VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

DETEKCE ANOMÁLIÍ V CHŮZI CHODCŮ

DETECTION OF ANOMALIES IN PEDESTRIAN WALKING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **ONDŘEJ POKORNÝ**
AUTHOR

VEDOUCÍ PRÁCE **Ing. TOMÁŠ GOLDMANN,**
SUPERVISOR

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav inteligentních systémů (UITS) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Pokorný Ondřej**
Program: Informační technologie
Název: **Detekce anomálií v chůzi chodců**
Detection of Anomalies in Pedestrian Walking
Kategorie: Zpracování obrazu

Zadání:

1. Seznamte se problematikou detekce anomálií v obraze. Především se zaměřte na řešení využívající neuronové sítě.
2. Sumarizujte informace o dostupných řešeních pro detekci anomálií pohybů chodců v obraze z kamer.
3. Navrhněte řešení pro detekci anomálii pohybů chodců ve videozáznamu. Jako základ řešení využijte neuronovou síť pro detekci skeletu.
4. Navržené řešení implementujte v programovacím jazyce Python a vytvořte k němu jednoduché uživatelské rozhraní.
5. Proveďte experimenty se záznamy z reálného prostředí. Vyhodnoťte úspěšnost detekce.

Literatura:

- STAAR, Benjamin; LÜTJEN, Michael; FREITAG, Michael. Anomaly detection with convolutional neural networks for industrial surface inspection. *Procedia CIRP*, 2019, 79: 484-489.
- ULLAH, Waseem, et al. CNN features with bi-directional LSTM for real-time anomaly detection in surveillance networks. *Multimedia Tools and Applications*, 2021, 80.11: 16979-16995.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Goldmann Tomáš, Ing.**
Vedoucí ústavu: Hanáček Petr, doc. Dr. Ing.
Datum zadání: 1. listopadu 2021
Datum odevzdání: 11. května 2022
Datum schválení: 3. listopadu 2021

Abstrakt

Cílem práce bylo vytvořit systém pro detekci anomálií v záznamech chůze chodců. Jako základ aplikace použijeme již existující řešení pro extrakci souřadnic skeletu chodce OpenPose. Pro následnou detekci z hodnot souřadnic jsem se zaměřil na řešení pomocí neuronových sítí. K řešení jsem použil obousměrnou LSTM neuronovou síť, která během experimentování měla nejlepší hodnoty detekce. Výsledná aplikace zvládá detekci tří anomálií a to skoku, dřepu a kliku. Výstupem je video, ve kterém jsou nápisem označeny sekvence, které obsahují anomálii. Celý systém je imlementovaný v jazyce Python a jeho běžně dostupných knihoven.

Abstract

The goal of this work was to create a system that would be able to detect anomalies in pedestrian walking. As the core of my application, I have used OpenPose, which is an application for detecting human skeletons. Then I used a bidirectional LSTM neural network to detect anomalies in video sequences. This architecture was chosen during the experiment because it outperformed other solutions. I trained my model to detect three types of anomalies. The output of my application is a video with marked sequences of anomalies. The whole system is implemented in Python.

Klíčová slova

strojové učení, umělá inteligence, neuronové sítě, anomálie, detekce anomálií, zpracování obrazu, počítačové vidění, detekce lidí, detekce skeletu, python, LSTM, rekurentní neuronové sítě, kamerový systém

Keywords

machine learning, artificial intelligence, neural networks, anomaly, anomaly detection, image processing, computer vision, human detection, skelet detection, python, LSTM, reccurent neural networks, surveillance system

Citace

POKORNÝ, Ondřej. *Detekce anomálií v chůzi chodců*. Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Tomáš Goldmann,

Detekce anomálií v chůzi chodců

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Tomáše Goldmanna. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Ondřej Pokorný
10. května 2022

Poděkování

Touto cestou bych chtěl poděkovat svému vedoucímu Ing. Tomáši Goldmannovi za poskytnutí cenných rad a za pomoc při psaní mé bakalářské práce.

Obsah

Kapitola 1

Úvod

Má práce se zabývá zpracováním kamerových záznamů chůze chodců a následnou detekcí abnormálních událostí jako je například pád člověka během chůze. K dosažení této funkcionality využijeme již existujících algoritmů, které detekují osoby v obrazu a extrahují souřadnice jejich skeletu. Právě data získaná ze skeletu člověka budeme následně využívat k detekci anomálií.

Zpracování obrazu je v dnešním světě na vzestupu především díky dostupnosti kamer a zvyšující se kvalitě jejich záznamů. Kromě kamerových systémů k tomuto rozvoji pomohl i vývoj telekomunikačních sítí, které usnadňují přenos dat z kamer. Kamery denně produkují velké množství dat, které již není možné zpracovat bez žádné automatizace, tudíž se klade velký důraz na zpracování obrazu a z něho získaných dat pomocí strojového učení.

Toto téma mě zaujalo, protože se dle mého názoru jedná o praktické a užitečné využití dat, které je velmi jednoduché získat, protože téměř každý dnes disponujeme zařízením, které je schopno pořídit nějaký video záznam. Téma mě také lákalo díky možnosti dozvědět se více o strojovém učení, konkrétně o neuronových sítí a to nejen z teoretického hlediska, ale i praktického při jejich návrhu a implementaci.

V následující kapitole se nejdříve zaměřím na anomálie samotné. Pokusím se vysvětlit, co to je anomálie a uvést pár příkladů využití detekce těchto hodnot v reálném světě. Následně si vysvětlíme vybrané základní principy strojového učení, které se k detekci anomálií používají. V další části popíšu a vysvětlím postup detekce anomálií pomocí neuronových sítí, zároveň se zde podíváme na jednotlivé kroky, které se budou muset před samotnou detekcí anomálií v chůzi chodců vykonat. Popíšeme zde již existující algoritmy pro detekci lidí v obraze, následně si popíšeme přístupy, které slouží k detekci skeletu lidí a jako poslední krok se podíváme, na to jak fungují samotné neuronové sítě. Po seznámení se s základními principy v neuronových sítí se zaměříme na neuronové sítě, které se používají pro detekci anomálií a abnormálních vzorů v datech. Ve třetí kapitole představím konkrétní návrh pro řešení našeho problému založený na algoritmech a principech, které jsme si vysvětlili v předcházejících kapitolách. Po návrhu následuje kapitola zaměřena na implementaci, kde popíši výsledné principy a postupy, které byly nakonec použity. A v poslední kapitole vyhodnotíme námi vytvořené řešení pomocí experimentů s reálnými videii a popíšeme také průběžné experimentování s různými architekturami neuronových sítí, které jsme používali během naší implementace.

Kapitola 2

Detekce anomálií v chůzi chodců

V této kapitole vysvětlím pojem anomálie a uvedu několik příkladů, kde se běžně anomálie detekují. Poté se zaměřím na detekci anomálií v chůzi chodců a to postupně od samotné detekce lidí, extrakci jejich skeletu až po samotnou detekci anomálií. Detekce abnormálních jevů v chůzi chodců se běžně využívá v mnoha systémech například pro detekci pádů v domech pro seniory, kde mají tyto systémy klíčovou roli při rychlé pomoci u různých úrazů.

2.1 Anomálie a jejich detekce

Tato část nám přiblíží obecnou problematiku detekce anomálií z hlediska strojového učení. Objasníme si pojem anomálie a přiblížíme si již existující přístupy a algoritmy k detekci anomálií a jejich využití v praxi.

Anomálie

Následující tři odstavce vychází z článku v časopise *ACM Computing Surveys* [1].

Anomálie jsou vzory v datech, které neodpovídají běžnému chování. Detekce anomálií má obvykle jako vstup nějakou kolekci dat, která obsahuje jak normální, tak abnormální hodnoty. Jejím hlavním cílem je rozpoznat dané anomálie ve vstupu a označit je tak, abychom je mohli následně nějakým systémem dále zpracovat. Systémů, které se snaží tyto detekovaná data dále zpracovat je v dnešním světě nespočetné množství. My si v následující části některé z těchto systémů představíme a zároveň se podíváme i na základní přístupy jak tyto systémy detekují anomální data.

2.1.1 Využití detekce anomálií

Detekce podvodů

Detekce podvodů se zaměřuje na odhalení kriminálních aktivit v komerčních organizacích jako jsou banky, pojišťovny nebo akciové trhy. Zaměřujeme se zde především na podezřelou aktivitu ze strany uživatelů a monitorujeme tak jejich chování. Pokud se něčí aktivita bude vymykat normálu, tak musíme tyto aktivity detekovat, aby nevznikla nějaká škoda ať už na straně společnosti nebo dalších uživatelů. Jako konkrétní příklad si zde můžeme uvést podvody s kreditními kartami. Pokud zákazník běžně nakupuje pomocí kreditní karty, můžeme si pomocí jeho běžných transakcí jednoduše modelovat jeho profil. Pokud někdo začne provádět nějaké neobvyklé transakce pomocí stejné kreditní karty, chceme tuto aktivitu za-

chytit. Mezi takové abnormální chování může patřit například výběr neobvykle velké částky peněz, nákup na neobvyklém geografickém místě nebo v neobvyklý čas.

Detekce vad strojů

Jako další důležitá aplikace detekce anomálií je detekce vad v průmyslové výrobě. Jako vstupní data jsou obvykle nějaké zaznamenané hodnoty ze senzorů strojů nebo kamer na nich umístěných. Zde se snažíme detekovat například praskliny povrchů, které by postupem času mohly vést k selhání celého systému nebo vážnějším poruchám. Mezi data získávaná ze senzorů se může zařadit například nahrávka zvuku motoru. Pokud máme k dispozici nahrávku správně fungujícího motoru bez poruch můžeme pak tyto dvě nahrávky porovnávat mezi sebou a detekovat jakékoliv abnormální chování. Takto můžeme detekovat poruchu v samém zárodku a předejít dalším problémům.

Detekce anomálií v medicínských datech

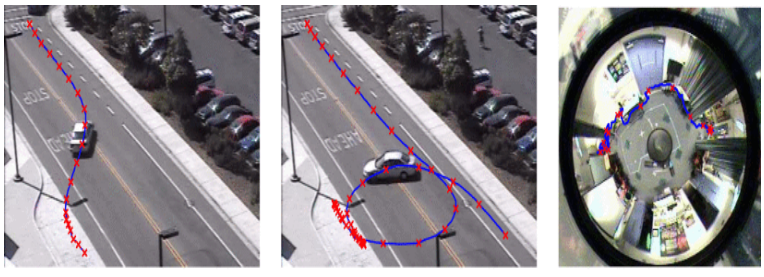
Tento odstavec vychází z knihy *Data Science – Analytics and Applications* [2].

Díky zvyšující se digitalizaci medicínských záznamů a obrazů máme dostatečnou kolekci dat, které jsou vhodné ke zpracování pomocí strojového učení. Detekce anomálií se tak běžně aplikuje například na rentgenové snímky plic. V těchto snímcích se mohou vyskytovat anomálie, které mohou znamenat přítomnost nějakého onemocnění jako je třeba zápal plic, nádorové nebo virové onemocnění.

Detekce anomálií v kamerových systémech

Následující část vychází z [3].

Kamerové systémy, které jsou v dnešním světě přítomny na každém rohu, mohou sloužit jako zdroj dat pro zpracování pomocí strojového učení. Pomocí získaných dat můžeme monitorovat a automaticky detekovat dopravní zácpy nebo dopravní nehody. Příklad abnormálního jevu nejen v dopravě je zobrazen zde 2.1. Pokud budeme sledovat nějaká veřejná místa můžeme automaticky detekovat anomální chování lidí jako jsou například konflikty mezi lidmi, odložené batohy na vlakových nádražích, či letištích nebo také krádeže předmětů v obchodech.

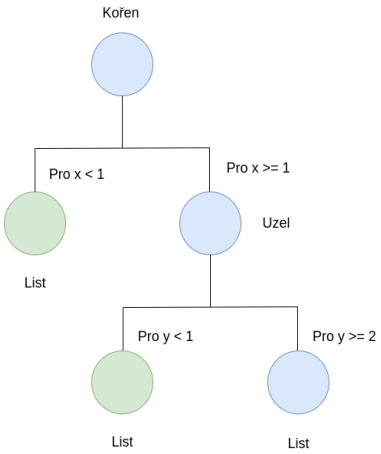


Obrázek 2.1: (a,b) Ukázka anomálního chování auta na vozovce. (c) Abnormální trajektorie člověka, který šel pouze po kraji místnosti. Převzato z [3].

