor obsahujúci časti ohľadne textu práce
- `src` ..... zdrojová forma práce vo formáte LaTeX
- `koledjoz_thesis.pdf` ..... text práce vo formáte PDF