UNIVERZITA PARDUBICE

FAKULTA ELEKTROTECHNIKY A

INFORMATIKY

DIPLOMOVÁ PRÁCE

2020 Bc. Zdeněk Novotný

Univerzita Pardubice

Fakulta elektrotechniky a informatiky

Podpora diagnostiky nemocí pohybového aparátu pomocí umělé neuronové sítě

Diplomová práce

2020 Bc. Zdeněk Novotný

Univerzita Pardubice
Fakulta elektrotechniky a informatiky
Akademický rok: 2019/2020

ZADÁNÍ DIPLOMOVÉ PRÁCE
(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Bc. Zdeněk Novotný**
Osobní číslo:	**I18247**
Studijní program:	**N2646 Informační technologie**
Studijní obor:	**Informační technologie**
Téma práce:	**Podpora diagnostiky nemocí pohybového aparátu pomocí umělé neuronové sítě**
Zadávající katedra:	**Katedra softwarových technologií**

Zásady pro vypracování

Postup: Cílem práce je návrh a implementace systému pro podporu diagnostiky nemocí pohybového aparátu na základě zpracování velkých dat pomocí umělé neuronové sítě. Student bude diplomovou práci řešit ve spolupráci s Fakultní nemocnicí Královské Vinohrady, která poskytne prvotní data pro návrh řešení, zajištění odborných konzultací na pracovištích, pro která je řešená problematika aktuální, a také asistenci při kvalitativních testech.
Teoretická část: Stručný popis chorob, na které je diagnostický nástroj mířen. Stručná rešerše existujících diagnostických nástrojů založených na metodách umělé inteligence. Popis senzorové techniky pro sběr dat. Metodika sběru dat. Popis softwarových nástrojů použitých pro řešení praktické části.
Praktická část: Analýza poskytnutých dat. Návrh a implementace softwarového nástroje pro podporu diagnostiky založeného na vybraném paradigmatu umělých neuronových sítí. Komplexní testování aplikace. Dokumentace k aplikaci včetně testovacího scénáře demonstrujícího použití.

Rozsah pracovní zprávy: **60 s.**
Rozsah grafických prací:
Forma zpracování diplomové práce: **tištěná/elektronická**

Seznam doporučené literatury:

AMATO, Filippo, Alberto LOPEZ, Eladia María PENA-MENDEZ, Petr VANHARA, Ales HAMPL a Josef HAVEL. Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine [online]. 2013, 11(2), 47-58 [cit. 2019-10-08]. DOI: 10.2478/v10136-012-0031-x. ISSN 1214021X.
MOEIN, Sara. Medical diagnosis using artificial neural networks. Hershey, PA: Medical Information Science Reference, an imprint of IGI Global, [2014]. ISBN 978-1466661462.
HAYKIN, Simon S., c2009. Neural networks and learning machines. 3rd ed. New York: Prentice Hall. ISBN 9780131471399

Vedoucí diplomové práce: **doc. Ing. Petr Doležel, Ph.D.**
Katedra řízení procesů

Datum zadání diplomové práce: **5. listopadu 2019**
Termín odevzdání diplomové práce: **15. května 2020**

Univerzita Pardubice Česká republika
L.S.

Ing. Zdeněk Němec, Ph.D.
děkan

prof. Ing. Antonín Kavička, Ph.D.
vedoucí katedry

V Pardubicích dne 15. listopadu 2019

Prohlašuji:

Tuto práci jsem vypracoval samostatně. Veškeré literární prameny a informace, které jsem v práci využil, jsou uvedeny v seznamu použité literatury.

Byl jsem seznámen s tím, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, zejména se skutečností, že Univerzita Pardubice má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 autorského zákona, a s tím, že pokud dojde k užití této práce mnou nebo bude poskytnuta licence o užití jinému subjektu, je Univerzita Pardubice oprávněna ode mne požadovat přiměřený příspěvek na úhradu nákladů, které na vytvoření díla vynaložila, a to podle okolností až do jejich skutečné výše.

Beru na vědomí, že v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, a směrnicí Univerzity Pardubice č. 7/2019 Pravidla pro odevzdávání, zveřejňování a formální úpravu závěrečných prací, ve znění pozdějších dodatků, bude práce zveřejněna prostřednictvím Digitální knihovny Univerzity Pardubice.

V Pardubicích dne 29. 5. 2020

Bc. Zdeněk Novotný

PODĚKOVÁNÍ

Děkuji tímto doc. Ing. Petru Doleželovi, Ph.D. za zprostředkování poutavého tématu z oblasti umělé inteligence a rovněž za jeho vedení a rady při realizaci celé diplomové práce. Dále děkuji doc. Ing. Janu Marešovi, Ph.D. a Fakultní nemocnici Královské Vinohrady za poskytnutí konkrétního problému z oblasti biomedicíny a potřebných podkladů k jeho řešení.

ANOTACE

Tato diplomová práce pojednává o automatizaci diagnostiky onemocnění pohybového aparátu pomocí zpracování velkých dat umělou neuronovou sítí. Nejprve je čtenář seznámen s předmětnou nemocí a současnou metodou diagnostiky, dále jsou popsány umělé neuronové sítě a provedena stručná rešerše již existujících diagnostických nástrojů na stejném základě. V závěru první části je pozornost věnována vybraným softwarovým nástrojům pro vlastní realizaci. Ve druhé části jsou analyzována poskytnutá data a dokumentován návrh a implementace výsledné aplikace.

KLÍČOVÁ SLOVA

biomedicína, diagnostika, Keras, nemoci pohybového aparátu, Python, Tensorflow, umělá inteligence, umělá neuronová síť, velká data

TITLE

Diagnostics Support of Musculoskeletal Diseases Using Artificial Neural Network

ANNOTATION

The diploma thesis discusses automatisation of musculoskeletal diseases using big data processing, based on an artificial neural network. The first part focuses on the disease itself and its current diagnostics method. Later on, artificial neural networks are introduced, as well as other present artificial neural network based diagnostics applications. The second part documents the analysis of provided data and implementation of the resulting application.

KEYWORDS

biomedicine, diagnostics, Keras, musculoskeletal diseases, Python, Tensorflow, artificial intelligence, artificial neural network, big data

OBSAH

SEZNAM ILUSTRACÍ A TABULEK

SEZNAM ZKRATEK A ZNAČEK

CPU – Central Processing Unit

GPU – Graphics Processing Unit

TPU – Tensor Processing Unit

ÚVOD

Vestibulární schwannom je nádorové onemocnění, které může způsobit mnohé problémy či dokonce smrt. V případě obtíží bývá přistoupeno k chirurgickému odstranění, při kterém však hrozí trvalé následky zejména na pohybovém ústrojí. V rámci této práce bude pozornost věnována skupině obličejových nervů, jejichž poškození vede ke ztrátě citu, paralýze či asymetriím v obličeji.

Proto jsou pacienti před operací a později během rehabilitace několikrát hodnoceni při provádění specifických obličejových cviků, které mají za úkol případné poškození odhalit a kvantifikovat. To má však určitá úskalí. Existuje totiž hned několik škál, podle kterých lze poškození hodnotit a hodnocení je vždy ovlivněno subjektivním názorem hodnotitele, tedy konkrétního doktora.

Pomocí rozboru obrazu by bylo možné tyto cviky, snímané na kameru a dále zpracované, hodnotit strojově a tím jak potlačit subjektivitu hodnocení, tak kvantifikovat kvalitu pohybů na jednotné škále. Cílem této práce je navrhnout a implementovat aplikaci, která umožní strojovou diagnostiku předzpracovaných obrazových dat s využitím umělé neuronové sítě. Řešení pro časosběrná data pohybu vybraných bodů pacientova obličeje poskytne výsledek, kvantifikující jeho poškození. Výstupem bude celé číslo z rozsahu <0; 9>.

Nejprve bude představen samotný vestibulární schwannom, jehož pooperační diagnostiku má výsledná aplikace za cíl, jeho příznaky a důsledky. Zmínka zde bude také o sběru obrazových dat, která slouží jako vstupy a pozornost bude věnována též umělým neuronovým sítím, které aplikace použije k určení výsledku.

Dále bude provedena stručná rešerše již existujících diagnostických aplikací, které se taktéž opírají o zpracování umělou inteligencí. A na závěr první části budou zmíněny také softwarové nástroje, bez kterých by se implementace neobešla.

Nakonec bude přistoupeno k návrhu samotné aplikace a k její implementaci. Výsledkem by měl být nástroj, který pro zadaná data pacienta poskytne výsledné hodnocení. Vzhledem k zaměření bude důraz kladen především na funkčnost, nicméně pro představu o realizovaných obličejových cvicích bude vytvořena též jednoduchá vizualizace.

1 Vestibulární schwannom

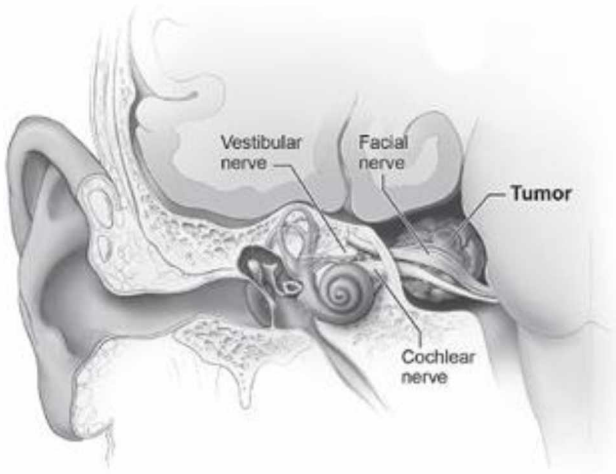
1.1 Popis

Jedná se o nezhoubný nádor v oblasti vnitřního ucha. Jak název napovídá, tvoří se na vestibulárním nervu a je způsoben zvýšenou produkcí tzv. Schwannových buněk, které tvoří obal nervu [2]. Roste obvykle pomalu a může se objevit v kterémkoliv věku, nejčastěji však mezi dvacátým a šedesátým rokem života [1]. Objevuje se zhruba u jednoho obyvatele ze sta tisíc ročně, častěji u žen, než u mužů [2].

Pokud je nádor malý, viditelně neroste a nepůsobí pacientovi potíže, je možné jeho pozorování pravidelnou magnetickou rezonancí. Problémy nastávají, když začne tlačit na okolní nervy [2].

1.2 Příznaky

V těsné blízkosti vestibulárního schwannomu se nachází nervy sluchové a rovnovážné. Mezi běžné projevy tak patří částečná ztráta sluchu, zvonění v uších, či ztráta rovnováhy a závrať. Jak nádor roste, může zasáhnout i obličejové nervy a způsobit tak znecitlivění a paralýzu obličeje. Takové příznaky negativně ovlivňují kvalitu každodenního života pacienta. Dalším růstem může dokonce začít tlačit na mozek a případně přivodit i smrt [1], [2].



Obrázek 1: Vestibulární schwannom ve vnitřním uchu, zdroj: [5]

1.3 Léčba

Vestibulární schwannom se obvykle léčí radioterapií, nebo chirurgickým odstraněním [1], [2]. K chirurgickému zásahu se přistupuje především tehdy, pokud je nádor větší než 2,5 cm a působí potíže [1]. Během operace je zničen vestibulární nerv a hrozí poškození ostatních blízkých nervů. Příznaky se tak zásahem mohou i skokově zhoršit [1], [2].

Zejména jsou ohroženy obličejové nervy. Důsledky jejich poškození jsou mnohé, od narušení pohyblivosti obočí nebo neschopnosti zavřít oči, přes zhoršení pohyblivosti úst a tím i horší artikulaci, po obličejové asymetrie, které vedou ke komplikacím v neverbální komunikaci [1].

1.4 Rehabilitace

Z výše uvedených důvodů absolvuje pacient pooperační rehabilitaci, která trvá několik měsíců. Součástí rehabilitace je i série obličejových cviků, pomocí níž je hodnocena pohyblivost obličeje. Tyto cviky jsou nejdříve hodnoceny týden před samotnou operací a jejich případné zhoršení, pozorované v rámci rehabilitace, vypovídá o rozsahu poškození obličejových nervů v důsledku zákroku. Pooperační hodnocení cviků probíhá s odstupem týdne, měsíce a tří měsíců od operace [1].

1.5 Diagnostika

Současná metoda hodnocení spočívá ve vizuální kontrole cviků lékařem. Činnost obličejových nervů lze hodnotit různými škálami, ze své podstaty je však výsledek vždy subjektivní [1]. Z toho vyplývá potřeba automatizace diagnostiky. Strojová metoda zaprvé potlačí vliv subjektivního hodnocení konkrétního doktora a zadruhé umožní zavést jednotnou a srozumitelnou škálu ke kvantifikaci stavu pacienta. V důsledku by mohla vést i k možnosti průběžné domácí sebediagnostiky pacientem, za předpokladu dostupnosti dostatečně přesných snímacích zařízení.

2 Sběr dat

2.1 Snímač

Ke strojové diagnostice jsou zapotřebí obrazová data pacienta. Tato data jsou v případě řešeném v této práci pořizována pomocí snímače Microsoft Kinect [1]. Kinect je pohybový snímač, disponující RGB kamerou, infrakamerou, čidly vzdálenosti a mikrofonem. V dnešní době se již nevyrábí, nicméně stále představuje kvalitní nástroj k pořizování stereoskopického vizuálního záznamu [23]. Snímač je během sběru dat umístěn staticky, tak, aby zabíral obličej pacienta [1].

2.2 Cviky

Pacient je vyzván k provedení celkem devíti cviků, například zavření očí, vycenění zubů, zvednutí obočí, úsměv, mračení, nafouknutí tváří a podobně. Průběh jeho cvičení je Kinectem zaznamenáván v podobě stereoskopického videa a ze záznamu je následně vytvořen trojrozměrný model pacientovy hlavy [1].

2.3 Body

Z modelu je vybráno celkem jedenadvacet sledovaných bodů. Jelikož je model trojrozměrný, je k dispozici poloha v čase a třech prostorových dimenzích. Tyto časosběry tvoří podklad pro následnou analýzu [1].

Snímané body jsou vyjmenovány v tabulce níže.

p	Kinect	position	*p*	Kinect	position
0	1104	left eye, bottom	1	241	left eye, top
2	210	left eye, inner	3	469	left eye, outer
4	346	left eyebrow, inner	5	222	left eyebrow, middle
6	1090	right eye, bottom	7	731	right eye, top
8	843	right eye, inner	9	1117	right eye, outer
10	803	right eyebrow, inner	11	849	right eyebrow, middle
12	18	nose, center	13	8	mouth, bottom
14	91	mouth, left	15	687	mouth, right
16	19	mouth, top	17	4	chin
18	28	forehead	19	412	left cheek
20	933	right cheek			

Tabulka 1: Snímané body, zdroj: [1]

2.4 Hodnocení

Důležitou složkou je také hodnocení kvalifikovaného lékaře. Lékař pacienta hodnotí dle svého odborného úsudku. Jím udělená známka se k sebraným datům přidružuje, spolu s dalšími informacemi, jako je identifikátor pacienta či datum pořízení záznamu, a slouží jako podklad

pro chybové učení (bude vysvětleno dále). Součástí je i předoperační hodnocení pacienta, které pomůže určit, nakolik se stav pacienta změnil.

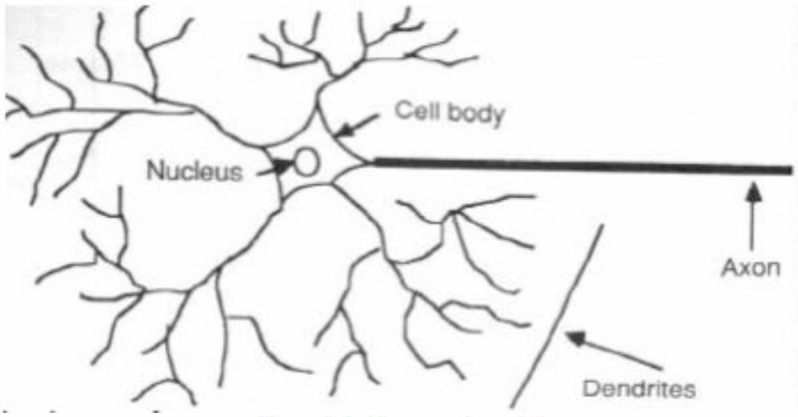
3 Neuronová síť

3.1 Neuron

Nervová buňka (neuron) je základním stavebním elementem biologického informačního systému. Umožňuje zpracování informací, jejich uchování a přenos a zabezpečuje tak základní funkce organismu, jakož i reakce na vzruchy, komunikaci nebo uvažování.

Neuron je tvořen tělem (soma), ze kterého vychází dendrity a axon. Vše je zabaleno buněčnou membránou. Dendrity jsou krátké a hodně větvené a představují vstupní spoje neuronu [3]. Každý neuron má kolem tisíce až deseti tisíc dendritů [4]. Axon je výstupním spojem, může být i desítky centimetrů dlouhý a je v neuronu vždy jen jeden [3].

Pro rychlejší přenos informací bývá neuron obalen takzvaným myelinem, tukovým obalem, který obsahuje již zmíněné Schwannovy buňky. Mezi úseky obalu jsou mezery, takzvané Ranvierovy zářezy, ve kterých se obnoví plná intenzita signálu [3].



Obrázek 2: Neuron, zdroj: [4]

3.2 Neuronová síť

Každý neuron provádí agregaci informací z dendritů. Pokud výsledná hodnota překročí určitý práh, vyšle signál axonem. Axon se na konci větví a na zakončení těchto výběžků jsou mnohdy takzvanými synapsemi napojeny další neurony. Toto spojení může vést na dendrit jiného neuronu, axon jiného neuronu, nebo i přímo na tělo jiného neuronu. Informace se tak předá z jednoho neuronu dalším. Takto propojené neurony tvoří neuronovou síť [3].

Každý neuron může být propojen s tisíci dalších neuronů [3], [7]. Při odumření neuronu dochází k reorganizaci neuronové sítě, při které si dendrity neuronů, které byly s odumřelým neuronem spojeny, najdou jiný neuron, na který se napojí [3]. Nervová soustava člověka je v podstatě obrovská neuronová síť, zahrnující cca 10^{11} až 10^{12} neuronů [7].

3.3 Inteligence

Definic inteligence je více. Například profesor William Stern, známý psycholog, definoval inteligenci takto: "Inteligence je všeobecná schopnost individua vědomě orientovat vlastní myšlení na nové požadavky, je to všeobecná duchovní schopnost přizpůsobit se novým životním úkolům a podmínkám." [6].

Tato definice (a podobně i ostatní uznávané definice [6]) podmiňuje inteligenci zpracováváním informací a přizpůsobováním se, tedy schopností učení.

3.4 Učení

Bez zkušeností jsou reakce na podněty reflexivní či instinktivní. Získáním zkušenosti lze při dalším výskytu stejné nebo podobné situace reagovat lépe. Učení umožňuje si situaci zapamatovat a později ji využít při rozhodování.

V podstatě je tento proces změnou vah v biologické neuronové síti. V situaci, kterou zažíváme, jsou aktivovány receptory, které slouží jako senzory okolí a jsou vstupní branou naší sítě. Receptory přemění vjemy na elektrické signály a vyšlou je jednotlivým neuronům. Jak již bylo naznačeno, pokud agregovaná hodnota vstupů nepřekoná práh aktivace konkrétního neuronu, je tento signál ztracen. Pokud ji překoná, neuron vyšle signál dalším neuronům. Takto se signály šíří neuronovou sítí a nakonec je generována určitá reakce.

Při průchodu signálu se ale navíc na několik sekund tvoří takzvané paměťové stopy a při opakovaném průchodu signálu se propustnost používaných synapsí mění. Paměťové stopy pak mohou vydržet i řádově hodiny. S odstupem času, typicky ve spánku, se působením paměťových stop modifikuje vnitřní struktura neuronů na řádově roky. Tento jev umožňuje efekt dlouhodobé paměti. Na základě pozměněných synapsí může neuronová síť v další podobné situaci generovat jinou reakci [3].