2.1.2 Detekce anomálií z hlediska strojového učení

Rozhodovací stromy

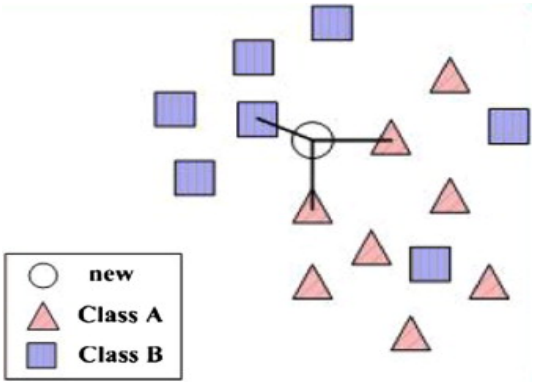
Rozhodovací stromy [4] jsou velmi efektivním algoritmem, který se používá ke klasifikaci hodnot do různých kategorií. Příklad takového stromu je na obrázku 2.2. Skládá se z několika komponent a to kořene, hran, uzlů a listů. Každý uzel má svůj atribut, který je pro něho nejvíce informativní ze všech ostatních atributů. Každá hrana z uzlu má svojí hodnotu pro daný atribut uzlu a každý list reprezentuje jednu z výsledných tříd, do kterých se snažíme rozřadit naše data. Na začátku bereme hodnotu z uzlu a snažíme se rozhodovat, na základě jejich atributů, do kterého dalšího uzlu se tato hodnota bude propagovat. Tento proces rozhodování se opakuje vždy v každém uzlu až dokud se nedostane do listu. Výsledkem tohoto algoritmu máme roztříděná data do různých kategorií, jako abnormální hodnoty bychom mohli označit ta data, která jsou obsažena v nejméně početné množině.



Obrázek 2.2: Ukázka rozhodovacího stromu

Algoritmus k-nejbližších sousedů

Algoritmus K-nejbližších sousedů [5] má jako vstup hodnoty x_1 až x_n a číslo K, které specifikuje počet shluků, které chceme ze vstupních hodnot vytvořit. Ještě před začátkem algoritmu musíme specifikovat funkci, která bude počítat vzdálenost mezi jednotlivými body a hodnoty c_1 až c_K, které budou reprezentovat středy shluku, které budeme tvořit. Velmi často se jako hodnotící funkce používá euklidovská vzdálenost. Na začátku budou středy shluků vybrány náhodně. Následně pro každou hodnotu x_i vypočítáme vzdálenost od středu každého shluku a zařadíme x_i do shluku, pro který je vzdálenost nejkratší. Po vložení hodnot do shluku musíme znovu vypočítat hodnotu středu shluku jako průměr všech hodnot ve shluku. Výsledkem algoritmu jsou shluky vstupních hodnot. Vizualizace vkládání nového prvku a jeho vzdáleností od ostatních již vložených hodnot je na obrázku 2.3. Jako abnormální hodnoty bychom poté mohli označit shluk, který bude tvořen nejmenším počtem hodnot.



Obrázek 2.3: Vložení prvku do shluku algoritmem K-nejbližších sousedů. Převzato z [6]

2.2 Anomálie v chůzi chodců

V této části si blíže představíme již existující postupy, algoritmy a technologie, které využijeme při detekci anomálií v chůzi chodců. Nejdříve se zaměříme na samotnou detekci lidí v obrazovém záznamu a představíme si základní přístupy k detekci lidí. Další část se bude věnovat následné detekci skeletu daného člověka a jeho extrakci z videa. A poté se zaměříme na detekci anomálií v obraze za pomocí neuronových sítí. V této části si nejdříve popíšeme principy neuronových sítí a poté se zaměříme na neuronové sítě, které bychom mohli následně využít pro řešení našeho problému detekce anomálií.

2.2.1 Detekce lidí v obraze

Tato část vychází z knihy *Computer Vision and Graphics: International Conference.* [7]

Detekce lidí v obraze je často náročný a klíčový úkol pro mnoho systémů. Detekci lidí v kamerových záznamech nám může ztížit několik faktorů, jako je například různé počasí, velikost lidí nebo jejich počet a umístění v obraze. Pomocí detekce se snažíme odpovědět na základní otázku, zda daný objekt v obrazu je nebo není člověk. Tento proces se obvykle skládá ze dvou kroků, první je detekce objektu a tím druhým je samotná klasifikace, zda se jedná o člověka nebo nikoliv.

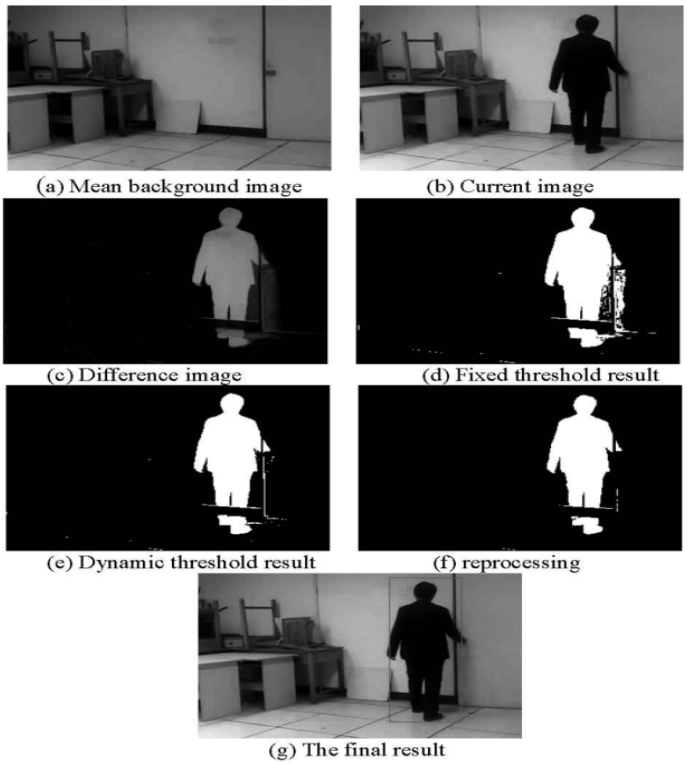
Techniky, které jsou popsány v disertační práci [8] využívané k rozpoznávání lidí v obrazu, bychom mohli rozdělit na dva typy. První přístup před samotným rozpoznáváním potřebuje zpracovat celý obraz a pomocí segmentace pozadí a výpočtu rozdílu mezi pozadím a popředím rozpozná člověka. Při tomto přístupu se tedy nejdříve detekuje objekt v popředí a poté se rozhoduje o tom, o jaký objekt se jedná. Toto rozhodování je založeno na různých charakteristikách daného objektu jako je například barva, tvar nebo pohyb. Druhý přístup vezme celý obraz a extrahuje z něho nejpravděpodobnější místa, kde by se mohly osoby s největší pravděpodobností nacházet. V tomto případě se rozhoduje na základě vlastností, které získáme z obrazu pomocí statické analýzy nebo neuronové sítě. Vlastnosti, podle kterých se rozhoduje se moc neliší od těch, které se používají i u prvního přístupu. Jedná se znovu hlavně o tvar, barvu nebo způsob pohybu objektu.

Algoritmy založené na odčítání pozadí

Tento odstavec je inspirován z [9].

Většinou se jedná o algoritmy, které se využívají pro staticky umístěné kamery a většina z nich je schopna detekovat objekty v reálném čase. Jednou z klíčových věcí, kterou musí tyto algoritmy zvládat, je pravidelné aktualizování pozadí, které se může měnit například vlivem počasí, intenzitou světla během dne nebo třeba jen pohybem mraků po obloze. Tyto algoritmy využívají k detekci pohybujícího se objektu v obraze rozdíl mezi jedním snímkem a následujícím snímkem. Přístup zobrazený na obrázku 2.4 se snaží na začátku vytvořit snímek pozadí, který se postupem času aktualizuje. Jako pozadí se určí buď první snímek nebo průměr jasu pixelů několika prvních snímků. Poté se počítá rozdíl jasu tohoto snímku s následujícími snímky, pokud rozdíl překročí stanovený práh, tak se jedná o pohybující se objekt. Tento výpočet je popsán rovnicí 2.1, kde k je číslo snímku, B_k je pozadí snímku, které budeme odčítat a F_k je aktuální snímek. Po provedení toho výpočtu se následně rozhodne, zda je v našem záznamu nějaký pohybující se objekt. Pokud ano odešle se tento objekt k následné klasifikaci, která určí, zda se jedná o člověka nebo nikoliv. Před samotnou klasifikací se ještě používají filtry k odstranění šumu a stínů, tak abychom dostali co nejlepší tvar pohybujícího se objektu. Po provedení těchto operací můžeme na základě tvaru objektu určit, zda se jedná o chodce nebo ne.

$$D_k(x,y) = \begin{cases} 1 & |F_k(x,y) - B_{k-1}(x,y)| > T + \Delta T \\ 0 & \text{else} \end{cases} \tag{2.1}$$



Obrázek 2.4: Ukázka zpracovávání obrazu algoritmem. Převzato z internetu [9].

Techniky využívající přímou detekci

Akira Utsumi and Nobuji Tetsutani [10] navrhli algoritmus, který je založený na přímé detekci člověka z videa. Detekce probíhá na základě použití geometrických struktur běžných pro detekované objekty. Lidské postavy většinou nejde lehce detekovat na základě barev, protože každý člověk může mít různě barevné oblečení, proto se tento algoritmus zaměřuje na vzdálenosti pixelů jednotlivých částí objektů. Detekce je založena na statistickém modelu, který určuje vzdálenosti jednotlivých částí těla jako je hlava, nohy nebo ruce od sebe. Tyto relativní vzdálenosti částí těla, na rozdíl od barev oblečení, jsou pro každého člověka podobné. Pro správnou funkčnost tohoto algoritmu potřebujeme dostatečně obsáhlou databázi obrázků člověka, abychom z ní mohli vypočítat průměrné hodnoty pro vzdálenosti jednotlivých částí lidského těla od sebe. Tento algoritmus nijak nepracuje s pozadím objektů, tudíž ho lze aplikovat i při dynamickém umístění kamery, například na automobilu.

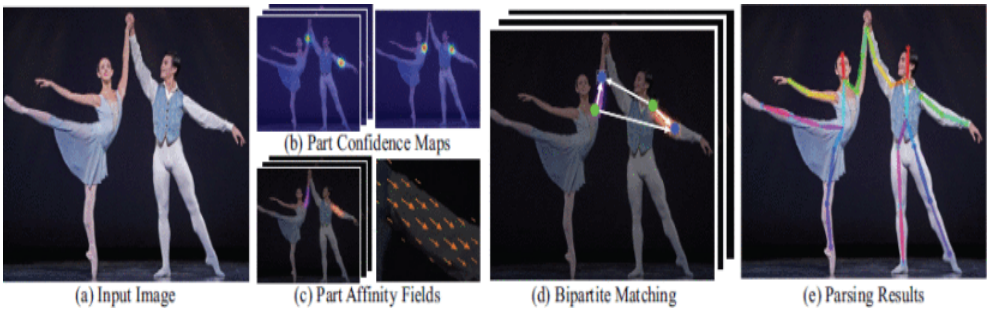
2.2.2 Detekce skeletu

Detekce skeletů chodců v záznamu videa je další z klíčových částí při detekci anomálií v chůzi chodců. Algoritmus, který bude detekci provádět, musí zvládnout rozpoznat části těla několika lidí, kteří se budou pohybovat v jednom snímku videa najednou. Tento proces vyžaduje překonání několika problémů jako je například různé rozmístění osob v obraze nebo jejich překryv během pohybu. Mnoho již existujících algoritmů je zaměřeno na detekci pouze jedné osoby nebo jsou výpočetně složité, takže se nedají použít v reálném čase. Zde se teď zaměříme na algoritmus, který je schopný v reálném čase detekovat několik osob najednou.

Algoritmus pro odhad pozice těla několika osob

Algoritmus [11], pro odhad pozice těla několika osob, jako jeden z prvních dokázal detekovat více lidí v obraze v reálném čase. Předchozí přístupy většinou nejdříve detekovali samotného člověka a poté se snažili na tento objekt aplikovat detekci skeletu pouze jedné osoby. Tento přístup není moc efektivní a zpracování obrazu s ním mohlo trvat až několik hodin, protože pro každou osobu musela být spuštěna samotná detekce skeletu. Algoritmus, který si zde představíme, je schopen najednou detekovat několik skeletu lidí v daném obraze, tudíž je mnohem efektivnější než předchozí přístupy.

Tento přístup rozděluje celý problém na několik podproblémů. V prvním kroce se nejdříve snaží detekovat všechny důležité části těla pro detekci skeletu, které se v obrazu nachází. Vytvoří se množina 2D souřadnic confidence map, které označují místa, kde se s vysokou pravděpodobností nachází nějaký z námi hledaných bodů. Zároveň se vytvoří i 2D pole vektorů, které reprezentují končetiny, které spojují dané body a mimo jiné určí i jejich pravděpodobný směr spojení. Tyto dvě množiny se v dalším kroku zpracují pomocí hladového algoritmu a vytvoří se grafová reprezentace lidského skeletu. Celý tento proces je zobrazen na obrázku 2.5.



Obrázek 2.5: Průběh zpracování obrazu algoritmem. Převzato z internetu [11].

