FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Softwarová aplikace založená na umělé inteligenci pro real time klasifikaci pohybu
Student:	Vojtěch Slavík
Vedoucí:	doc. Ing. Patrik Kutílek, MSc., Ph.D.
Studijní program:	Informatika
Obor / specializace:	Znalostní inženýrství
Katedra:	Katedra aplikované matematiky
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Cílem práce je návrh, programování a testování software pro klasifikaci tréninkových ukazatelů z pohybových dat MoCap 9DOF MEMS, které poskytují 3D data o akceleraci a úhlové rychlosti. Aplikace bude navržena pro účely využití v rehabilitaci. Na základě analýzy současného stavu navrhněte metody zpracování kinematických veličin zaznamenaných MoCap systému během tréninku pro klasifikaci pohybu. Dále navrhněte a v praktické části práce implementujte vhodné metody strojového učení pro klasifikaci dat pohybu. Praktická část práce bude zahrnovat vývojové diagramy, návrh programu využívající neuronové sítě, vlastní programování v jazyce Python nebo MatLab, verzovaný kód na GitHub a dokumentaci kódu. Navržené metody otestujte a statisticky vyhodnoťte aplikovatelnost vámi navrženého systému a metod pro praxi.

Elektronicky schválil/a Ing. Karel Klouda, Ph.D. dne 20. října 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Bakalářská práce

Softwarová aplikace založená na umělé inteligenci pro real time klasifikaci pohybu

Vojtěch Slavík

Katedra aplikované matematiky
Vedoucí práce: doc. Ing. Patrik Kutílek, MSc., Ph.D.

22. června 2022

Poděkování

Nejdříve bych chtěl poděkovat mému vedoucímu doc. Ing. Patrikovi Kutílkovi, MSc., Ph.D. a jeho kolegům Ing. Janovi Hejdovi, Ph.D a Bc. Jánovi Hýblovi za jejich velikou pomoc při tvorbě této práce. Dále také chci poděkovat Ing. Alešovi Příhodovi za poskytnutí popisů rehabilitačních cviků, a hlavně děkuji svým rodičům, kteří mě celý můj život plně podporují.

Prohlášení

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

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 22. června 2022 . . . . . .

Odkaz na tuto práci

Slavík, Vojtěch. *Softwarová aplikace založená na umělé inteligenci pro real time klasifikaci pohybu.* Bakalářská práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Klasifikace pohybu je populární vědeckým cílem, a to především pohybu člověka. Kategorizace se převážně provádí pomocí shlukových funkcí nebo neuronových sítí, či jejich kombinací. Tato práce umožňuje klasifikaci osmi různých rehabilitačních cviků využívající paži, a to za pomoci tří různých metodik, jejichž výsledky jsou v této práci porovnány podle rychlosti a přesnosti. Metody dosahují přesnosti 95,87%, 91,90% a 43,60%, a práce tak umožňuje přesnou a spolehlivou klasifikaci. Pro snímání pohybu je používán senzor MoCap 9 DOF MEMS, z kterého jsou využita 3D data o akceleraci a úhlové rychlosti.

Klíčová slova Klasifikace pohybu, Umělá inteligence, Neuronové sítě, Senzory, Rehabilitace, Snímání pohybu

Abstract

Motion classification is a popular scientific goal, especially human movement. The classification is mainly performed using cluster functions, neural networks or the combination of both. This work enables classification of eight different rehabilitation exercises using the arm, with the help of three different methodoligies, the results of which are compared in this work according to speed and accuracy. The methods achieve accuracies of 95,87%, 91,90% a 43,60%, and

the work thus enables accurate and reliable classification. The MoCap 9 DOF MEMS sensor is used for motion sensing, from which 3D data on acceleration and angular velocity are used.

Keywords Motion classification, Artificial intelligence, Neural networks, Sensors, Rehabilitation, Motion capture

Obsah

Seznam obrázků

Seznam tabulek

Kapitola 1

Úvod

Rehabilitační cviky jsou efektivní metodou pro zlepšení fyzického stavu pacienta. Tyto techniky jsou nejlépe provázeny pod dohledem experta, který dohlíží na kvantitu a kvalitu cviků, jenž pacient provádí. Bohužel není vždy možné, aby expert neustále dohlížel na provedení těchto cviků, čímž se snižuje účinnost rehabilitace.

Tato práce má za cíl nahradit roli experta při určování, jaké cviky pacient provádí a tím potencionálně odlehčit pracovní zátěž odborníka. Tato substituce je dosažena pomocí systému, který v reálním čase samostatně, spolehlivě a přesně vyhodnocuje jaký pohyb pacient provádí neboli pomocí real-time klasifikace pohybů. Konkrétně se jedná o kategorizaci dat nasnímaných z jednoho sensoru, který poskytuje data o akceleraci, úhlové rychlosti a magnetické orientace o frekvenci 100 Hz.

Pro dosažení co nejlepší přesnosti této aplikace byly implementovány tři různé metody kategorizace, které byly otestovány, porovnány a nakonec ohodnoceny, zda je možné tyto metody použít v praxi.

Hlavní motivace pro výběr této práce bylo použití umělé inteligence, jelikož mě toto téma přijde velmi aktuální a zajímavé. Dále mě zaujalo, že by se práce dala využít v medicíně pro náhradu experta, což mi přijde jako skvělá motivace pro tvorbu práce.

1.1 Cíle práce

Cílem práce je návrh, programování a testování aplikace pro klasifikaci pohybových dat nasnímaných z inerciálního pohybového senzoru, za účelem použití v rehabilitaci. Na základě analýzy současného stavu budou navrženy tři metody klasifikace, které budou statisticky porovnány a otestovány zda jsou přesné a spolehlivé.

Praktická část práce bude také zahrnovat vývojové diagramy, verzovaný kód na GitHub a dokumentaci.

Kapitola 2

Současný stav problematiky

Klasifikace pohybu je populární a relevantní obor, ohledně kterého vzniklo mnoho prací. Tento vysoký počet studií je dán neexistencí univerzální metody, která by dosahovala optimálních výsledků. Pro každý typ klasifikace pohybu je potřeba pečlivě vybrat typ senzoru, jaké příznaky z dat využívat, a hlavně jakou metodu zvolit pro samotnou kategorizaci dat. V mé práci jsem se rozhodl implementovat a testovat tři různé metody klasifikace. v této kapitole jsou tyto metodiky představeny, společně se souvisejícími pracemi, které tyto metody využívají.

2.1 Klasifikace pohybu

Klasifikaci pohybu lze rozdělit do dvou částí. Získání dat pomocí senzoru, které obsahují správné a potřebné informace ohledně pohybu pro klasifikaci a následné určení o jaký typ pohybu se jedná pomocí strojového učení aplikovaném na těchto datech. Pro kategorizaci pohybu existuje několik typů senzorů – optické, inerciální, magnetické, mechanické a akustické [3].

Každý z těchto typů má své výhody a nevýhody a jsou vhodné pro rozdílné klasifikace. V této práci byl použitý inerciální senzor. Tyto snímače používají akcelerometry, které měří zrychlení objektu, gyroskopy snímající úhlovou rychlost a magnetometry pro určení orientace objektu. Zde využitý senzor používá všechny tři tyto součástky a je označovaný jako typ 9DOF – 9 degrees of freedom – jelikož snímá o pohybu devět příznaků.

Přínos a důvod použití inerciálního snímače je lehká aplikace. Stačí pouze senzor upevnit na určené místo – zde na pravém zápěstí – a spustit snímání. Inerciální senzory není ovšem vždy vhodné použít, jelikož může vzniknout kumulativní chyba při odhadu polohy pomocí integrace akcelerace a úhlové rychlosti.[3] Tato vada se ale v této práci neprojevuje, jelikož se neurčuje absolutní poloha snímaného objektu.

Jak již bylo zmíněno, za účelem co nejlepší přesnosti a použitelnosti jsou v této práci implementovány a testovány tři různé metody klasifikování, a

to pomocí neuronové sítě (NS) využívající expertní vyhodnocení, samoučící neuronové sítě a shlukové analýzy (SA). v této kapitole je vysvětleno, jakým způsobem tyto metodiky fungují a dále jsou představeny konkrétní modely NS a algoritmy SA.

2.2 Neuronové sítě

„*Umělá neuronová síť, často jen nazývaná neuronová síť, je matematický (nebo výpočtový) model, který je inspirován strukturou a funkcí biologických neuronových sítí v mozku.*“ [4] (překlad vlastní). Tyto sítě byly poprvé vytvořeny už v roce 1943 v [5], ovšem popularita NS vzrostla až v nedávných letech z důvodu vyšší výkonnosti hardwaru a lepší dostupnosti velkých datasetů.

Neuronové sítě jsou složeny z umělých neuronů, které jsou společně propojeny a poskytují si informace o dané úloze. Každý tento neuron může mít libovolný počet vstupů, ale pouze jeden výstup, který lze ovšem poslat i více neuronům. Všechny vstupy mají také danou váhu, která udává, jak moc je vstup „důležitý“ oproti ostatním.

Neurony jsou v neuronových sítích seskupeny do vrstev, které mohou být vstupní, skryté nebo výstupní. Každá NS obsahuje právě jednu vstupní vrstvu, za kterou je připojený libovolný počet skrytých vrstev, a nakonec jedna vrstva výstupní.

Vrstva vstupní pouze předává vstupy do zbytku sítě, což jsou ve valné většině skryté vrstvy, kde jsou provedeny veškeré výpočty. Výstupní vrstva se zase stará o transformaci vstupů na finální výstup.

NS tedy mohou mít libovolný počet skrytých vrstev, které obsahují libovolný počet neuronů a mohou být také různých typů. Tato různojakost umožňuje aplikaci na několik druhů úloh – predikce, klasifikace, shlukování nebo asociace. Také to ale znamená, že úkol vytvoření korektní struktury NS pro danou úlohy je pracný a složitý úkol.

Základní proces každého neuronu je transformace vstupů na výstup. Ten u neuronů začíná, jakmile má neuron k dispozici všechny vstupy. Nejdříve jsou tyto vstupy vynásobený váhami a sečteny. Dále je k této hodnotě přičtený stanovený práh, a nakonec je součet transformován pomocí aktivační funkce a poslán na výstup. Neboli:

$$f(\sum_{i=1}^{n} w_i \cdot x_i + o) \tag{2.1}$$

Aktivační funkce tedy rozhoduje, zda je neuron aktivován. Důvod použití je možnost nelineární transformace vstupu. Bez aktivační funkce by totiž NS byla schopna pouze té lineární. To by znamenalo, že by se NS nemohla naučit žádné komplexní úlohy bez ohledu na počet neuronů.

Existují tři typy aktivačních funkcí: binární, lineární a nelineární. Binární funkce mají práh. Pokud je přijatý vstup větší nežli tento práh, je neuron aktivován, pokud ne tak není neuron aktivován a výstup není poslán dál. Tato funkce tedy umožňuje posílání dvou typů výstupu, což například znamená, že se nedá použít pro klasifikací více než dvou tříd. Matematicky je tato funkce pospána takto:

$$f(x) = \begin{cases} 0 & x < 0 \\ 1 & x \geq 1 \end{cases} \tag{2.2}$$

Lineární aktivační funkce neprovádí žádnou změnu přijatému vstupu. Pouze vezme vstup a ten pošle dál v úplně stejné podobě. Nevýhoda použití těchto funkcí je jejich konstantní hodnota derivace. To způsobuje, že není možné použití zpětné šíření chyby, jelikož nemá derivace vztah k vstupu. Neboli:

$$f(x) = x \tag{2.3}$$

Nelineární funkce umožňují správně použití zpětné šířen chyby, jelikož má derivace funkce má vazbu k vstupu. Navíc umožňují použití několik vrstev neuronů, jelikož výstup bude nelineární kombinací vstupů. Jednou z nelineárních funkcí je Tanh:

$$f(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}} \tag{2.4}$$

Která je často používána ve skrytých vrstvách, jelikož je střední hodnota výstupů blízko 0, čímž zlehčuje učení pro další vrstvy.