3.5 Umělá inteligence

I umělou inteligenci lze definovat mnoha způsoby. Marvin Minsky, který do tohoto oboru sám značně přispěl, ji definoval takto: „Artificial intelligence is the science of making machines do things that would require intelligence if done by men.". Expertní skupina Evropské komise pro umělou inteligenci, High-Level Expert Group on Artificial Intelligence, ve své poslední zprávě navrhuje následující, poněkud výřečnou definici: „Systémy umělé inteligence (AI) jsou softwarové (ale také hardwarové) systémy vytvořené lidmi, kterým je dán komplexní úkol jednat ve fyzické nebo digitální dimenzi za pomoci vnímání svého okolí sběrem dat,

interpretace sbíraných strukturovaných nebo nestrukturovaných dat, odůvodňování znalostí nebo zpracovávání informací získaných z dat a vybírání nejlepšího jednání za účelem dosažení stanoveného cíle. Systémy AI mohou využívat symbolická pravidla nebo se učit číselné modely a mohou také přizpůsobit své chování na základě analýzy toho, jak jejich předcházející chování ovlivnilo jejich prostředí. " [12].

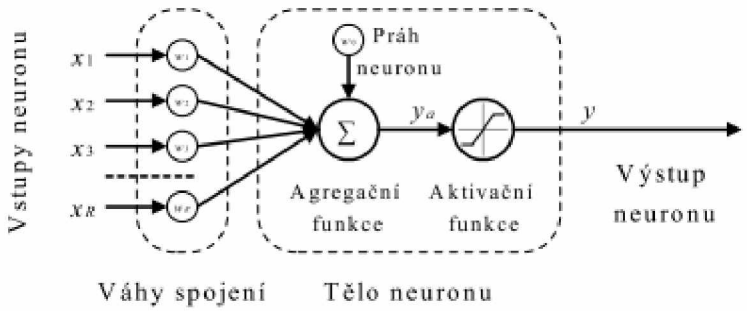
3.6 Umělý neuron

Umělý neuron je implementací matematického modelu neuronu. Níže na obrázku je znázorněn formální model neuronu, také známý dle svých autorů jako McCullochův-Pittsův. Modelů umělých neuronů existuje samozřejmě více, ale zde si pro vysvětlení vystačíme s McCullochovým-Pittsovým. Biologické funkce skutečného neuronu jsou v umělém nahrazeny funkcemi matematickými [3], [7].

Dendrity jsou nahrazeny vektorem hodnot. Jedná se typicky buď o booleovské hodnoty, nebo reálná čísla, avšak vstupní prostor se může lišit v závislosti na konkrétní implementaci [3].

Na hranici neuronu jsou jednotlivé hodnoty vstupního vektoru ohodnoceny vstupními vahami. Vstupní váhy nahrazují synapse skutečných neuronů a určují míru, kterou se konkrétní vstup neuronu participuje na jeho výstupu [3].

Práh umělého neuronu nemusí odpovídat prahu skutečného neuronu. Zatímco v biologickém neuronu slouží k zániku slabých signálů a je tedy od signálu v podstatě odečten, v umělém neuronu představuje obecný posun intenzity signálu. Jedná se tedy o hodnotu, která ovlivňuje výsledek a přitom sama nezávisí na hodnotě vstupního vektoru [3], [7].



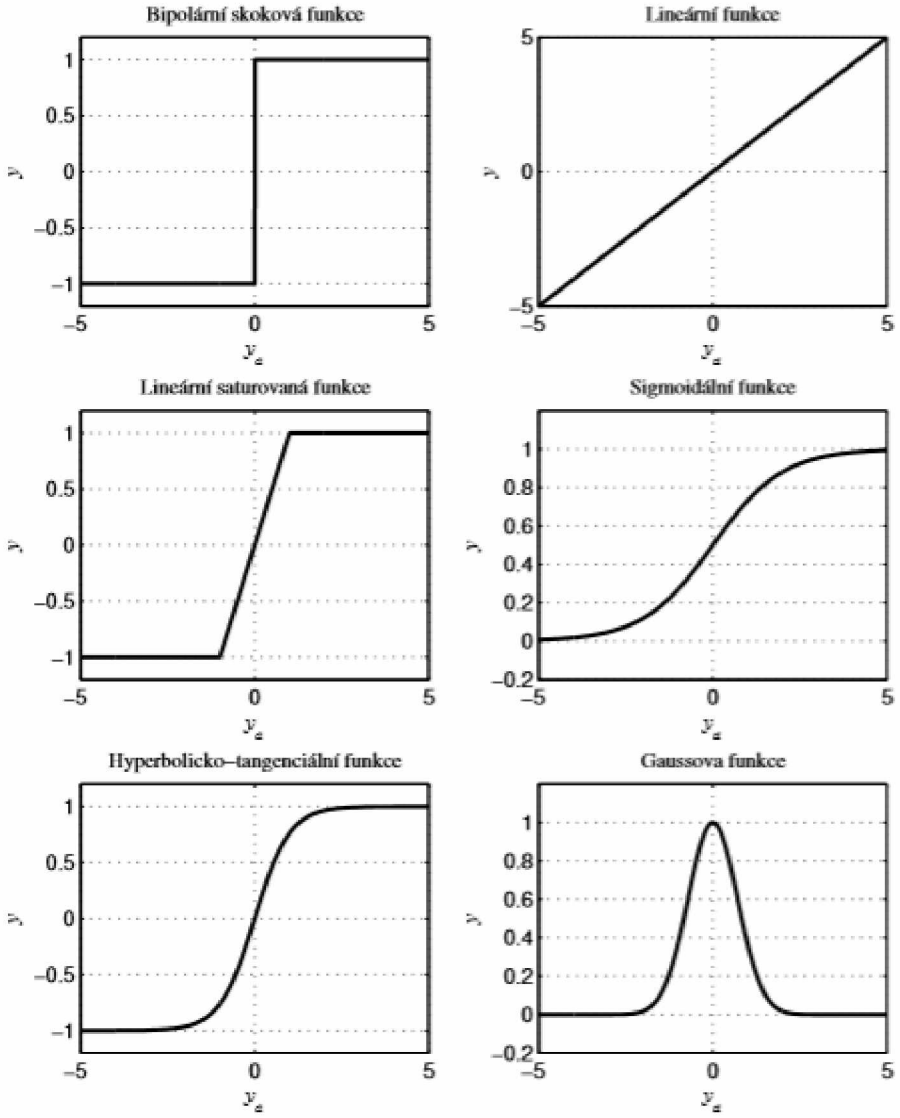
Obrázek 3: Formální model umělého neuronu, zdroj: [3]

Pomocí agregační funkce je z vektoru (již vahami ohodnocených) vstupů vypočítána jediná hodnota. V případě uvedeného formálního modelu, kde je vážení vstupů definováno jako součin a agregační funkce coby suma, ji lze vyjádřit jako

$$y_a = \sum_{i=0}^{R} x_i w_i; i \in \{0, 1, \dots n\}, x_0 \in \{-1,1\}, zdroj: [3]$$

Stojí přitom za povšimnutí, že w_0 je prahem neuronu, nikoli skutečnou vahou vstupu. V praxi ovšem může mít agregační funkce, stejně jako vážení vstupů, úplně jiný předpis.

Agregované vstupy jsou následně předány aktivační funkci, která z nich určí výslednou hodnotu signálu. Aktivační funkce je volena v souladu s požadavky na výstup. Použít lze rozličné lineární i nelineární, diskrétní i spojité funkce. Některé běžně používané jsou popsány na následujícím obrázku.



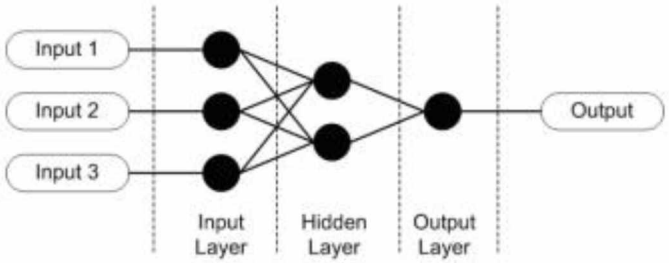
Obrázek 4: Vybrané aktivační funkce, zdroj: [3]

Vhodnou volbou aktivační funkce je vymezen obor hodnot výstupu a může též zvýrazňovat jeho podstatné rysy [3]. Některé zdroje doporučují používat pouze nelineární aktivační funkce [7].

3.7 Umělá neuronová síť

Analogicky k biologické neuronové síti, umělá neuronová síť vzniká propojováním umělých neuronů. Agregovaný a aktivovaný signál jednoho neuronu slouží jako vstup dalším neuronům, nebo jsou vstupní zpracována několika „sousedními" neurony a výstupem je vektor jejich výsledků.

Topologie umělých neuronových sítí se může značně lišit, v závislosti na transformaci, pro kterou je jednotlivá síť navržena. Obvykle rozlišujeme jednotlivé vrstvy [3]. Vstupní vrstvu, která slouží pro příjem vstupních signálů a rozměrem tak nutně odpovídá vektoru (či matici, tensoru) vstupu. Na druhé straně je vrstva výstupní, která poskytuje výsledek transformace signálu. Rozměry výstupní vrstvy udávají rozměry výsledku transformace a její aktivační funkce určuje obor hodnot.



Obrázek 5: Neuronová síť, zdroj: [4]

Mezi vstupní a výstupní vrstvou mohou být ještě takzvané skryté vrstvy, které provádějí další transformace. Za umělou neuronovou síť lze považovat i samostatný neuron, který tak zastává všechny vrstvy sítě najednou [3]. Příkladem je neuronová síť typu jednoduchý perceptron, ke které se ještě vrátíme. Pojem „hluboké učení" se naopak používá pro sítě s více než třemi vrstvami, včetně vstupní a výstupní vrstvy. Týká se tedy sítí s více než jednou skrytou vrstvou.

Další důležitou informací o topologii je způsob propojení neuronů, ze kterého vyplývá i tok signálu sítí. Neurony mohou mít spoje mezi sousedními vrstvami, v rámci jedné vrstvy, nebo dokonce zpětné vazby. Z hlediska toku lze umělé neuronové sítě dělit na dopředné (jednosměrné) a rekurentní (obsahující zpětnovazební smyčky) [3].

3.8 Učení umělé neuronové sítě

Tak, jako skutečný neuron agreguje vstupní signály a při překonání prahu vyšle výsledný signál axonem, umělý neuron vstupy agreguje, aktivuje a poskytne výstup. A analogicky k učení neuronu, který upraví své váhy na používaných synapsích podle paměťových stop, umělý neuron musí upravit vstupní váhy, aby poskytovaly správný výsledek.

Učení umělých neuronových sítí lze rozdělit na učení s učitelem a učení bez učitele. Učení bez učitele, neboli samoorganizace, se používá u umělých neuronových sítí, realizujících shlukovou analýzu. Takové sítě hledají ve vstupních datech podobné vzory a podle nich je rozřazují do shluků. Při učení se používá princip podobný zmíněným paměťovým stopám. Váhy neuronu, které hrály při klasifikaci významnou roli, se posílí, zatímco ostatní oslabí. Pokud tedy následně přijde na vstup signál s podobným vzorem, je vyšší šance, že bude zařazen do stejného shluku [3].

Učení s učitelem, nebo též chybové učení, je mnohem častější. Neuronu jsou poskytnuta data, která má transformovat a ke kterým existuje i požadovaný výstup. Požadovaný výsledek je porovnán s predikovaným výsledkem, a pokud byla predikce špatně, neuronu jsou upraveny váhy tak, aby příště predikoval lépe [3].

Na úrovni sítě je princip stejný, ovšem změny je třeba distribuovat všem dotčeným neuronům. Při učení bez učitele se váhy upravují již při průchodu signálu. Při chybovém učení je třeba implementovat takzvané zpětné šíření chyby, v rámci kterého jsou úpravy vah distribuovány od výstupní vrstvy, kde je chyba odhalena, zpět směrem ke vstupní vrstvě [3].

Stejně jako předloha, i umělá neuronová síť vyžaduje ke kvalitnímu naučení opakování. Počet opakování záleží na konkrétním případu, není však výjimkou učit síť tisíce epoch. Jedna epocha představuje jednu generaci učení, tedy jeden průchod trénovacích vzorků.

Obvykle se doporučuje rozdělení dat na trénovací, validační a testovací množinu. Největší, testovací množina, by měla pokud možno pokrývat co nejšíře spektrum možných vstupů. Touto množinou je síť trénována, s její pomocí si upravuje váhy. Po každé epoše je síti předložena validační množina k predikci a sleduje se, zda se její predikce zlepšuje. V okamžiku, kdy se predikce na validační množině začne naopak zhoršovat, je již síť za svým vrcholem a učí se trénovací vzorky „nazpaměť“. Trénování má tedy smysl, jen dokud se výsledky na validační množině zlepšují. Po natrénování sítě přichází řada na testovací množinu, jejíž úspěšnost predikce je ukazatelem, nakolik je síť prakticky použitelná. [3]

Ve složitějších příkladech lze narazit na sofistikovanější úpravy v síti, než je prostá úprava vah. Řeč je například o úpravě koeficientů aktivačních funkcí, nebo dokonce o změnách ve vnitřní topologii sítě. [3]

3.9 Uplatnění umělé neuronové sítě

Princip predikce umělé neuronové sítě byl již zmiňován. Jednotlivé neurony zpracovávají signály pomocí svých vnitřních struktur a důsledkem vzájemné interakce v rámci sítě poskytují sofistikovanější predikce. Princip chybového učení umožňuje upravovat váhy se znalostí pouze vstupu a výstupu. Není vůbec zapotřebí znát postupy, které k rozhodnutí vedly.

To je zásadní rozdíl oproti expertním systémům, které naopak vyžadují podrobné zmapování všech vnitřních rozhodovacích procesů. Tudíž lze s výhodou umělé neuronové sítě využívat pro procesy, u kterých podrobné zmapování není možné, nebo se nevyplatí.

Oproti stochastickým algoritmům pak nabízejí rychlejší zpracování. Na rozdíl od stochastických metod hledání minima, které se projevují náročnými výpočty v momentě definice všech vstupů, u umělých neuronových sítí je náročnost výpočtů soustředěna na fázi učení, zatímco predikce pro konkrétní vstupy již bývá rychlá.

Není proto překvapením, že se umělé neuronové sítě stále výrazněji prosazují v rozličných oborech, včetně bankovnictví, dopravy, elektroniky, medicíny, sociálního inženýrství, telekomunikací, výroby, vzdělávání a dalších. Řeší zejména třídní klasifikaci, shlukovou analýzu, rozpoznávání vzorů a predikci. [3]

4 Umělé neuronové sítě a zpracování časosběrných dat

4.1 Přehled

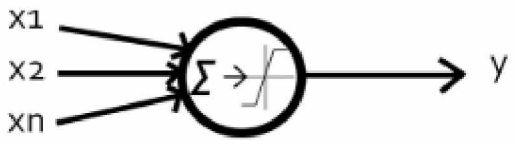
Pro klasifikaci časosběrných dat lze použít rozličné topologie umělé neuronové sítě. Kvalita predikce různých topologií je částečně funkcí vstupu. Záleží na rozměrech časové řady, šumu i na případné vzájemné závislosti jednotlivých složek [8]. Zde si uvedeme alespoň několik možných přístupů.

4.2 Jednoduchý perceptron

Jednoduchý perceptron rozhodně nepatří mezi typické klasifikátory časosběrných dat. Jedná se v podstatě o implementaci formálního modelu neuronu. Představuje velice jednoduchou síť o jediném prvku. Složitější perceptronové sítě, tedy seskupení vzájemně propojených perceptronů, však patří mezi nejpoužívanější umělé neuronové sítě [3]. Je proto správné si jej alespoň zmínit.

Jedná se o jediný neuron, který představuje všechny vrstvy neuronové sítě. Musí pojmout celý vstupní vektor resp. matici, proto musí mít tolik vstupů, kolik má vstupní vektor resp. matice hodnot. Pro každý vstup má samozřejmě i váhu. Vážené vstupy spolu s prahem neuronu jsou sumovány lineárně váženou agregační funkcí a přepočítány typicky skokovou aktivační funkcí [3]. Výstup jednoduchého perceptronu je tedy z rozsahu {0; 1}, případně {-1; 1}, v podstatě booleovská hodnota. Rozdělí množinu vstupů na dvě třídy.

Některé zdroje uvádějí jednoduchý perceptron i s jinou, např. lineární aktivační funkcí [3]. Změnou aktivační funkce se samozřejmě změní i prostor predikovaných hodnot.



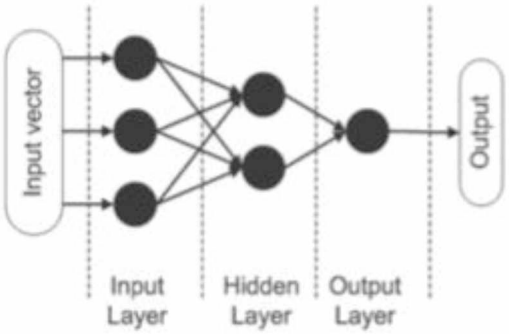
Obrázek 6: Jednoduchý perceptron, zdroj: vlastní

4.3 Vícevrstvý perceptron

Jedná se o jednu z nejpoužívanějších topologií umělých neuronových sítí [8]. Vícevrstvý perceptron se někdy nazývá též plně propojená umělá neuronová síť, nebo dopředná vícevrstvá umělá neuronová síť [3], [8]. Jak z názvů vyplývá, síť je složena z perceptronů, které nejsou propojeny mezi sebou v rámci jedné vrstvy, ale každý neuron z libovolné vrstvy je jednosměrně

propojen s každým neuronem ve vrstvě přímo následující, a to směrem od vstupu k výstupu, nikdy naopak [8].

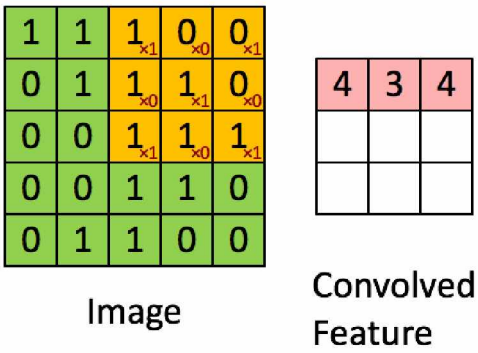
Každý neuron první skryté vrstvy tedy dostane informace o všech vstupech. Vzhledem ke své jednoduchosti je vícevrstvý perceptron velmi oblíbený např. coby aproximátor či prediktor [3]. Nevýhoda při zpracování časové řady je ta, že se v síti ztrácí souvislost mezi hodnotami jednotlivých časových kroků [8].



Obrázek 7: Vícevrstvý perceptron, zdroj: [4]

4.4 Konvoluční neuronová síť

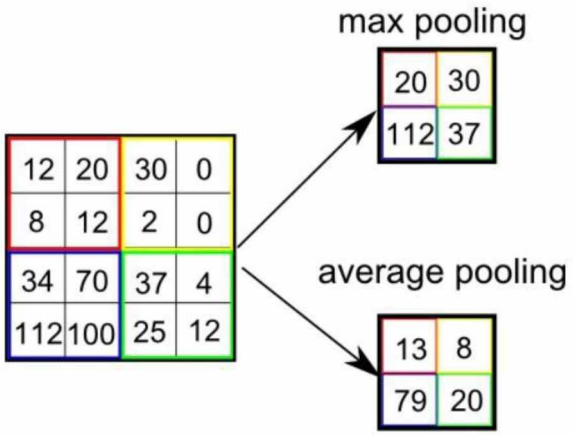
Konvoluce je jednou ze základních operací v teorii signálů. Lze si ji představit jako postupné posouvání filtru časovou řadou. Hodnoty z oblasti o velikosti filtru jsou váženy váhami filtru a agregovány. V závislosti na nastavení konvoluce může, ale nemusí dojít k redukci rozměrů dat. [10]



Obrázek 8: Konvoluce, zdroj: [10]

Konvoluce slouží k potlačení nedůležitých odchylek. Agregací sousedních hodnot se projevují jen významné odchylky, např. hrany v případě obrázků. Ostatně právě v rozboru obrazu se konvoluce velmi často používá [8][10].

Po konvoluční vrstvě často následuje vrstva poolingu, která data dále agreguje. Slouží k potlačení šumu a redukuje rozměry. Používá se max pooling, který z dat o velikosti filtru vybere maximum, nebo average pooling, který data o velikosti filtru zprůměruje [10].