Algoritmus pro extrakci skeletu založený na LSTM neuronové síti

Následující část, zabývající se extrakcí skeletu, vychází z práce [12], která se zaměřuje na detekci a extrakci skeletu z videí tanců. Získaná data z těchto videí by poté měla být využita za účelem asistence při učení tance nebo při hodnocení tanečníků na soutěžích. V tomto případě byla hlavní výzva, mimo jiné klasické překážky jako je různorodost lidských těl a stylů pohybu, extrahovat souřadnice z obvykle rychle pohybujících se osob s dostatečnou přesností a citlivostí. K detekci využili LSTM (anglicky *Long short-term memory*) neuronovou síť, která se hojně využívá pro detekci vzorů v sekvencích dat, v tomto případě sekvencí videa. Rozšířili ji o optimalizaci hejnem částic (anglicky *Particle swarm optimization*), čímž vylepšili hledání optimálních parametrů neuronové sítě a tím i její výkonnost.

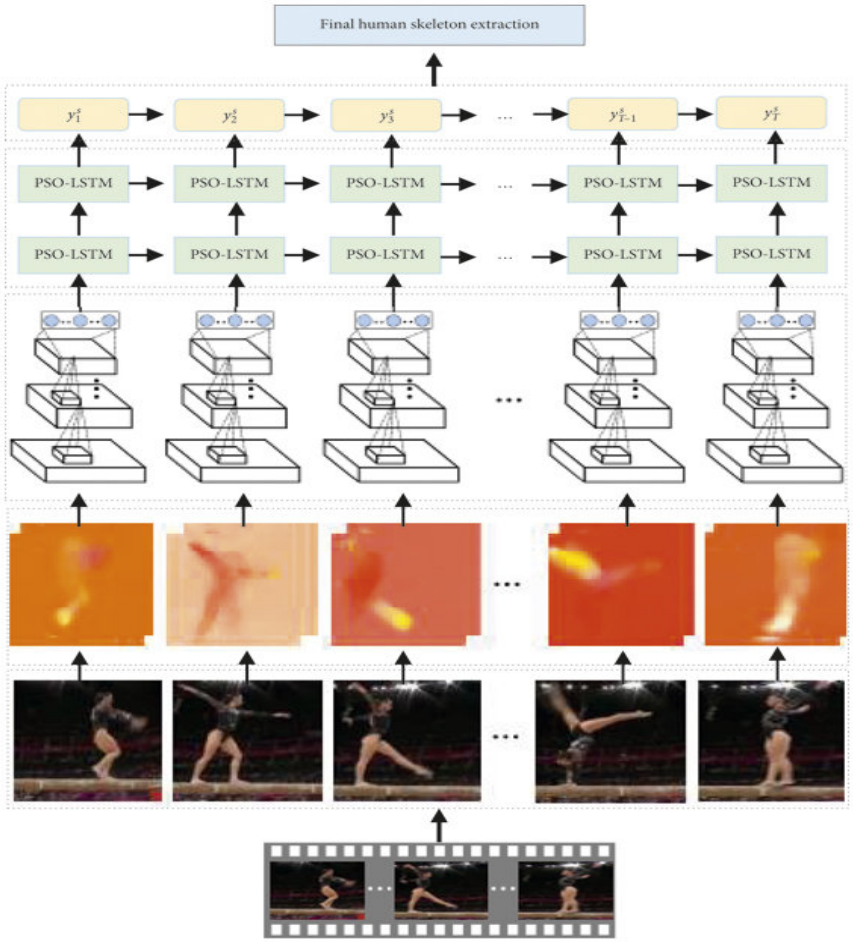
Algoritmus ze vstupního videa nejdříve vytvoří vzorky snímků, které obsahují vlastnosti lidského skeletu, poté se tyto vzorky snaží rozřadit do tříd pro jednotlivé body lidského těla. Vytvoří se zde množina *confidence map* pro každý bod, které se následně zpracují pomocí LSTM neuronové sítě a jako výsledek se vytvoří finální podoba extrahovaného skeletu lidského těla. Tento proces je na obrázku 2.6. Následně toto řešení otestovali na veřejně dostupných datasetech a získali velmi dobré hodnoty přesnosti detekce při velmi rychlém zpracování.

2.3 Detekce anomálií pomocí neuronových sítí

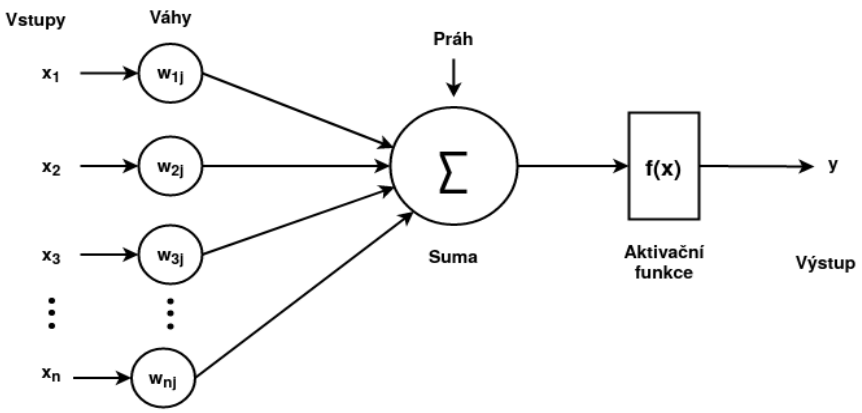
2.3.1 Neuronové sítě

Neuronová síť (anglicky *neural network*) [13] je výpočetní systém, který se inspiroval biologickými neuronovými sítěmi v lidském mozku. Neuronové sítě se skládají z několika vrstev umělých neuronů, které simulují chování biologických neuronů. Stejně jako neurony v mozku, které jsou spolu propojeny pomocí synapsí, jsou spolu propojeny i umělé neurony v neuronové síti. Každé toto spojení má vlastní váhu, podle které neuronová síť klade důraz na daný uzel. Každý umělý neuron má svojí definovanou hodnotu prahu, sčítá hodnoty všech signálů, které do něho přichází a pokud tyto hodnoty přesáhnout hodnotu prahu, tak se aktivuje. Celé toto chování je modelováno pomocí matematických funkcí.

Model umělého neuronu, který je zobrazen na obrázku 2.7, se skládá z několika hodnot x_1 až x_n na vstupu, své váhy prahu a jednoho výstupu y. Výstup neuronu y může vést do dalšího neuronu nebo může být výsledkem celé neuronové sítě. V neuronu bude probíhat výpočet jeho potenciálu a poté výpočet hodnoty výstupu pomocí aktivační funkce $f(x)$.



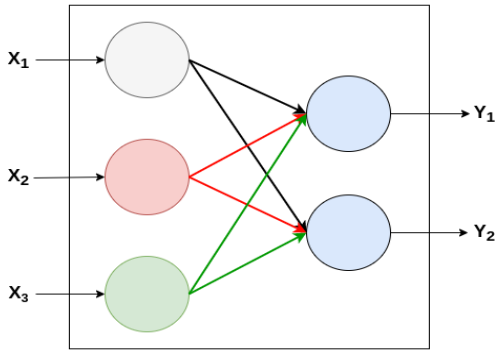
Obrázek 2.6: Průběh zpracování obrazu algoritmem. Převzato z internetu [12].



Obrázek 2.7: Model neuronu s aktivační funkcí.

Plně propojená neuronová vrstva

Plně propojená neuronová vrstva [14] se skládá z jednotlivých neuronů, jejichž vstup je výstup z každého neuronu, který je obsažen v předchozí vrstvě. Výstup neuronu je poté opět směřován do každého neuronu v následující vrstvě. Takové propojení je zobrazeno na obrázku 2.8. Dochází zde tedy ke transformaci X vstupů na Y výstupů. Neuronové sítě, které obsahují několik takto plně propojených vrstev se nazývají hluboké neuronové sítě. Tento typ neuronových sítí se obvykle dokáže mnohem efektivněji a lépe naučit reprezentovat i velmi složité funkce, než klasické neuronové sítě.



Obrázek 2.8: Schéma plně propojené neuronové vrstvy.

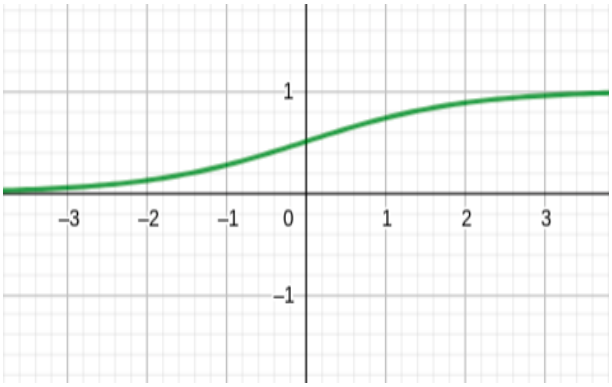
Aktivační funkce

Aktivační funkce [15] jsou důležité pro transformaci vstupního signálu na výstupní. Aktivační funkce se aplikuje na výstup z neuronu, který provede sumu všech vstupů a jejich vah. Její výstup se obvykle dále předává do další vrstvy neuronové sítě. Pokud by v neuronových sítích nebyla použita žádná aktivační funkce, její výstup by vypadal jako jednoduchá lineární funkce. Lineární funkce nejsou dostatečně komplexní a neumožňují nám rozpoznávat pravidelnosti mezi vstupy a výstupy, to je důvod proč se v neuronových sítích používají aktivační funkce. Nejčastěji používané aktivační funkce jsou funkce **sigmoid**, **tanh**, **ReLU** a různé jejich další modifikace.

Aktivační funkce **sigmoid** je nelineární funkce, která transformuje vstupy na hodnoty v rozmezí od 0 do 1, jak můžeme vidět na obrázku 2.9. Pokud na vstupu do funkce sigmoid přijde velké číslo, tak se výsledná hodnota bude blížit číslu 1, u velkých záporných čísel se výsledek bude blížit hodnotě 0. Funkce je tedy nejvíce citlivá na změnu hodnoty kolem střední hodnoty 0,5, což může u některých algoritmů způsobit, že se bude hůře odhadovat nová hodnota váhy a tudíž se bude hůře zvyšovat výkonnost modelu. Předpis funkce sigmoid je zde 2.2.

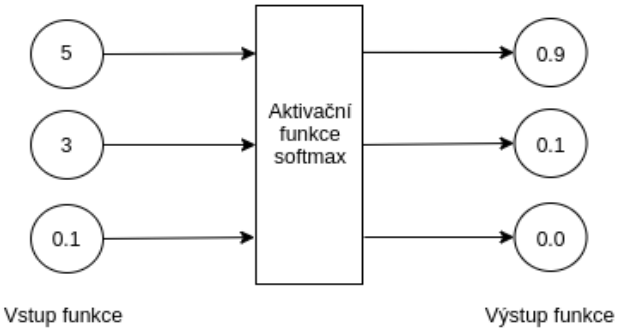
$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.2}$$

Funkce **softmax** [16] je kombinace několika funkcí sigmoid, které jak je zmíněno výše transformují vstup do rozmezí od 0 do 1. Tento výstup může být interpretovaný jako pravděpodobnost nastání nějakého jevu, proto se tato funkce často používá pro binární klasifikaci.



Obrázek 2.9: Průběh funkce sigmoid.

Funkce softmax se používá pro klasifikaci nějakého problému, který má více tříd, takže nelze použít binární klasifikace pomocí funkci sigmoid, takový příklad je zobrazen zde 2.10. Výstup softmax funkce jsou tedy hodnoty od 0 do 1, pro každou třídu, která se může na výstupu objevit, tyto hodnoty se opět mohou interpretovat jako pravděpodobnosti nastání nějakého jevu.

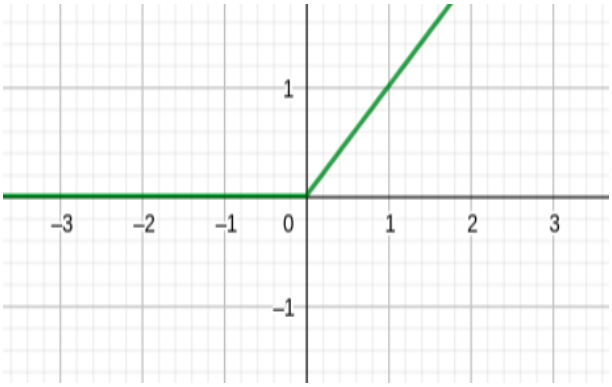


Obrázek 2.10: Příklad výstupu funkce softmax.

Mezi další populární aktivační funkce patří **ReLU** [17], která svým průběhem 2.11, připomíná funkci lineární. Tato funkce pro všechny záporné vstupy vrací na výstup 0 a pro všechny kladné vstupy včetně nuly vrací dané číslo. Mezi jeden z velkých kladů této funkce, je její jednoduchá implementace a nenáročný výpočet, kdy se nemusí použít žádný exponenciální výpočet. Předpis pro tuto aktivační funkci je zde 2.3.