Dále je potřeba představit aktivační funkci Softmax, která je při klasifikaci velmi často používána. Matematicky lze vyjádřit takto:

$$f(x_i) = \frac{exp(x_i)}{\sum_j exp(x_j)} \tag{2.5}$$

Tato funkce je používána pro určení pravděpodobnost, i do jaké třídy objekt patří při klasifikaci. Například při klasifikaci tří tříd by vrstva používající tuto aktivační funkci poslala tři výstupy, jejichž součet je roven 1. Jinak řečeno posílá tato funkce na výstup pravděpodobnosti každé třídy.

Další součástí každé NS je ztrátová funkce. Tato funkce počítá rozdíl mezi vyprodukovaným a správným výstupem NS. Dá se taky říct, že určuje, jak kvalitní jsou výsledky NS. Tyto funkce jsou velice důležité, jelikož určují, jak se mají váhy neuronů změnit.

Jedna z těchto funkcí je categorical crossentropy (kategorická entropie), která je používána u klasifikace několika tříd. Ztráta této funkce je vypočítána následovně:

$$f(x) = -\sum_{i=1}^{C} x_i \cdot \log \hat{x_i} \tag{2.6}$$

Kde x_i je i-tá hodnota vyprodukovaného výstupu, $\hat{x_i}$ je i-tá hodnota očekávaného výstupu a C je počet tříd.

Další často používaná ztrátová funkce je mean absolute error (střední absolutní chyba):

$$f(x) = \frac{1}{N}\sum_{i=1}^{N}|x_i - \hat{x_i}| \tag{2.7}$$

Kde N je počet prvků ve vektoru x, x_i jsou prvky vektoru x neboli vytvořený výstup a $\hat{x_i}$ jsou prvky vektoru předpokládaného výstupu. Tato funkce se používá například při regresy.

Při tvorbě struktury NS je potřeba také brát v potaz charakteristiku datasetu, především přítomnost či nepřítomnost očekávaných výstupů jednotlivých vzorků datasetu. Pokud nejsou výstupy součástí datasetu, musí se použít samoučící NS (NS bez učitele), které využívají takzvané učení bez učitele. Jestliže dataset výstupy obsahuje, může se pak implementovat i NS využívající expertní vyhodnocení (NS s učitelem), které využívají učení s učitelem. Pro klasifikaci pohybu se dají použít oba dva tyto typy a v následujících kapitolách jsou popsány.

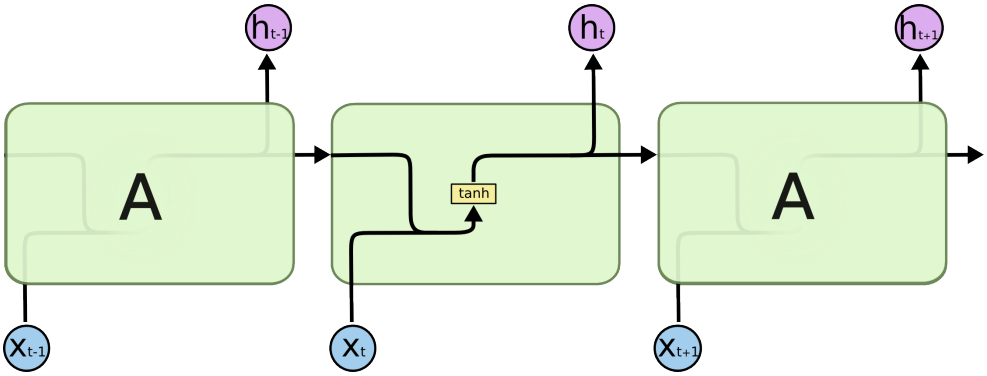
2.2.1 Neuronové sítě využívající expertní vyhodnocení

Tyto NS používají již zmíněné učení s učitelem. Tento styl trénování sítě používá zpětnou vazbu. NS nejdříve vypočítá výstup, poté ho porovná s očekávaným výstupem uvedeným v datasetu a určí svoji chybu pomocí ztrátové funkce. Na základě této chyby se poté NS změní s cílem lepšího výsledku. Obvykle tato metoda dává lepší výsledky nežli učení bez učitele, na druhou stranu je potřeba mít určené výstupy v datasetu.

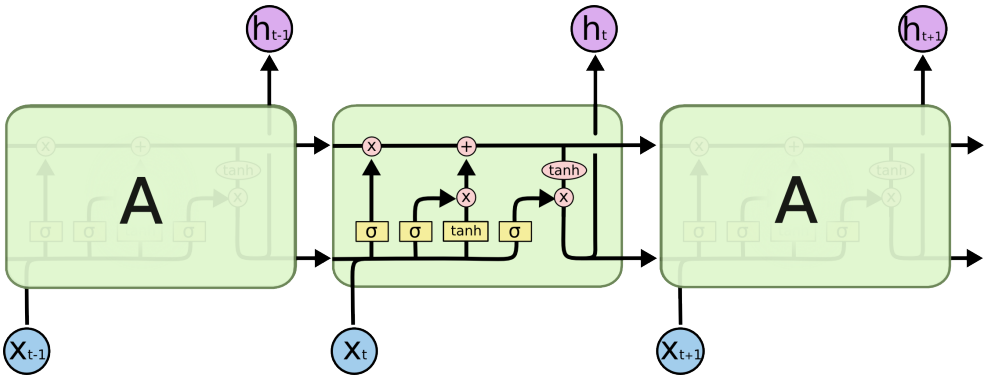
NS využívající expertní vyhodnocení jsou jednou z nejčastěji používaných metod pro klasifikaci lidského pohybu, jelikož dosahují velkých přesností. Na příklad v [6] je dosaženo na datasetu přesnost 96,88% nebo v [7] přesnost 97,6%. Toto časté využití znamená, že existuje mnoho prací, ze kterých lze čerpat. Pro lepší přehlednost lze tyto zdroje rozdělit do tří typů. Práce využívající umělé NS (dopředné NS), konvoluční NS nebo rekurentní NS.

Umělé NS jsou tvořeny skupinou neuronů na každé vrstvě, a co ji charakterizuje je její dopřednost – vstupy se zpracovávají pouze ve směru vpřed. Výhoda tohoto modelu je možnost naučení jakékoliv nelineární funkce. Tento typ NS je použitý například v [8], kde klasifikuje data získané pomocí analýzy hlavních komponent a dosahuje přesnosti 90,1% nebo také v [9], kde jsou získaná data z několika akcelerometrů kategorizována s přesností 83-90%.

Jako konvoluční NS jsou označovány sítě, které obsahují takzvanou konvoluční vrstvu. Tato vrstva extrahuje relevantní prvky z výstupu, což je například velmi užitečné u úloh, kde se zpracovává obraz nebo video. Proto jsou často využívané i při klasifikaci lidského pohybu, například v [10] je dosažená přesnost 92,1% a v [11] přesnost 95,7%.



Obrázek 2.1: Jednoduchá RNS [1]



Obrázek 2.2: LSTM architektura [1]

Rekurentní neuronové sítě (RNS) obsahují rekurentní spojení u neuronů. To umožňuje NS práci se sekvenčními daty, například s časovými řady, jelikož je vytvořen koncept paměti, který poskytuje NS informace o předešlých vstupech. Potencionální problém u mnoho úloh je zapomenutí předešlých vstupů, jelikož má většina rekurentních NS pouze krátkodobou paměť.

Long short-term memory (LSTM) je typ rekurentní NS, který tento problém řeší, a proto je velmi často používán při klasifikaci pohybu. Například v [12] byla dosažená přesnost 96% , v [13] 88.66-98,33% a v [14] 92,67%.

Hlavní vlastnost LSTM (a důvod proč řeší problém s dlouhodobou pamětí) jsou takzvané LSTM Cells (LSTM buňky). Všechny RNS mají podobu opakujících se modelů neuronové sítě. Například modelů, které používají jednoduchý Tanh vrstvu 2.1.

LSTM taky mají tuto podobu, ale nemají pouze jednu vrstvu, ale čtyři 2.2.

Co je v této architektuře důležité je vrchní horizontální čára, která zobrazuje Cell state. Cell state uchovává informace a chová se jako spoj mezi jednotlivými modely. LSTM umožňuje přidávat či odebírat informace u Cell state pomocí tří různých Gates (bran).

1. Forget Gate: Tato brána rozhoduje, jestli se z Cell state budou odstraňovat informace, případně jaké. Pomocí Sigmoidní funkce je vyprodukován výstup v rozmezí 0 a 1, kde 0 znamená, aby byla informace kompletně vymazána a 1 opak. Zde je matematický vzorec odpovídající této operaci:

$$f(t) = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{2.8}$$

Kde W_f je váha, h_{t-1} a x_t jsou vstupy a b_f je stanovený práh.

2. Input Gate: Určuje, jaké nové informace budou uloženy do Cell state. Nejdřív je pomocí Sigmoid funkce určeno jaké hodnoty budou změněny, a poté Tanh funkce vytvoří vektor potencionálních hodnot $\hat{C}_t$.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2.9}$$

$$\hat{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_c) \tag{2.10}$$

3. Output Gate: Tato brána provádí změnu v Cell state, kterou určily předchozí dvě brány.

$$C_t = f_t * C_{t-1} + i_t * \hat{C}_t \tag{2.11}$$

Kde C_{t-1}, reprezentuje předchozí Cell state.

Nakonec je vytvořený výstup h_t na základě Cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{2.12}$$

$$h_t = o_t * \tanh(C_t) \tag{2.13}$$

V této práci bylo pro metodu klasifikaci s využitím NS s učitelem použitá LSTM. Důvodem je paměť, která umožňuje přesnou klasifikaci, a také existence velkého počtu studií používající LSTM, z kterých se dá čerpat pro návrh metody.

2.2.2 Samoučící neuronové sítě

U této metody je využité učení bez učitele. Jinak řečeno, není při trénování očekávaný výstup znám, a při učení jsou očekávané výstupy buď vytvořeny (například autoencoder) nebo jsou hledány vzory v datech. Jak už bylo zmíněno, veliká síla těchto NS, je možnost použití na datasetech, které nemají označené vzorky. u určitých úloh je totiž velmi pracné nebo i nemožné určit očekávané výstupy. Samozřejmě je ale možné použít tento typ NS i na úlohy, které mají očekávané výstupy, právě za účelem nalezení různých vzorů v datech.

Narozdíl od NS využívající expertní vyhodnocení není tento typ NS často používán pro klasifikaci lidského pohybu, což je dáno tím, že ve valné většině těchto úloh není těžké dodat očekávaný výstup do naměřených dat. Většina těchto prací využívá typ NS jmenující se autoencoder.

Autoencoder (AE) je kodér, který se skládá z kodéru, který komprimuje data a dekodéru který dekomprimuje data, tak aby byly co nejvíce podobné své originální podobě. i když je tento typ NS uváděný jako samoučící NS (například v [15]), využívá i přesto učení s učitelem. Narozdíl od NS využívající expertní vyhodnocení si totiž autoencoder vytváří očekávané výstupy sám. Tyto výstupy jsou identické vstupům NS, jelikož se AE snaží co nejlépe zrekonstruovat vstupy.