Obrázek 9: Pooling, zdroj: [10]

Konvoluce spolu s poolingem představují nástroj, kterým lze výrazně zmenšit rozměry dat a přitom zachovat důležité informace. V praxi se často používá několik konvolučních a poolingových vrstev střídavě za sebou, čímž jsou z dat postupně extrahovány jen ty nejvýraznější rysy. Po vrstvách konvoluce a poolingu následuje vrstva perceptronů, která umožní rozeznávání spojitostí v agregovaných datech [8][10].

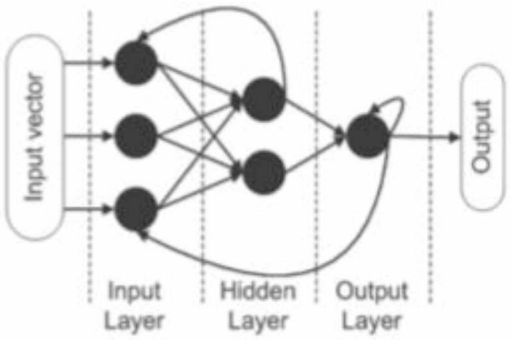
4.5 Plně konvoluční síť

Termín plně konvoluční síť se používá pro sítě, které vychází pouze z konvoluce. Průběžný pooling mezi jednotlivými konvolučními vrstvami se v plně konvoluční síti nepoužívá, rozměr dat tak zůstává zachován. Až na samém konci je vrstva globálního poolingu, který zajistí požadovaný počet výstupů. Výhodou tohoto přístupu je zejména možnost variabilních rozměrů vstupů [8].

4.6 Rekurentní neuronová síť

Rekurentní neuronové sítě disponují určitou formou dočasné paměti. Využívají k určení výstupu nejen aktuální hodnoty, ale též hodnoty z minulosti. To z nich dělá silný nástroj pro predikci dynamických dat, kde je mnohdy zapotřebí zohlednit i předchozí průběh. Z podstaty jsou stavěné spíše na predikci dalších kroků, než pro klasifikaci časosběru jako celku. Přesto existují zmínky o rekurentních neuronových sítích, aplikovaných coby klasifikátory [8][11]. Vycházejí z toho, že signály, procházející rekurentní neuronovu sítí, aktivují odpovídající neurony. Kombinace aktivovaných výstupních neuronů natrénované sítě tak zřejmě vypovídá

o vlastnostech vstupní sekvence a z jejích vlastností lze usoudit příslušnost k třídě. Klasifikace je v podstatě založena na pozorování aktivace neuronů při průchodu sítí [11].



Obrázek 10: Rekurentní neuronová síť, zdroj: [4]

5 Diagnostické nástroje založené na metodách umělé inteligence

Umělá inteligence se prosazuje v mnoha oborech, medicínu nevyjímaje. Používá se například k spektrální analýze, modelování psychických poruch, predikci vývoje onemocnění a zejména k analýze rozličných obrazů a signálů [13].

Oblast diagnostiky v tomto ohledu samozřejmě nezůstává pozadu. Známý systém Watson od firmy IBM radí onkologům s léčbou [14]. V souvislosti s šířením COVID-19 byl naučen také pro poskytování rad ohledně této nemoci [15].

Přestože nejde přímo o diagnostiku, stojí v souvislosti se zmíněnou nemocí COVID-19 jistě za zmínku také systém BlueDot, který před jejím šířením varoval v době, kdy naši političtí představitelé teprve připravovali projevy o tom, že není třeba se čehokoli obávat [16].

Uplatnění našla umělá inteligence i v diagnostice rakoviny kůže. Dosahuje zde dokonce lepších výsledků, než dermatologové [17]. Podobně předčila i radiology v diagnostice rakoviny prsu aplikace vyvinutá v rámci projektu Google Health [18]. Dalším relevantním projektem od Googlu je systém DeepMind, který by měl diagnostikovat oční choroby [17], [19].

Společnost L'Oréal v rámci své kampaně nabízí spolu s technologiemi rozšířené reality také aplikaci pro diagnostiku pleti pomocí rozboru obrazu [20].

Vysoká škola chemicko-technologická v Praze, ve spolupráci s Fakultní nemocnicí Královské Vinohrady, vyvíjí systém pro pooperační diagnostiku nemocí pohybového aparátu pomocí zpracování obrazu. Zmíněný systém zahrnuje robotické snímání chůze člověka a statické snímání obličeje, jejichž záznamy jsou dále analyzovány. Z tohoto projektu vychází i tato diplomová práce [1].

Pomocí umělé neuronové sítě se daří úspěšně diagnostikovat podezření na akutní infarkt myokardu na základě EKG a anamnézy či klasifikovat onemocnění páteře z analýzy krve [13].

V čínské Národní laboratoři pro nové softwarové technologie v Nanjingu byl vyvinut systém pro identifikaci rakovinových jaterních buněk ze snímků vzorků biopsie. Systém využívá celý soubor umělých neuronových sítí ve dvou stupních. První stupeň vyhledává zdravé buňky a buňku označí za zdravou pouze tehdy, když se na její nezávadnosti shodnou všechny sítě prvního stupně. Druhý stupeň klasifikuje buňky, které byly odhaleny coby nádorové, do několika typů rakoviny, případně jako zdravou. V tomto stupni rozhoduje většina [13].

V Epileptickém centru Univerzity Johnse Hopkinsona v Baltimoru diagnostikovali záchvaty umělou neuronovou sítí pomocí dat z encefalogramu s velmi dobrými výsledky [13].

Na univerzitě v Michiganu byla umělá neuronová sít používána k předpovídání diabetu po tři roky s 6142 účastníky a překonala ostatní klasifikační a regresní modely výslednou citlivostí 68,04 % [13].

V souvislosti s diabetem byly vyvinuty i další systémy, používající například rekurentní neuronovou síť k predikci hypoglykémie u pacientů s diabetem prvního typu. [22]

Firma Orr vyvinula poplašný systém k identifikaci selhání v dýchacím cyklu. Sestával ze dvou umělých neuronových sítí, jedné pro identifikaci selhání a druhé pro jeho klasifikaci. Síť selhání detekuje na 96 % a správně klasifikuje na 95,6 % [13].

Tento výčet by mohl být ještě mnohem delší, ale pro představu postačí. Cílem bylo ukázat, jak široce se umělé neuronové sítě v lékařské diagnostice uplatnily a dále uplatňují. Jak je vidět, usnadňují práci lékařům a mnohdy je dokonce v některých ohledech předčí.

6 Softwarové nástroje pro vývoj

6.1 TensorFlow

TensorFlow je otevřená platforma pro strojové učení, vyvinutá společností Google. Jméno dostala podle objektů, se kterými pracuje, tenzorů. Tato platforma umožňuje relativně rychlou a snadnou tvorbu umělých neuronových sítí [23].

Není tedy zapotřebí implementovat síť se všemi jejími váhami, prahem, agregační a aktivační funkcí a obzvláště komplikovaným učením. Tensorflow poskytuje nástroje pro deklarativní přístup [23].

Kromě CPU dokáže pracovat i na GPU [24], což je zásadní pro zpracování v rozumném čase. Pro použití prostředků grafické karty stačí nainstalovat aplikační rozhraní CUDA a několik knihoven. A samozřejmě je také nutné disponovat odpovídající grafickou kartou, s výpočetní kapacitou alespoň 3, 0. Výpočetní kapacita jednotlivých grafických karet s CUDA jádry je dohledatelná na [26].

Další možností je delegování výpočtů na TPU [24], což je speciální procesor, zaměřený na operace s tenzory [27]. TPU byl vyvinut společností Google speciálně pro vývoj umělé inteligence ve spojení s platformou Tensorflow [27].

Alternativou je například framework PyTorch, za jehož vývojem stojí Facebook a který poskytuje velmi podobné funkcionality [25]. I PyTorch je otevřeným softwarem [25]. V této diplomové práci je však zvolen TensorFlow.

6.2 Keras

Keras je aplikační rozhraní pro hluboké učení, které obaluje platformu TensorFlow. Je napsán v programovacím jazyce Python. Poskytuje důležité abstrakce a struktury pro ještě snazší a rychlejší tvorbu umělých neuronových sítí [24].

Tam, kde TensorFlow umožňuje pouze deklarovat proměnné a jejich tok skrz agregační a aktivační funkce, namísto vlastní implementace, Keras zachází ještě dál a umožňuje deklarovat celé vrstvy sítě. Informace o počtu neuronů, aktivační funkci a podobných specifících stačí předat pomocí parametrů [24].

Je volně dostupný pod licencí MIT s povinností uvedení autorského práva a licence při distribuci. Povinnosti je učiněno za dost v příloze B.

6.3 NumPy

NumPy je další knihovna programovacího jazyka python. Slouží k vědeckým výpočtům, obsahuje nástroje pro tvorbu a operace nad maticemi, včetně různých funkcí a transformací. Obsahuje také nástroje pro generování náhodných čísel a pro integraci kódů v programovacích jazycích C, C++ či Fortran [28].

Knihovna NumPy je dostupná volně pod licencí BSD s povinností uvedení autorského práva, podmínek šíření a zřeknutí se odpovědnosti. Tyto náležitosti jsou uvedeny v příloze B.

6.4 Matplotlib

Vizualizace jsou realizovány knihovnou matplotlib. Ta umožňuje komplexní vykreslování v ploše i prostoru a to včetně animací a interaktivních vizualizací. Jedná se o otevřený software [29].

6.5 Tqdm

Jelikož v některých částech aplikace může docházet k značným prodlevám, byla integrována též knihovna tqdm, která poskytuje ukazatel postupu. Ten slouží coby ukazatel činnosti programu v jinak zdánlivě nečinném místě a rovněž poskytuje představu o době, potřebné k dokončení [30].

Tqdm je volně dostupný pod licencí MIT s povinností uvedení autorského práva a licence při distribuci. Ta je uvedena v příloze C.

6.6 Python

Python je interpretovaný, vyšší programovací jazyk. Je objektově orientovaný a poskytuje užitečné datové struktury, dynamické typování a vazby, díky čemuž v něm lze vyvíjet velmi rychle [31]. Mnohdy je používán i coby skriptovací jazyk nebo pro kompletaci jiných skriptů/kódů do jednoho spustitelného programu. Umožňuje přímé spouštění programů, psaných jazyce C a C++ [32].

Python je oblíbený také pro vysokou míru abstrakce a pro vynucování čitelně psaného kódu. Speciální význam zde mají i bílé znaky, takže nedodržení odřádkování nebo odsazení změní význam kódu, případně jej učiní neinterpretovatelným [32].

Nechybí ani garbage collector, který po programátorovi průběžně uklízí. Interpret Pythonu má navíc přímo integrovaný debugger, takže při výjimkách poskytuje relevantní hlášky s výpisem

zásobníku volání [31]. Všechny tyto aspekty dělají z Pythonu velmi pohodlný nástroj pro vývoj. Jedná se o otevřený software [31], [32].

6.7 Kivy

Pro vytvoření jednoduchého grafického uživatelského rozhraní je použit framework Kivy. Jedná se o otevřenou knihovnu. Využívá OpenGL ES 2. [33] Existuje samozřejmě mnoho dalších možností, například PyQt5 (Qt je velmi rozšířený grafický framework pro C++, nabízí rozsáhlou dokumentaci a podporu), Tkinter, Pyside2 (opět Qt) a další. Kivy byl vybrán, protože je považován za velmi moderní a v žebříčcích grafikých knihoven pro Python se umisťuje velmi vysoko [34], [35], [36].

6.8 IntelliJ IDEA

Jedná se o velmi komplexní integrované vývojové prostředí, zaměřené především na vývoj v jazyce Java. Nabízí pokročilou kompletaci příkazů a bloků, analýzu kódu a silné refaktorizační nástroje [17]. Samotné vývojové prostředí je také napsáno v Javě, takže je multiplatformní.

Podle výrobce používá IDEA 65 % vývojářů Javy [17]. Používání shledal autor velmi intuitivní, pohodlné a efektivní. Zmíněné vývojové prostředí rozhodně patří ke špičce [38], [39], [40].

Velkou devízou je jeho modulárnost, zajištěná velkým množstvím plug-inů. Nechme stranou různé lintery, spellcheckery, kontroly pokrytí, nástroje pro databázi či soubory, cloudové služby a podobně. IntelliJ IDEA poskytuje zásuvné moduly pro velké množství frameworků a programovacích jazyků, nevyjímaje samozřejmě ani Python.

Společnost JetBrains, která toto vývojové prostředí vyvíjí, vyprodukovala i vývojové prostředí přímo pro Python, jménem PyCharm, ovšem podle samotného výrobce se jedná o vývojové prostředí s užším zaměřením a jeho jedinou zmiňovanou výhodou je jednoduchost [41]. Zdá se tedy, že pro uživatele, zvyklého na prostředí IDEA, nemá migrace na PyCharm smysl.

Jedná se o komerční software, nicméně JetBrains poskytují studentům (a učitelům, ba dokonce i hromadně školám) studijní licence zdarma [42].

6.9 Visual Studio Code

Jedná se o produkt firmy Microsoft, ovšem známému vývojovému prostředí se Code podobá jen z části. Sám výrobce o něm hovoří jako o editoru kódu [43]. Narozdíl od robustního Visual Studia vládne v Code minimalistický design. Většina operací probíhá pomocí klávesových

zkratek, a pokud je uživatel zapomene, použije fulltextový vyhledávač. Jednoduchost přináší především přehled a rychlost. Editor je navíc multiplatformní [43].

I Visual Studio Code disponuje velkým množstvím zásuvných modulů [43], nicméně při používání se ukázaly limity ‚pouze editoru', kdy pro některé jazyky nelze vytvořit plnohodnotný projekt.

Pro jazyk Python však žádný projekt ani není potřeba. Stačí vytvořit složku, sepsat zdrojové soubory, z terminálu připravit virtuální prostředí Pythonu (Visual Studio Code jej ihned nabídne k použití, je-li umístěno v aktuální složce) a spustit. Vygenerují se dva soubory. První je soubor s nastavením, který nese například cestu k virtuálnímu prostředí. Druhý obsahuje instrukce pro spuštění. V něm je třeba nastavit především správnou cestu k spouštěnému souboru a dalším spuštěním se již program rozběhne.

Visual Studio Code je zdarma pro osobní i komerční použití [44]. Jeho kód je otevřený pod MIT licencí [44]. V rámci diplomové práce byla použita varianta VSCodium, což je komunitní projekt kompilace zdrojových kódů s vypnutou telemetrií a která představuje jednoduchý způsob, jak na linuxových systémech získat pravidelně aktualizovanou aplikaci [46].

7 Analýza poskytnutých dat

7.1 Struktura

O způsobu sběru dat již bylo pojednáno v samostatné kapitole. Výstupem jsou soubory ve formátu CSV. Každý soubor obsahuje časový průběh jednoho bodu. První řádek je hlavička, každý další odpovídá jednomu časovému kroku. V prvním sloupci je zaznamenán čas každého kroku a následně souřadnice jednoho bodu v pořadí x, y, z. V níže uvedené tabulce je uveden velmi krátký vzorek. Mnohé sekvence obsahují stovky kroků.

t	x	y	z
26.2279075	-191.528216004372	316.970944404602	721.873581409454
26.2937294	-198.443159461021	340.512573719025	692.972719669342
26.3861127	-204.133495688438	370.575040578842	683.418393135071
26.4569661	-202.986881136894	407.209008932114	684.837579727173
26.521348	-192.023485898972	406.929761171341	669.333159923553

Tabulka 2: Data bodu, zdroj: zdrojová data

7.2 Metadata

Metadata jsou zakódována v názvu souboru. Ten má formát „<identifikátor pacienta> <datum pořízení záznamu> <identifikátor bodu> <hodnocení pacienta> <předoperační hodnocení pacienta> <identifikátor cviku>_<časové razítko>.dat.csv“. Například výše uvedená data jsou ze souboru „00000000000 2019-05-01 ForeheadCenter eval=1 bf=0 01_oboci_2019-05-01-17-07-25.dat.csv“, tedy jde o polohu středu čela, pořízenou v 17:07:25 dne 1. 5. 2019, u pacienta číslo 0, který byl ohodnocen známkou 1 a předoperační hodnocení měl 0.

7.3 Body, cviky, pacienti

V kapitole o sběru dat bylo rovněž zmíněno, že takových bodů je z pacientova obličeje snímáno jedenadvacet. Vzhledem k tomu, že v každém souboru je právě jeden bod, na jediný cvik je zapotřebí dvacet jeden soubor. Tyto soubory mají zřejmě shodná jména, s výjimkou identifikátoru bodu (viz výše) a shodnou délku, neboť mají všechny původ v tom samém záznamu.

Jak již bylo stanoveno, každý pacient je hodnocen na základě devíti cviků. Pro každou kombinaci identifikátoru pacienta a data pořízení (jelikož kontroly jsou prováděny v pevných odstupech) by tak teoreticky mělo být k dispozici devět sad po jedenadvaceti souborech. Níže se však ukáže, že praxe je poněkud odlišná.

7.4 Kontroly

Více kontrol jednoho pacienta nemá smysl brát v potaz. Z hlediska načtení dat je zapotřebí identifikovat soubory, které přísluší jedné kontrole. Z hlediska učení a predikce je zase důležitá vazba mezi záznamem a výsledným hodnocením. Změny mezi jednotlivými kontrolami u konkrétního pacienta v průběhu rehabilitace nemají pro predikci hodnocení žádnou hodnotu. Dokonce se lze domnívat, že hodnocení by měla být nezávislá, a tak ani nesmějí vycházet z hodnocení předchozího.

Instanci kontroly libovolného pacienta tak lze zobecnit na hodnocení pacienta a o souboru jednotlivých hodnocení pacientů uvažovat jako o souboru pacientů. Tato abstrakce sice znemožní extrahovat z dat detaily spojené s vývojem pacienta, ale nabízí reprezentaci kontrol coby seznam nezávislých objektů pacientů, z nichž každý nese časosběrná data cviků a výsledné hodnocení, které je výhradně funkcí časosběrných dat. Tedy vhodný vstup pro chybové učení.

Zaveďme si nyní definice:

1) Cvik: Nechť soubor $S_{a,b,c,d}$ je soubor formátu CSV ze vstupních dat, kde a je identifikátor bodu, b je identifikátor pacienta, c je identifikátor cviku a d je datum pořízení. Bodem $B_{a,b,c,d}$ se rozumí časový průběh hodnot ve třech osách, extrahovaný ze souboru $S_{a,b,c,d}$. Cvik $C_{b,c,d}$: $\forall a$ pak budiž struktura, uchovávající všechny body, náležící stejné kombinaci identifikátoru pacienta b, identifikátoru cviku c a data pořízení d.
2) Pacient: Nechť cvik $C_{a,b}$ je cvikem dle předchozí definice, kde a je identifikátor pacienta a b je datum pořízení. Pak pacientem $P_{a,b}$ se rozumí struktura, která zahrnuje všechny cviky $C_{a,b}$, pro a, b.
3) Hodnocení pacienta: Hodnocení (někdy též ohodnocení) pacienta $H \in \langle 0,9 \rangle$ je celé číslo, kvantifikující poškození pacientova pohybového aparátu. Zdravému pacientovi přísluší hodnocení 0, maximálnímu poškození odpovídá číslo 9.

7.5 Neúplní pacienti

Pro orientaci, zdrojová data v době tvorby práce čítají 17220 souborů CSV. Jednoduchým výpočtem lze určit očekávaný počet pacientů (tak, jak byli právě definováni).

$$\frac{\frac{17220}{21}}{9} = 91,\bar{1}$$

Z výsledku je ovšem patrné, že data nejsou zcela konzistentní. V modelovaném systému je přípustný pouze celočíselný počet pacientů. Nelze klasifikovat pacientovu netriviální podmnožinu.

7.6 Úplní pacienti

Později, v oddílu věnovaném implementaci, bude tento problém samozřejmě řešen. V rámci řešení budou odhalena některá fakta, které jsou relevantní již pro analýzu dat. Ukáže se, že kompletních pacientů (dle definice výše) je k dispozici 94. To je více, než výše stanovený počet pacientů a tím i více, než počet možných úplných pacientů (pacientů se všemi cviky). Po nápravě zbude pacientů (již s jistotou úplných) 89, což není o moc méně, nicméně vzhledem k rozměrům a předpokládané variabilitě vstupních dat je zde na místě podezření, že vstupů je k natrénování umělé neuronové sítě nedostatečný počet.