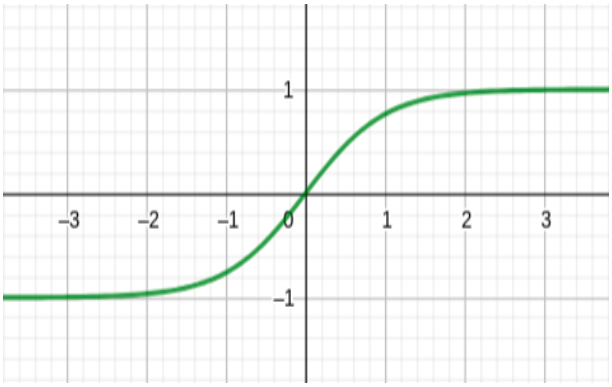
$$f(x) = \begin{cases} x & \text{pokud } x \geq 0 \\ 0 & \text{jinak} \end{cases} \tag{2.3}$$

Hyperbolický tangens [18] je aktivační funkce, která je často používána jako alternativa k funkci sigmoid. Obvykle se k použití funkce tanh přejde v momentě, kdy se objeví problém s krajními hodnotami funkce sigmoid. Tato funkce nám pro vstupy generuje výstupy v intervalu od -1 do 1, takže pokud touto funkcí nahradíme funkci sigmoid, budeme pro velké záporné hodnoty na vstupu místo 0 dostávat hodnotu -1. Předpis funkce 2.4, který na první pohled vypadá složitěji než předchozí funkce generuje výsledky, které vypadají jako obrázek 2.12.



Obrázek 2.11: Průběh funkce ReLU.

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



Obrázek 2.12: Průběh funkce tanh.

Učení neuronové sítě

Učení neuronové sítě [19] probíhá pomocí modifikace vah a prahů jednotlivých neuronů, tak abychom dostali co nejlepší výsledky pro daný vstup. Tento proces se nazývá *backpropagation* a skládá se ze dvou částí. První část je takzvaná *forward propagation*, kdy probíhá výpočet výstupu neuronové sítě pomocí jednotlivých neuronů. Druhá část je *backward propagation* během které se od výstupu směrem ke vstupu propaguje chyba daného výpočtu. Učení dělíme na dva druhy a to učení s učitelem (anglicky *supervised learning*) a učení bez učitele (anglicky *unsupervised learning*). Při učení s učitelem máme testovací data, která dáváme na vstup neuronové sítě a pro tyto data známe hodnoty, které bychom měli dostat na výstupu. Hodnoty na výstupu poté porovnáme s očekávanými výstupními hodnotami. Z rozdílu mezi těmito daty vypočítáme hodnotu chyby (anglicky *loss*), kterou se budeme snažit snížit pomocí upravení vah a prahů neuronů. Tento postup opakujeme dokud se hodnota chyby dostatečně nesníží. Při učení bez učitele máme soubor dat, který obsahuje pouze data pro vstupy, ale na rozdíl od učení s učitelem už pro ně neobsahuje data výstupní. Tento přístup se většinou používá pro shlukování nebo označení dat podle nějaké statistické vlastnosti, kterou mají tato data společnou.

Problémy během učení neuronové sítě

Následující tři odstavce vychází z [20].

Při učení neuronové sítě se snažíme generalizovat nějaký specifický problém z příkladů, které dáváme na vstup. To vše za účelem, aby byla neuronová síť schopna řešit podobný problém, se kterým se ještě nikdy předtím nesetkala. Během tohoto procesu učení může dojít k několika problémům, dva konkrétní z nich bych zde chtěl představit, a to konkrétně *overfitting* a *underfitting* neuronové sítě.

Overfitting je jeden z nejběžnějších problémů, se kterým se můžeme během trénování neuronových sítí setkat. Dochází k němu pokud se model naučí detaily a šum v tréninkových datech do takové míry, že to začne negativně ovlivňovat celkovou výkonnost modelu. Model generalizuje detaily některých vstupních dat a snaží se je aplikovat na nová vstupní data, pro které ale tyto koncepty nemusí platit a v tom případě dojde k ovlivnění výsledků modelu. Jedním způsobem jak zjistit a následně předejít tomuto jevu je rozdělení dat na tréninkové a validační data. Validační data se použijí až po trénování modelu na ověření jeho validity, takže získáme objektivní obrázek o tom, zda je daný model správný nebo ne.

Jako další příklad problému, který může nastat během učení neuronové sítě je *underfitting*. Dochází k němu pokud model nedokáže modelovat tréninková data ani generalizovat data, se kterými se ještě nesetkal. Narozdíl od *overfittingu* jde jednoduše detekovat, protože model bude mít špatnou výkonnost už na tréninkových datech. Jeden ze způsobu jak tento problém řešit je zkusit změnit nějaký z použitých algoritmů nebo se pokusit udělat náš model více komplexní například přidáním další vrstvy.

2.3.2 Neuronové sítě používané pro detekci anomálií v obraze

Vychází z časopisu *EURASIP Journal on Advances in Signal Processing* [21].

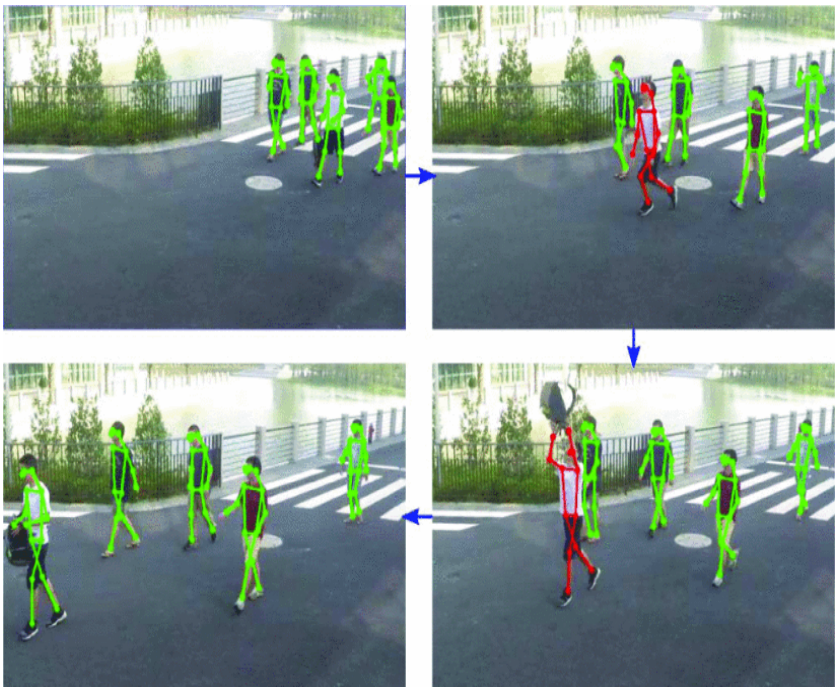
V této části se seznámíme s existujícími řešeními pro detekci anomálií pomocí neuronových sítí. Detekce anomálií v obrazu má mnohá využití a je to kritická část mnohých aplikací. Může se využít například pro detekci přetížení dopravních cest, kontrolu bezpečnostních kamer, charakteristiku lidské chůze nebo detekci pádů chodců. Samotná detekce objektů a jejich klasifikace může být provedena některým z výše zmíněných algoritmů nebo jejich kombinací a detekce anomálií může být navazujícím krokem zpracování těchto dat.

Tato část textu byla inspirována knihou *Anomaly Detection Principles and Algorithms.*[22]. Anomálie je odchylka od nějakého normálu. Mnoho procesů, které v přírodě existují se řídí nějakými pravidly, pomocí níž můžeme stanovit různé hypotézy, které popisují normální průběh. Ovšem i tyto procesy se mohou odchýlit od tohoto normálního stavu a mohou nastat různé anomálie ve výsledcích těchto procesů. Naším úkolem bude tyto anomálie detekovat a označit. Výsledky některých procesů můžeme aproximovat různými rozloženími, kde poté můžeme rozpoznat, zda některé hodnoty na rozdíl od ostatních do tohoto rozložení nezapadají a tudíž se bude jednat o anomálii. Výsledky jiných procesů je zase lepší místo aproximace rozložením shlukovat do skupin, kdy si musíme určit hodnotu vzdálenosti, kdy ještě nová hodnota bude patřit do již existujícího shluku nebo se bude jednat o nový shluk hodnot či anomálii. Nyní si představíme konkrétní příklady algoritmů, jež se zaměřují na detekci anomálií v záznamu chodců.

Algoritmus využívající trajektorie skeletu pro detekci anomálií

Následující část vychází ze sborníku konference *Conference on Computer Vision and Pattern Recognition* [23].

Algoritmus, který se snaží detekovat anomálie z trajektorie lidského skeletu, trajektorii skeletu vyjadřuje jako 2D souřadnice v obraze. Následná detekce probíhá podle více faktorů, u pohybu chodců to může být jejich umístění v obraze, rychlost nebo držení těla. Algoritmus rozděluje detekci na 2 části, první sleduje celkový pohyb těla v obraze a nese informace o tvaru, velikosti a pohybu lidského těla. Druhý se zaměřuje pouze na jednotlivé části těla, jejich deformace během pohybu a ignoruje umístění těla v celkovém obraze. K detekci anomálií se poté využívá Encoder-Decoder. Tento princip se velmi často používá k detekci anomálií. Skládá se ze tří částí, kdy první je tvořena dekodérem, který vezme daný vstup a zakóduje ho do nějaké skryté, ale předem dané reprezentace. Takto zpracovaná data dále putují do koderu, který je vezme a snaží se zpětně vytvořit data, která byla na vstupu. Naše neuronová síť se před nasazením musí trénovat na datech, které neobsahují žádné nenormální data, tím zajistíme, že bude náš kodér transformovat zakódovaná data na data bez anomálií. Po provedení tohoto procesu se vezme námi vytvořený výstup a porovná se s daty získanými ze záznamu. Pokud se budou tyto data mezi sebou nějak významně lišit, označíme tento snímek za anomálii. Příklad takového výstupu je na obrázku 2.13.



Obrázek 2.13: Příklad detekce anomálie v chůzi chodce. Převzato z [23].

Algoritmus detekce anomálií pomocí extrakce příznaků

Následující část vychází z knihy *Computer Vision and Graphics* [24].

Algoritmus slouží především k detekci pádů lidí v záznamu kamery. Nejčastěji se využívá pro detekci pádů seniorů v jejich domovech, kdy systém rozpozná pád a zavolá danému člověku pomoc. Algoritmus, pro detekci lidí v záznamu, využívá výše popsanou techniku *background-subtraction* 2.2.1. Algoritmus pak používá klasické sledování objektu v obrazu a extrahuje z něho příznaky, pomocí níž rozhoduje zda daná osoba spadla nebo nikoliv. V tomto přístupu se ještě navíc pro zlepšení detekce daného člověka využívá sledování pozice jeho hlavy, to především v případě, kdy nemůžeme jednoduše sledovat celé jeho

tělo a získat tak veškeré nutné vlastnosti ke klasifikaci. Mezi příznaky, které se používají k detekci člověka, který leží na zemi se zde používá například poměr šířky a výšky obdélníku, který ohraničuje daného člověka nebo porovnání reálné výšky osoby s aktuální výškou v získaném záznamu. Pokud není možné detekovat celé tělo člověka, bude se k detekci využívat vzdálenost hlavy osoby od podlahy v místnosti. Tento algoritmus je velice účinný především pro statické kamery, které jsou umístěny ve vnitřních prostorách budov.

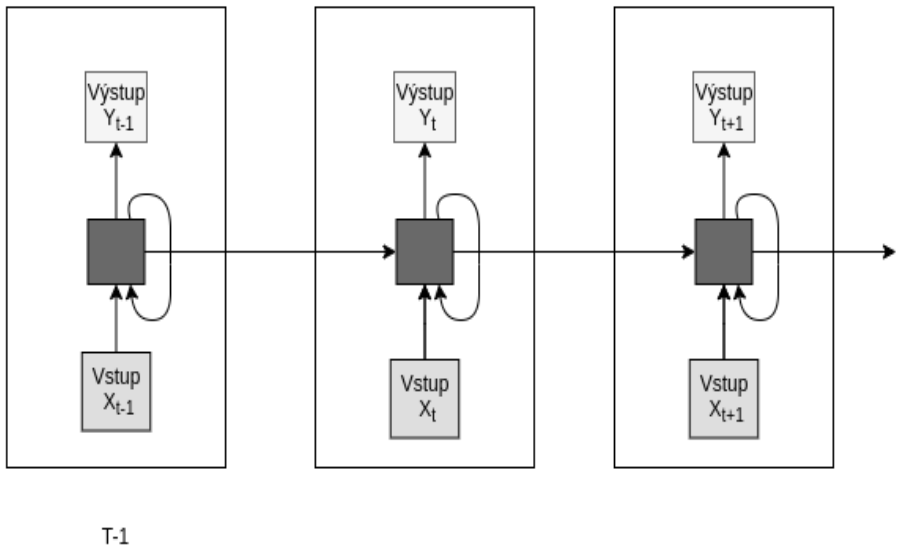
2.3.3 Neuronové sítě používané pro klasifikaci vzoru v datech

Pokud budeme chtít detekovat nějaké konkrétní anomálie, které se mohou v našich datech náhodně opakovat, budeme moc využít toho, že každá anomálie bude mít svůj nějaký typický průběh. Tyto průběhy se pak následně budeme snažit detekovat pomocí speciálních druhů neuronových sítí, která se zaměřují na detekci vzorů v datech. Tento problém řeší především rekurentní neuronové sítě a jejich různá rozšíření, proto si teď přiblížíme principy tohoto typu neuronových sítí, které by se nám mohly hodit na řešení našeho problému.