Při učení AE jsou data nejdříve zkomprimována kodérem, poté dekomprimována dekodérem, a nakonec je změřena takzvaná chyba rekonstrukce, která udává, jak moc jsou dekomprimované data podobné své originální podobě (neboli podobné očekávanému výstupu). Tento cyklus se pořád opakuje s cílem co nejvíce snížit tuto chybu. Ideálním výsledkem AE, je tedy komprimace a dekomprimace dat bez jakékoliv změny.

V práci [16] byla představena architektura dosahující přesnosti 75,1%, která použila autoencoder. Nejdříve je naučená síť využívající autoencoder. Poté je vyjmut kodér, který transformuje data na menší dimenzi, a nakonec je použitý náhodný les pro klasifikaci těchto zakódovaných dat. Podobná struktura je použita i v této práci, s rozdílem klasifikátoru (zde je použitá jednoduchá vrstva perceptronů), a také použití takzvaného LSTM autoencoder (LAE). Tento typ AE je používán na sekvenční data (například v [17], [18], [19]), a proto je dobrou volbou pro tuto úlohu. LAE byl použit například v [20] a [17].

Motivace pro volbu použití této sítě, je možné nalezení vzorů v datech při použití komprimovaných dat pomocí kodéru, a také možné odstranění šumu, což by mohlo zlepšit přesnost klasifikace.

2.3 Shluková analýza

Shluková analýza je statistická metoda, jejímž cílem je rozdělit data do shluků tak, aby si prvky patřící do stejné skupiny byly bližší nežli objekty z jiných shluků. Používá se především pro klasifikaci, kde každý shluk reprezentuje třídu, do které může prvek patřit.

Dvě nejdůležitější vlastnosti u každé SA je použitý typ vzdálenosti a metoda nalezení shluků. Vzdálenost určuje, jak vypočítat podobnost objektů. Například: vzdálenost euklidovská $\sqrt{\sum_{i=1}^{p}(x_i - y_i)^2}$ a vzdálenost manhattanská $\sum_{i=1}^{p}|x_i - y_i|$, kde x a y jsou vektory (neboli objekty) o stejném počtu prvků.

Nejčastěji používané vzdálenosti používané pro časové řady jsou euklidovská a dynamic time warping (DTW). Obecně je euklidovská vzdálenost používána pro nalezení průměru všech objektů v shlucích. Ovšem tento přístup

není ideální pro časové řady, jak je například ukázáno v [21]. DTW tento problém ovšem nemá a je použitý například v [22], ale na druhou stranu je výpočetně náročnější.

Matematicky jde optimalizační problém DTW vyjádřit následovně:

$$DTW_q(x, x') = \min_{\pi \in A(x,x')} \left(\sum_{(i,j)\in\pi} d(x_i, x'_j)^q \right)^{\frac{1}{q}} \tag{2.14}$$

Kde π je sekvence párů $((i_0, j_0), \ldots, (i_{K-1}, j_{K-1}))$ délky K a $A(x, x')$ obsahuje všechny přístupné cesty. Aby cesta byla přístupná musí splňovat dvě podmínky.

1. Začátek a konec časové řady jsou spárované a platí: $\pi_0 = (0, 0)$, $\pi_{K-1} = (n-1, m-1)$.

2. Sekvence monotónně roste a platí: $i_{k-1} \leq i_k \leq i_{K-1} + 1$, $j_{k-1} \leq j_k \leq j_{K-1} + 1$

Metody shlukové analýzy lze dále rozdělit na hierarchické a nehierarchické. Hierarchické metody používá dříve nalezené shluky, které buď dělí (divizní přístup) nebo spojuje (aglomerativní přístup). Metody nehierarchické vždy rozdělí objekty do disjunktních shluků. Tyto shluky jsou poté postupně upravovány, ale disjunkce je pořád dodržována.

Veliká nevýhoda hierarchických metod je jejich výpočetní náročnost. v aglomerativním algoritmu je potřeba provést $n * (n-1)/2$ porovnání v každém kroku, v divizním 2^{n-1}, kde n je počet shluků. To znamená, že se nehodí pro velké datasety, a tím pádem ani pro tuto úlohu.

Co se týče nehierarchických metod pro klasifikaci pohybu, tak jsou nejčastěji používány K-means [23] a K-medoids [24]. Obě dvě tyto metody lze formalizovat jako optimalizační úlohu, která se snaží minimalizovat danou účelovou funkci. K-means i K-medoids se snaží minimalizovat celkový součet kvadrátů vzdáleností objektů od středů jejich shluků, neboli minimalizuje tuto sumu:

$$\sum_{i=1}^{k} \sum_{x\in S_i} ||x - \mu_i||^2 \tag{2.15}$$

Kde S_i jsou shluky, x jsou objekty a μ_i jsou středy shluků.

Rozdíl těchto metod je ve vypočítávání μ_i. K-means vypočítává středy shluků jako geometrické středy, zatímco k-medoids vybírá jako střed shluků jeden z prvků tohoto shluku. Tento objekt je vybrán, tak aby minimalizoval tuto sumu.

$$\sum_{x\in S_i} ||x - x_i||^2 \tag{2.16}$$

K-medoids algoritmus tedy funguje následovně:

1. Nejdříve je vybrán počet shluků jako N.
2. Náhodně se vybere N objektů, které jsou zvoleny jako středy.
3. Všechny objekty jsou přiřazeny k nejbližšímu středu.
4. Vypočtou se nové středy shluků podle výše zmíněné metody.
5. Opakování kroku 3 a 4 dokud nejsou nalezeny ideální středy a přiřazování objektů do shluků zůstává stejné.

Důsledkem je rozdílného vypočítávání středů shluků mezi K-means a K-medoids je větší robustnost K-medoids a jak je ukázáno v [25] je to daleko vhodnější způsob nežli K-means pokud je použito DTW. Tato výhoda byl důvod volby K-medoids využívající DTW.

Dále je potřeba zmínit se o problému vysoké dimenzionality dat při používání shlukové analýzy, který může způsobit projevení jevu prokletí dimenzionality. Vysoká dimenzionalita totiž způsobuje, že jsou všechny objekty velmi daleko od sebe a při shlukování jde jen ztěžka poznat, které body jsou blízko sobě.

Například pokud by každý objekt měl 80 příznaků s deseti možnými hodnotami, byl by počet všech možných objektů roven 10^80, což je rovno počtu všech atomů v pozorovatelném vesmíru. Shluková analýza by tedy musela v tomto případě porovnávat vzdálenosti například několika stovek atomů rozházených po celém vesmíru. Je asi zřejmé, že v tomto případě by se nenašli žádné shluky částic, které by si byly opravdu blízko.

Pro řešení tohoto problému je potřeba snížit počet dimenzí neboli redukovat počet příznaků. Například lze určité příznaky odebrat úplně, nebo několik příznaků spojit do jednoho a podobně. Ovšem pokud tato možnost není lze použít jednu z metod redukce dimenzionalit.

Těchto metod existuje několik – principal component analysis (metoda hlavních komponent), backward elimination, forward selection, low wariance filter, high correlation filter. Tyto metody využívají hypotézu variet, která říká, že většina reálných mnohorozměrných dat je ve skutečnosti rozprostřena podél variet menší dimenze.

Cílem těchto metod je tedy nalézt vhodné zobrazení z původního prostoru do nového prostoru menší dimenze. Metoda hlavních komponent (MHK) je často využívaná metoda redukcí dimenzionalit, například v [26] a [27], která využívá lineární projekce na lineární varietu.

Tato lineární projekce je vlastně ortogonální projekce, která funguje následovně: Mějme q dimenzionální podprostor V prostoru $\mathbf{R}^p$. Každý bod $x \in \mathbf{R}^p$ lze jednoznačně rozložit do součtu $x = v_x + u_x$, kde v_x je bod V a u_x je kolmý vektor na V. Poté se bod v_x nazývá ortogonální projekcí bodu x na podprostor V.

Otázkou je jak najít podprostor V pro různá q. V této metodě je daný požadavek, že pro každé q musí minimalizovat kvadratickou chybu projekce originálního datasetu na ten s q dimenzemi.

Pro získání tohoto minima je nedříve potřeba provést středování datasetu. Vznikne tak dataset kde každý nový bod je definován jako $y_i = x_i - \overline{x}$, kde $\overline{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$.

Poté se může říci, že podprostor V je tvořen prvními q vektory báze tvořené vlastními vektory $b_1, \ldots, b_p$ příslušejícím vlastním číslům matice $\frac{1}{n-1}\mathbf{X}^T\mathbf{X}$, kde $\mathbf{X}$ je vystředovaný dataset v maticové formě. Metoda MHK tedy tímto způsobem získá podprostor V, který má q dimenzí, a na té poté i jednotlivé ortogonální projekce, čímž se redukuje dimenze na q.

Kapitola 3

Návrh metod klasifikace pohybu

Cílem této práce je zpracování dat z 9DOF senzoru a následné klasifikace pomocí tří různých metod. v této kapitole se budeme zabývat implementací této aplikace napsané v jazyce Python.

3.1 Práce s daty

Konzolová aplikace pro komunikaci a přenos dat z pohybového senzoru mi byla poskytnuta společně se senzorem. Aplikace využívá Modbus protokol implementovaný knihovnou PyModus. Při běhu softwaru jsou snímaná data ukládána do bufferu, které jsou po ukončení aplikace uživatelem transformována do struktury DataFrame z knihovny Pandas a následně uložena na disk. Každý tento DataFrame obsahuje devět příznaků – tři příznaky zrychlení, magnetické orientace a úhlové rychlosti.

Pro trénování a testování přesnosti modelů byl vytvořen dataset obsahující 8 typů rehabilitačních pohybů. Tyto cviky byly vybrány ze dvou .pdf souborů (*LTV - pacient po fraktuře hlavičky radia po zhojení.pdf* a *Pacient s frakturou humeru akutní stádium - V.Scherrens LTV.pdf*), které jsou umístěny v adresáři společně s LaTeX zdrojovými kódy. Výchozí pozice, popisy a zdroje cviků jsou popsány zde:

1. Vzpřímený sed na židli; hlezenní, kolenní a kyčelní kloub svírá 90°; paže podél těla
   - Při nádechu vzpažit, při výdechu vrátit paže zpět
   - Fraktura hlavičky
2. Vzpřímený sed na židli; hlezenní, kolenní a kyčelní kloub svírá 90°; paže podél těla, v loketním 90°
   - Provádění střídání pronace a supinace (Dlaň dole, dlaň nahoře)

- Fraktura hlavičky

3. Vzpřímený sed na židli; hlezenní, kolenní a kyčelní kloub svírá 90°; paže ve vzpažení
   - Paže krčíme v loktech a jdou za hlavu, paže poté znova propínáme
   - Fraktura hlavičky
4. Vzpřímený stoj, od stěny stojíme na vzdálenost délky paže, levá paže je v předpažení a dlaň opřená o stěnu
   - Pravá paže se z flekčního postavení propíná a dlaň směřuje podél těla
   - Fraktura hlavičky
5. Vzpřímený sed na židli; hlezenní, kolenní a kyčelní kloub svírá 90°; hlava v prodloužení páteře, paže podél těla
   - Pravou paži vzpažíme a provedeme úklon k levé straně
   - Fraktura hlavičky
6. Vleže na zádech, DK pokrčené v kolenou, ruce podél těla
   - Ruce předpažit a ohnout v loketním kloubu, prsty položit na ramena, paže vrátit do původní polohy stejným pohybem
   - Fraktura humeru
7. Vleže na zádech, DK pokrčené v kolenou, ruce podél těla
   - Ruce upažit, poté vzpažit a vrátit do původní pozice
   - Fraktura humeru
8. Výchozí poloha ve stoji, ruce podél těla
   - Pravou ruku upažíme a "kreslíme" malé kruhy do vzduchu
   - Fraktura humeru

Dále byl také nasnímán pohyb devátý, který reprezentuje klidový stav senzoru.