7.7 Hodnocení pacientů

Z hlediska učení je také důležité, aby umělé neuronové síti v rámci trénování byly pokud možno předkládány příklady z celého oboru hodnot. (Ideálně i z celého definičního oboru, ovšem to je vzhledem k počtu pacientů nemožné zcela zřejmě.) V tomto případě, s klasifikací na deset tříd, to znamená požadavek na pacienty s každým možným výsledným ohodnocením, tedy 0 až 9.

Hodnocení zbylých devětaosmdesáti pacientů má následující četnosti:

Na zbylých devětaosmdesáti pacientech je ve skutečnosti následující rozdělení hodnocení:

Známka	0	1	2	3	4	5	6	7	8	9
Četnost před operací	56	33	0	0	0	0	0	0	0	0
Četnost po operaci	0	40	19	12	3	2	13	0	0	0

Tabulka 3: četnost jednotlivých hodnocení ve vstupních datech, zdroj: vstupní data

Předoperační hodnocení je zde jen pro zajímavost, důležitá jsou hodnocení pooperační. Z tabulky je vidět, že téměř polovina pacientů má hodnocení 1, zatímco hodnocení 0, 7, 8 a 9 nemá naopak nikdo. Je zřejmé, že požadavek na pokrytí celého oboru hodnot není splněn. Nicméně, vzorky jsou použitelné pro návrh a implementaci umělé neuronové sítě, určené k jejich zpracování.

8 Návrh a implementace

8.1 Přehled

Návrh a implementaci lze rozdělit do několika částí. Nejdříve se podíváme na samotné načtení vstupních dat a jejich reprezentaci v rámci programu. Smyslem načtení dat je jejich extrakce ze souborů, a zpracování do formátu, ve kterém s nimi lze dále manipulovat. Vzhledem k formátu vstupu se nejedná o elementární problém. Je třeba navrhnout odpovídající struktury k uchování dat a operace, které tyto struktury spolehlivě naplní a také zkontrolují. Jak již bylo nastíněno v předchozí kapitole, neúplné vzorky pacientů nelze bezpečně klasifikovat, nemá tedy smysl je uchovávat a budou odebrány.

S načtenými daty vykonáme ještě drobnou odbočku. Pro představu o tom, co vlastně hodláme zpracovávat, bude implementována též jednoduchá vizualizace. Výsledkem by měla být animace jednotlivých cviků ze zdrojových dat. Vizualizace rozhodně nemá za cíl jakkoliv se participovat na predikci. Jedná se pouze o nástroj, kterým bude možné znázornit hodnocené pohyby.

Data budeme samozřejmě chtít poskytnout umělé neuronové síti. To však nelze provést přímo. Nejprve musí být transformovány do formátu, ve kterém je umělá neuronová síť frameworku Keras dokáže zpracovat. Kromě těchto nutných transformací je také na místě zvážit optimalizační změny, neboť s malou trénovací množinou, kterou zde disponujeme, je každé potenciální vylepšení vhodné. S transformacemi dat souvisí také definice vstupní vrstvy sítě, pro kterou se ostatně transformace realizují.

Následně již lze přistoupit k samotné klasifikační síti. Zde je třeba učinit zásadní rozhodnutí, a sice která topologie bude použita. Kromě topologie vrstev je důležitý i počet neuronů ve vrstvách a počet vrstev. Je také nutné zvolit vhodnou výstupní vrstvu. Všeobecně ideální počty a ostatní vlastnosti neexistují. Liší se v závislosti na problému, který umělá neuronová síť řeší. Snaha o optimalizaci je předmětem experimentů. Případně, s dostatkem prostředků, lze použít například evoluční algoritmy.

Nakonec bude celá aplikace obalena uživatelským rozhraním. Dostupné bude základní textové rozhraní pro trénování a predikci z příkazové řádky a grafické uživatelské rozhraní, které výčet rozšíří o vizualizaci.

Jednotlivé třídy budou představovány vždy pro právě probíranou část a některé jejich atributy a metody v dané části ještě nemají opodstatnění, proto mohou působit matoucně. Proto jsou

v každé části uvedeny pouze relevantní vlastnosti daných tříd a ty, které v tomto výčtu chybí, jsou později zmíněny coby jejich rozšíření. Do značné míry tak výklad tříd koresponduje s jejich vývojem. Průběžně jsou přiloženy UML diagramy, které usnadní pochopení a řídí se stejnou zásadou.

8.2 Načtení dat

Prvním krokem je samozřejmě načtení pacientů ze zdrojových souborů. K tomu je potřeba definovat vhodné třídy a metody.

8.2.1 Cvik

Třída Exercise (v souboru exercise.py) představuje jeden cvik. Uchovává jméno cviku, jeho průběh a počet řádků, což je v podstatě délka záznamu v časových krocích. Průběh má formát dvourozměrného pole o šířce trojnásobku počtu bodů, navýšené o jednu. Na poskytnutých datech je výsledná šířka 64: jedenadvacet bodů, násobených třemi dimenzemi a první sloupec je vyhrazen pro časové razítko kroku. Délka průběhu odpovídá počtu kroků, a tedy i počtu řádků vstupního souboru, resp. počtu řádků poníženému o jednu, pokud soubory obsahují hlavičku.

Z hlediska načítání je stěžejní metoda pro přidání bodu, add_point, která očekává v parametrech jméno bodu a průběh jeho polohy. Pokud se jedná o první výskyt cviku, je nastavena délka, pole inicializováno nulami a první sloupec naplněn časovými razítky. Následně je jméno bodu (platí i pro každý další bod) přeloženo na číselný index a podle něj je celý průběh bodu zařazen do průběhu cviku.

8.2.2 Pacient

Pacient je reprezentován třídou Patient (patient.py). Ta nese pacientův identifikátor, datum kontroly, předoperační i pooperační hodnocení a samozřejmě také seznam cviků. (V Pythonu je výchozí strukturou pro ukládání většího počtu hodnot dynamický seznam, oproti obvyklejšímu statickému poli v jiných jazycích [45]. Polem je v tomto textu myšlena struktura z balíčku NumPy.)

Je zde, pochopitelně, metoda pro přidání bodu, add_point_to_an_exercise, která tuto úlohu deleguje na odpovídající cvik. Důležitá je také metoda get_exercises. Ta interní seznam cviků transformuje v jediné pole tím, že je všechny poskládá za sebe. Výsledné pole nezahrnuje časová razítka a má tak šířku <počet bodů>*3*<počet cviků>, což v případě poskytnutých dat vychází na 567 sloupců.

8.2.3 Pacienti

Seznam pacientů zastřešuje pomocná třída Patients (loader.py). Stejně, jako je ve třídě Patient metoda, která deleguje přidání bodu na Exercise, je zde metoda add_point, která potenciálně přidá pacienta (jedná-li se o jeho první výskyt) a deleguje přidání bodu na něj. Dále poskytuje informaci o počtu pacientů a manipulační metody, tedy přidání pacienta (add), seznam pacientů (to_array), pacienta dle indexu(get) nebo vyhledání pacienta dle identifikátoru(find).

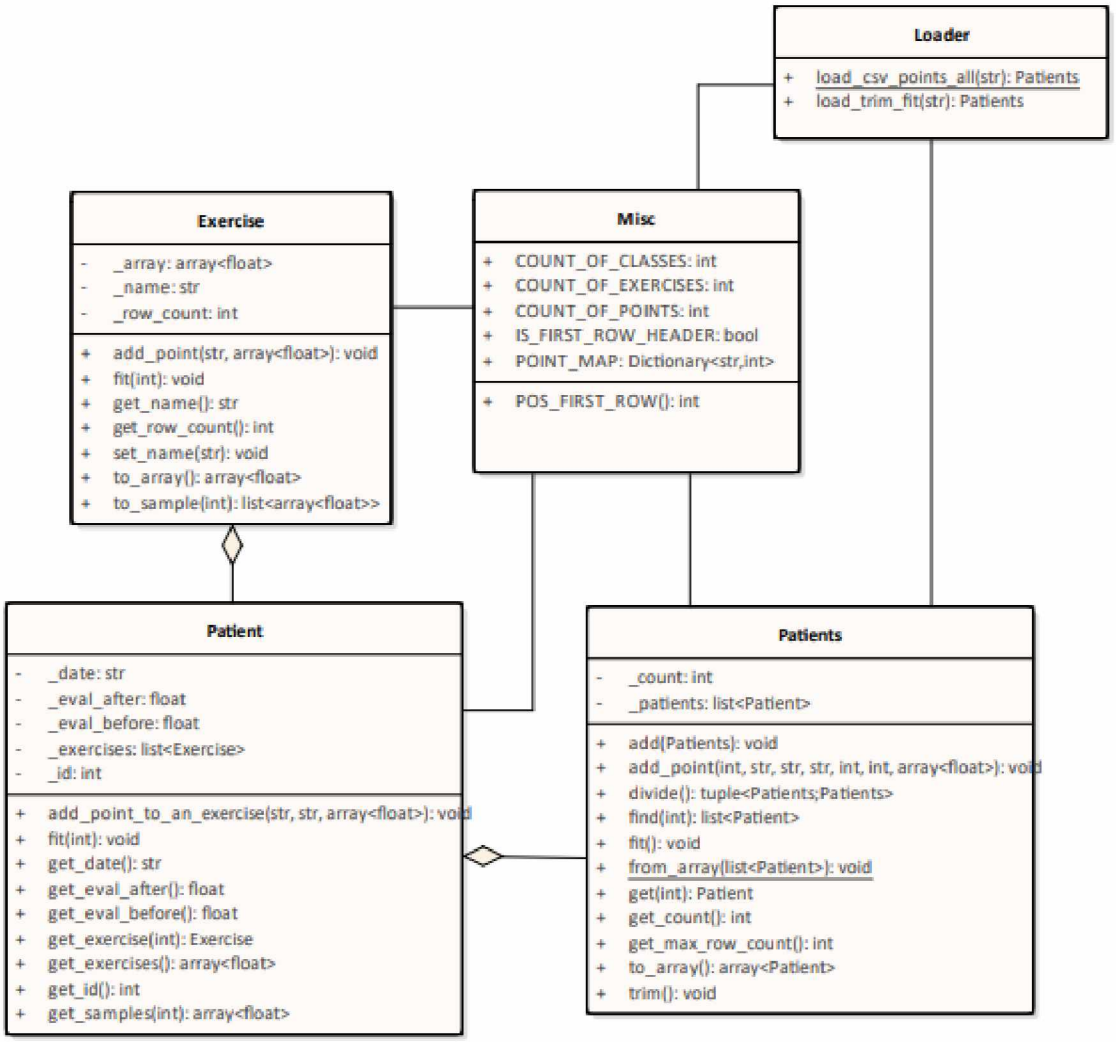
8.2.4 Načítání

Celé načítání má na starost třída Loader (loader.py), která prostřednictvím statické metody load_cs_points_all iteruje přes soubory ve složce, předané parametrem. Z názvu každého souboru získá metadata a z jeho obsahu data bodu. Správné rozřazení dat už deleguje na instanci třídy Patients.

Někteří načtení pacienti však neobsahují všechny cviky. Takového pacienta nelze vyhodnotit. Samozřejmě se můžeme pokusit o hodnocení samostatných cviků, které by měly odpovídat výsledné známce, ale od sítě se očekává klasifikace celého pacienta. To by vyžadovalo podvržení cviků například nulami, čímž bychom ovšem síť učili hodnotit i takový vstup nesouvisející známkou. Vhodnější je takového pacienta zcela vyřadit. O vyřazení nekompletních pacientů se stará metoda trim třídy Patients.

8.2.5 Konstanty

V předchozích odstavcích lze postřehnout větší množství potenciálně měnitelných vlastností, které jsou v kódu zachyceny v podobě třídy Misc (constants.py), Ta obsahuje konstanty jako počet cviků, počet bodů, počet výsledných známek, mapu jmen cviků na čísla či mapu jmen bodů na čísla, informace o tom, zda je v souborech hlavička sloupců a podobně. Hodnoty z této třídy jsou používány ve zbytku programu, což umožňuje rychlé změny globálního charakteru, například v případě nového cviku, nebo rozhodnutí o snímaní více bodů. Je však potřeba si uvědomit, že při takové změně bude nutné provést nové trénování umělé neuronové sítě, a to výhradně na datech, která odpovídají novému formátu.

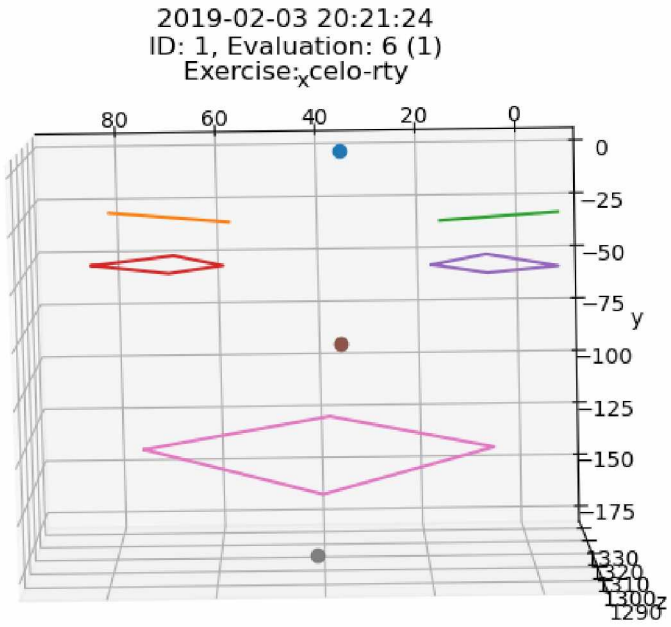


Obrázek 11: Diagram tříd, načtení dat, zdroj: vlastní

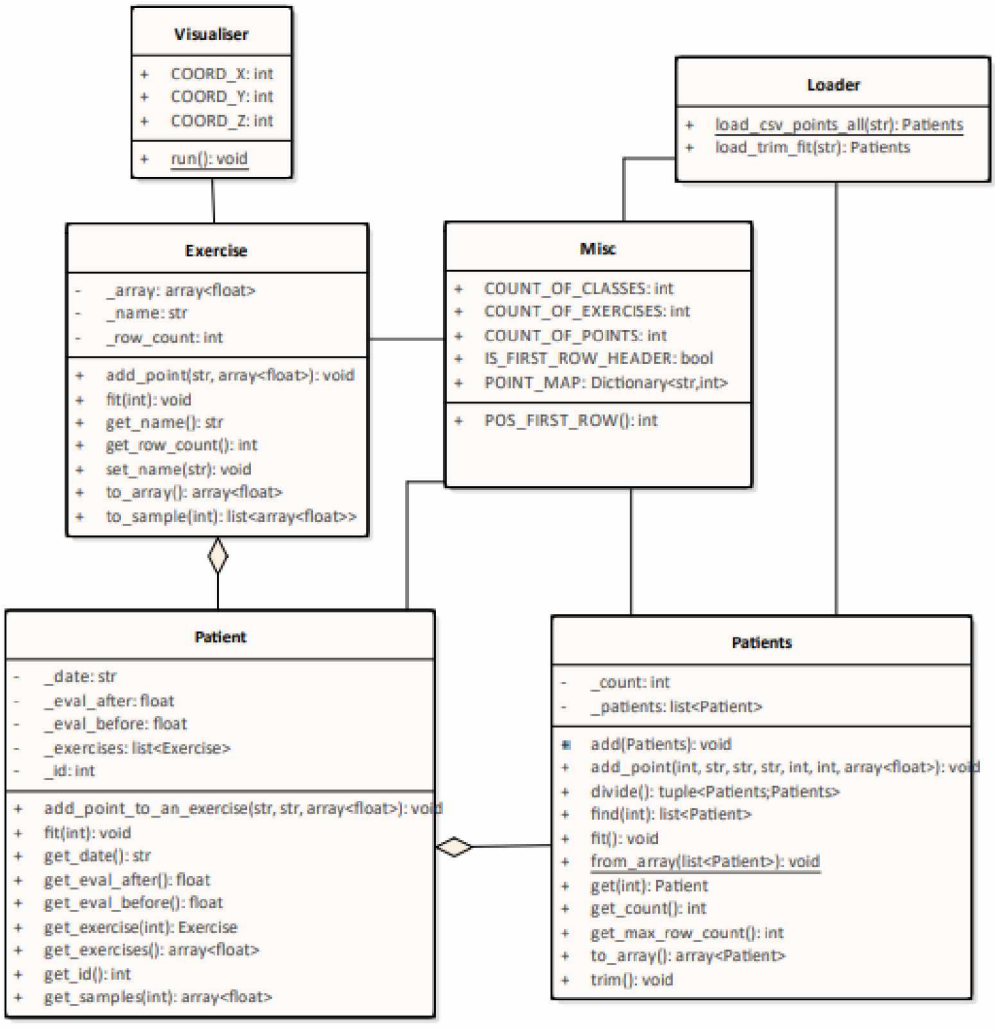
8.3 Vizualizace dat

Nyní již nic nestojí v cestě vizualizaci cviků. Cílem v rámci této práce rozhodně není věrná reprezentace, ta by mohla být předmětem samostatného projektu. Zde postačí, když bude zřejmé, o jaký obličejový cvik se jedná. Bodů je relativně málo a nabízí se jejich proložení křivkami.

K tomuto účelu vznikla třída Visualiser, která disponuje jedinou metodou – run. Ta pro cvik, předaný parametrem, spustí animaci pomocí knihovny Matplotlib.



Obrázek 12: Jednoduchá vizualizace proložením úseček body, zdroj: vlastní



Obrázek 13: Diagram tříd, vizualizace dat, zdroj: vlastní

8.4 Transformace dat

8.4.1 Požadovaný formát

Data jsou již dispozici ve formátu, ve kterém jsou srozumitelná člověku a snadno manipulovatelná. Nicméně síť frameworku Keras má jiné požadavky. Pokud má jediný vstup, vyžaduje tenzor z knihovny Tensorflow, nebo pole z knihovny NumPy, případně strukturu poli podobnou („array-like“) [47]. V případě více vstupů očekává seznam těchto tenzorů, či polí. Každá položka seznamu pak představuje celou vstupní množinu nebo dávku generátoru jednoho vstupního filtru.

Vstupem do transformace je seznam pacientů (třída Patients), kterou lze konvertovat na pole tříd Patient. Třída Patient nám také zpřístupňuje všechny cviky ve formě jediného pole. Problém je, že každý cvik má jinou délku. To nekoresponduje s fixními rozměry pole. Knihova NumPy si s tímto umí poradit, vstup konvertuje, ovšem jednotlivé záznamy uchová jako instance typu Object, nikoli číselná pole. Typicky se tento přístup může hodit – vzniklé pole obsahuje všechny hodnoty vstupu a dokonce je nadále indexovatelné na všech dimenzích, ovšem pro síť frameworku Keras je takový vstup netransparentní, bere jej jako jednodimenzionální a vyvolá výjimku.

8.4.2 Sjednocení rozměrů

Je nutná unifikace rozměrů jednotlivých cviků, aby bylo možné pacienty zpracovat do akceptovatelné formy. K tomu disponuje třída Patients metodou get_max_row_count, vracející délku nejdelšího cviku za všechny pacienty, a metodou fit, která (delegací na třídu Patient a ta delegací na třídu Exercise) zajistí samotnou změnu délky pro každý cvik. Chybějící hodnoty jsou doplněny nulami shora. Časové razítko je vypuštěno, lze předpokládat konstantní vzorkovou frekvenci snímače a délku počítat v krocích. Pro ilustraci je níže uvedený cvik, původně délky 5, doplněný na délku 7; zobrazeny jsou jen první tři sloupce, tedy souřadnice prvního bodu.

0	0	0	...
0	0	0	...
-191.528216004372	316.970944404602	721.873581409454	...
-198.443159461021	340.512573719025	692.972719669342	...
-204.133495688438	370.575040578842	683.418393135071	...
-202.986881136894	407.209008932114	684.837579727173	...
-192.023485898972	406.929761171341	669.333159923553	...

Tabulka 4: Cvik po změně délky, zdroj: vlastní

Upravené cviky lze seřadit do jediného číselného pole, jehož struktura je srozumitelná pro umělou neuronovou síť. V rámci transformace vstupních dat již zbývá jen optimalizace.