Rekurentní neuronové sítě

Tato část vychází z knihy *Recurrent Neural Networks: Design and Applications* [25].

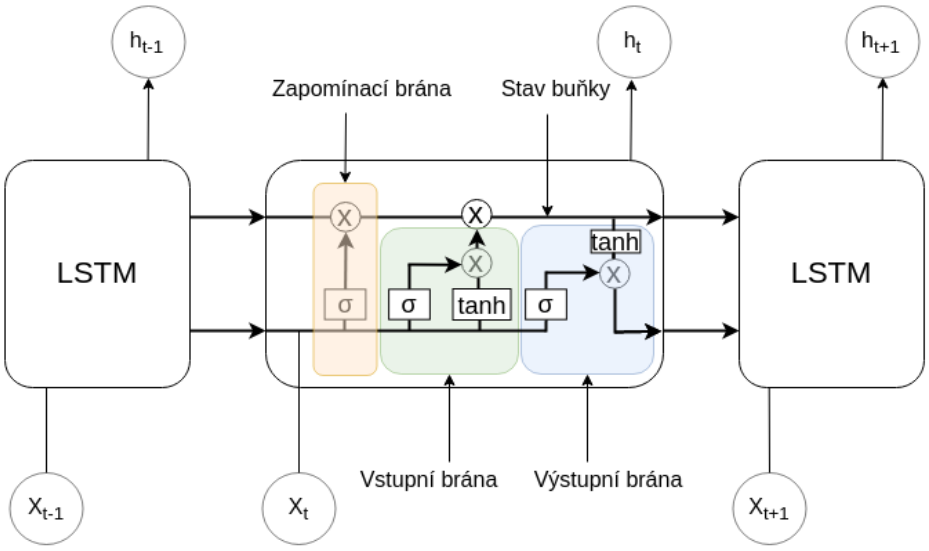
Rekurentní neuronové sítě (anglicky *Recurrent neural networks*) jsou navrženy tak, aby se jednoduše učili sekvenční nebo časově závislé vzory. Jsou to sítě, které obsahují ve své architektuře mimo jiné i spojení pomocí zpětné vazby. Každý neuron v této síti má jako vstup výstup z neuronu z předchozí vrstvy a zároveň výstup ze sebe sama. Výstup z neuronu je tedy směřován jak do dalšího neuronu, tak zpět do toho neuronu, ze kterého pochází a tím zajišťuje již zmíněnou zpětnou vazbu. Díky zpětné vazbě můžeme tedy reflektovat závislosti mezi vstupními daty, alespoň jeden krok zpět. Tento princip je zobrazen na obrázku 2.14.



Obrázek 2.14: Architektura rekurentní neuronové sítě. Prostřední obdélník značí vnitřní stav buňky. Smyčka reprezentuje zpětnou vazbu.

Neuronové sítě typu LSTM

LSTM (anglicky *Long Short-Term Memory*) [26] je speciální typ rekurentní neuronové sítě, který by měl řešit problém mizejícího gradientu funkce při učení neuronové sítě a rozšiřovat základní architekturu rekurentních neuronových sítí. Skládá se z buňky a třech druhů bran. Buňka si v každém časovém kroku udržuje hodnotu tzv. shovaného vektoru h a paměťový vektor m, který se stará o kontrolu aktualizací aktuálního stavu a výstupu. Brány LSTM se dělí na vstupní bránu, zapomínací bránu a výstupní bránu. Vstupní brána rozhoduje, zda se bude aktualizovat paměťový vektor. Zapomínací brána bude resetovat paměťový vektor buňky a výstupní brána rozhoduje zda je stav dané buňky viditelný pro její okolí. Díky těmto komponentů je LSTM schopná se naučit závislosti v po sobě jdoucích datech a tudíž je vhodná například ke klasifikaci nebo předpovědi sekvencí. Na rozdíl od základní rekurentní neuronové sítě, nám tedy umožňuje se učit závislosti v sekvencích až o délce paměťového vektoru, to je její hlavní výhoda oproti původní síti, která si pamatovala zpětně pouze jednu hodnotu. Schéma LSTM je zobrazeno na obrázku 2.15, kde x jsou v vstupní data v čase t a h jsou výstupní hodnoty z neuronu.



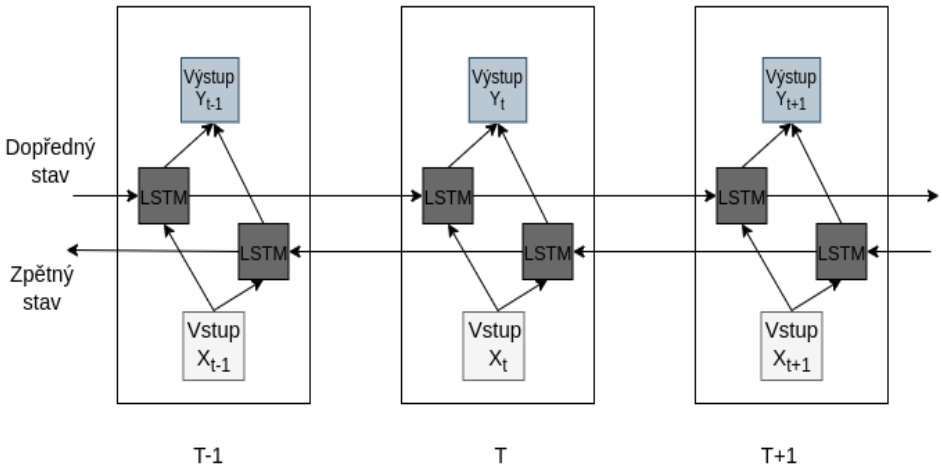
Obrázek 2.15: Architektura LSTM neuronové sítě.

Obousměrné LSTM neuronové sítě

Obousměrné LSTM neuronové sítě (anglicky *Bidirectional LSTM neural networks*) [27] patří mezi další typy neuronových sítí, které rozšiřují rekurentní neuronové sítě. Stejně jako LSTM se používají k problémům, které můžeme reprezentovat pomocí sekvencí. Mezi takové problémy spadá například rozpoznávání a detekce řeči, predikce časových sekvencí nebo překlad vět z jednoho jazyka do jiného.

Díky její struktuře, která je zobrazena na obrázku 2.16 můžeme využívat informace ze vstupní sekvence jak z budoucnosti tak z minulosti. To je jejich hlavní výhoda oproti jiným rekurentním sítím, které nedokáží pracovat s celou sekvencí najednou. Tato funkcionalita je zajištěna pomocí speciálního propojení jednotlivých částí sítě. Nachází se zde tak zvaný dopředný stav, který se stará o propagaci sekvence směrem dopředu do budoucnosti, zpětný

stav naopak bere informace ze sekvence, která se se už na vstupu nacházela a propaguje je zpět, což nám umožňuje tyto hodnoty využít i v minulosti. Tato architektura dosahuje velmi kvalitních výsledků při klasifikaci sekvencí i pro velmi složité vzory.



Obrázek 2.16: Zakladni architektura obousměrné neuronové sítě.

Kapitola 3

Návrh řešení

V této kapitole popíšu a vysvětlím návrh řešení, použité přístupy a zvolené technologie, které na řešení našeho problému aplikujeme. Nejdříve si přiblížíme zvolené technologie a celkový postup, jak by měla výsledná aplikace fungovat z vysokoúrovňového pohledu, poté se podíváme detailněji na detekci lidí a následnou extrakci jejich skeletu, popíšeme si a vysvětlíme transformaci získaných souřadnic na vstup neuronové sítě, samotnou detekci anomálií v takto zpracovaných datech a na konec si popíšeme zpracování těchto dat a vytvoření výsledného videa.

3.1 Popis zvolených technologií

Pro implementaci jsem zvolil programovací jazyk **Python** [28]. Jedná se o dynamicky interpretovaný, vysokoúrovňový jazyk, který nám umožňuje využít různá programovací paradigmata. I když se často řadí mezi skriptovací jazyky, dovoluje nám vývoj různých druhů aplikací. Díky jeho jednoduché syntaxi a široké využitelnosti se z něho stal velice populární jazyk. V našem projektu budeme využívat verzi Python 3.9.2. Tento jazyk jsme mimo jiné zvolili kvůli jeho bohaté nabídce různých rozšiřujících knihoven. Hlavní knihovny, které využijeme při našem řešení si představíme níže.

OpenCV

OpenCV[1] je open source knihovna, jejíž funkce jsou určené především k usnadnění práce v oblasti strojového učení a počítačového vidění. Knihovna obsahuje až 2500 optimalizovaných algoritmů, které zahrnují jak klasické tak takzvané state-of-the-art algoritmy. Tyto algoritmy mohou být použity pro sledování objektů v záznamech videí, extrakci 3D modelů z obrazu nebo k jednoduchým úpravám vstupních snímků.

Tensorflow

Tensorflow[2] je open source knihovna určená pro hluboké učení, má mnoho různých aplikací, ale především se zaměřuje na definování, trénování a nasazení modelů neuronových sítí. Je vyvíjena společností Google a podporuje různé programovací jazyky, ale především Python, Javascript a C++.

[1] OpenCV - https://opencv.org/
[2] Tensorflow - https://www.tensorflow.org/

Keras

Keras[3] je vysokoúrovňové API pro hluboké učení, které běží na Tensorflow platformě pro strojové učení. Je implementované v programovacím jazyce Python s důrazem na zjednodušení a zrychlení možností experimentovat s hlubokými neuronovými sítěmi a strojovým učením. Díky keras můžeme během našeho vývoje neuronové sítě jednoduše využívat tensorové operace na CPU, GPU nebo TPU, ale také můžeme spouštět naše modely v prohlížeči nebo na mobilním zařízení.

Tkinter

Tkinter[4] je vestavěný modul jazyka Python, který se používá k vytváření uživatelského grafického rozhraní pro aplikace, napsané právě v jazyce Python. Patří mezi velice oblíbené a často používané knihovny hlavně díky tomu, že je již zabudovaná v jazyce Python, což velmi urychluje a usnadňuje její použití. Zajišťuje nám objektově orientované rozhraní k nástrojům pro tvorbu oken, tlačítek, ikon a podobně.

Matplotlib

Matplotlib[5] je knihovna určená pro vytváření statických, interaktivních nebo animovaných vizualizací v Pythonu. Zjednodušuje uživateli vytváření a následné úpravy grafů nebo obrázků vysoké kvality, které jsou určeny do technických dokumentací nebo odborných publikací.

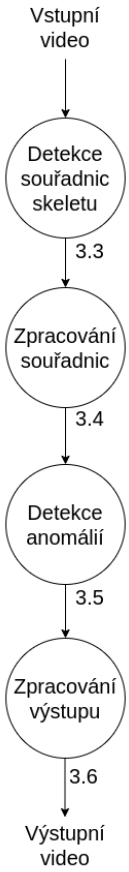
Google Colaboratory

Google Colaboratory[6] je produkt firmy Google, který nám poskytuje prostředí pro spouštění a vytváření kódu v jazyce Python přímo v prohlížeči. Toto prostředí je vhodné především pro strojové učení, analýzu dat a studijní účely. Colab je hostován na Jupyter notebooku a nevyžaduje žádné pokročilé nastavování a zároveň nám zprostředkovává přístup k výpočetním zdrojům v podobě GPU.

3.2 Návrh schématu aplikace

Výsledná aplikace by měla nejdříve nechat uživatele vybrat video, na kterém bude chtít provést detekci anomálií v chůzi chodců. Následně se na tomto videu spustí detekce souřadnic skeletu lidí, které se na videu nachází, ty se poté budou muset předzpracovat pro vstup neuronové sítě. Neuronová síť provede detekci anomálií v takto zpracovaných datech a výsledek detekce se bude muset připravit pro výsledné zpracování a označení na videu, které bude naší aplikací zobrazeno. Tento celý proces je vizualizován na obrázku zde 3.1. Podrobnější popis a vysvětlení jednotlivých kroků se nachází v následujících odstavcích.