Jelikož byl v této práci použitý pouze jeden senzor umístěný na pravém zápěstí viz 3.1, zúžil se výběr potencionálních cviků na klasifikaci. Není totiž možné kategorizovat cviky, při kterých se nehýbe pravým zápěstí, jelikož by senzor neposkytoval žádné relevantní informace.

Dále bylo také potřeba vybrat cviky, kterou si jsou dostatečně odlišné. Pokud by totiž dva pohyby byly moc podobné, co se týče pohybu pravého zápěstí, tak by mohlo dojít k špatné klasifikaci.

Obrázek 3.1: Poloha umístění senzoru

Zde vybraných 8 cviků tyto dvě požadavky splňují. Jedná se o pohyby pravého zápěstí a paže, a zároveň si jsou dostatečně odlišné, aby si je klasifikátor nespletl, jak je také ukázáno v maticích záměn 4.1 a 4.2.

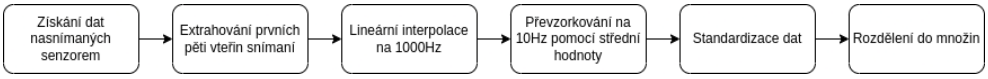
Pohyby byly nasnímány od 12 různých osob, za účelem naučení správné klasifikace při různé rychlosti a způsobu provedení cviku, a také při rozdílné velikosti pacienta. U každé osoby byly nasnímány tři vzorky každého cviku dlouhé jednu minutu. Celkově bylo tedy nasnímáno 36 různých vzorků pro každý cvik.

Pro předání dat do klasifikátorů je z minutového snímání vždy vybrán právě jeden K vteřinový úsek z prostředku intervalu. To znamená, že tento interval může začínat v půlce cviku nebo na konci, což umožňuje klasifikátoru provést korektní real-time klasifikaci.

První překážka implementace byl rozdílný počet řádků stejně časově dlouhých vzorků. Při selekci například tří sekundových intervalů z dvou minutových snímání, se často stalo, že tyto vzorky měli odlišný počet řádků. To bylo způsobeno nekonzistentní frekvencí senzoru. I když má totiž senzor 100 Hz, tak nebyly všechny řádky přesně 10 ms po sobě. Pro vyřešení problému bylo potřeba využít změny frekvence signálu neboli převzorkování.

Nejdříve je na data aplikována lineární interpolace, který změní frekvenci na 1 Hz. Při této metodě jsou chybějící hodnoty vypočítány pomocí lineárních křivek, které spojují existující hodnoty. Tím se zaručí, že pokud byly vzorky stejné časové délky, budou mít i stejný počet řádků, jelikož všechny řádky budou přesně 1 ms od sebe.

Ovšem v této úloze nebylo potřeba mít takto velkou frekvenci. Pro klasifikaci lidského pohybu je ideální frekvence 10 Hz, jelikož nemá podstatný



Obrázek 3.2: Proces transformace dat

vliv na výsledky klasifikace a zrychluje běh aplikace [28], [29]. Dalším krokem tedy je data převzorkovat na 10 Hz, kde jsou řádky vypočítány jako střední hodnota příslušných původních řádků.

Poté bylo také potřeba data standardizovat, aby model mohl lépe interpretovat hodnoty v datasetu. Přeškálování dat pomocí standardizace způsobí změnu střední hodnoty na 0 a směrodatné odchylky na 1. Standardizace probíhala na každém příznaků odděleně, aby byly vnímány stejně důležitě, a to pomocí tohoto vzorce:

$$x_n ew = \frac{x_i - \overline{x}}{s} \tag{3.1}$$

Nakonec je dataset rozdělen do 60% pro trénovací množinu, 13,3% pro validační množinu a 26,6% pro testovací množinu.

Celý proces manipulace s daty je zobrazen na diagramu 3.2.

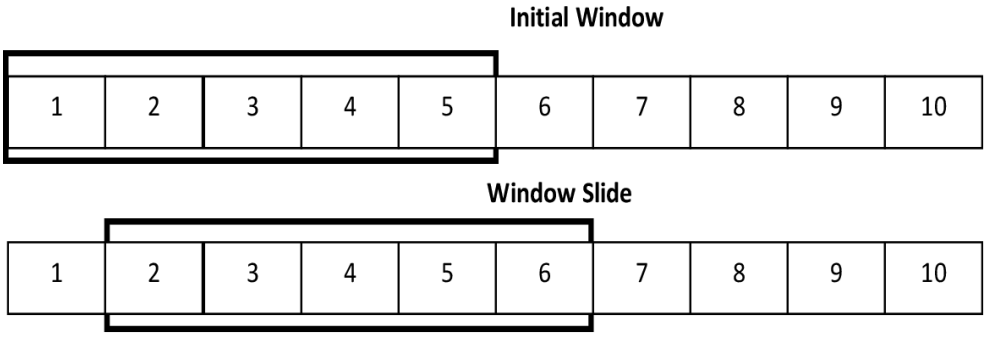
Dále bylo potřeba zvolit ideální délku intervalů, které jsou předávány na klasifikaci. Zvolená délka pohybu nesmí být moc krátká, aby se nestalo, že nebude zachycen celý pohyb, ale zároveň nesmí být moc dlouhá, aby při real-time klasifikaci nebyla dlouhá prodleva mezi dokončením pohybu a výsledné kategorizace.

Tato délka byla určena pomocí statistického vyhodnocení různých délek intervalů v kapitole 4, kde bylo zjištěno, že ideální délka je 5 sekund pro NS s učitelem, 4 sekundy pro NS bez učitele a 4 pro K-medoids, a to z důvodu dosažené velké přesnosti, a také relativně krátké délce.

Poté bylo potřeba vytvořit novou metodu komunikace, kterou lze použít na real-time klasifikaci. Nelze totiž použít způsob jako při vytváření datasetů, jelikož je při real-time klasifikaci potřeba kontinuální a okamžité posílání dat na kategorizaci při běhu aplikace. Proto byla implementována nová metoda využívající funkčnost vláken, kde se na hlavním vláknu se data klasifikují a na druhém probíhá komunikace se senzorem.

Data získaná se senzoru jsou umístěna do dvou bufferů. První buffer je využíván stejně jako u předchozí metody komunikace – data jsou do bufferu vkládána a na konci jsou uložena na disk. V druhém jsou však data ihned transformována do DataFrame a pomocí struktury queue sdílená s hlavním vláknem, který queue kontroluje, a jakmile obsahuje dostatečný počet dat je zahájena klasifikace a následně jsou data odstraněna.

Ovšem v případě, že jsou všechna kategorizovaná data odstraněna, může nastat situace, kdy se v první polovině úseku provádí jiný pohyb nežli v druhé.



Obrázek 3.3: Sliding window [2]

v tomto případě by model nebyl schopný klasifikovat oba dva pohyby, ale pouze jeden. Pro vyřešení tohoto problému byla v této práci použita technika sliding window (posuvné okno).

Tato technika funguje následovně. Mějme okno konstantní délky K, v tomto případě je K rovno zvolené délky intervalu, což je 5 vteřin. Ze senzoru přichází postupně datový tok. Toto okno je postupně posouváno datovým tokem neboli vždy zachytí interval 5 vteřin, poté je posunuto o fixní délku T a data „vlevo“ jsou odstraněna. Tento proces je ukázán zde: 3.3, kde je okno posunuto o délku 1.

Hlavní otázkou je jaké zvolit T pro ideální průběh klasifikace. Pokud je T zvoleno příliš malé, zvedne se výrazně požadavek na hardware. v opačném případě se zase může stát, že pohyb nebude klasifikován. Zde bylo T zvoleno jako polovina snímaného intervalu, neboli 2,5 vteřin, stejně jako v [11].

3.2 Neuronové sítě využívající expertních hodnocení

Jeden z důvodů pro volbu Python jako programovacího jazyka je velká dostupnost knihoven užitečných pro ulehčení implementace neuronových sítí. Keras je knihovna zaměřená na hluboké učení a obsahuje hned několik modelů pro implementaci neuronových sítí. Při implementaci aplikace byl využit model Sequential, který umožňuje tvorbu sítě postupným přidáváním vrstev.

Při tvorbě jakékoliv neuronové sítě je hlavní otázka, jaký model vybrat neboli jaké zvolit vrstvy a jejich aktivační funkce a jak je propojit. Jako model neuronové sítě jsem zvolil Bidirecional Long short-term memory (BLSTM) představený v [13] a v [14]. BLSTM obsahuje dvě LSTM architektury (představeny v 2.2.1), jedna přijímá vstup v dopředném směru a druhé ve směru vzad. To umožňuje modelu lépe pochopit kontext dat, čímž se zvyšuje přesnost při dynamické klasifikaci.

Dále byla také použita vrstva Dropout se standardní 0.5 mírou odpadnutí, která omezuje problém přeučení modelu [30] . Síť je ukončena dvěma vrstvami Dense, které jsou jednoduché plně propojené vrstvy, které dokončují klasifikaci, a to pomocí aktivační funkce Softmax 2.5. Ostatní Dense a BLSTM vrstvy obsahují aktivační funkci Tanh 2.4, z důvodu již zmíněného ulehčení učení pro skryté vrstvy.

Jako ztrátová funkce neboli funkce, která udává jak dobrý je výsledek sítě, byla použita kategorická entropie 2.6, jež je standartně používána pro klasifikaci více než dvou tříd.

Problém snad každé neuronové sítě je možnost přeučení (overfitting). Při přeučení sítě, sice dosahuje model dobré přesnosti na poskytnutých datech, ale pokud jsou předaná nová data na klasifikaci, tak síť provede klasifikaci s špatnou přesností. Použitá vrstva Dropout sice omezuje tento problém, ale neřeší ho.

Jednou z možností je zvolit limit epoch. Epocha značí cyklus, při kterém projde NS trénovací dataset a obnoví se váhy NS. Při omezení epoch na malý počet se zamezí přeučení sítě, ovšem není vhodné zvolit tento limit moc malý, protože pak by se síť nestačila naučit vůbec, což by také způsobilo špatnou přesnost. Bylo by tedy potřeba model otestovat na různých hodnotách epoch, a učit právě tu nejlepší.

V této práci bylo ovšem provedeno jiné řešení, a to pomocí metody Early Stopping (ES). Při této metodě lze nastavit libovolně velký počet epoch, jelikož ES učení zastaví samo. ES totiž kontroluje při každém epochu, zdali se model zlepšuje na validačním datasetu. Pokud se NS přestane zlepšovat, učení je přerušeno.

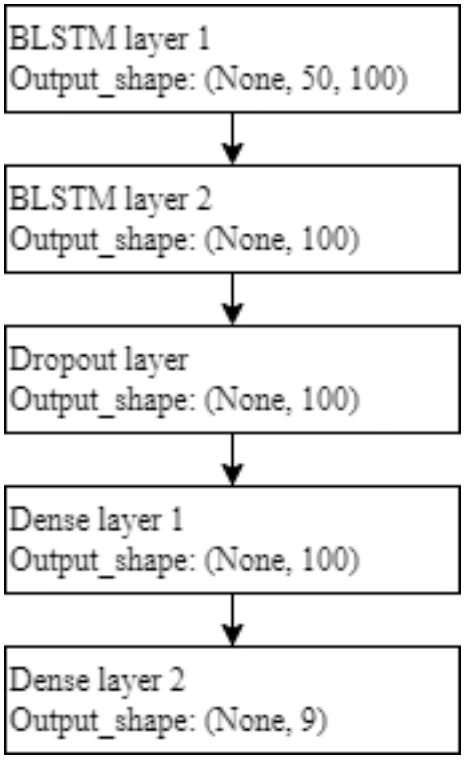
Důvod proč tato metoda funguje je použití validačního datasetu a ne testovacího. Pokud by se totiž použil testovací dataset, neměl by ES žádný efekt, co se týče omezení přeučení, jelikož by testovací dataset nebral jako nová data.