8.4.3 Normalizace hodnot

První optimalizace cílí na problém, který je dobře patrný ve výše uvedené Tabulce 3. Každou souřadnici bodu v čase lze chápat jako součet počáteční hodnoty a přírůstku. O tvaru a pohyblivosti obličeje přitom vypovídají pouze přírůstky. Patrně nechceme, aby se hodnocení pacienta odvíjelo od jeho vzdálenosti od senzoru nebo od jeho výšky a podobně. Přírůstky k počáteční hodnotě lze vypočítat odečtením původní hodnoty. Tato úprava byla přidána do výše představené metody fit. Pro ilustraci je níže opět uveden cvik po aplikaci zmíněné úpravy. Vychází z ilustrační Tabulky 3, i zde je zobrazen pouze první bod z celé kolekce bodů ve cviku.

0	0	0	...
0	0	0	...
0	0	0	...
-6,914943456649	23,541629314423	-28,900861740112	...
-12,605279684066	53,604096174240	-38,455188274383	...
-11,458665132522	90,238064527512	-37,036001682281	...
-0,495269894600	89,958816766739	-52,540421485901	...

Tabulka 5: Cvik tvořený přírůstky, zdroj: vlastní

8.4.4 Rozdělení množin

Další optimalizace vychází z rozdělení ohodnocení pacientů, které již bylo zmíněno v rámci oddílu analýzy dat. Ohodnocení pacientů totiž není rovnoměrné a některé vzorky dokonce zcela chybí. Pro připomenutí:

Ohodnocení pacienta	Počet výskytů
0	0
1	40
2	19
3	12
4	3
5	2
6	13
7	0
8	0
9	0
10	0

Tabulka 6: Rozdělení ohodnocení pacientů, zdroj: vlastní

V souvislosti s tímto neduhem je třeba si uvědomit, že později, v oddílu testování, budeme síť trénovat na vstupních datech. Kvalitu natrénování budeme chtít také zkontrolovat,

kvantifikovat. Pro tuto kontrolu je zapotřebí validační množina, kterou bude tvořit zhruba čtvrtina vstupních dat.

Pokud bychom množinu pacientů rozdělili indexem nebo zcela náhodně, riskujeme, že některé méně četné třídy v trénovací množině vůbec nebudou a síť tudíž nedostane možnost se je naučit. Proto je třída Patients rozšířena o metodu divide, která z původní instance vygeneruje dvě nové, trénovací a validační. V trénovací instanci je vždy alespoň 75 % vzorků každé třídy a ve validační všechny zbylé.

8.4.5 Vzorkovací frekvence

Budeme-li vnímat vstupní data jako kolekci signálů, nabízí se ještě další optimalizace. Na vstupních signálech se totiž mohou vyskytovat nežádoucí odchylky, případně prostě jen nedůležité odchylky, které by mohly výsledek zkreslovat. Snížením vzorkovací frekvence vliv malých výkyvů potlačíme a na průbězích signálů budou patrné jen dlouhodobé trendy [48].

O snížení vzorkovací frekvence cviku se stará jeho metoda to_sample a třída Patient se rozrostla o metodu get_samples. Analogicky k řazení původních cviků do jediného pole v metodě get_exercises, metoda get_samples vytvoří jediné pole všech cviků se sníženou vzorkovací frekvencí.

Stanovit obecně vhodnou frekvenci samozřejmě není možné a nelze riskovat, že jejím snížením naopak ztratíme důležité detaily. Řešením je zachování originálu a jeho vyhodnocování paralelně s upravenou verzí. Neuronová síť v tomto případě bude zpracovávat několik datových sad. Pro účely práce bylo zvoleno poloviční a pětinové vzorkování, což spolu s originálem znamená tři sady vstupů a tím i tři vstupní filtry umělé neuronové sítě. Níže je ukázána Tabulka 4 po transformaci polovičním vzorkováním.

0	0	0	...
0	0	0	...
-12,605279684066	53,604096174240	-38,455188274383	...
-0,495269894600	89,958816766739	-52,540421485901	...

Tabulka 7: Data s polovičním vzorkováním, zdroj: vlastní

8.4.6 Generátor

Poslední optimalizace je čistě praktická. Knihovna Keras si poradí s učením z pole a můžeme tedy umělé neuronové síti předat k učení jednorázově celou trénovací množinu. V případě poskytnutých dat to rozhodně nepředstavuje problém, ale je očekávatelné, že časem bude pacientů přibývat. Složitější umělé neuronové sítě mohou být paměťově naročné a v kombinaci

s velkým množstvím trénovacích dat by mohlo dojít k vyčerpání vyhrazené paměti. Tento problém lze redukovat rozdělením vstupních dat do dávek.

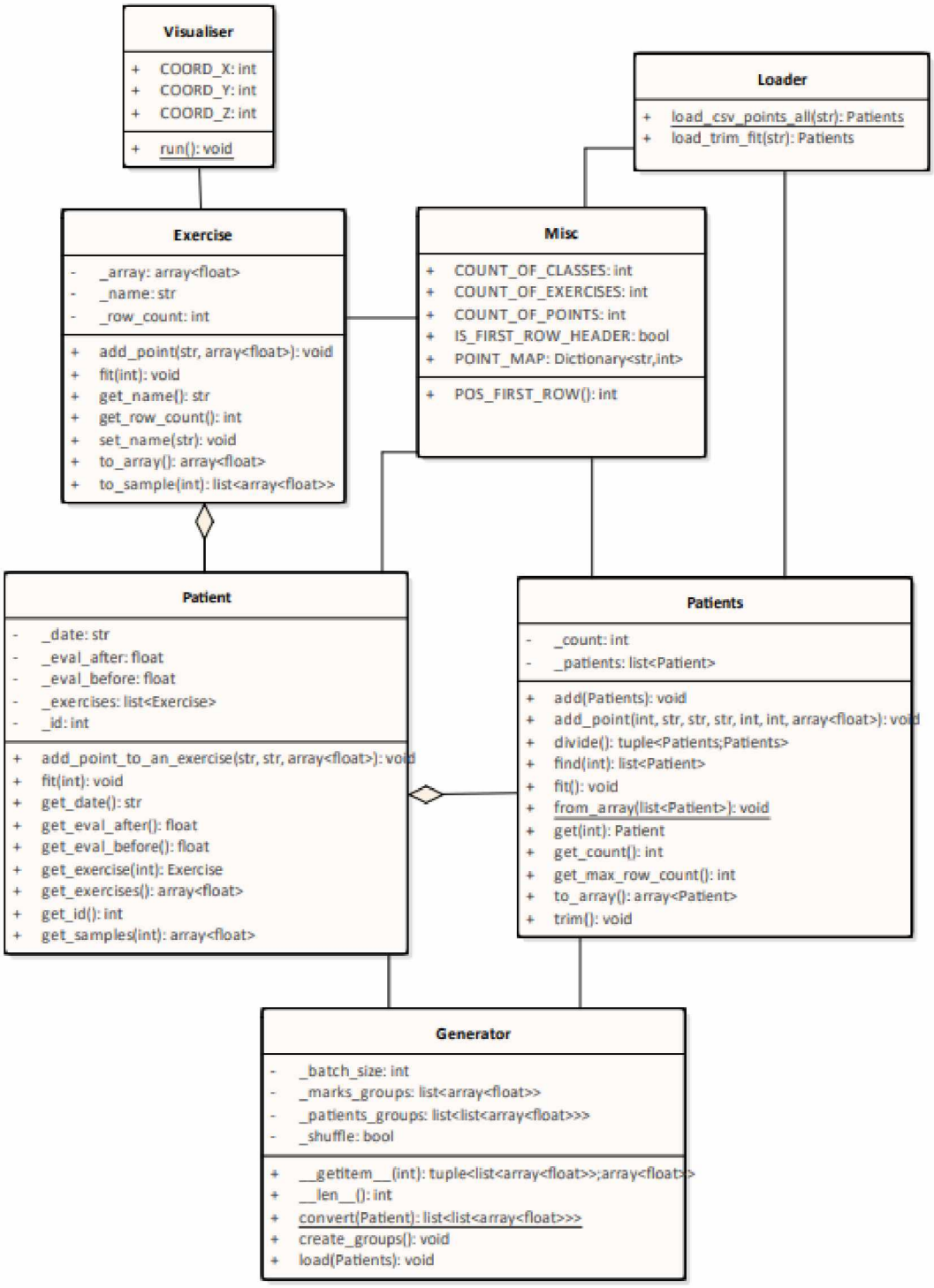
Místo vstupního pole bude umělé neuronové síti předán generátor, který poskytuje pacienty v dávkách o předem určené velikosti. V programu je reprezentován třídou Generator (generator.py). Třída Generator je potomkem třídy tensorflow.keras.utils.Sequence, což je nutné, aby byla akceptovatelným vstupem. Další nutnou podmínkou je definice metod __len__, vracející počet dávek a __getitem__, která poskytuje dávku dle indexu v parametru. Umělá neuronová síť, které je generátor předán, pomocí těchto dvou metod iteruje přes dávky.

Vzhledem k aplikovaným optimalizacím má generátor na starost více než jen rozdělení vstupů na několik skupin. V parametrech očekává pacienty formou třídy Patients, velikost dávky a informaci, zda mají být pacienti zpřeházeni. Nejprve si ze třídy Patients vyžádá pole pacientů, které zpřehází, bylo-li to požadováno. Následně přes jednotlivé pacienty iteruje a vytváří celkem čtyři seznamy. V prvním seznamu jsou pacienti s pětinovým vzorkováním, ve druhém s polovičním vzorkováním a ve třetím v původním tvaru. Čtvrtý seznam je vyhrazen pro výsledná hodnocení.

Formát hodnocení musí odpovídat očekávanému výstupu z umělé neuronové sítě, aby mohl být použit k učení. Jak již bylo zmíněno, výstup sítě je daný výstupní vrstvou. V tomto konkrétním případě bude výstupem pole příslušností k jednotlivým třídám, blíže je výstup vysvětlen v následujícím oddílu. Ohodnocení známkou x je vyjádřeno polem, ve kterém je na x-tém indexu 1 a všude jinde 0. Např. známka 3 je na výstupu pole [0, 0, 0, 1, 0, 0, 0, 0, 0, 0].

Následně jsou seznamy konvertovány na pole a tato pole uložena do dávek. Každá dávka obsahuje dvojici (tuple), kde prvním prvkem je trojice vstupních polí a druhým pole výsledných hodnocení.

Kromě metod generátoru obsahuje třída ještě statickou metodu convert, která samostatného pacienta konvertuje do stejného tvaru, s výjimkou hodnocení. Slouží pro přípravu vstupu k predikci, protože i ta nyní vyžaduje všechna tři pole.



Obrázek 14: Diagram tříd, transformace dat, zdroj: vlastní

8.5 Umělá neuronová síť

8.5.1 Topologie

S připraveným generátorem je na čase přejít k samotné síti. V teoretické části bylo představeno několik obvyklých topologií. Asi nejnadějnější se jeví konvoluční neuronová síť, která je

oblíbená zejména ve zpracování obrazu, ale používá se i pro zpracování časových řad. Její hlavní výhodou je schopnost učení výrazných rysů, což dává naději pro rozpoznání anomálií v datech. Vhodnou alternativou je také dopředná vícevrstvá síť, která je jednoduchá a nabízí obecně kvalitní rozpoznávání vazeb, ovšem ignoruje souvislost mezi jednotlivými časovými kroky.

Primárně bude implementována konvoluční neuronová síť, nicméně paralelně k ní i dopředná vícevrstvá, která bude sloužit pro srovnání. Obě sítě ostatně budou mít mnoho společného. Topologie se liší zejména skrytými vrstvami. Vstupní vrstva nutně odpovídá rozměrům vstupních dat, zatímco výstupní vrstva závisí na očekávaném výstupu.

8.5.2 Vstupní vrstva

Vstupní vrstva zahrnuje tři vstupní body. Každý vstupní bod je v podstatě samostatná umělá neuronová síť. I tato „vstupní" síť má svou vstupní vrstvu, jejíž rozměry odpovídají vstupním polím. Liší se tedy délkou, anžto délka se odvíjí od vzorkovací frekvence. Po vstupní vrstvě následuje konvoluce. Je zvolena jednodimenzionální konvoluce, neboť v tomto případě není žádoucí, aby se filtry „posunovaly" v rámci jednoho kroku. Tomu odpovídá i šířka, stejná jako šířka vstupní vrstvy. Konvoluci si tak lze představit jako filtr, s šířkou celého kroku a definovanou délkou, který klouže po krocích shora dolů. U konvoluční vrstvy je třeba definovat několik volitelných parametrů, a sice:

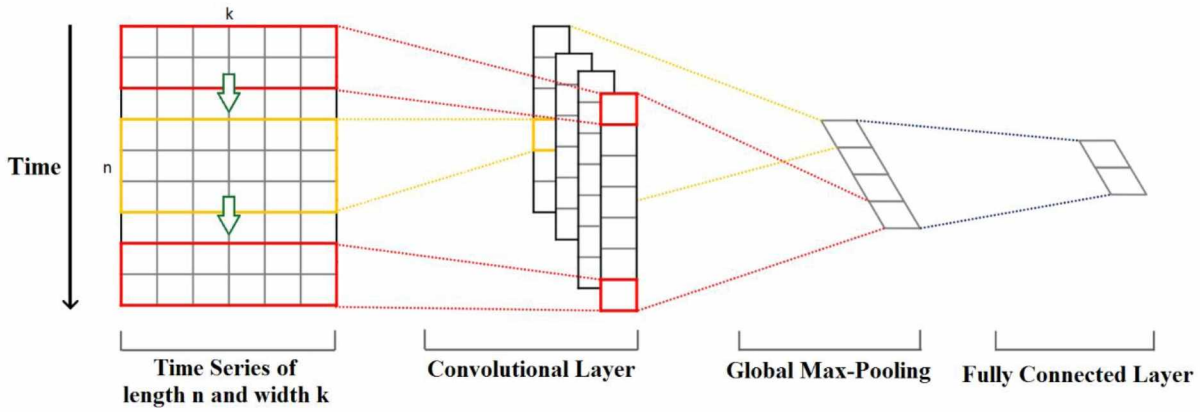
- Počet filtrů – počet filtrů, které provádí konvoluci. Dá se říci, že čím více filtrů, tím více rysů se síť dokáže naučit. Správný počet bude předmětem pokusů.
- Délka filtru – určuje v tomto konkrétním případě počet kroků, které se vejdou do filtru. Správná délka bude předmětem pokusů.
- Vycpávka (padding) – nastaveno na „same". To znamená, že testovaná data mají přesah, doplněný nulami, díky čemuž se rozměr dat nesníží [49].
- Aktivační funkce – nastaveno na „tanh", tedy hyperbolický tangens. Ten je spojitý na intervalu (-1; 1), takže nebude třeba hodnoty normalizovat před předáním dalším vrstvám.

Další je vrstva poolingu. Zvolen je jednodimenzionální globální max pooling, který z každého filtru ponechá jen jediný, nejsilnější rys. Výstup je předán plně propojené vrstvě, která v něm hledá souvislosti. Parametry jsou:

- Počet neuronů – teoreticky, čím více neuronů ve vrstvě, tím přesnější predikce. Správný počet bude předmětem pokusů.

- Aktivační funkce – i zde použit hyperbolický tangens.

Následuje vrstva dropout, která zajistí, že náhodně vybrané neurony budou v jednotlivých fázích učení vynechány. Zabrání tomu, aby na sobě propojené neurony příliš závisely [50]. Pomáhá tak redukovat učení sítě „nazpaměť".



Obrázek 15: Jednodimenzionální konvoluce vstupních sítí, zdroj: [48]

8.5.3 Skrytá vrstva, sjednocení

Tímto způsobem jsou vystavěny tři sítě, které představují vstup hlavní sítě. Jejich výstupy se nejdříve spojí do jediného pole. To zajišťuje vrstva zřetězení (concatenate). Spojené výstupy jsou opět předány plně propojené vrstvě (se stejnými parametry jako ve vstupních sítích) a nakonec výstupní vrstvě. Ta je reprezentovaná další plně propojenou vrstvou.

8.5.4 Výstupní vrstva

Výstupní plně propojená vrstva má však již jiný formát. Musí odpovídat očekávanému vstupu. V případě hodnocení pacientů diskrétními známkami z rozmezí <0; 9>, které zde očekáváme, můžeme hovořit o třídní klasifikaci. Jednotlivými třídami se rozumí očekávané známky a pro každého pacienta bude umělá neuronová síť predikovat příslušnou třídu.

Pro třídní klasifikaci se obvykle používá aktivační funkce softmax [8]. Zjednodušeně řečeno, tato aktivační funkce spočítá příslušnost vstupu pro konkrétní třídu. Přitom příslušnost je vyjádřena racionálním číslem z rozmezí (0; 1) a suma všech příslušností se rovná jedné. [8] Jelikož vyžadujeme kvantifikaci příslušnosti pro každou třídu, počet výstupních neuronů musí odpovídat počtu tříd. V případě hodnocení pacientů jich bude deset.

8.5.5 Konfigurace

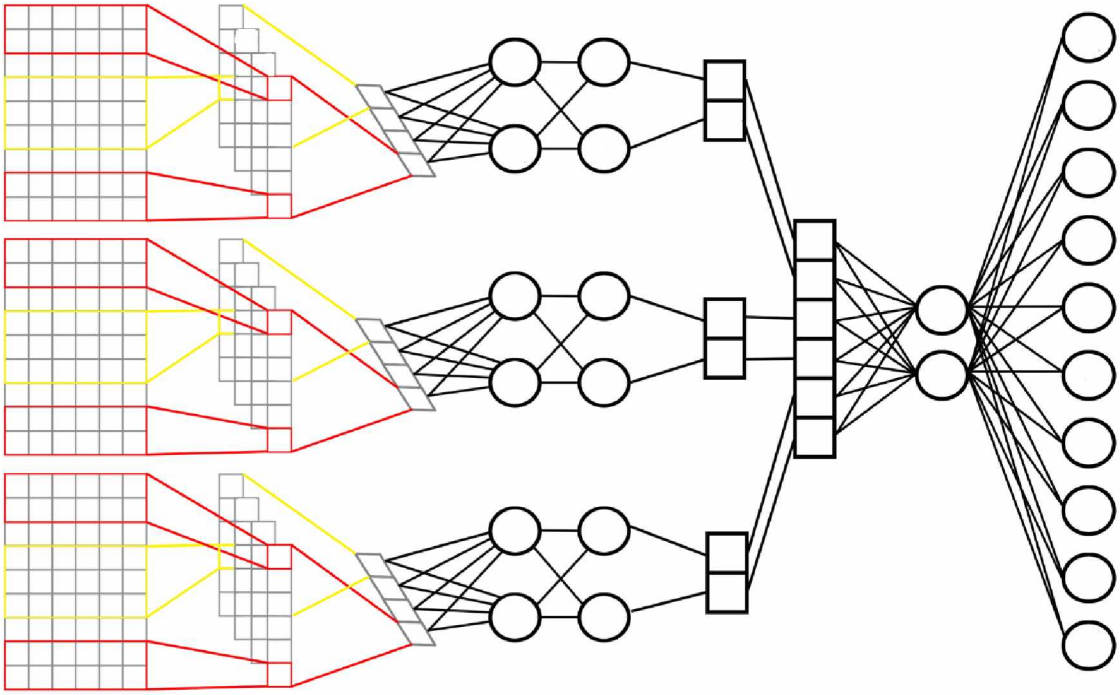
Nesmíme zapomenout na učení sítě, od výstupu lze totiž odvodit i vhodnou ztrátovou funkci. Pro podobné problémy se obvykle používá některá z cross-entropy funkcí. Funkce cross-entropy, níže značena E coby chybovost, se výrazně zvyšuje s odchylkou od správného určení a indikuje obzvláště vysokou ztrátu pro případy, kdy je špatná třída zvolena s vysokou příslušností [9]. Matematický předpis je následující:

$$E = -\sum_{c=1}^{m} y_{o,c} \log(p_{o,c}), \mathrm{m} \in \mathrm{N}, \mathrm{y} \in \{0;\ 1\}, \mathrm{p} \in < 0;\ 1 >$$

Kde *m* značí počet tříd, *y* je indikátor správnosti třídy a *p* je predikovaná příslušnost pozorování (vstupu) *o* k třídě *c* [9].

Jelikož v řešeném případě nepřipouštíme stav, kdy jeden pacient přísluší částečně více než jedné známce, je vhodná ztrátová funkce categorical cross-entropy, která při výpočtu ztráty zohledňuje předpoklad příslušnosti k právě jedné třídě. Tato funkce je parametrem kompilace sítě, při které je síť nakonfigurována a připravena k použití. Dalšími parametry je optimizátor (zvolen RMSprop) a metrika.