[3] Keras - https://keras.io/about/
[4] Tkinter - https://tkdocs.com/
[5] Matplotlib - https://matplotlib.org/
[6] Google Collab - https://colab.research.google.com/



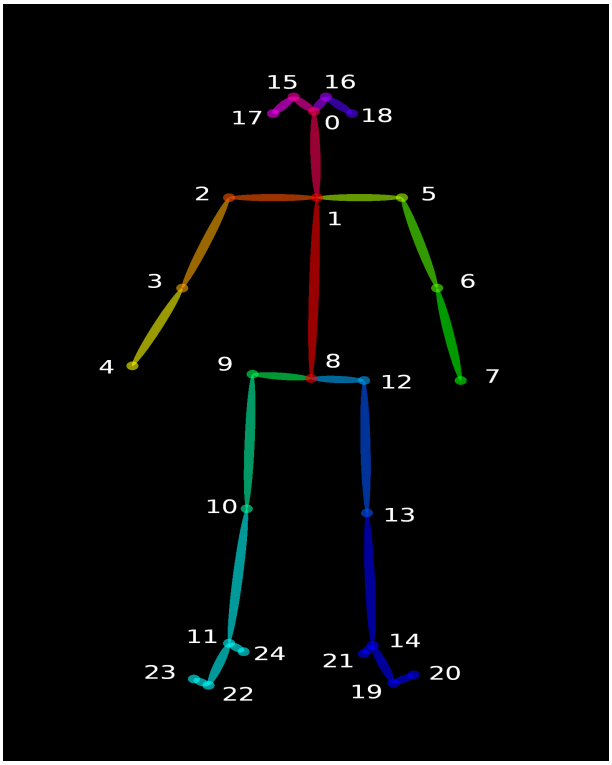
Obrázek 3.1: Návrh schématu aplikace.

3.3 Detekce lidí a extrakce souřadnic skeletu

Detekce lidí a extrahování jejich skeletu je klíčovou částí této práce. Pokud nedokážeme dostatečně kvalitně získat klíčové body lidského těla pro následné zpracování, tak dojde k zanesení velkých chyb a nepřesností, které se významně projeví na výsledku našeho programu.

Základ aplikace bude tvořen již existujícím řešením OpenPose, které dokáže detekovat lidský skelet ve videu v reálném čase a s dostatečnou kvalitou. K dosažení této funkcionality využívá konvoluční neuronovou síť, podrobnější popis jak probíhá samotná detekce je popsán zde 2.3.2. Tato aplikace nám nabízí několik možností, jak si výslednou detekci připravit pro naše další použití. Na výběr je ze dvou druhů modelů, které reprezentují výsledný skelet. Naše aplikace využije model 3.2, který detekuje 25 různých částí lidského těla, další model nabízí detekci pouze 18 bodů. OpenPose nám pro každý tento bod vrací 2D souřadnice a hodnotu pravděpodobnosti, že se tento bod opravdu na těchto souřadnicích nachází. Tato aplikace rovněž nabízí normalizaci extrahovaných hodnot, opět zde máme volbu z více možností intervalů do kterých se budou výsledné hodnoty transformovat. Zde jsme zvolili normalizace hodnot v rozmezí od 0 do 1. Normalizace hodnot pro nás bude důležitá zejména z hlediska následného trénování neuronové sítě, která je schopna dosahovat vyšších přesností pro normalizovaná data než pro data bez normalizace. Výsledky detekce

se uloží do souboru ve formátu *JSON*. Tento formát byl zvolen především kvůli jednoduché manipulaci pomocí jazyka Python.



Obrázek 3.2: Openpose model chodce. Převzato z [29].

3.4 Předzpracování vstupních dat pro neuronovou síť

Po detekci lidského skeletu ze vstupního videa získáme pro každý detekovaný bod hodnotu 2D souřadnic a pravděpodobnost výskytu daného bodu právě na těchto souřadnicích X a Y. Předzpracování všech hodnot nebude nijak brát v potaz hodnotu pravděpodobnosti dané predikce, pro detekci se nebudou používat pouze hodnoty ze snímků, kde není detekována žádná souřadnice lidského těla, tedy na snímku by se žádná osoba neměla nacházet. Díky tomu, že nám OpenPose nabízí i hodnotu pravděpodobnosti nabízí se zde myšlenka na odfiltrování hodnot detekce s nízkou pravděpodobností, v našem případě by toto způsobilo více škody než užitku vzhledem k chybějícím snímkům, které by nebyly zahrnuty v detekci, ale na výsledném videu by se nacházely a tím by nám do zobrazení anomálie zanášely další chybu a detekce by působila méně přesně. Naše aplikace pro detekci anomálií bude využívat pouze hodnoty souřadnic nosu, krku a trupu. Na obrázku modelu 3.2 to jsou body 0, 1, a 8, tudíž si musíme ze získaných hodnot pro každý snímek obrazu extrahovat právě tyto hodnoty. Souřadnice nosu, krku a trupu byly zvoleny, především kvůli tomu, že se jejich souřadnice během chůze různých osob příliš neliší například na rozdíl od rukou nebo ramenou, kdy záleží na dané osobě a jejím stylu chůze. Ale pokud dojde k nějaké anomálii během chůze, tak se hodnoty těchto bodů výrazně liší od normálního průběhu. Také tímto trochu eliminujeme ztrátu dat během detekce, kde například hodnoty souřadnic

rukou nebo nohou nejsou detekovány s dostatečnou přesností, kvůli tomu že se zrovna nacházeli v zákrytu za tělem.

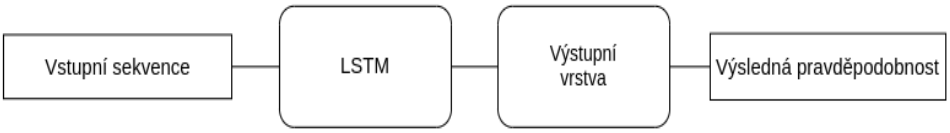
Z hodnot souřadnic si následně vytvoříme sekvence o velikosti 30 snímků, což je 1 sekunda videa. Každá sekvence tedy obsahuje 30 hodnot X a Y pro námi zvolené body. Délka sekvence se s největší pravděpodobností během implementace ještě změní, protože různé anomálie mají různou dobu trvání, tudíž je obtížně určit ideální hodnotu bez jakéhokoliv experimentování. Takto vytvořené sekvence poté použijeme jako vstup pro neuronovou síť, která roztřídí sekvence do tříd, do kterých náleží.

3.5 Detekce anomálií

Po studiu několika existujících řešení 2.3.2 a současných trendech v detekci anomálií a klasifikaci vzorů v datech jsme pro naše řešení zvolili LSTM neuronovou síť. Tento typ neuronových sítí se běžně používá ke klasifikaci nebo detekci anomálií v sekvencích dat, které jsou na sobě závislá.

3.5.1 Neuronová síť navrhnutá pro naše řešení

Pro základní řešení jsem navrhl neuronovou síť, která se skládá ze dvou vrstev. První je vstupní LSTM vrstva, která obsahuje 30 neuronů, přijímající námi navrhnuté sekvence dat. Vstupní sekvence budou ve tvaru $(N \times L \times S)$, kde N je počet sekvencí vytvořených ze vstupního videa, L je počet snímků a S je počet vzorku v dané sekvenci. V našem návrhu bude L nastaveno na 30 snímků a S na 6, protože budeme pracovat s X a Y souřadnicemi 3 bodů. Další vrstva bude výstupní vrstva, která se bude skládat z jednoho neuronu. V této vrstvě bude použita aktivační funkce sigmoid 2.3.1, která nám bude vracet hodnoty pravděpodobnosti, zda se ve vstupní sekvenci nachází anomálie nebo nikoliv. Tato architektura byla inspirována řešením pro klasifikaci sekvencí. Vstupní sekvence byla slova z recenze na film, tyto slova byla zakódována do číselné reprezentace a následně se snažila určit, zda se jedná o kladnou nebo zápornou recenzi[7].



Obrázek 3.3: Návrh základní neuronové sítě.

3.6 Zobrazení výstupu

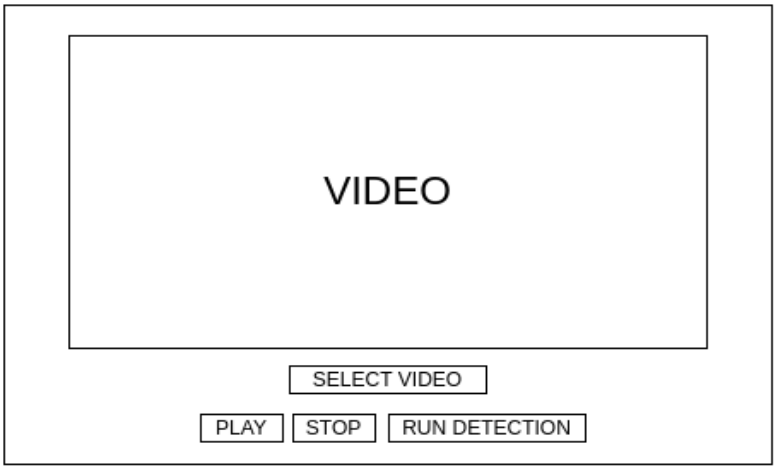
Výstupem aplikace bude video, které na daném chodci pomocí aplikace OpenPose zvýrazní jeho skelet a na základě detekce anomálií námi vytvořenou neuronovou sítí označí snímky s nalezenou anomálií nápisem *ANOMALY*. Jako výstup z neuronové sítě dostaneme označené sekvence o délce 30 snímků, takže ještě před tímto zobrazením budeme muset vypočítat všechny čísla snímků, které obsahují nějaký abnormální jev. To zda se jedná o anomálii

[7] https://machinelearningmastery.com/sequence-classification-lstm-recurrent-neural-networks-python-keras/

nebo nikoliv bude na základě stanoveného prahu a výsledné detekce, pokud je práh menší než hodnota detekce označíme poté snímek jako abnormální. Základní hodnota prahu pro detekci bude nastavena na hodnotu 60, tato hodnota se s největší pravděpodobností bude během implementace a následného experimentování s naším řešením měnit tak, aby výsledek detekce byl co nejpřesnější.

3.7 Návrh grafického rozhraní

Pro výslednou aplikaci budeme také vytvářet jednoduché grafické rozhraní. Toto rozhraní bude obsahovat tlačítka pro ovládání videa, především jeho spouštění a zastavování. Kromě těchto dvou bude nabízet také tlačítko pro vybrání videa, po jehož stisknutí by se mělo zobrazit okno, ve kterém se vybere video. Poslední tlačítko, které se bude v rozhraní nacházet bude pouštět samotnou detekci anomálií. Návrh rozložení tlačítek je vizualizován zde 3.4, při jeho tvorbě jsem cílil především na jednoduché ovládání samotné aplikace.



Obrázek 3.4: Návrh grafického rozhraní.

Kapitola 4

Implementace

Výsledná aplikace byla implementována v programovacím jazyku Python verze 3.9.2. Pro implementaci bylo využito několika knihoven, mezi nejdůležitější patří knihovny *OpenCV*, *tensorflow*, *keras* a *Tkinter*. Pro vizualizaci získaných výsledků a průběhu jednotlivých anomálií jsem využil knihovnu *matplotlib*, k trénování neuronové sítě prostředí *Google Colaboratory*, díky němuž jsem měl přístup ke grafické kartě, která mi zajistila dostatečný výpočetní výkon. Kromě výsledné aplikace jsem mimo jiné vytvořil také dataset, který jsem použil k následnému trénování neuronové sítě.

4.1 Tvorba datasetu pro trénování neuronové sítě

Pro trénování neuronové sítě jsem vytvořil vlastní dataset, který se skládá z 42 minut záznamů videí, která obsahují normální chůzi chodců a anomálie jako skok, dřep a klik. Z těchto videí jsem poté vytvořil celkově 3732 sekvencí o délce 20 snímků. Abychom dosáhli co nejlepších výsledků, tak na těchto videích se vyskytuje několik osob. Tyto osoby se lišily ve své výšce, pohlaví ale i stylu a rychlosti jejich chůze, čímž dělají naše získaná data z těchto videí více obecná. Pomocí aplikace Openpose 2.3.2 jsem extrahoval 2D souřadnice částí lidského těla. Tyto souřadnice jsem uložil do souboru typu *csv*, kde na každém řádku jsou hodnoty právě pro jeden snímek. Poté jsem takto vytvořený soubor prošel a za pomocí natočených videí jsem se snažil označit a roztřídit sekvence s danou anomálií. Tímto procesem vznikly soubory se souřadnicemi a k nim náležející třídy, do které dané snímky patří. Trénování na tomto datasetu tedy probíhá tak, že na vstup neuronové sítě přicházejí hodnoty souřadnic z několika po sobě jdoucích řádcích našeho souboru, počet řádků se odvíjí od délky sekvence s kterou pracujeme. Výstup neuronové sítě je pravděpodobnost, která se snaží odhadnout do jaké třídy dané snímky patří. Při trénování se snažíme detekovat anomální sekvence, které jsem ručně označil jako abnormální, tedy obsahující nějaký námi hledaný vzor.

	Skok	Dřep	Klik	Chůze
Počet sekvencí	514	533	485	2200
Celková délka v minutách	5,7	5,9	5,38	24,5
Procentuální zastoupení	13,77	14,28	12,99	58,95

Tabulka 4.1: Obsah vytvořeného datasetu.