U ES je potřeba určit dva důležité parametry. Monitor, který určuje, jestli ES sleduje přesnosti, či ztrátovou funkci modelu a patience. Patience udává kolik epoch má ES počkat při zastavení růstu (nebo klesání) sledované hodnoty.

Zde bylo zvoleno monitorování ztrátové funkce, jelikož lépe reprezentuje, jak kvalitní síť doopravdy je oproti jednoduché přesnosti. A jako patience bylo zvoleno 20, jelikož umožňuje síti, aby opravdu našla co nejlepší přesnost.

Pro co nejlepší přesnost modelu bylo potřeba dále určit hyperparametry. Jedním z těchto parametrů je jaký optimalizátor vybrat a jaká bude jeho learning rate (rychlost učení). Stejně jakov [14], byl zde použitý optimalizátor Adam s learning rate 0.001. Tento optimalizátor je algoritmus, který optimalizuje techniku gradientní sestup.

Nakonec byl optimalizován počet vrstev BLSTM (L) a počet neuronů v každé vrstvě (N). Často se totiž stává, že není vhodné mít pouze jednu BLSTM vrstvu, ale je lepší jich mít více. Proto byly otestované různé kombi-



Obrázek 3.4: Architetura BLSTM NS

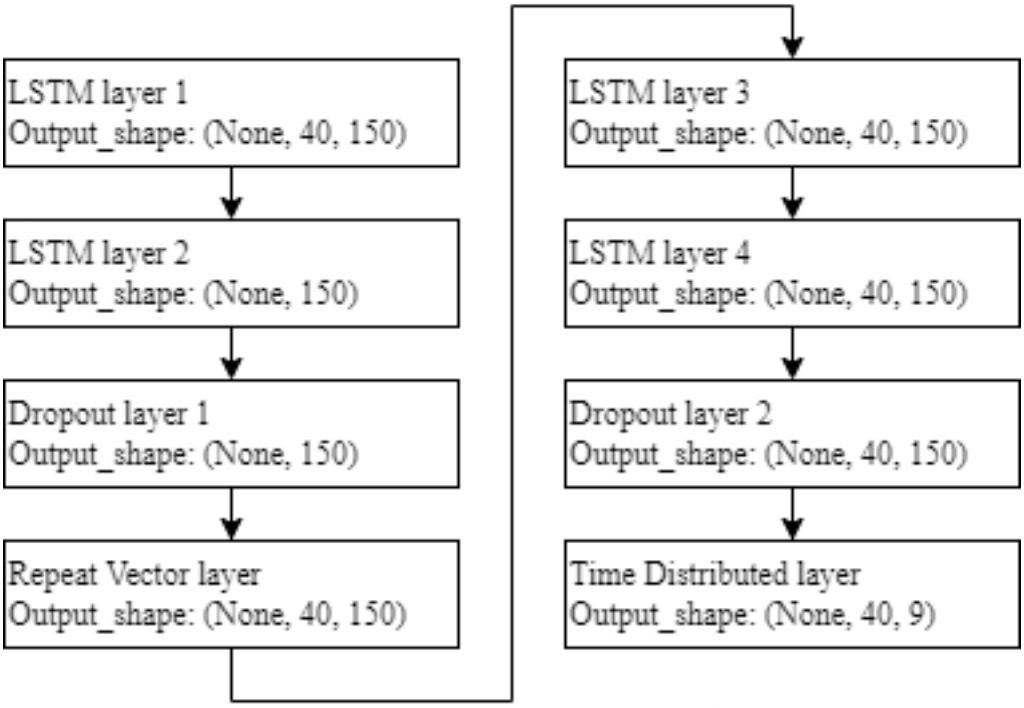
nace L a N viz kapitola 4.1, kde bylo zjištěno, že nejlepší kombinace je $N = 2$ a $L = 100$.

Pro implementaci více BLSTM vrstev za sebou bylo ovšem potřeba nastavit, aby BLSTM vrstvy, jejichž výstup pokračoval do další BLSTM vrstvy, posílaly nejenom poslední Cell state, ale celou sekvenci. To je potřeba, už jen kvůli tomu, že by se jinak zmenšila dimenze a další BLSTM vrstva by s takovým vstupem nemohla pracovat. Tento problém byl vyřešený nastavením parametru Return_Sequence, který je dostupný u poskytovaných vrstev knihovnou Keras, na True.

Finální diagram neuronové sítě je zobrazen na 3.4.

3.3 Samoučící neuronové sítě

Zpočátku bylo úmyslem vytvořit metodu klasifikace, která bude využívat pouze samoučící neuronovou síť, ovšem nebyl nalezený žádný model, který by se mohl použít na tuto úlohu a zároveň dosahoval dobrých výsledků. Jedna z možností bylo použít Self-organizing map (SOM), která bez učitele vytváří shluky, podobně jako například K-means. Tento model je ale převážně využíván pro detekci anomálií v datech a při klasifikaci by nejspíše dosahoval



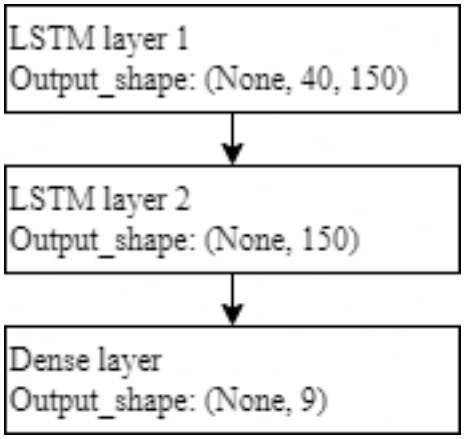
Obrázek 3.5: Architektura LAE

špatných výsledků. Proto jsem se rozhodl použít kombinaci samoučící NS – Long short-term memory autoencoder (LAE) – a NS s učitelem.

Jak bylo zmíněno výše LAE je kodér, který využívá architekturu kodéru a dekodéru LSTM pro komprimování dat pomocí kodéru a dekódování těchto dat do originální podoby pomocí dekodéru. Narozdíl od standardního AE zachycuje, podobně jako LSTM, LAE časové závislosti, což je u tohoto problému potřeba.

Jednoduchá struktura LAE, kde jsou použité pouze dvě LSTM vrstvy je zobrazena na 3.5. Stejně jako u první metodiky je zde použita vrstva Dropout pro omezení přeučení. První vrstva LSTM je kodér LAE neboli část sítě, která komprimuje data. Vrstva RepeatVector a druhá LSTM je zase dekodér LAE, kde RepeatVector přidá navíc dimenzi, aby data mohla být znovu použita v LSTM vrstvě.

Jako poslední vrstva je použitá Time Distributed Dense vrstva, která zajistí vazbu vstupu a výstupu jednu ku jedné. Což je žádané, jelikož chceme jako výstup originální nekomprimovaná data. Všechny vrstvy Dense a LSTM této architektury obsahují aktivační funkci Tanh 2.4. Ztrátová funkce modelu je střední absolutní chyba 2.7, která udává, jak velká chyba nastala při rekonstrukci dat.



Obrázek 3.6: Architektura kodéru s klasifikační vrstvou

Samozřejmě je zde řešen problém klasifikace, a proto je potřeba NS upravit. z LAE je tedy vyjmut kodér, na který je napojená vrstva perceptronů, která provádí klasifikaci. Diagram této sítě je zobrazený zde 3.6.

Vytvořený model pro klasifikaci tedy funguje následovně. Nejdříve je naučen model LAE. Poté je vytvořen model nový, který se skládá z naučeného kodéru vyjmutého z LAE a jedné vrstvy neuronů použité s aktivační funkcí softmax pro klasifikaci. i zde obsahují ostatní vrstvy funkci Tanh 2.4. Model druhý je poté naučený s použitím učitele a ztrátovou funkcí kategorická entropie 2.6, jelikož se už jedná o klasifikační model.

Při učení druhého modelu bylo také potřeba zabránit učení kodér části NS. To znamená, že se učila pouze poslední vrstva a váhy u kodéru zůstaly stejné. Pokud by tento krok nebyl proveden, chovala by se NS jako standardní LSTM síť s učitelem, jelikož by nebyly zachovány váhy u kodéru. Tuto funkčnost umožňuje knihovna Keras nastavit pomocí parametru trainable na False.

Použití kodéru v modelu znamená, že model klasifikuje komprimovaná data. To může mít jak pozitivní, tak i negativní efekt na přesnost klasifikace. Při komprimaci se mohou například ztratit potřebné informace pro klasifikaci. Na druhou stranu může komprimace odstranit neužitečné části dat, například šum.

Stejně jako u NS s učitelem byla použita metoda Early Stopping k zamezení přeučení, a to při učení jak LAE, tak i finální NS. Postup volby parametrů byl také stejný jako u NS s učitelem. Jako optimalizátor byl zvolen Adam s learning rate 0.001.

Dále byl podobně v jako předchozí kapitole určen optimální počet vrstev LSTM a neuronů v nich na 2 a 150 viz kapitola 4.2.

Jak si lze všimnout architektura je skoro stejná jako u NS s učitelem. Hlavní rozdíl mezi těmito metodami je způsob, jakým byly NS trénovány neboli jak se nastavovali jejich váhy.

3.4 Metody shlukové analýzy

Jako shlukový algoritmus byl vybrán K-medoids implementovaný pomocí scikit-learn-extra, což je rozšíření známe knihovny scikit-learn, která se zabývá strojovým učením.

Typ implementované vzdálenosti je dynamic time warping (DTW), který byl vytvořen specificky pro měření podobností časových řad. Algoritmus DTW pro dva objekty a a b je popsán zde 1.

Algorithm 1 DTW algoritmus pro dva objekty

$A \leftarrow a.size$
$B \leftarrow b.size$
$C \leftarrow cumdist(a, b)$
$M \leftarrow np.ones((A + 1, B + 1)) \cdot \infty$
$M[0, 0] \leftarrow 0$
for a_i in a **do**
  for b_i in b **do**
    $MIN \leftarrow \min(M[a_i, b_i + 1], M[a_i + 1, b_i], M[a_i, b_i])$
    $M[a_i + 1, b_i + 1] \leftarrow C[a_i, b_i] + MIN$
  end for
end for
return $M[A, B]$

Volbou k-medoids a DTW ovšem vznikl problém vysoké komplexity. K-medoids má komplexitu $\mathcal{O}(N^2 * K * T)$ oproti k-means komplexitě $\mathcal{O}(N * K * T)$, kde N je počet vzorků, T počet iterací a K počet shluků. Navíc je i komplexita metriky DTW – $\mathcal{O}(N * M)$ – větší než například u euklidovská vzdálenosti – $\mathcal{O}(\log \min N, M)$.

Tato vyšší komplexita znamenala výrazné zpomalení při vytváření modelu. Narozdíl od modelu neuronové sítě s učitelem, kde se síť naučila do deseti minut, se tento model učil skoro celý den. Toto byl také důvod proč bylo v této práce nakonec použitý DTW algoritmus implementován knihovnou tslearn, který byl daleko rychlejší nežli DTW algoritmus, který byl napsán bez pomocí knihoven. To mělo za účinek, že čas učení byl v řádu minut a byl tedy přijatelný.