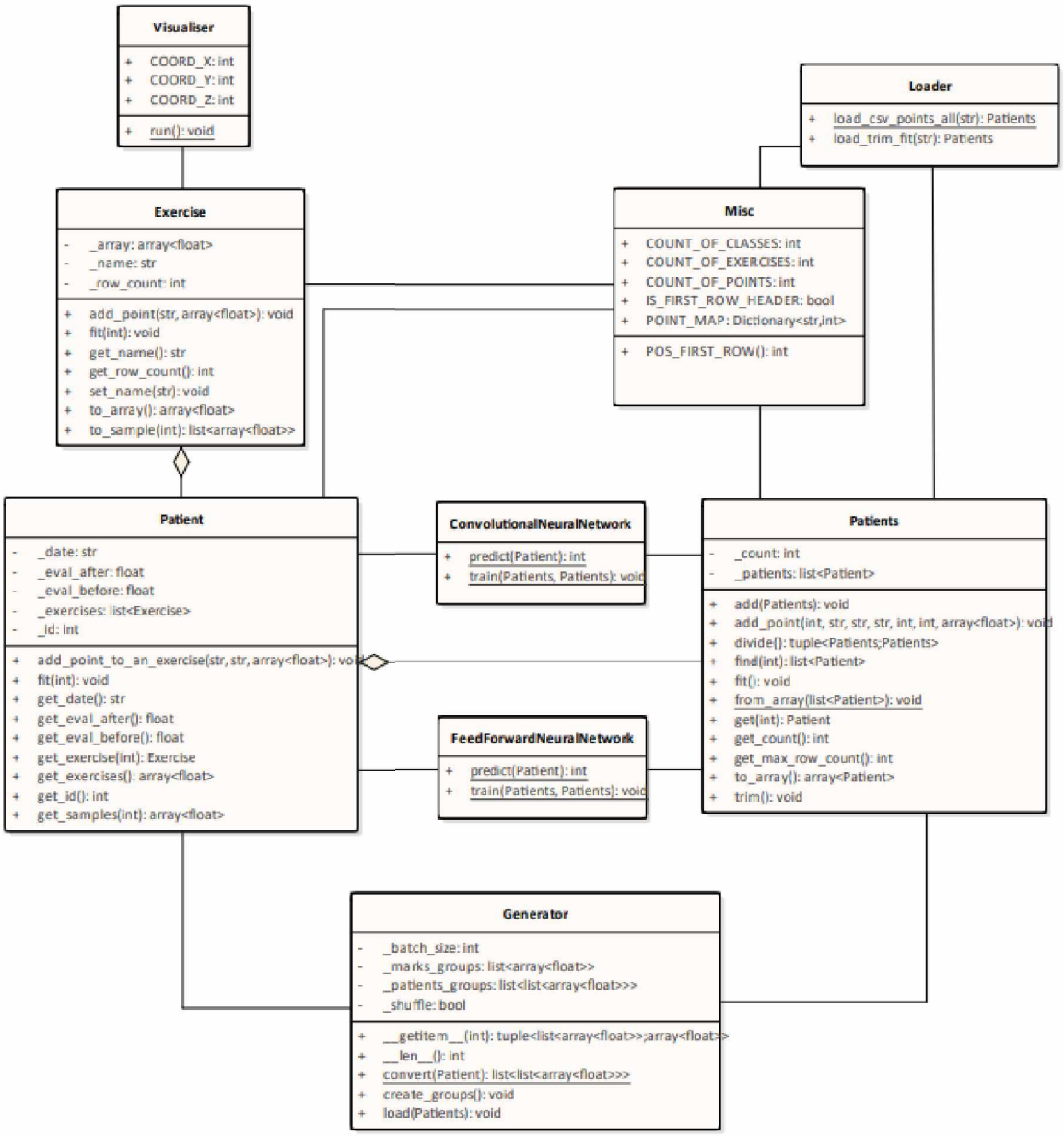
Zvolena je metrika „accuracy“, která kvantifikuje přesnost sítě číslem z rozsahu <0; 1>. Při učení sítě je po každé dávce vrácena ztráta a přesnost. Ztráta představuje výsledek ztrátové funkce a kvantifikuje odchylky sítě od správné klasifikace při tréninku. Přesnost vyjadřuje podíl správně klasifikovaných vzorků při validaci po každé epoše trénování, který je pro člověka poněkud srozumitelnější. Obecně, nízká ztráta vypovídá o kvalitní síti, zatímco vysoká přesnost o kvalitním natrénování [51]. Dosažení minima ztráty a maxima přesnosti bude předmětem pokusů.



Obrázek 16: Model výsledné síťě, zjednodušený, zdroj: [48], modifikovaný

8.5.6 Kontrolní síť

Dopředná vícevrstvá neuronová síť je implementována úplně stejně, pouze místo konvoluční a poolingové vrstvy je ve vstupních sítích použita vrstva plně propojená.



Obrázek 17: Diagram tříd, neuronové sítě, zdroj: vlastní

8.6 Uživatelské rozhraní

8.6.1 Textové uživatelské rozhraní

Aplikace je v první řadě experiment a ve skutečnosti ani nezahrnuje požadavek na líbivé grafické prostředí. Výchozím rozhraním je konzole. Konzolová aplikace (aidi_cli.py) je napsána přímo v pythonu a v podstatě jen deleguje práci.

8.6.2 Grafické uživatelské rozhraní

Přestože grafické uživatelské rozhraní aplikace přímo nevyžaduje, pro pohodlnou manipulaci je také dostupné (aidi_gui.py, aidi_gui.kv). Jak již bylo uvedeno v oddílu softwarových

nástrojů, k vyvoření je použita knihovna kivy. Celé grafické rozhraní je definováno ve dvou souborech.

V souboru s příponou KV, který musí mít stejné jméno, jako třída, až na postfix „App“, je popsána celá hierarchická struktura prvků rozhraní. V souboru s příponou PY je pak obslužný program, který tuto strukturu načte a spustí. Oficiální návrhář pro kivy je označen za zavržený a neudržovaný, neboť se jeho vývoj nevyplatí [21].

9 Testování

9.1 Cíle

Se sítí připravenou k použití se dostáváme do fáze testování. V tomto oddíle bude zdokumentována série testů, která má za cíl:

- dokázat, že navržená umělá neuronová síť je funkční;
- dokázat, že diagnostika navrženou umělou neuronovou sítí je možná;
- zjistit, zda je diagnostika v současnosti proveditelná;
- zvolit finální parametry sítě.

Síť je považována za funkční, pokud pro zadané vstupy poskytne výstupy a pokud je schopná tréninku pomocí připraveného generátoru.

Diagnostika je považována za možnou, pokud trénování sítě ovlivňuje klasifikaci a síť tedy v datech nachází souvislosti, které se učí.

Za v současnosti proveditelnou je diagnostika považována, pokud učení sítě produkuje zlepšení klasifikace do takové míry, že dokáže věrně predikovat ohodnocení validační množiny.

V této části také dojde k výběru parametrů sítě, jejichž hodnoty byly v rámci návrhu a implementace ponechány otevřené. Jedná se zejména o počet filtrů a neuronů skrytých vrstev a vůbec počet skrytých vrstev. Optimalizačním ukazatelem bude minimalizace ztráty a maximalizace přesnosti.

9.2 Metodika

Proběhnou testy umělé neuronové sítě a její případné úpravy v souladu s formulovanými cíli testování. Testy budou probíhat na předmětné konvoluční neuronové síti a paralelně též na kontrolní dopředné vícevrstvé neuronové síti.

Pro účely testování bude používán generátor v deterministickém režimu, tedy bez náhodného zpřeházení pacientů. Obě sítě budou trénovány a testovány během jednoho spuštění programu, na stejné načtené množině pacientů. Tím se zamezí i odchylkám v případě nedeterministického pořadí načítaných souborů, jejichž výběr je svěřen knihovně os.

Průběh každého dílčího testu je následující: ve třech pokusech proběhne pokaždé tisíc epoch učení neuronové sítě na trénovací množině a poté hodnocení na validační množině. Výsledky trénování i hodnocení jsou evidovány. Počet epoch se v závěru testování může změnit, v souladu s cílem optimalizace parametrů sítě.

Výstupem testování je série tabulek. Každá tabulka odpovídá jednomu nastavení parametrů umělých neuronových sítí. Jsou uvedeny výsledné ztráty a přesnosti na testovací a validační množině, pro jednotlivé pokusy a pro obě sítě.

9.3 Výsledky

9.3.1 1 vrstva, 10 filtrů, 10 neuronů, délka filtru 8, 16, 24

		Pokus	1	2	3
Dopředná vícevrstvá	Trénovací	Ztráta	0,0192	0,0000	0,0000
		Přesnost	0,9851	1	1
	Validační	Ztráta	4,3784	10,7683	7,1403
		Přesnost	0,2727	0,1818	0,3636
Konvoluční	Trénovací	Ztráta	0,1173	0,0753	0,1017
		Přesnost	0,9403	0,9701	0,9552
	Validační	Ztráta	3,2232	3,5301	4,2814
		Přesnost	0,3636	0,3182	0,3182

Tabulka 8: 1 vrstva, 10 filtrů, 10 neuronů, délka filtru 8, 16, 24, zdroj: vlastní

9.3.2 1 vrstva, 50 filtrů, 50 neuronů, délka filtru 8, 16, 24

		Pokus	1	2	3
Dopředná vícevrstvá	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	8,9976	6,3715	7,4063
		Přesnost	0,3182	0,2273	0,3182
Konvoluční	Trénovací	Ztráta	0	0,0188	0,0000
		Přesnost	1	0,9701	1
	Validační	Ztráta	11,1176	7,1512	13,5941
		Přesnost	0,3182	0,1818	0,2273

Tabulka 9: 1 vrstva, 50 filtrů, 50 neuronů, délka filtru 8, 16, 24, zdroj: vlastní

9.3.3 1 vrstva, 100 filtrů, 100 neuronů, délka filtru 8, 16, 24

		Pokus	1	2	3
Dopředná vícevrstvá	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	6,2946	5,7037	6,0173
		Přesnost	0,4091	0,2727	0,4091
Konvoluční	Trénovací	Ztráta	0	0,0000	0
		Přesnost	1	1	1
	Validační	Ztráta	13,3315	14,8280	12,3779
		Přesnost	0,2273	0,2273	0,2727

Tabulka 10: 1 vrstva, 100 filtrů, 100 neuronů, délka filtru 8, 16, 24, zdroj: vlastní

9.3.4 1 vrstva, 200 filtrů, 200 neuronů, délka filtru 8, 16, 24

		Pokus	1	2	3
Dopředná vícevrstvá	Trénovací	Ztráta	0	0,0000	0,0000
		Přesnost	1	1	1
	Validační	Ztráta	6,2173	3,9490	5,7697
		Přesnost	0,3636	0,5	0,4091
Konvoluční	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	13,4876	15,0199	14,1734
		Přesnost	0,2727	0,2273	0,2727

Tabulka 11: 1 vrstva, 200 filtrů, 200 neuronů, délka filtru 8, 16, 24, zdroj: vlastní

9.3.5 Vyhodnocení první části

První sada testů je zaměřena na počet neuronů. V tomto případě bylo přistoupeno k velkému kroku v počtu neuronů, aby se ukázalo, zda má smysl použít jich např. stovky.

Z výsledků je patrné, že konvoluční neuronová síť je funkční ve smyslu, uvedeném výše. Je rozhodně schopná predikovat výstupy na základě vstupu a trénovat z generátoru. Jelikož se přesnost predikce na trénovací množině ve všech případech blíží jedné, lze také soudit, že síť se na poskytnutých datech učí a může tedy být použita k diagnostice. Bohužel, z výsledků též vyplývá, že klasifikace na testovací množině nebyla přesvědčivá ani v jedné konfiguraci.

U vícevrstvého perceptronu se výsledky v rámci jednotlivých testů výrazně liší. Pro 200 neuronů každé vstupní sítě i společné sítě sice v jednom případě dosáhly dokonce poloviční přesnosti, ale vypsáním jednotlivých predikcí bylo zjištěno, že síť se jednoduše naučila naprostou většinu vstupů hodnotit známkou 1, která je v poskytnutých datech zastoupena téměř z poloviny. Síť tak sice nachází souvislost, kterou dosáhne relativně dobrého výsledku, ovšem způsobem, který není přípustný.

Vzhledem k faktu, že takto predikující síť neposkytuje relevantní srovnání k předmětné síti, je dopředná vícevrstvá neuronová síť z dalšího testování vypuštěna. Předmětem dalších pokusů je optimalizace parametrů konvoluční neuronové sítě, k zajištění kvalitnější predikce.

Konvoluční síť poskytuje poněkud vyrovnanější výsledky. Zajímavé je, že se zvyšujícím se počtem neuronů a filtrů se její přesnost snižuje, na rozdíl od přesnosti kontrolní sítě. Jelikož žádný jiný parametr sítě měněn nebyl a opačná tendence u perceptronu vylučuje souvislost s počtem neuronů plně propojené sítě, je na vině patrně počet filtrů.

Proto bude otestována síť s kombinací malého počtu filtrů a velkého počtu neuronů. Dále bude testován vliv délek filtrů na kvalitu predikce.

9.3.6 1 vrstva, 16 filtrů, 100 neuronů, délka filtru 8, 16, 24

		Pokus	1	2	3
Konvoluční	Trénovací	Ztráta	0,0581	0,0271	0,0482
		Přesnost	0,9701	0,9851	0,9701
	Validační	Ztráta	4,9434	6,8729	5,5106
		Přesnost	0,4545	0,4545	0,3182

Tabulka 12: 16 filtrů, 100 neuronů, délka filtru 8, 16, 24, zdroj: vlastní

9.3.7 1 vrstva, 16 filtrů, 100 neuronů, délka filtru 4, 8, 12

		Pokus	1	2	3
Konvoluční	Trénovací	Ztráta	0,0000	0,0193	0,0192
		Přesnost	1	0,9701	0,9851
	Validační	Ztráta	12,3504	6,2219	6,5905
		Přesnost	0,1364	0,3636	0,4091

Tabulka 13: 16 filtrů, 100 neuronů, délka filtru 4, 8, 12, zdroj: vlastní

9.3.8 1 vrstva, 16 filtrů, 100 neuronů, délka filtru 16, 32, 48

		Pokus	1	2	3
Konvoluční	Trénovací	Ztráta	0,0187	0,0453	0,0190
		Přesnost	0,9701	0,9552	0,9851
	Validační	Ztráta	9,2765	7,7833	7,8863
		Přesnost	0,1818	0,0455	0,4091

Tabulka 14: 1 vrstva, 16 filtrů, 100 neuronů, délka filtru 16, 32, 48, zdroj: vlastní

9.3.9 Vyhodnocení druhé části

Snížení počtu filtrů přineslo zřejmé zlepšení. Na druhou stranu, zmenšení i zvětšení filtrů přesnost snížilo a proto zůstanou na velikosti 8, 16 a 24.

Výsledky však stále nejsou použitelné v praxi. Přesnost není ani poloviční. Další možnost, jak predikci potenciálně zlepšit, je přidání skrytých vrstev.

9.3.10 2 vrstvy o 16 filtrech, 100 neuronů

		Pokus	1	2	3
Konvoluční	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	5,5643	6,1475	6,9884
		Přesnost	0,5	0,4091	0,4091

Tabulka 15: 2 vrstvy o 16 filtrech, 100 neuronů, zdroj: vlastní

9.3.11 2 vrstvy o 16 filtrech, dvě vrstvy sjednocení

		Pokus	1	2	3
Konvoluční	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	7,7642	8,1129	9,0588
		Přesnost	0,3636	0,4091	0,2727

Tabulka 16: 2 vrstvy o 16 filtrech, dvě vrstvy sjednocení, zdroj: vlastní

9.3.12 2 vrstvy o 16 filtrech a zdvojená plně propojená vrstva vstupu

			2 vrstvy o 16 filtrech a zdvojená p, p, v, vstupu		
Konvoluční		Pokus	1	2	3
	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	7,8389	6,7058	8,1535
		Přesnost	0,3182	0,2727	0,3182

Tabulka 17: 2 vrstvy o 16 filtrech a zdvojená plně propojená vrstva vstupu, zdroj: vlastní

9.3.13 Vyhodnocení třetí části

Přidáním druhé konvoluční vrstvy se síť v predikci opět zlepšila, přínos dalších plně propojených již žádný užitek neindikuje. Přestože se predikce zlepšila, Nelze ji považovat za kvalitní. To je zřejmě velkou měrou vinou nedostatku trénovacích vzorků.

Další trénovací vzorky sice zatím k dispozici nejsou, lze však zjednodušit řešený problém a tím vrstvě klasifikaci ulehčit. Nabízí se klasifikace menšího množství tříd. Pro účely testování předpokládejme jednodušší situaci, kdy je zapotřebí pacienty dělit na zdravé a ostatní. Zdravého pacienta definujme jako pacienta s ohodnocením 0 až 1. Tím se obor hodnot omezuje na dvojici tříd.

K vyhodnocení problému je použita zatím nejúspěšnější síť, tedy síť se dvěmi konvolučními vrstvami po šestnácti filtrech velikosti 8, 16 a 24.

9.3.14 Zjednodušená klasifikace

Konvoluční		Pokus	1	2	3
	Trénovací	Ztráta	0	0	0
		Přesnost	1	1	1
	Validační	Ztráta	3,8725	3,2748	2,9729
		Přesnost	0,5	0,5909	0,6364

Tabulka 18: Zjednodušená klasifikace, zdroj: vlastní

9.3.15 Vyhodnocení čtvrté části

Při klasifikaci do pouze dvou možných skupin, kterým navíc pacienti náleží zhruba polovičním podílem, se přesnost poprvé dostává nad polovinu. Nicméně zběžným pohledem na predikované hodnoty se ukázalo, že skupiny výsledků ani zdánlivě neodpovídají skupinám očekávaného hodnocení. Dat je zřejmě příliš málo na to, aby se projevila korelace.

10 Dokumentace aplikace

10.1 Textové rozhraní

Konzolová aplikace umožňuje trénink a ohodnocení pacientů. Na vstupu očekává operátor a cestu ke složce, kde jsou uloženy vstupní soubory.