4.2 Popis výsledného schématu aplikace

Aplikace, kterou jsem vytvořil, vykonává postupně stejné kroky k dosažení výsledného videa, jako jsem navrhnul před její implementací. Nějaké drobné změny se projevily například ve finálním vzhledu uživatelského rozhraní, ve finální architektuře neuronové sítě a ve struktuře dat, které dáváme na vstup neuronové síti. Tyto změny však nemají vliv na postup a schéma celkové aplikace, které bylo navržen a zobrazeno zde 3.1. Celý tento proces je popsán v návrhu aplikace 3.2. Příklad výsledného výstupu je zobrazen na obrázku 4.2.

Výsledný program je implementován ve třech souborech, který každý obsahuje jednu třídu, která implementuje danou funkcionalitu. V souboru *main.py* se nachází třída *App*, která nám definuje grafické rozhraní a obsahuje veškerou logiku pro obsluhu tlačítek, které se nachází v uživatelském rozhraní, také zajišťuje zobrazení snímků námi zvoleného videa. Je to hlavní část naší aplikace, která řídí a spouští veškeré akce.

Třída *DisplayVideo*, která se nachází v souboru *displayvideo.py* implementuje otevření a čtení videí. K dosažení této funkcionality se používá knihovna *OpenCV* a to především její funkce *VideoCapture*, pomocí níž se vytvoří objekt videa. Poté se na tento objekt volá funkce *read*, která nám vrací jednotlivé snímky. Takto získané snímky se poté vrací na další zpracování do třídy *App* a *AnomalyDetector*.

V posledním souboru *anomalydetector.py* se nachází třída *AnomalyDetector*, která se stará o samotnou detekci anomálií z videa. Implementuje spouštění detekce skeletu pomocí *OpenPose*, následně získané data předzpracuje pro vstup neuronové sítě, spustí detekci a připraví její výsledek pro zobrazení na výstup. Během tohoto procesu aplikace vytváří dočasné pomocné soubory ve složce *tmp*. Tato složka obsahuje výsledky detekce skeletu před jejím dalším zpracováním ve formátu *JSON*. Tyto soubory se načtou a vytvoří se z nich jedno ucelené pole sekvencí pro vstup neuronové sítě. Po zpracování neuronovou sítí se podle jejích výsledků vytvoří pole, které obsahuje čísla snímků videa, které přesahují námi stanovený práh pro detekci anomálie. Takto vytvořené pole se poté využije v třídě *App* a to ve funkci *update*, kde se vytváří výsledné snímky, pokud se zrovna dané číslo snímku nachází v námi vytvořeném poli, na výsledný snímek se vloží nápis *ANOMALY*.

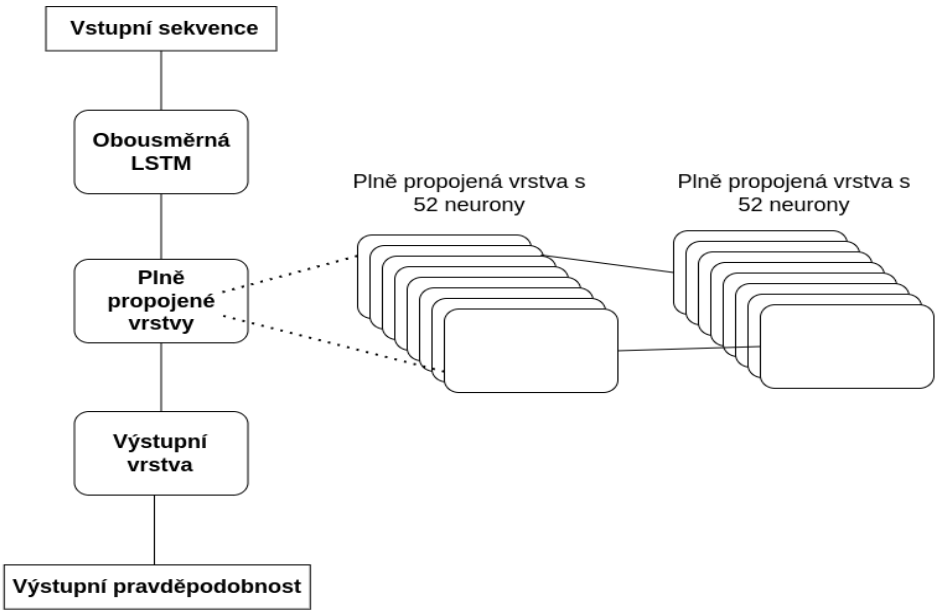
4.3 Implementace neuronové sítě pro detekce anomálií

Výsledná neuronová síť, která je použita k detekci se výrazně liší od původně navrhované neuronové sítě 3.5.1, která obsahovala pouze dvě vrstvy. Taková neuronová síť se v průběhu práce projevila jako nedostatečná, vzhledem k množství vzorů, pro které jsme neuronovou síť trénovali.

Během implementace jsem také změnil délku sekvence z 30 snímků na 20 snímků. Tato změna je z důvodu vizualizace výsledků detekce na videích. Při délce sekvence 30 snímků se výsledné videa jevila jako velmi nepřesná, to převážně v případech, kdy anomálie byla mnohem kratší než jedna sekunda, ale naše neuronová síť označila celou sekvenci jako abnormální. Také jsem experimentoval s kratšími sekvencemi, ale ty nám neumožňovaly dostatečnou přesnost detekce, protože neobsahovaly zásadní části průběhu anomálií. Experimenty, kterými jsem nakonec vybral délku 20 jsou popsány zde 5.1.2.

Jako vstupní vrstva je místo klasické *LSTM* neuronové sítě použita *Obousměrná LSTM neuronová síť*, jejíž princip jsem vysvětlil zde 2.3.3. Tato první vrstva bude obsahovat 26 neuronů, počet neuronů odpovídá délce vstupních sekvencí. Její výstup bude následně směrován do dalších 2 vrstev. Tyto vrstvy jsou plně propojeny mezi sebou. Obsahují každá 52 neuronů, cože je dvojnásobný počet neuronů předchozí vrstvy. Všechny tyto vrstvy k

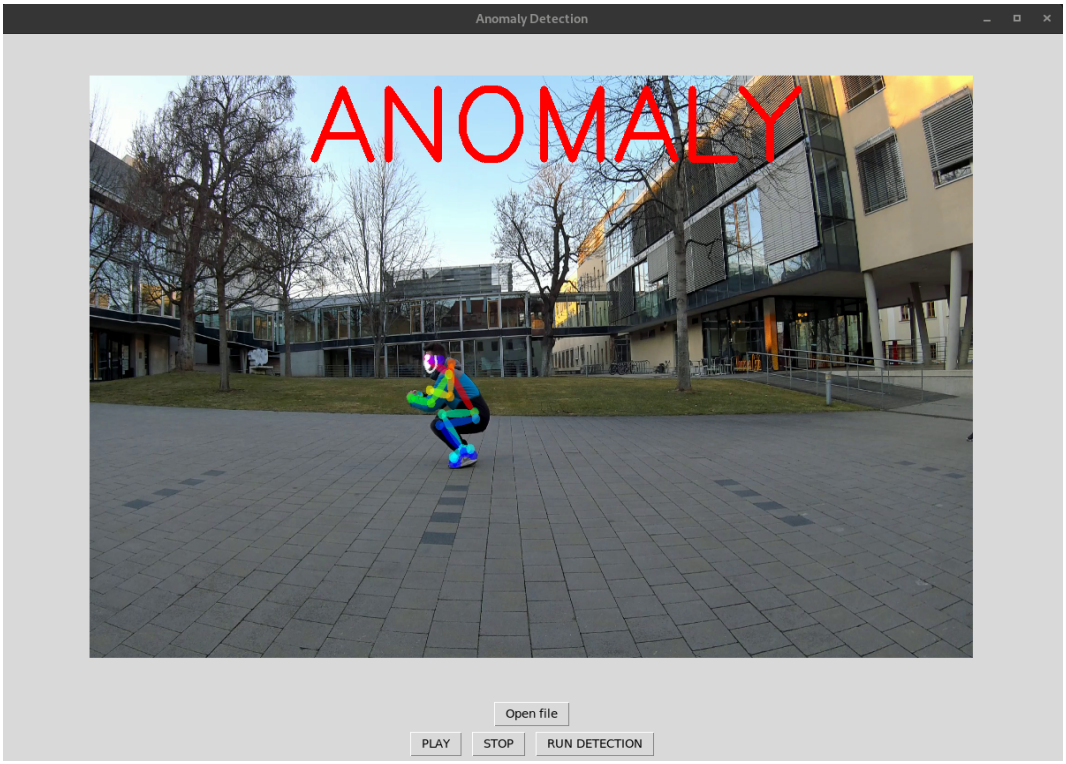
jejím aktivacím používají aktivační funkci *relu* 2.3.1. Poslední výstupní vrstva obsahuje čtyři neurony a k aktivaci používá funkci softmax 2.3.1, která se používá ke klasifikaci do více než jedné třídy.



Obrázek 4.1: Architektura neuronové sítě.

4.4 Tvorba grafického rozhraní

Výsledné grafické uživatelské rozhraní se příliš neliší od původního návrhu 3.4. Rozhraní obsahuje tlačítko *Play* a *Stop*, které slouží ke spuštění a zastavení vybraného videa. Tlačítko *Select video* slouží ke zvolení videa, na kterém proběhne detekce anomálií. Po jeho stisknutí se uživateli zobrazí vyskakovací okno, kde si bude moc jednoduše vybrat dané video ve svém souborovém adresáři. Jako základní cesta pro vyhledání videa je zvolená složka, ve které je umístěna aplikace, pro detekci lze zvolit pouze soubory, které končí koncovkou *mp4* nebo *avi*. Poslední tlačítko, které se v rozhraní nachází je *Run detection*, které musí být stisknuto před spuštěním videa, pokud se tak nestane aplikace přehraje pouze vybrané video bez spuštění jakékoliv detekce. Příklad výsledného výstupu, který je zobrazen v grafickém rozhraní je na obrázku 4.2.



Obrázek 4.2: Příklad výstupu z aplikace.

Kapitola 5

Experimenty

V této kapitole provedu různé experimenty a vyhodnotím úspěšnost mého řešení. Experimenty jsem rozdělil na dvě části, první se zabývá samotným experimentováním s různými neuronovými sítěmi a jejich úspěšnostmi detekce různých druhů anomálií nebo jejich kombinací. V podstatě zde zachycuji postupný vývoj neuronové sítě, který nastal během její implementace a průběžného testování. Některé neuronové sítě, které jsem používal byly přesnější a úspěšnější než jiné a to vlivem několika faktoru, například postupným přidáváním druhů anomálií, které jsem se snažil detekovat jsem musel vytvářet robusnější řešení, které se dokáže naučit více vzorů. Druhá část se poté bude zabývat experimentování s finální verzí natrénované neuronové sítě, kdy budu vyhodnocovat její přesnost a kvalitu detekce pomocí naší aplikace na různých videích.

5.1 Experimenty s neuronovými sítěmi pro detekci anomálií

Experimenty v této části probíhají na mnou vytvořeném datasetu, kdy jsem ho rozdělil na trénovací a testovací data. Trénovací data obsahovala 80% a testovací zbylých 20% našeho datasetu. Poté jsem navrhnul architekturu neuronové sítě, spustil jsem trénování. Vyhodnocení probíhalo na základě hodnoty *loss* a *accuracy*, jak na trénovacích tak na testovacích datech. Naším cílem bylo dosáhnout co nejnižší hodnoty *loss* 2.3.1 a co největší hodnoty *accuracy* na obou druhů datech.

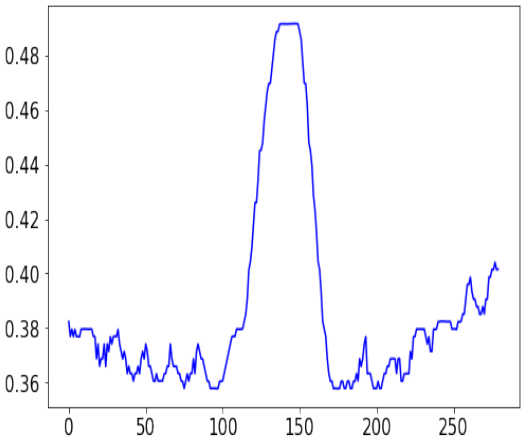
5.1.1 Základní neuronová síť

Pro první experimentování jsem použil neuronovou síť, která byla složena ze 2 vrstev. První byla LSTM vrstva, která obsahovala 30 neuronů, druhá byla vrstva výstupní s jedním neuronem a aktivační funkcí sigmoid. Jedná se o architekturu, kterou jsem navrh před začátkem implementace a experimentování. Její vizualizace je zde 3.3. Délka sekvence je rovněž stejná jako je navrhnutá hodnota a to 30 snímků.