Pro použití algoritmu K-medoids bylo potřeba vzorky, které byly ukládány v 2D podobě, transformovat do vektoru. To bylo uděláno jednoduše pomocí funkcionalit numpy knihovny, ovšem po transformaci měl každý vzorek $5 * 10 * 9 = 450$ příznaků, jelikož je úsek dlouhý pět vteřin, data jsou v 10 Hz a je 9 příznaků. Tento obrovský počet příznaků znamenal, že se zde objevilo prokletí dimenzionality zmíněné v kapitole 2.3.

Vyřešení tohoto problému je jednoduché ale ne snadné; je potřeba snížit dimenzionalitu. Jedna z možností by bylo snížit počet příznaků, které jsou používány ze senzoru. Ovšem v této úloze je nezbytné mít k dispozici infor-

mace o akceleraci, úhlové rychlosti i magnetické orientace, a to ve všech třech směrech.

Dále by šlo snížit frekvenci z 10 Hz, například na 1 Hz, čímž by se počet příznaků snížil na 45. To by znamenalo, že by prokletí dimenzionality mělo mnohem menší efekt, ale na druhou stranu by bylo ztraceno mnoho informací ohledně prováděného pohybu.

Řešením v této práci bylo použití metody redukce dimenzí, a to metodu hlavních komponent (MHK) představenou v kapitole 2.3. MHK umožňuje redukci dimnezionality, při které se sice samozřejmě některé informace ztratí, ale je to daleko lepší řešení nežli zmenšení frekvence.

Tato metoda byla implementována pomocí knihovny scikit-learn a počet komponent byl určen pomocí určení minimálního možného rozptylu (neboli kolik z původních informací se zachovalo) na $0,8$. Tento parametr byl zvolen jako kompromis mezi zachováním informací a počtu vytvořených komponent. Je totiž žádané mít komponent co nejméně, ale zároveň se nesmí ztratit příliš mnoho informací.

U shlukových algoritmů je dále vždy nutné udat počet shluků N, které chceme najít. Na tento problém dává smysl určit N jako počet typů pohybů, které jsou klasifikovány. Ovšem s tímto přístupem spoléháme na to, že vzorky rozdílných typů pohybů jsou podle DTW dostatečně vzdálené, a zároveň že jsou vzorky stejných pohybů dostatečně blízko.

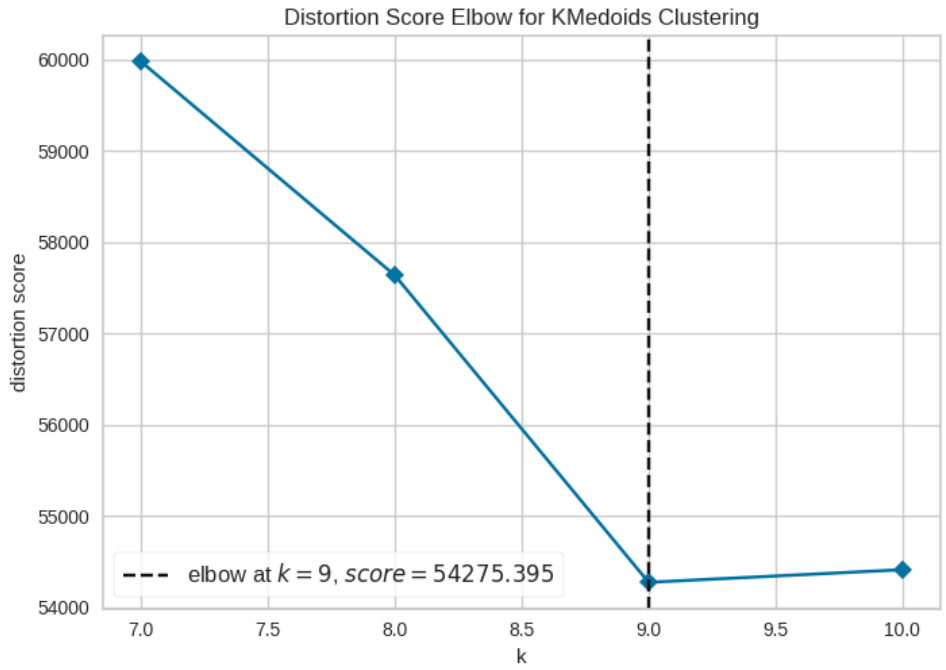
Může se totiž stát, že bude tento model vnímat dva pohyby pomocí metriky DTW natolik blízké, že je nepůjde rozdělit na dva shluky s přijatelnou přesností, nebo na druhou stranu může model vnímat vzorky jednoho pohybu natolik vzdálené, že je rozdělí do shluků dvou. Pro vyřešení problému je tedy nutné zjistit tento ideální N počet shluků.

Knihovna Yellowbrick poskytuje třídu KElbowVisualizer, pomocí které lze najít optimální N. Tato třída používá metodu elbow neboli loket. Při této metodě je model trénován s různým počtem shluků a vždy je zaznamenaný příslušný distortion score (DS). Na 3.7, je zobrazen tento graf.

Jak si lze všimnout, DS se zpočátku velmi rychle snižuje, ale s rostoucím počtem shluků se klesání zpomaluje. Pro určení N je potřeba najít právě ten počet shluků, po kterém se klesání DS výrazně zpomalí neboli najít takzvaný loket v grafu. Ten je vyznačený v grafu černou čárkovanou čarou a je roven devíti.

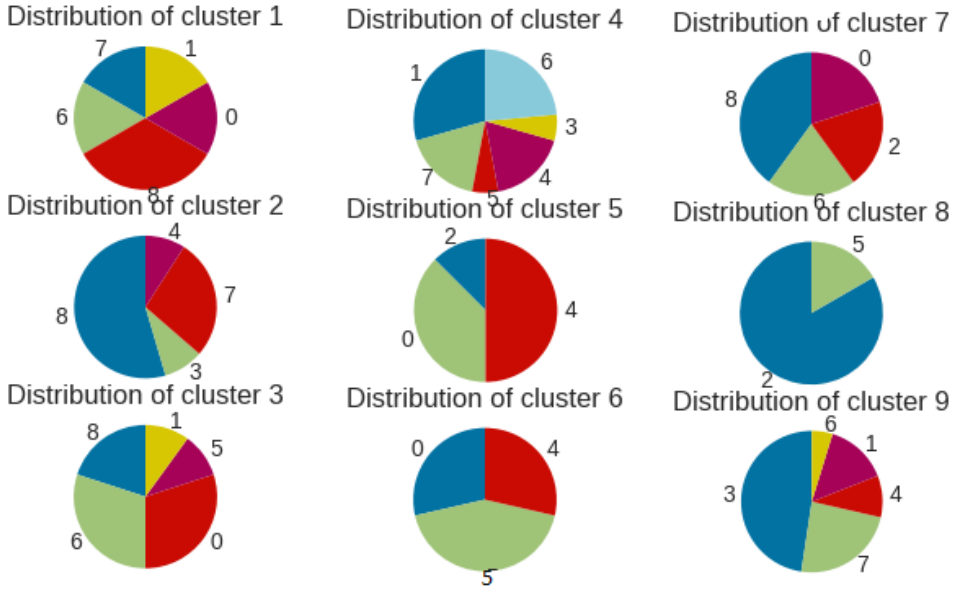
Nakonec je potřeba určit, jak jsou vzorky v shlucích umístěny. Neboli zjistit, jestli vzorky v shlucích jsou převážně jednoho typu pohybu a případně kterého, aby se při klasifikaci pohybů mohlo vypsat o jaký typ pohybu jde, a ne jenom číslo shluku.

Na 3.8 jsou vykresleny grafy, které zobrazují reprezentaci jednotlivých pohybů v shlucích po natrénování modelu. Ke každému shluku je dále přiřazen typ pohybu, a to podle toho které třídy pohybu obsahoval shluk nejvíce vzorků. Bohužel se tedy může stát, že pro jeden pohyb bude reprezentovaný



Obrázek 3.7: KElbow graf

vícero shluky anebo naopak, jeden pohyb nebude reprezentovaný vůbec, a tím pádem nebude moct být klasifikován.



Obrázek 3.8: K-Medoids grafy reprezentující jednotlivé pohyby

Kapitola 4

Aplikace a metody testování navržených metod

Naučené modely lze aplikovat na data dvěmi způsoby. V prvním způsobu se klasifikuje uložený .csv soubor na disku. V tomto případě je soubor rozdělen na 5 vteřinové úseky, které jsou postupně klasifikovány a výsledky jsou vypsány.

Druhá možnost je použití real-time klasifikace. Jak již bylo zmíněno v kapitole 3.1, dostává v tomto případě model data instantně. Jakmile dostane dostatečný počet dat na klasifikaci, učiní tak a přesune se pomocí techniky sliding window na nová data.

Pro testování metod při klasifikaci byl použitý vytvořený dataset popsaný v kapitole 3.1. Jak již bylo zmíněno, vždy bylo z minutového intervalu vyjmut právě jeden 5vteřinový úsek z prostředka. To má za následek, že úsek může začínat v jakékoliv fázi cviku neboli je simulována real-time klasifikace.

Dá se tedy říct, že výsledky na tomto datasetu, určují i přesnost při real-time klasifikaci, zanedbáme-li možnost, kdy se při real-time klasifikaci klasifikují v jednou úseku dva typy cviků. Bohužel, tento scénář nebyl testován, jelikož je v tomto případě těžké hodnotit přesnost modelu.

Jak je napsáno v kapitole 3.1, bylo nejdříve potřeba určit optimální délku intervalu. v tabulce 4.1 jsou vyhodnoceny různé délky intervalů u jednotlivých metod, kde počet BLSTM vrstev u NS s učitelem byl 2, počet LSTM vrstev v kodéru a dekodéru byl také 2 a počet neuronů v nich 175.

Vypsaná čísla jsou střední hodnotou přesností při dvaceti různých, ale daných rozdělení dat do trénovací, validační a testovacích množin. Všechny metody a kombinace tedy pracovaly se stejnými množinami dat, které ovšem byly v každém z dvaceti měření rozdílné.

Dohromady bylo na každé kombinaci provedeno 1 940 klasifikací – v testovací množině bylo vždy 97 vzorků a testování probíhalo dvacetkrát.

Délky intervalů byly zvolené s ohledem na jejich délku – není vhodné, aby při real-time klasifikaci byl úsek moc dlouhý. Délka 12 a 15 sekund byly

Tabulka 4.1: Testování délky intervalů

Typ NS	1s	2s	3s	4s	5s	6s	8s	12s	15s
BLSTM	82,78	84,48	89,33	90,93	**94,35**	94,06	94,09	94,18	94,09
LAE	68,60	81,34	90,93	**91,29**	84,48	78,25	79,02	74,95	76,85
K-medoids	28,66	39,80	36,29	**43,60**	34,40	43,58	43,89	30,72	32,99

Tabulka 4.2: Testování hyperparametrů BLSTM NS

# vrstev/neuronů	100	125	150	175	200	250	300
1	93,20	93,35	93,40	92,89	93,55	93,61	94,64
2	**95,87**	94,90	95,82	93,35	94,74	95,10	**95,87**
3	95,10	94,90	94,28	95,56	94,17	93,30	92,68

otestovány pro určení zdali, by se přesnost zlepšila, ale nebylo by vhodné je použít i v případě nejlepší přesnosti kvůli příliš dlouhé prodlevě kategorizace.

V dalších třech podkapitolách jsou testovány jednotlivé metody pomocí stejného postupu jako při testování délky intervalů pomocí provedení 1 940 klasifikací. Dále je také u každé metody zobrazena matice záměn (confusion matrix).