Přepínač --help resp. -h vytiskne stručnou nápovědu k použití.

```
(venv) noamd@nomadovoLaptop ~/skola/diplomka/aidi $ python aidi_cli.py
Using TensorFlow backend.
AI diagnostic application
Usage:
        aidi_cli.py <operation> <path>
Operation:
        --train, -t     Train the net, using patient files provided in the <path> directory.
        --predict, -p   Predict results for patients, found in the <path> directory.
        --help, -h      Print this help.
Path:
        A directory, where the patient(s) data are stored.
        Expects one or more complete patients.
(venv) noamd@nomadovoLaptop ~/skola/diplomka/aidi $
```

Obrázek 18: Konzolová aplikace, nápověda, zdroj: vlastní

K trénování slouží přepínač --train, případně -t. Dalším parametrem je předána cesta zdrojové složky. Ze složky jsou načteni všichni pacienti a je zahájeno trénování. Průběžně je vypisován průběh.

```
(venv) noamd@nomadovoLaptop ~/skola/diplomka/aidi $ python aidi_cli.py -t csv_points/
Using TensorFlow backend.
INF: Loading csvs.
100%|##########################################################################################################| 17220/17220 [00:52<00:00, 329.80it/s]
INF: Loaded a total of 95 patients.
INF: Trimming patients.
INF: Trimmed to a total of 89 patients.
INF: Fitting exercises.
INF: Fitted all the exercises.
INF: Flattening exercises.
INF: Flattened all the exercises and performing down-sampling..
INF: The original sample is 678 steps long.
INF: Created a down-sample by 2 with the length of 339.
INF: Created a down-sample by 5 with the length of 135.
Epoch 1/1000
6/6 [==============================] - 5s 785ms/step - loss: 1.9171 - accuracy: 0.3034
Epoch 2/1000
6/6 [==============================] - 4s 720ms/step - loss: 1.4528 - accuracy: 0.4607
Epoch 3/1000
6/6 [==============================] - 4s 720ms/step - loss: 1.4604 - accuracy: 0.4270
Epoch 4/1000
3/6 [==============>...............] - ETA: 2s - loss: 1.4330 - accuracy: 0.4792
```

Obrázek 19: Konzolová aplikace, trénování, zdroj: vlastní

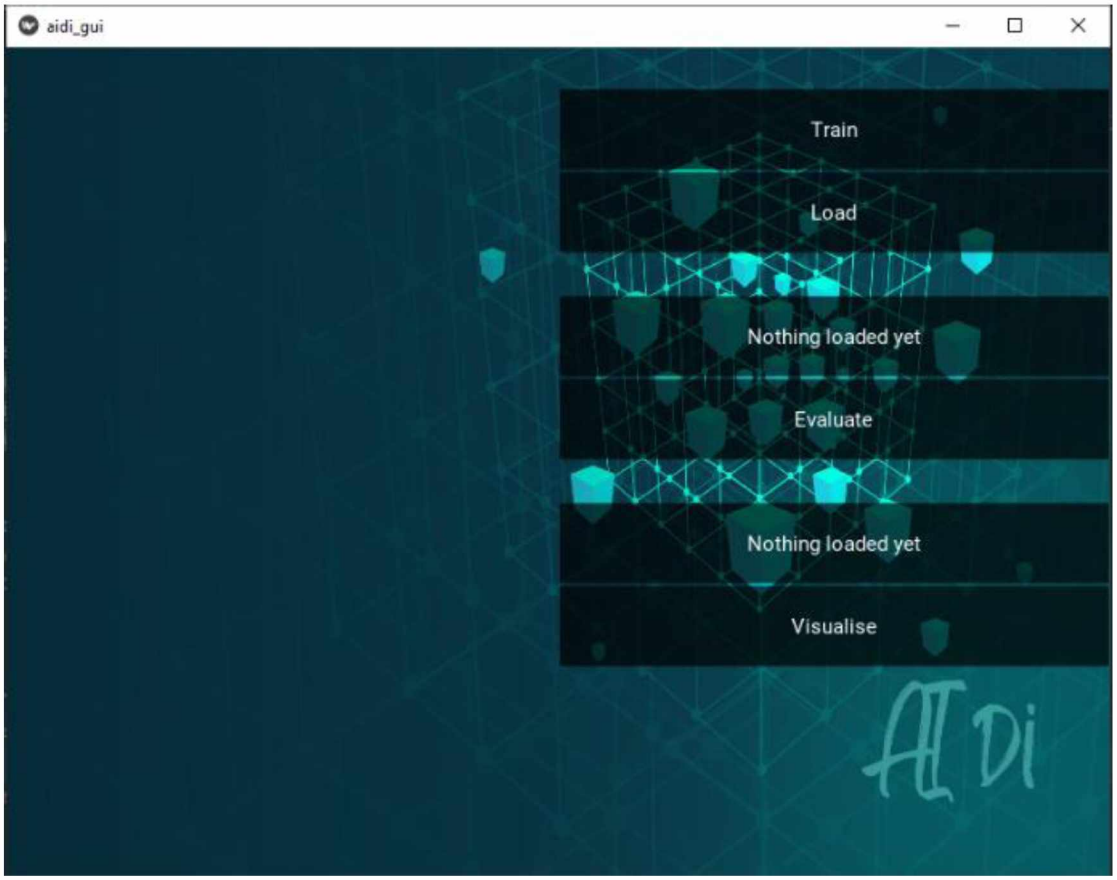
Následně je možné přejít k hodnocení. To se spouští přepínačem --predict, nebo -p. V dalším parametru je opět očekávána cesta ke zdrojové složce. Všichni pacienti ve složce jsou načteni a zpracováni. Informace o pacientech jsou jednotlivě vypisovány spolu s hodnocením.

```
(venv) noamd@nomadovoLaptop ~/skola/diplomka/aidi $ python aidi_cli.py -p csv_tryit
Using TensorFlow backend.
INF: Loading csvs.
100%|##########################################################################################################| 189/189 [00:00<00:00, 314.48it/s]
INF: Loaded a total of 1 patients.
INF: Trimming patients.
INF: Trimmed to a total of 1 patients.
INF: Fitting exercises.
INF: Fitted all the exercises.
Patient:
        Id: 1
        Date: 2019-02-03
        Pre-operative evaluation: 0
        Evaluation: 1
(venv) noamd@nomadovoLaptop ~/skola/diplomka/aidi $
```

Obrázek 20: Konzolová aplikace, predikce, zdroj: vlastní

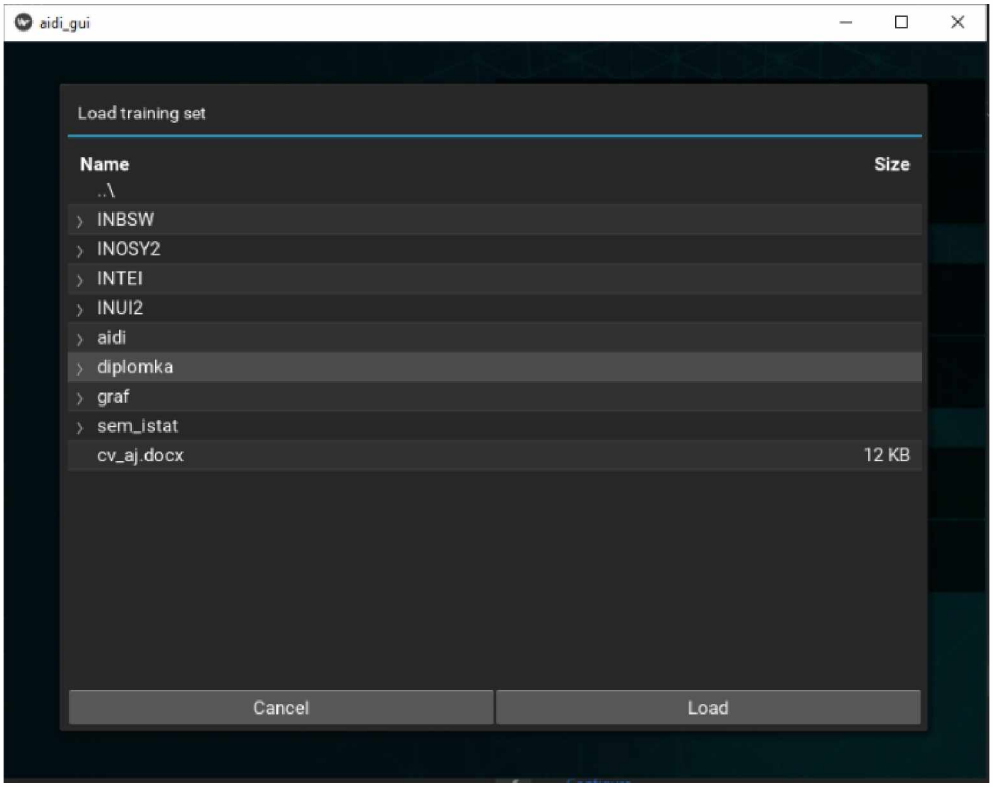
10.2 Grafické rozhraní

Grafické uživatelské prostředí slouží pro pohodlnější obsluhu. Oproti konzolové variantě navíc poskytuje vizualizaci cviků.



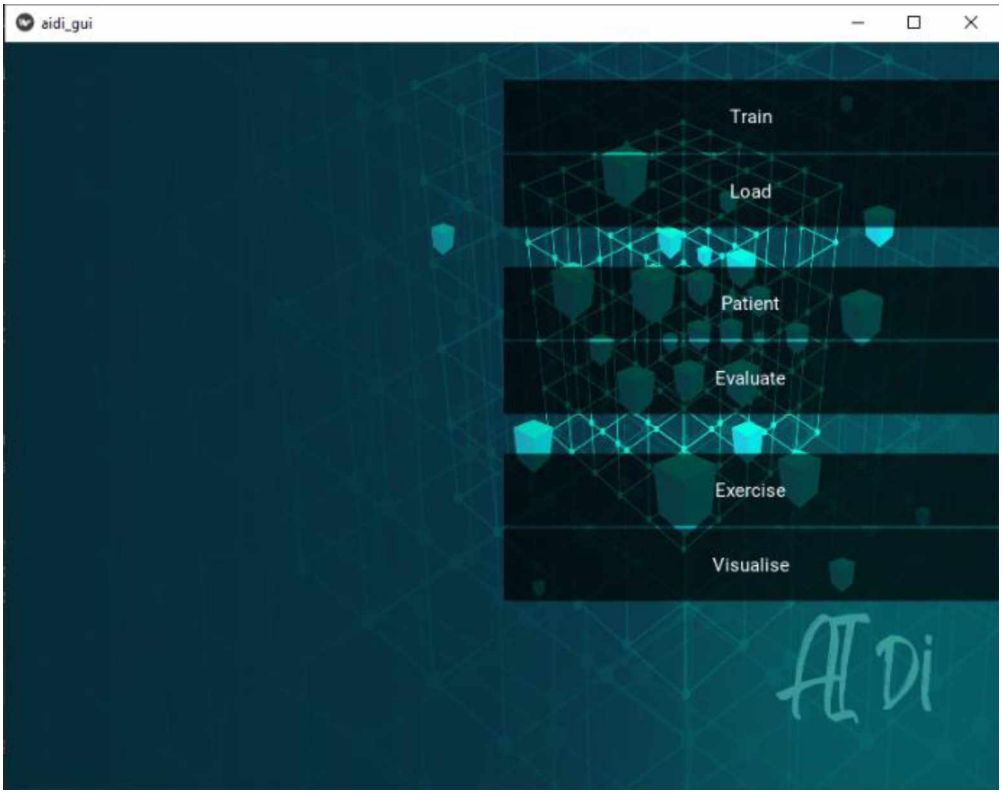
Obrázek 21: Grafické uživatelské rozhraní, zdroj: vlastní

Nejdřív je třeba síť natrénovat. Klikutím na tlačítko Train se otevře dialog, ve kterém stačí vybrat složku s daty a trénování započne.

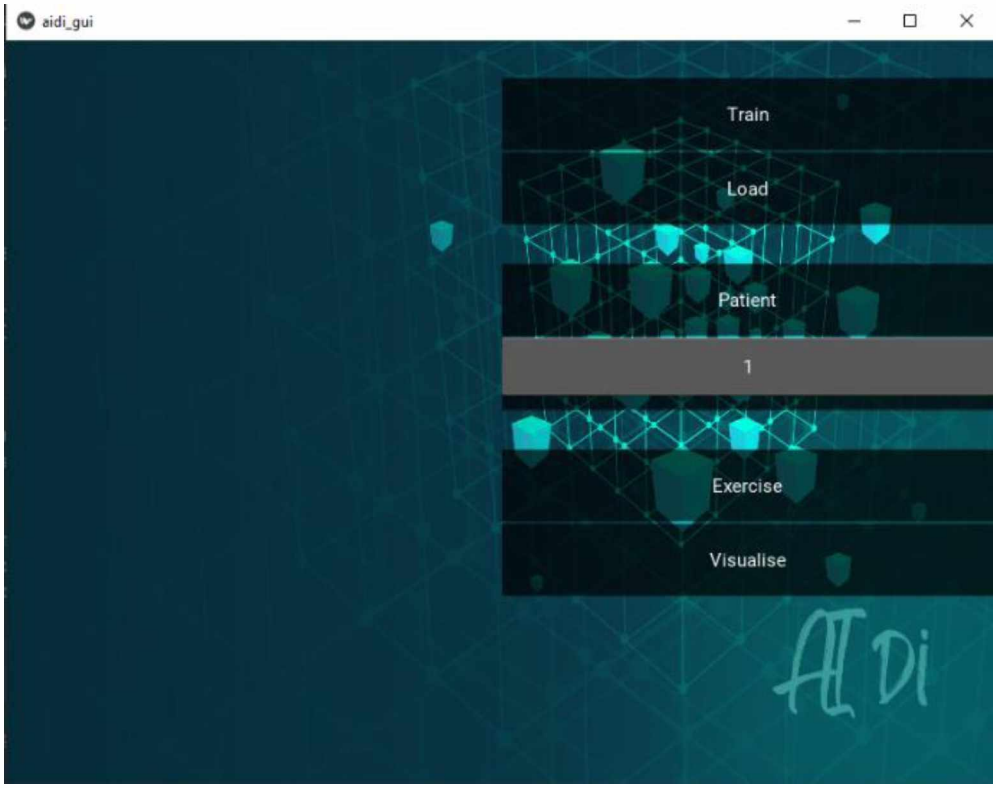


Obrázek 22: Dialog pro výběr zdrojových dat, zdroj: vlastní

Analogicky probíhá načtení dat tlačítkem Load. Tato data nejsou určena k testování, ale k prohlížení. Po načtení se zpřístupní funkcionalita predikce a vizualizace.

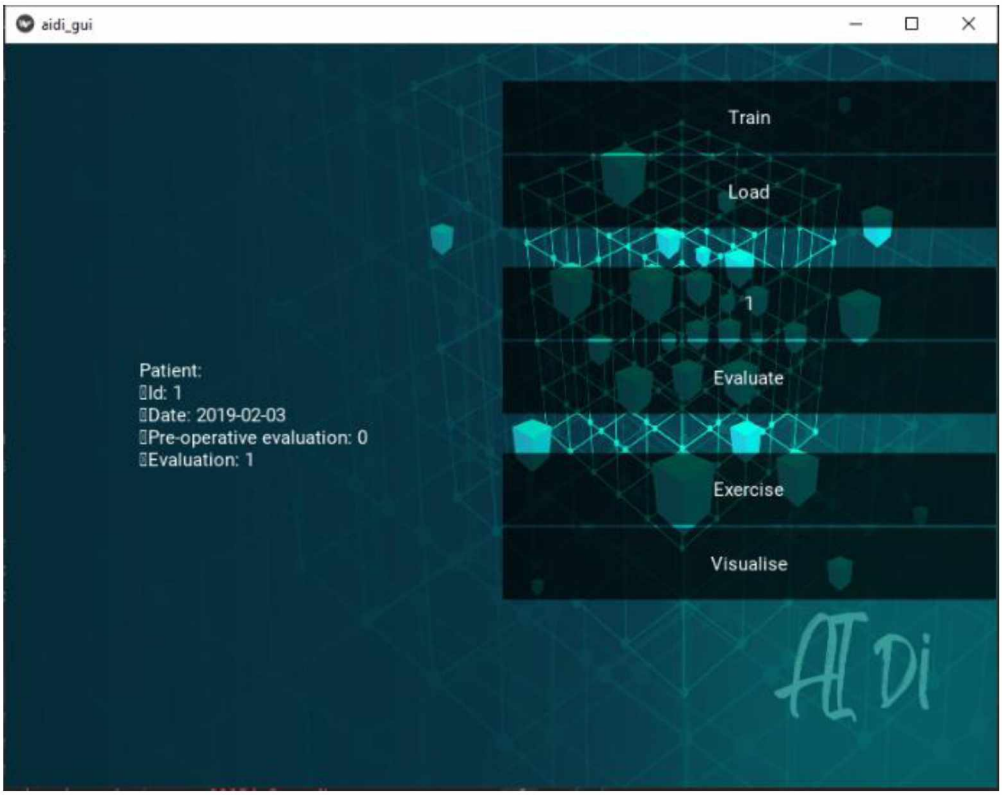


Obrázek 23: Rozhraní po načtení dat, zdroj: vlastní

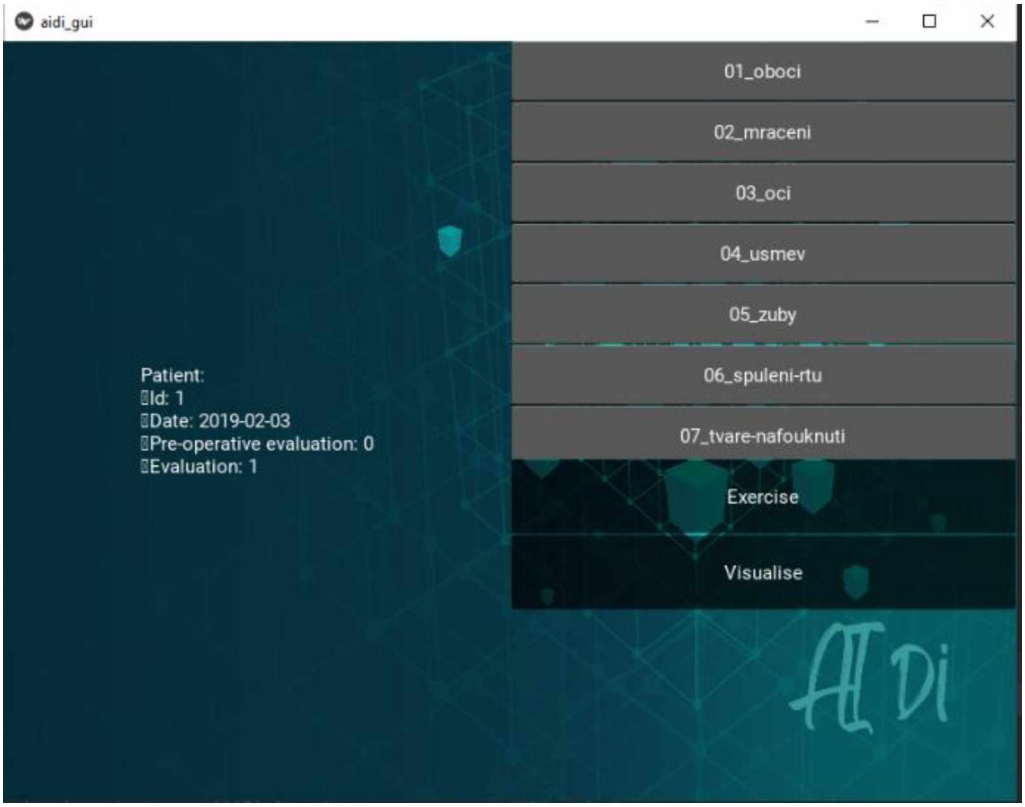


Obrázek 24: Výběr pacienta, zdroj: vlastní

Pacienta si uživatel vybere ze seznamu. Zde je pouze jediný pacient s identifikačním číslem 1. Následně, stiskem tlačítka Evaluate, proběhne hodnocení pacienta.

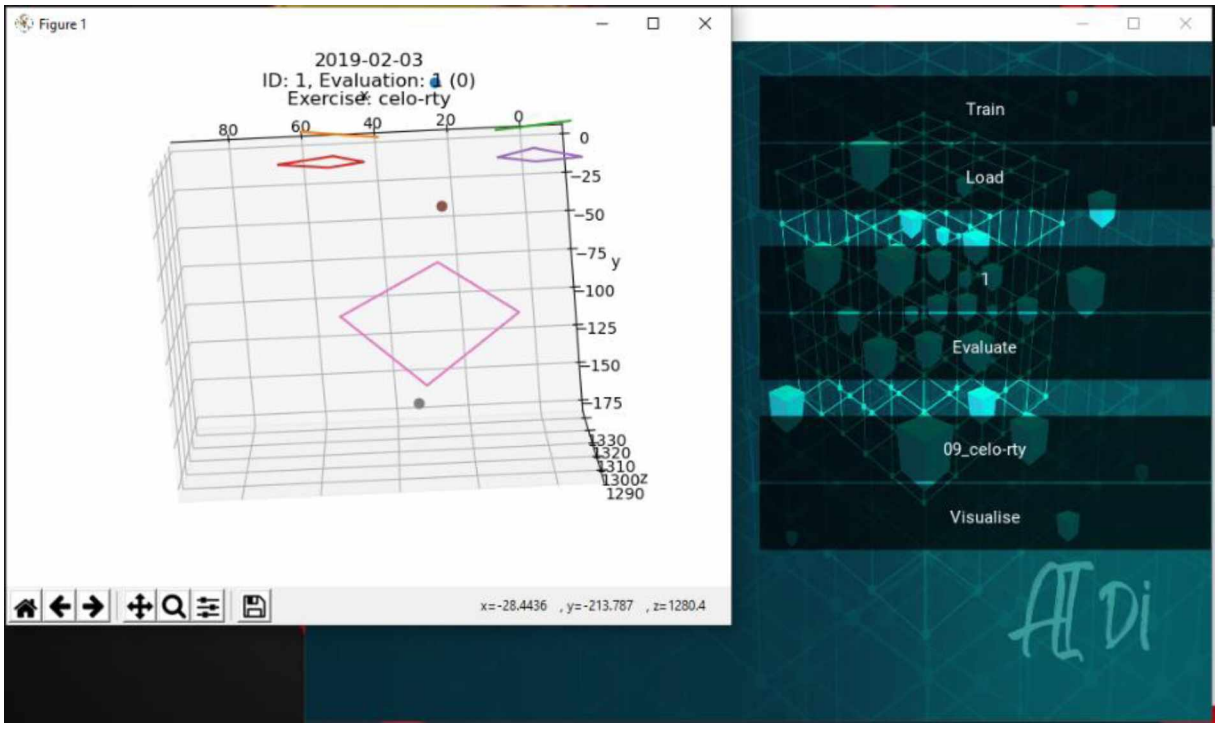


Obrázek 25: Hodnocení pacienta, zdroj: vlastní



Obrázek 26: Výběr cviku, zdroj: vlastní

Stejně probíhá i výběr cviku. Tlačítkem Visualise se spustí jednoduchá vizualizace.