Matice záměn udává, jaké třídy byly klasifikovány správně a jaké špatně. v této úloze mohou tedy ukázat, jestli nějaké pohyby vnímá klasifikátor jako podobné.

4.1 Neuronové sítě využívající expertních hodnocení

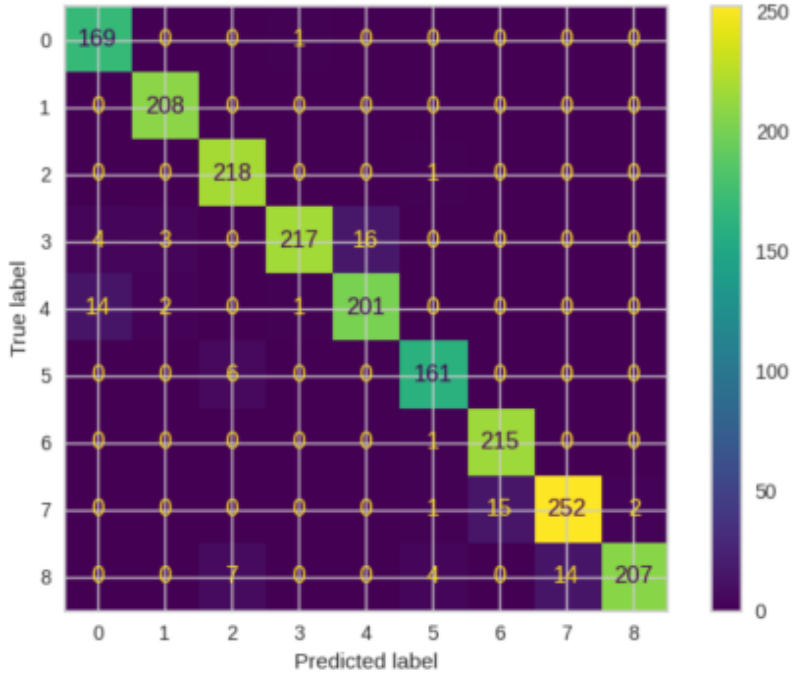
U této metody bylo potřeba testovat různé kombinace počtu BLSTM vrstev (L) a neuronů v nich (N) pro dosažení co největší přesnosti. Proto bylo stejně jako v [14] otestovány hodnoty L (1, 2, 3) a N (100, 125, 150, 175, 200, 250, 300) viz tabulka 4.2.

Nejlepší kombinace je tedy $L = 2$, $N = 100$, která sice dosahuje stejné přesnosti s $L = 2$, $N = 300$, ale je méně výpočetně náročná. Tato NS s učitelem dosáhla na klasifikaci 1 940 pohybů přesnosti 95,87% se směrodatnou odchylkou 1,85.

Nakonec byla také vytvořena matice záměn viz 4.1.

4.2 Samoučící neuronové sítě

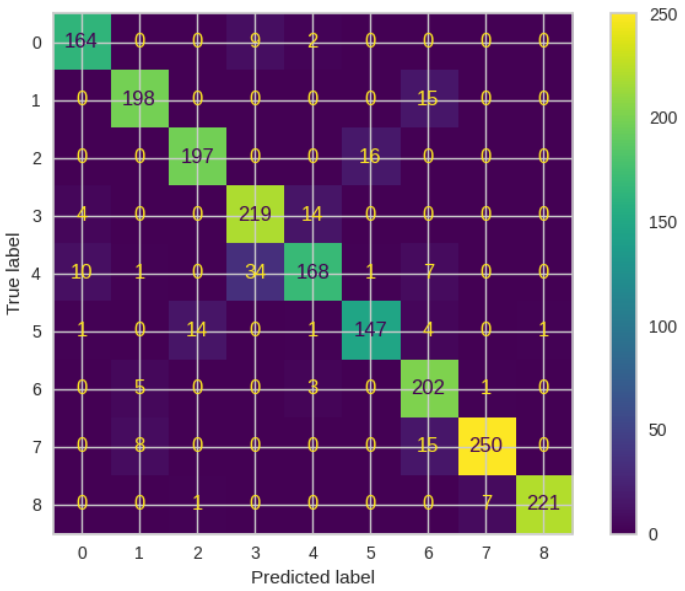
Způsob a počet testování byl zde stejný jako u NS s učitelem viz tabulka 4.3. Zde je ovšem nejlepším parametrem $L = 2$ a $N = 150$, které dosahují přesnosti 91,90% se směrodatnou odchylkou 2,82.



Obrázek 4.1: Matice záměn BLSTM NS

Tabulka 4.3: Testování hyperparametrů LAE NS

# vrstev/neuronů	100	125	150	175	200	250	300
1	82,06	80,22	83,71	89,01	88,24	88,86	90,10
2	90,30	89,32	**91,90**	91,29	90,92	91,30	91,75
3	89,28	89,69	89,28	90,72	91,34	90,50	89,78



Obrázek 4.2: Matice záměn LAE NS

Tabulka 4.4: Absence pohybů při volbě shluků

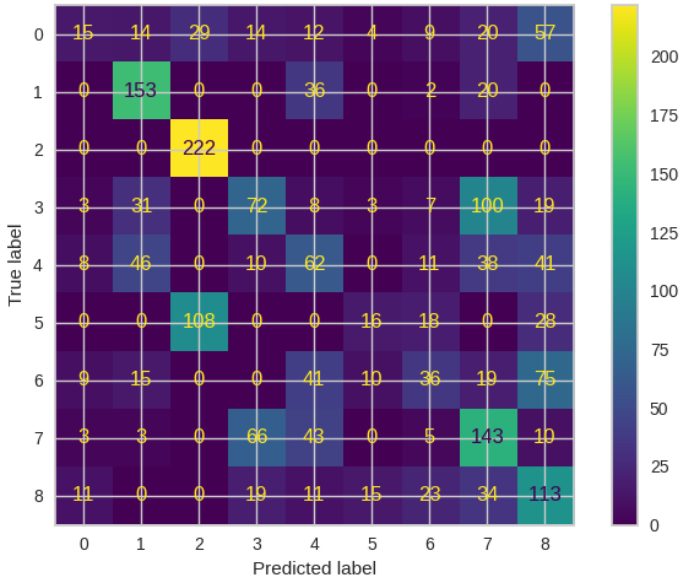
Číslo pohybu	1	2	3	4	5	6	7	8	9
Počet absencí	17	0	0	13	7	16	12	5	1

Dále byla také vytvořena matice záměn viz 4.2.

4.3 Metody shlukové analýzy

U této metody dosáhla klasifikační přesnost při dvaceti měření hodnoty 43,60% a směrodatná odchylka byla rovna 1,31. Zajímavě bylo u každého měření rozhodnotu, že ideální počet shluků je roven 6. V tabulce 4.4 je ukázáno kolikrát každý pohyb nebyl zvolen jako jeden ze šesti shluků.

I zde byla vytvořena matice záměn 4.3.



Obrázek 4.3: Matice záměn K-medoids

Kapitola 5

Výsledky a diskuze

V této kapitole jsou diskutovány výsledky získané z testování a je vyhodnoceno, zda se dá klasifikátor použít v praxi pro nahrazení experta, při určování typu rehabilitačního cviku.

Dále je také ukázáno, jaké cviky jsou nejčastěji špatně klasifikovány, pomocí využití maticí záměn.

5.1 Neuronové sítě využívající expertních hodnocení

Jak si lze všimnout u 4.2 přesnost se obecně o trochu zlepšila při přidaní druhé BLSTM vrstvy, ovšem přidání třetí už nemělo pozitivní vliv. Vliv počtu neuronů se zdá být celkem náhodný, jelikož nejde říci, že čím více tím lépe nebo naopak.

Přesnost 95,87% dosažená NS s učitelem je výborný výsledek, který určitě lze použít v praxi, jelikož se při této přesnosti může na model spolehnout. Tento výsledek je nejlepší ze tří zde implementovaných metod, a to co se týče přesnosti, tak i směrodatné odchylky.

Zde dosažená přesnost je výborná i v porovnání s ostatními prácemi, které používají LSTM s učitelem ke klasifikaci. Například [12], [13] a [14] dosahují přesnosti 96%, 88.66-98,33% a 92,67%. Nejdem ovšem říci, že zde použitá architektura je lepší nebo horší, jelikož obrovsky záléží na poskytnutých datech, které jsou klasifikovány.

Matice záměn dále ukazuje, že síť převážně chybovala při klasifikaci pohybů 3, 4, 7 a 8. i u těchto cviků, ale model ve valné většině klasifikuje správně, což poukazuje, že cviky byly vybrány dostatečně odlišené.

5.2 Samoučící neuronové sítě

Tabulke 4.3 ukazuje, že stejně jako u BLSTM NS se při přidání druhé LSTM vrstvy přenost zlepší, ale třetí už nemá pozitivní vliv na přesnost klasifikace. Počet neuronů nemá výrazný vliv na přesnost kromě variant, kde je počet LSTM vrstev rovno jedné.

Tato NS nečekaně dosahuje slušné přesnosti 91,90%. Nečekaně, jelikož oproti LSTM NS existuje daleko méně studií, které se touto architekturou zabývají. Tato metoda se dá v praxi použít, avšak není ideální, jelikož je přesnost celkem nízká. Tyto výsledky tedy ukazují, že i tato struktura NS se dá použít pro klasifikaci. Přeci jenom zde byla za kóderem použitá jednoduchá vrstva neuronů pro klasifikaci. Tato vrstva by se dala nahradit jiným klasifikátorem, jako například Náhodný Les, což by mohlo přesnost zlepšit. Toto téma je přenechané do budoucích prací.

U matice změn si lze všimnout, že chyb už je více nežli u BLSTM architekrury. Nejvíce tento model chyboval při klasifikaci pohybů 3 a 4, a podobně jako u BLSTM nastal problém i s klasifikací pohybu 7.

5.3 Metody shlukové analýzy

Bohužel tato metoda dosáhla nejhorší přesnosti, a to pouhých 43,60%. Tato tedy není vhodná pro použití v praxi, jelikož není dostatečně spolehlivá. Tento horší výsledek je nejspíše dán už charakteristikou samotné shlukové analýzy. Není totiž standardní použít SA na shlukování, kde už jsou jednotlivé třídy pojmenovány neboli na shlukování s učitelem. Dále se zde také musely redukovat dimenze, čímž se ztratily relevantní informace o pohybu, a tím pádem jistě klesla i přesnost.

Jak si lze všimnout u matici záměn 4.3 bylo zde opravdu mnoho špatných klasifikací u každého pohybu. Jediné pohyby, které byly správně klasifikovány byly typu druhého a třetího. Na druhou stranu první pohyb byl správně kategorizován pouze patnáctkrát. Tento výsledek by se také dal vyvodit z tabulky 4.4, kde druhý a třetí pohyby vždy měly vlastní shluk. První pohybu byl na druhou stranu shluk přiřazen pouze třikrát z dvaceti měření.

Z tohoto výsledku ovšem nelze odvodit, že si jsou pohyby příliš podobné, a tím pádem jsou špatně klasifikovány, jelikož matice 4.1 a 4.2 dokazují, že tomu tak není. Chyba je v tomto případě u klasifikační metody.

Kapitola 6

Závěr

Cílem této práce bylo vytvořit systém umožňující přesnou real-time klasifikaci rehabilitačních pohybů, a to pomocí tří různých metodik.

V práci bylo vybráno osm různých rehabilitačních cviků využívající paži určené pro klasifikaci a jeden pohyb, kde paže byla v klidovém stavu. Za pomoci dvanácti dobrovolníků, kteří prováděli pohyby, byl vytvořen dataset obsahující 324 pohybů. Dataset byl postačující pro naučení modelu s postačující přesností při real-time klasifikaci, ale v budoucnu by bylo prospěšné dataset rozšířit pro dosažení větší rozmanitosti, a tak i větší přesnosti.

Jako první metoda klasifikace byla zvolena neuronová síť s využitím expertního učení, konkrétněji Bidirecional Long short-term memory. Tento model dosáhl na datasetu vysoké přesnosti 95,87% a dal by se tedy použít v praxi. Druhá metoda využila spojení samoučící neuronové sítě autoencoder a jedné vrstvy neuronů použité pro klasifikaci, kde byl model naučen s použitím učitele. Přesnost na datasetu byla oproti první metodě menší, a to 91,90%, ale i tak by se dala použít v praxi. Třetí metoda využila shlukovou analýzu pomocí algoritmu K-Medoids, metriky Dynamic Time Warp a metody KElbow, která určila optimální počet shlulů. Výsledky byly zde nejhorší. Přesnost na datasetu dosáhla 43,60%, což není přijatelné pro spolehlivou klasifikaci, a tak se nedá použít v praxi. Z výsledků těchto tří metod byl tedy určena jako nejlepší metoda využití neuronové sítě s učitelem, který splnil požadavky pro přesnost a spolehlivost systému.

Vytvořená práce zahrnuje pouze rehabilitační cviky využívající paži, jelikož je používán pouze jeden senzor. Proto by bylo možné ji obohatit o více senzorů, které by umožnily klasifikovat více komplexní cviky. Dále by šlo také otestovat jiné klasifikátory u druhé metody, což by mohlo zlepšit přesnost kategorizace.

Literatura

[1] Understanding LSTM Networks. 2022, [Online; cit. 2022-6-14]. Dostupné z: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[2] Hota, H. S.; Handa, R.; Shrivas, A. K.: Time Series Data Prediction Using Sliding Window Based RBF Neural Network. 2017.

[3] Field, M.; Stirling, D.; Naghdy, F.; aj.: Motion capture in robotics review. In *2009 IEEE International Conference on Control and Automation*, 2009, s. 1697–1702, doi:10.1109/ICCA.2009.5410185.

[4] Suzuki, K.: *Artificial Neural Networks*. Rijeka: IntechOpen, 2013, str. ix, doi:10.5772/3409. Dostupné z: https://doi.org/10.5772/3409

[5] Mcculloch, W.; Pitts, W.: A Logical Calculus of Ideas Immanent in Nervous Activity. *Bulletin of Mathematical Biophysics*, ročník 5, 1943, ISSN 0007-4985.

[6] Zeng, M.; Nguyen, L. T.; Yu, B.; aj.: Convolutional Neural Networks for human activity recognition using mobile sensors. *Proceedings of the 6Th IEEE International Conference on Mobile Computing, Applications and Services (Mobicase)*, 01 2014: s. 197–205.

[7] Ayrulu-Erdem, B.; Barshan, B.: Leg Motion Classification with Artificial Neural Networks Using Wavelet-Based Features of Gyroscope Signals. *Sensors*, ročník 11, č. 2, 2011: s. 1721–1743, ISSN 1424-8220, doi:10.3390/s110201721. Dostupné z: https://www.mdpi.com/1424-8220/11/2/1721

[8] Yu, H.; min Sun, G.; xing Song, W.; aj.: Human motion recognition based on neural network. In *Proceedings. 2005 International Conference on Communications, Circuits and Systems, 2005.*, ročník 2, 2005, str. 982, doi:10.1109/ICCCAS.2005.1495271.

[9] Mantyjarvi, J.; Himberg, J.; Seppanen, T.: Recognizing human motion with multiple acceleration sensors. In *2001 IEEE International Conference on Systems, Man and Cybernetics. e-Systems and e-Man for Cybernetics in Cyberspace (Cat.No.01CH37236)*, ročník 2, 2001, s. 747–752 vol.2, doi:10.1109/ICSMC.2001.973004.

[10] Um, T. T.; Babakeshizadeh, V.; Kulić, D.: Exercise motion classification from large-scale wearable sensor data using convolutional neural networks. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, s. 2385–2390, doi:10.1109/IROS.2017.8206051.

[11] Huang, J.; Lin, S.; Wang, N.; aj.: TSE-CNN: A Two-Stage End-to-End CNN for Human Activity Recognition. *IEEE Journal of Biomedical and Health Informatics*, ročník 24, č. 1, 2020: s. 292–299, doi:10.1109/JBHI.2019.2909688.

[12] Rego Drumond, R.; Marques, B.; Vasconcelos, C.; aj.: PEEK - An LSTM Recurrent Network for Motion Classification from Sparse Data. 01 2018, s. 215–222, doi:10.5220/0006585202150222.

[13] Yang, L.; Chen, J.; Zhu, W.: Dynamic Hand Gesture Recognition Based on a Leap Motion Controller and Two-Layer Bidirectional Recurrent Neural Network. *Sensors*, ročník 20, č. 7, 2020, ISSN 1424-8220, doi:10.3390/s20072106. Dostupné z: `https://www.mdpi.com/1424-8220/20/7/2106`

[14] Hernández, F.; Suárez, L. F.; Villamizar, J.; aj.: Human Activity Recognition on Smartphones Using a Bidirectional LSTM Network. In *2019 XXII Symposium on Image, Signal Processing and Artificial Vision (STSIVA)*, 2019, s. 1–5, doi:10.1109/STSIVA.2019.8730249.

[15] Baldi, P.: Autoencoders, Unsupervised Learning, and Deep Architectures. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, *Proceedings of Machine Learning Research*, ročník 27, editace I. Guyon; G. Dror; V. Lemaire; G. Taylor; D. Silver, Bellevue, Washington, USA: PMLR, 02 Jul 2012, s. 37–49. Dostupné z: `https://proceedings.mlr.press/v27/baldi12a.html`

[16] Liu, H.; Taniguchi, T.: Feature Extraction and Pattern Recognition for Human Motion by a Deep Sparse Autoencoder. In *2014 IEEE International Conference on Computer and Information Technology*, 2014, s. 173–181, doi:10.1109/CIT.2014.144.

[17] Nguyen, H.; Tran, K.; Thomassey, S.; aj.: Forecasting and Anomaly Detection approaches using LSTM and LSTM Autoencoder techniques with the applications in supply chain management. *International Journal of Information Management*, ročník 57, 2021: str. 102282, ISSN 0268-4012,

doi:https://doi.org/10.1016/j.ijinfomgt.2020.102282. Dostupné z: https://www.sciencedirect.com/science/article/pii/S026840122031481X

[18] Xu, X.; Yoneda, M.: Multitask Air-Quality Prediction Based on LSTM-Autoencoder Model. *IEEE Transactions on Cybernetics*, ročník 51, č. 5, 2021: s. 2577–2586, doi:10.1109/TCYB.2019.2945999.

[19] Tang, T.; Jia, J.; Mao, H.: Dance with Melody: An LSTM-Autoencoder Approach to Music-Oriented Dance Synthesis. In *Proceedings of the 26th ACM International Conference on Multimedia*, MM '18, New York, NY, USA: Association for Computing Machinery, 2018, ISBN 9781450356657, str. 1598–1606, doi:10.1145/3240508.3240526. Dostupné z: https://doi.org/10.1145/3240508.3240526

[20] Malhotra, P.; Ramakrishnan, A.; Anand, G.; aj.: LSTM-based Encoder-Decoder for Multi-sensor Anomaly Detection. 2016. Dostupné z: http://arxiv.org/abs/1607.00148

[21] Chu, S.; Keogh, E.; Hart, D.; aj.: *Iterative Deepening Dynamic Time Warping for Time Series*. s. 195–212, doi:10.1137/1.9781611972726.12. Dostupné z: https://epubs.siam.org/doi/abs/10.1137/1.9781611972726.12

[22] Adistambha, K.; Ritz, C. H.; Burnett, I. S.: Motion classification using Dynamic Time Warping. In *2008 IEEE 10th Workshop on Multimedia Signal Processing*, 2008, s. 622–627, doi:10.1109/MMSP.2008.4665151.

[23] Biswas, D.; Cranny, A.; Gupta, N.; aj.: Recognizing upper limb movements with wrist worn inertial sensors using k-means clustering classification. *Human Movement Science*, ročník 40, 2015: s. 59–76, ISSN 0167-9457, doi:https://doi.org/10.1016/j.humov.2014.11.013. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0167945714002115

[24] Vo, Q. V.; Hoang, M. T.; Choi, D.: Personalization in Mobile Activity Recognition System Using K-Medoids Clustering Algorithm. *International Journal of Distributed Sensor Networks*, ročník 9, č. 7, 2013: str. 315841, doi:10.1155/2013/315841.

[25] Niennattrakul, V.; Ratanamahatana, C. A.: On Clustering Multimedia Time Series Data Using K-Means and Dynamic Time Warping. In *2007 International Conference on Multimedia and Ubiquitous Engineering (MUE'07)*, 2007, s. 733–738, doi:10.1109/MUE.2007.165.

[26] Oszust, M.; Wysocki, M.: Recognition of signed expressions observed by Kinect Sensor. In *2013 10th IEEE International Conference on Advanced Video and Signal Based Surveillance*, 2013, s. 220–225, doi:10.1109/AVSS.2013.6636643.

[27] Oszust, M.; Wysocki, M.: Clustering and Classification of Time Series Representing Sign Language Words. In *Artificial Intelligence and Soft Computing*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, ISBN 978-3-642-38610-7, s. 218–229.

[28] Khusainov, R.; Azzi, D.; Achumba, I. E.; aj.: Real-Time Human Ambulation, Activity, and Physiological Monitoring: Taxonomy of Issues, Techniques, Applications, Challenges and Limitations. *Sensors*, ročník 13, č. 10, 2013: s. 12852–12902, ISSN 1424-8220, doi:10.3390/s131012852. Dostupné z: `https://www.mdpi.com/1424-8220/13/10/12852`

[29] Bersch, S. D.; Chislett, C. M. J.; Azzi, D.; aj.: Activity detection using frequency analysis and off-the-shelf devices: Fall detection from accelerometer data. In *2011 5th International Conference on Pervasive Computing Technologies for Healthcare (PervasiveHealth) and Workshops*, 2011, s. 362–365, doi:10.4108/icst.pervasivehealth.2011.246119.

[30] Baldi, P.; Sadowski, P. J.: Understanding Dropout. In *Advances in Neural Information Processing Systems*, ročník 26, editace C. Burges; L. Bottou; M. Welling; Z. Ghahramani; K. Weinberger, Curran Associates, Inc., 2013. Dostupné z: `https://proceedings.neurips.cc/paper/2013/file/71f6278d140af599e06ad9bf1ba03cb0-Paper.pdf`

Příloha A

Seznam použitých zkratek

NS Neuronová síť

LSTM Long short-term memory

BLST Bidirectional long short-term memory

SA Shluková analýza

MHK Metoda hlavních komponent

RNS Rekurentní neuronová síť

DTW Dynamic time warping

AE Autoencoder

LAE LSTM Autoencoder

DS Distortion score

Příloha **B**

Obsah přiloženého CD

- `README.md` ..... Markdown soubor s popisem práce
- `BP_Slavik_Vojtech_2022.pdf` ..... bakalářská práce v pdf
- `data` ..... vytvořený dataset
- `docs` ..... dokumentace kódu
- `src` ..... zdrojový kód aplikace
  - `train_model.py` ..... trénování modelů
  - `classify.py` ..... klasifikace pohybů