Obrázek 27: Vizualizace cviku, zdroj: vlastní

11 ZÁVĚR

V rámci diplomové práce byl navržen a implementován nástroj pro diagnostiku onemocnění pohybového aparátu pomocí zpracování velkých dat umělou neuronovou sítí. Konkrétním předmětem diagnostiky byly shledány pooperační komplikace, spojené s vestibulárním schwannomem. Proto bylo toto nádorové onemocnění představeno, spolu s jeho příznaky, dopady a současnou metodou pooperační diagnostiky.

Nebylo pochyb, že zde prostor pro automatizaci je. Po stručném popisu sběru dat byla pozornost věnována umělým neuronovým sítím a jejich existujících aplikací v lékařské diagnostice. Byly též představeny použité softwarové nástroje, které vývoj s umělými neuronovými sítěmi umožnují či usnadňují.

Ve druhé části byla provedena podrobná analýza poskytnutých dat a na jejich základě vypracován a dokumentován návrh aplikace, která byla rovněž implementována. Výsledkem je nástroj, který predikuje hodnocení pacienta na základě pohybu jeho obličeje.

V závěru práce byla provedena série testů, které vyzkoušely síť z hlediska obecné funkčnosti i z hlediska kvality predikce. Bohužel se ukázalo, že predikce sítě je prozatím nedostatečná. Na její zhruba poloviční přesnost se spoléhat nelze.

Na základě schopnosti naučit se trénovací vzorky však lze tvrdit, že vyvinutá konvoluční umělá neuronová síť dokáže na vstupních datech rozeznávat vzory a má tak potenciál klasifikovat pacienty pomocí hlubokého učení, za předpokladu, že bude k dispozici dostatečná trénovací množina. To je samozřejmě především otázkou času.

Byl tedy položen základ aplikace, která může v budoucnu usnadnit život lékařům i pacientům. Otevírá možnost diagnostiky bez přítomnosti lékaře, možná i z pohodlí domova, pomocí dalších zařízení pro záznam stereoskopického videa. Kromě toho také redukuje vliv subjektivního hodnocení lékaře.

Zajímavým rozšířením by byla například mobilní aplikace, kterou by se pacienti mohli kdykoliv diagnostikovat sami. Dále by aplikace mohla udržovat historii kontrol a poskytovat pacientovi informace o zlepšování, případně zhoršování jeho stavu. Obecně by také bylo dobré, kdyby vyhodnocování probíhalo on-line. Pacient by tak dostával zpětnou vazbu okamžitě a mohl by s její pomocí třeba i pracovat na zlepšení v rámci rehabilitace.

Změnami v konstantách lze aplikaci snadno přizpůsobit změnám. Je kupříkladu možné upravit počet výstupních tříd, upravit počet cviků či snímaných bodů a podobně.

Aplikováním objektově orientovaného paradigmatu se však aplikace stala i do značné míry modulární. Lze ji logicky členit na tři části, a sice na modul načítání, modul vyhodnocování a modul obsluhy.

První část zahrnuje třídu pro načítání souborů a kontejnerové třídy pro uchování a modifikaci dat. Představuje tedy nástroj pro načtení specifických zdrojových dat do konzistentní, přehledné a dále zpracovatelné formy. Jejím nahrazením lze zobecnit klasifikaci aplikací na libovolná jiná vstupní data.

Vyhodnocující modul je tvořen umělou neuronovou sítí (v tomto případě konvoluční neuronovou sítí) a generátorem. Stará se tedy o konverzi načtených objektů do akceptovatelného formátu pro klasifikaci a klasifikaci samotnou. Nahrazením lze pro hodnocení použít jinou topologii sítě, nebo i úplně jiný přístup. Kromě umělé neuronové sítě je použitelný například expertní systém.

I neuronové sítě ale nabízejí další alternativy, které se do rámce této diplomové práce nevešly. Řeč je zejména o složitých sítích typu ResNet, InceptionTime či HIVE-COTE. Takové topologie by mohly (na odpovídajícím technickém vybavení) dosahovat lepších výsledků než představená síť. Bez ohledu na zvolenou topologii je však prvořadě zapotřebí násobně více trénovacích vzorků.

Konečně, modul obsluhy zahrnuje obě uživatelská rozhraní a jeho nahrazením lze síť upravit pro specifické požadavky uživatele.

12 POUŽITÁ LITERATURA

1. KOHOUT, Jan, Jan CRHA, Kateřina TRNKOVÁ, Karel ŠTÍCHA, Jan MAREŠ a Martin CHOVANEC. ROBOT-BASED IMAGE ANALYSIS FOR EVALUATING REHABILITATION AFTER BRAIN SURGERY. MENDEL: International Conference on Soft Computing. Brno: Brno University of Technology, Faculty of Mechanical Engineering, 1995-, 2018(24), 159-164. ISSN 1803-3814.
2. ZVĚŘINA, Eduard. Neurinom akustiku – vestibulární schwannom – osobní pohled na nejmodernější postupy v jeho léčbě. Časopis lékařů českých. 1862-, 2010(149), 269-276. ISSN 0008-7335.
3. DOLEŽEL, Petr. Úvod do umělých neuronových sítí pro studenty technických vysokých škol. Pardubice: Univerzita Pardubice, 2016. ISBN 978-80-7560-022-6.
4. SUZUKI, Kenji. Artificial Neural Networks - Methodological Advances and Biomedical Applications. Rijeka: InTech, 2011. ISBN 978-953-307-243-2.
5. Vestibular Schwannoma (Acoustic Neuroma) and Neurofibromatosis. National Institute on Deafness and Other Communication Disorders [online]. Bethesda: NIDCD Information Clearinghouse, 2015 [cit. 2020-05-28]. Dostupné z: https://www.nidcd.nih.gov/health/vestibular-schwannoma-acoustic-neuroma-and-neurofibromatosis
6. PRAUS, Petr. Inteligence a její měření. Časopis Mensy České republiky [online]. Praha: Mensa České republiky, c2008 [cit. 2020-05-28]. Dostupné z: https://casopis.mensa.cz/veda/inteligence_a_jeji_mereni.html
7. CHALUPNÍK, VITALIJ. Biologické algoritmy (4) - Neuronové sítě. ROOT.CZ: Informace nejen ze světa Linuxu [online]. Praha: Internet Info, c1998-2020, 25. 4. 2012 [cit. 2020-05-28]. Dostupné z: https://www.root.cz/clanky/biologicke-algoritmy-4-neuronove-site/
8. ISMAIL FAWAZ, Hassan, Germain FORESTIER, Jonathan WEBER, Lhassane IDOUMGHAR a Pierre-Alain MULLER. Deep learning for time series classification: a review. Data Mining and Knowledge Discovery [online]. c2020, 2. 3. 2019, 2019(33), 917–963 [cit. 2020-05-28]. ISSN 1384-5810. Dostupné z: https://doi.org/10.1007/s10618-019-00619-1
9. Loss Functions. Read the Docs: Create, host, and browse documentation. [online]. Read the Docs, Inc & contributors, c2020 [cit. 2020-05-28]. Dostupné z: https://ml-cheatsheet.readthedocs.io/en/latest/loss_functions.html

10. SAHA, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. Towards Data Science: Sharing conceps, ideas and codes [online]. A Medium Corporation, c2020, 15. 12. 2018 [cit. 2020-05-28]. Dostupné z: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53
11. HÜSKEN, Michael a Peter STAGGE. Recurrent neural networks for time series classification. Neurocomputing [online]. 2003, 2003(50), 223-235 [cit. 2020-05-28]. Dostupné z: https://www.sciencedirect.com/science/article/abs/pii/S0925231201007068?via%3Dihub
12. Umělá inteligence – definice dle Expertní skupiny na AI. SEDLAKOVA LEGAL [online]. c2020, 23. 2. 2020 [cit. 2020-05-28]. Dostupné z: https://www.sedlakovalegal.cz/definice-umela-inteligence/
13. GÖRLICHOVÁ, Lucie. Umělé neuronové sítě v lékařské diagnostice [online]. Brno, 2006 [cit. 2020-05-28]. Dostupné z: https://is.muni.cz/th/tqkvr/Text_prace.pdf. Diplomová práce. Masarykova univerzita v Brně, Přírodovědecká fakulta, Katedra analytické chemie. Vedoucí práce Marta Farková.
14. Umělá inteligence pomáhá při diagnostice a léčbě. Zdravotnický deník: zdravé je vědět [online]. Praha: Media Network s.r.o, c2020, 8. 9. 2017 [cit. 2020-05-28]. Dostupné z: https://www.zdravotnickydenik.cz/2017/09/umela-inteligence-pomaha-pri-diagnostice-lecbe/
15. COVID-19 response automation: IBM Watson Assistant. IBM [online]. [cit. 2020-05-28]. Dostupné z: https://www.ibm.com/watson/covid-response
16. NIILER, ERIC. An AI Epidemiologist Sent the First Warnings of the Wuhan Virus. WIRED [online]. Condé Nast, c2020, 25. 1. 2020 [cit. 2020-05-28]. Dostupné z: https://www.wired.com/story/ai-epidemiologist-wuhan-public-health-warnings/
17. Jak dokáže umělá inteligence měnit zdravotnictví. Otevřené zdravotnictví: Zdravotnicví, věc veřejná [online]. c2018, 1. 2. 2019 [cit. 2020-05-28]. Dostupné z: https://www.otevrenezdravotnictvi.cz/novinky/jak-dok%C3%A1%C5%BEe-um%C4%9Bl%C3%A1-inteligence-m%C4%9Bnit-zdravotnictv%C3%AD.html
18. Umělá inteligence předčila doktory v diagnostice rakoviny prsu. Computer World: Deník pro IT profesionály [online]. Praha: Internet Info DG, 4. 1. 2020 [cit. 2020-05-28]. Dostupné z: https://computerworld.cz/technologie/umela-inteligence-predcila-doktory-v-diagnostice-rakoviny-prsu-55793

19. Using AI to predict retinal disease progression. DeepMind [online]. London: DeepMind Technologies Limited, 18. 5. 2020 [cit. 2020-05-28]. Dostupné z: https://deepmind.com/blog/article/Using_ai_to_predict_retinal_disease_progression
20. NOVÁ ÉRA KOSMETIKY V ČR: L'ORÉAL PŘINÁŠÍ ČESKÝM SPOTŘEBITELŮM SLUŽBY POHÁNĚNÉ UMĚLOU INTELIGENCÍ A ROZŠÍŘENOU REALITOU. L'Oréal [online]. Praha: L'Oréal Česká Republika, S R.O., 19. 6. 2019 [cit. 2020-05-28]. Dostupné z: https://www.loreal.cz/novinky/novinky/2019/june/limitlessbeauty
21. THOMA, Martin. Kivy Designer. Github [online]. San Francisco: github, 2020 [cit. 2020-05-28]. Dostupné z: https://github.com/kivy/kivy-designer
22. AMATO, Filippo, Alberto LOPEZ, Eladia María PENA-MENDEZ, Petr VANHARA, Ales HAMPL a Josef HAVEL. Artificial neural networks in medical diagnosis. Journal of Applied Biomedicine [online]. 2013, 11(2), 47-58 [cit. 2019-10-08]. DOI: 10.2478/v10136-012-0031-x. ISSN 1214021X.
23. What is TensorFlow? Introduction, Architecture & Example. Guru99 [online]. c2020 [cit. 2020-05-28]. Dostupné z: https://www.guru99.com/what-is-tensorflow.html
24. CHOLLET, François, et al. About Keras. Keras: Simple. Flexible. Powerful. [online]. [cit. 2020-05-28]. Dostupné z: https://keras.io/about/
25. PASZKE, Adam, Sam GROSS, Francisco MASSA, et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library [online]. Curran Associates, 2019 [cit. 2020-05-28]. Dostupné z: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf
26. CUDA GPUs. NVIDIA Developer [online]. c2020 [cit. 2020-05-28]. Dostupné z: https://developer.nvidia.com/cuda-gpus
27. Understanding Tensor Processing Units. GeeksforGeeks: A computer science portal for geeks [online]. Noida: GeeksforGeeks [cit. 2020-05-28]. Dostupné z: https://www.geeksforgeeks.org/understanding-tensor-processing-units/
28. OLIPHANT, Travis E. A guide to NumPy. USA: Trelgol Publishing, 2006 [cit. 2020-05-28].
29. HUNTER, John D. Matplotlib: A 2D Graphics Environment. Computing in Science & Engineering [online]. 2007, 9(3), 90-95 [cit. 2020-05-28]. DOI: 10.1109/MCSE.2007.55. ISSN 1521-9615. Dostupné z: http://ieeexplore.ieee.org/document/4160265/

30. Tqdm. Github [online]. San Francisco: GitHub, 2020 [cit. 2020-05-27]. Dostupné z: https://github.com/tqdm/tqdm/blob/master/LICENCE
31. What is Python? Executive Summary. Python [online]. Python Software Foundation, 2020 [cit. 2020-05-27]. Dostupné z: https://www.python.org/doc/essays/blurb/
32. What is Python? Opensource.com [online]. Red Hat, 2019 [cit. 2020-05-27]. Dostupné z: https://opensource.com/resources/python
33. Kivy - Open source Python library for rapid development of applications that make use of innovative user interfaces, such as multi-touch apps. Kivy.org [online]. kivy, 2020 [cit. 2020-05-27]. Dostupné z: https://kivy.org/#home
34. Python Top GUI Frameworks For Developers in 2020. Morioh [online]. Tallinn, Estonia, 2020 [cit. 2020-05-27]. Dostupné z: https://morioh.com/p/72aabaff0e7e
35. What are the best Python GUI frameworks/toolkits? Slant [online]. Parli, 2019 [cit. 2020-05-27]. Dostupné z: https://www.slant.co/topics/6620/~python-gui-frameworks-toolkits#1
36. The 6 Best Python GUI Frameworks for Developers. Resellerclub [online]. ResellerClub, 2020 [cit. 2020-05-27]. Dostupné z: https://blog.resellerclub.com/the-6-best-python-gui-frameworks-for-developers/
37. InteliiJ IDEA. Jetbrains [online]. jetbrains, 2020 [cit. 2020-05-27]. Dostupné z: https://www.jetbrains.com/idea/promo/ultimate/?gclid=EAIaIQobChMIlcSRnfjL6QIVxITVCh1ugAL6EAAYASAAEgJzVfD_BwE
38. Best IDE Software - A List of the Top 10. Keycdn [online]. Switzerland: proinity, 2017 [cit. 2020-05-27]. Dostupné z: https://www.keycdn.com/blog/best-ide
39. 10 Best IDE Software. Websitesetup [online]. websitesetup, 2020 [cit. 2020-05-27]. Dostupné z: https://websitesetup.org/best-ide-software/
40. What are the best IDEs? Slant [online]. Parli, 2020 [cit. 2020-05-28]. Dostupné z: https://www.slant.co/topics/10021/~ides
41. PyCharm vs. IntelliJ IDEA Python plugin FAQ. Confluence [online]. Atlassian Confluence, 2018 [cit. 2020-05-28]. Dostupné z: https://confluence.jetbrains.com/display/PYH/PyCharm+vs.+IntelliJ+IDEA+Python+plugin+FAQ
42. Free Educational Licenses. Jetbrains [online]. JetBrains, 2020 [cit. 2020-05-28]. Dostupné z: https://www.jetbrains.com/community/education/#students
43. Getting Started. Visual studio code [online]. Microsoft, 2020 [cit. 2020-05-28]. Dostupné z: https://code.visualstudio.com/docs

44. Visual Studio Code FAQ. Visual studio code [online]. Microsoft, 2020 [cit. 2020-05-28]. Dostupné z: https://code.visualstudio.com/docs/supporting/FAQ
45. Python Lists. W3schools.com [online]. Refsnes Data, 2020 [cit. 2020-05-28]. Dostupné z: https://www.w3schools.com/python/python_lists.asp
46. Free/Libre Open Source Software Binaries of VSCode [online]. [cit. 2020-05-28]. Dostupné z: https://vscodium.com/
47. ABADI, Martín, Paul BARHAM, Jianmin CHEN, et al. TensorFlow: A System for Large-Scale Machine Learning [online]. 12th USENIX Symposium on Operating Systems Design and Implementation. USENIX Association, 2015 [cit. 2020-05-28]. ISBN 978-1-931971-33-1. Dostupné z: https://www.usenix.org/system/files/conference/osdi16/osdi16-abadi.pdf
48. GRANAT, Margarita. How to Use Convolutional Neural Networks for Time Series Classification. Towards Data Science: Sharing conceps, ideas and codes [online]. A Medium Corporation, c2020, 2019 [cit. 2020-05-28]. Dostupné z: https://towardsdatascience.com/how-to-use-convolutional-neural-networks-for-time-series-classification-56b1b0a07a57
49. PERERA, Ayeshmantha. What is Padding in Convolutional Neural Network's(CNN's) padding. Medium [online]. Medium, 2018 [cit. 2020-05-28]. Dostupné z: https://medium.com/@ayeshmanthaperera/what-is-padding-in-cnns-71b21fb0dd7
50. BUDHIRAJA, Amar. Dropout in (Deep) Machine learning. Medium [online]. Medium, 2016 [cit. 2020-05-28]. Dostupné z: https://medium.com/@amarbudhiraja/https-medium-com-amarbudhiraja-learning-less-to-learn-better-dropout-in-deep-machine-learning-74334da4bfc5
51. THOMA, Martin. How to interpret "loss" and "accuracy" for a machine learning model. Stackoverflow [online]. Stack Exchange, 2016 [cit. 2020-05-28]. Dostupné z: https://stackoverflow.com/questions/34518656/how-to-interpret-loss-and-accuracy-for-a-machine-learning-model

13 PŘÍLOHY

Příloha A: Licence – Keras

Příloha B: Licence – NumPy

Příloha C: Licence – Tqdm

Příloha A: Licence - Keras

COPYRIGHT

All contributions by François Chollet:

Copyright (c) 2015 - 2019, François Chollet.

All rights reserved.

All contributions by Google:

Copyright (c) 2015 - 2019, Google, Inc.

All rights reserved.

All contributions by Microsoft:

Copyright (c) 2017 - 2019, Microsoft, Inc.

All rights reserved.

All other contributions:

Copyright (c) 2015 - 2019, the respective contributors.

All rights reserved.

Each contributor holds copyright over their respective contributions.

The project versioning (Git) records all such contribution source information.

LICENSE

The MIT License (MIT)

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.

Příloha B: Licence – NumPy

Příloha C: Licence – Tqdm

`tqdm` is a product of collaborative work.

Unless otherwise stated, all authors (see commit logs) retain copyright

for their respective work, and release the work under the MIT licence

(text below).

Exceptions or notable authors are listed below

in reverse chronological order:

* files: *

  MPLv2.0 2015-2020 (c) Casper da Costa-Luis

  [casperdcl](https://github.com/casperdcl).

* files: tqdm/_tqdm.py

  MIT 2016 (c) [PR #96] on behalf of Google Inc.

* files: tqdm/_tqdm.py setup.py README.rst MANIFEST.in .gitignore

  MIT 2013 (c) Noam Yorav-Raphael, original author.

[PR #96]: https://github.com/tqdm/tqdm/pull/96

Mozilla Public Licence (MPL) v. 2.0 - Exhibit A

This Source Code Form is subject to the terms of the

Mozilla Public License, v. 2.0.

If a copy of the MPL was not distributed with this file,

You can obtain one at https://mozilla.org/MPL/2.0/.

MIT License (MIT)

Copyright (c) 2013 noamraph

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.