Detekce dřepu podle jedné souřadnice

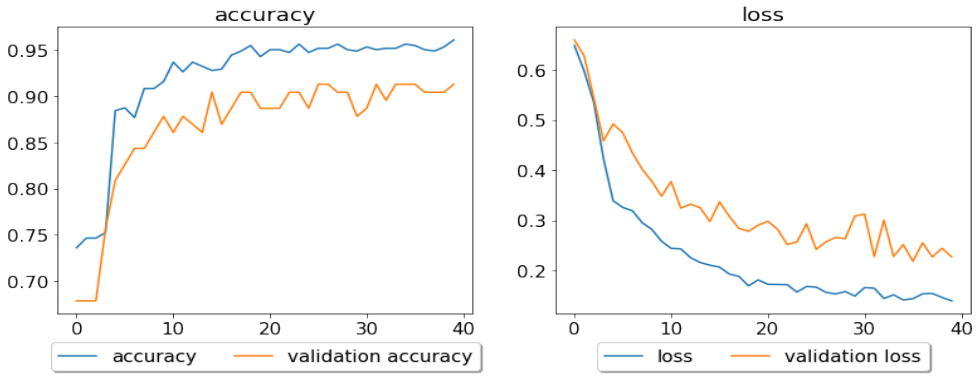
Jako první anomálii, kterou jsem se snažil pomocí neuronové sítě detekovat byl dřep. K prvnímu experimentování jsem využil pouze souřadnici hlavy a to konkrétně její Y hodnotu. Její průběh v čase má charakteristický průběh hodnot a lze jednoduše určit v jaké bodě dochází ke změně z chůze na dřep. Pro názornost jsem průběh této souřadnice vizualizoval na grafu zde 5.1. Na první pohled se zdá, že hodnoty znázorněné v grafu by měli vypadat

spíše opačně, tedy měly by nejdříve klesnout na nižší hodnoty a poté se vrátit do normálu. Zde tomu tak není, protože jsme při detekci pomocí *OpenPose* použili normalizaci hodnot na interval od *0* do *1* s tím, že hodnoty *(1,1)* se nachází v pravém dolním rohu a *(0,0)* v levém horním.



Obrázek 5.1: Hodnota Y v čase během chůze narušené dřepem.

Při detekci anomálie v podobě dřepu na základě jedné souřadnice si vedla tato neuronová síť velice dobře. Během trénování se hodnota přesnosti detekce pohybovala až okolo *95%* a hodnota ztrátové funkce byla ke konci trénování na hodnotách okolo *12%*. Pro testovací data byla přesnost o něco nižší, ale stále se pohybovala okolo *90%*. Průběhy všech hodnot během trénování a validace jsou uvedeny zde 5.2.

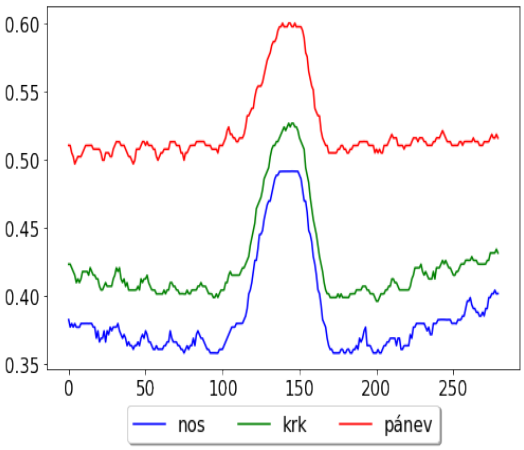


Obrázek 5.2: Průběh trénování.

Detekce dřepu na základě více bodů skeletu

V následujícím experimentu jsem k souřadnicím hlavy pohybujícího se chodce přidal ještě hodnoty krku a pánve. Jejich ukázkový průběh je zobrazen zde 5.3. Tím jsem získal více hodnot pro trénování detekce, zároveň se tak změnila i sekvence, která slouží jako vstup pro neuronovou síť. Používané sekvence jsou nyní nastaveny na tvar, který byl navrhnut před implementací, jedná se tedy o $(N \times L \times S)$, kde L je 30 snímků a S je 6 vzorků pro jeden snímek. Tyto body na lidském těle by se měli během dřepu oproti normální chůzi zásadně

lišit, souřadnice X, by měla zůstat téměř neměnná po celou dobu dřepu a hodnoty Y by se na rozdíl od běžné chůze měli výrazně změnit a opět vrátit na původní hodnoty začátku pohybu.

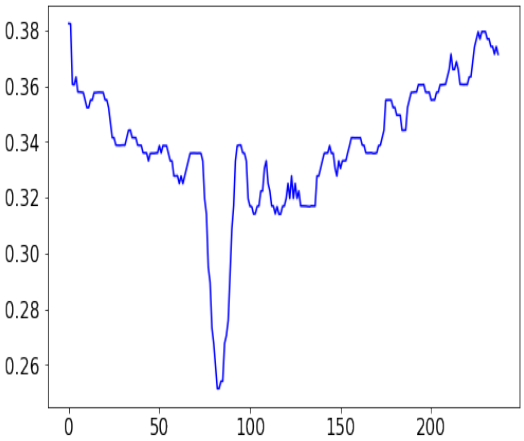


Obrázek 5.3: Hodnoty třech souřadnic Y v čase během chůze narušené dřepem.

Po přidání dalších souřadnic částí lidského těla se predikce pomocí naší neuronové sítě téměř nezměnila a výsledky, které jsem získával jsou skoro totožné jako při použití pouze jedné souřadnice. A proto jsem se rozhodl pomocí stejné neuronové sítě zkusit detekovat více jak jednu anomálii. K trénování jsem tedy navíc použil i data určená pro učení detekce skoku.

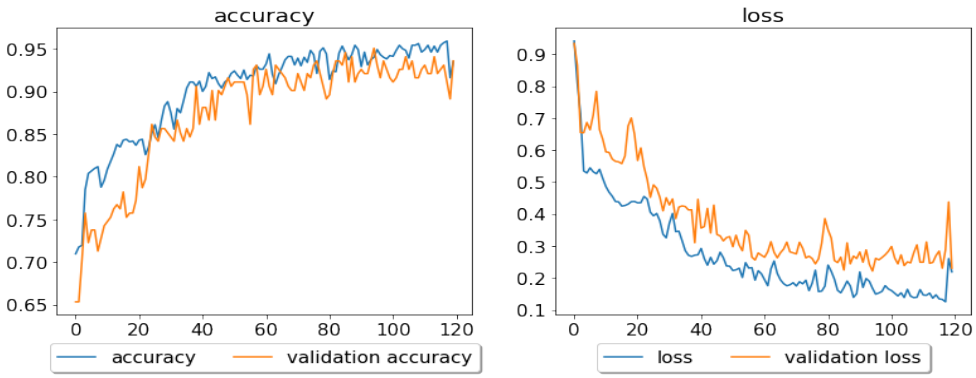
Detekce dřepu a skoku

Naše neuronová síť ukázala dobré výsledky při detekci jedné anomálie, proto se v tomto experimentu pokusím použít stejnou architekturu pro detekci dvou anomálií. Druhá anomálie bude skok, jehož průběh je zobrazen na zde 5.4. Oproti dřepnutí se liší ve směru změny, její velikosti a doby trvání, kdy skok obvykle trvá kratší časový interval.



Obrázek 5.4: Hodnota Y v čase během chůze narušené skokem.

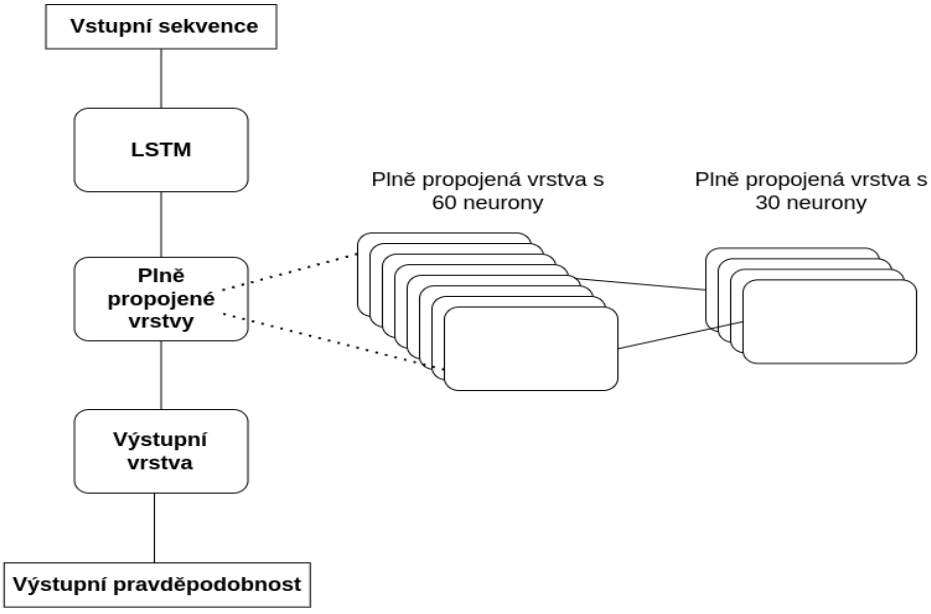
Při trénování si naše neuronová síť opět nevedla vůbec špatně. Výsledné hodnoty *loss* a *accuracy* jsou dost podobné hodnotám při učení detekce pouze jedné anomálie.



Obrázek 5.5: Průběh trénování.

5.1.2 LSTM neuronová síť rozšířená o plně propojené vrstvy

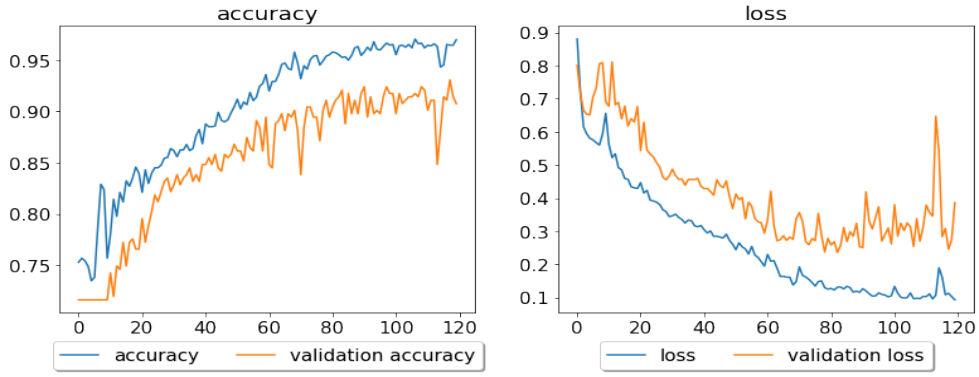
Následující experimenty budou probíhat s neuronovou sítí, jejíž architektura bude obsahovat 4 vrstvy. Neuronová síť je vizualizována na obrázku 5.6. První bude LSTM vrstva s 30 neurony, na ní budou navazovat 2 plně propojené vrstvy 2.3.1, které budou obsahovat 60 a 30 neuronů. V těchto vrstvách bude použita aktivační funkce relu 2.3.1. Poslední výstupní vrstva bude obsahovat 3 neurony a aktivační funkci softmax 2.3.1. Vstupní sekvence bude zkrácena z 30 snímků na 20, protože vzor skoku v datech trvá kratší časový interval, takže ve výstupní videu detekce by se výsledky jevily jako velmi nepřesné.



Obrázek 5.6: Architektura neuronové sítě.

Detekce dřepu a skoku

V prvním experimentu s nově navrženou neuronovou sítí jsem se pokusil stejně jako v posledním experimentu provést detekci dřepu a skoku a použil jsem opět pouze 3 souřadnice lidského těla, tudíž se nebude měnit ani vstupní sekvence.

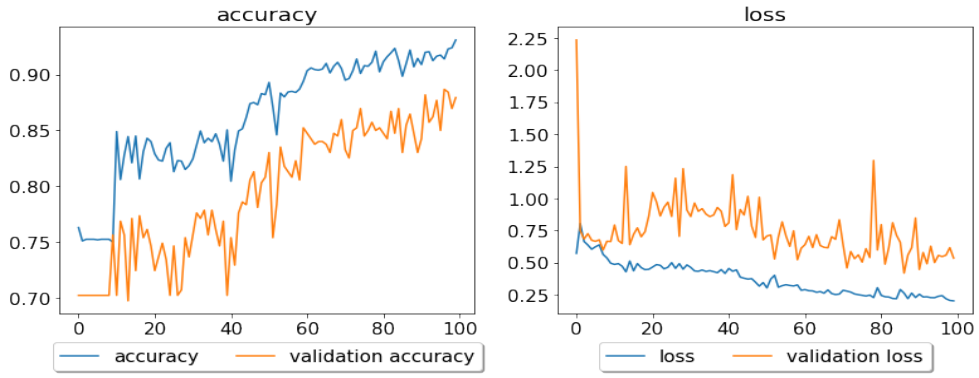


Obrázek 5.7: Průběh trénování.

Výsledek experimentu je podobný jako výsledek předchozího, jediný malý rozdíl, kterého si mezi těmito grafy můžeme všimnout je větší rozdíl mezi trénovacími a validačními daty, pro obě sledované hodnoty.

Detekce dřepu a skoku pomocí kratších sekvencí

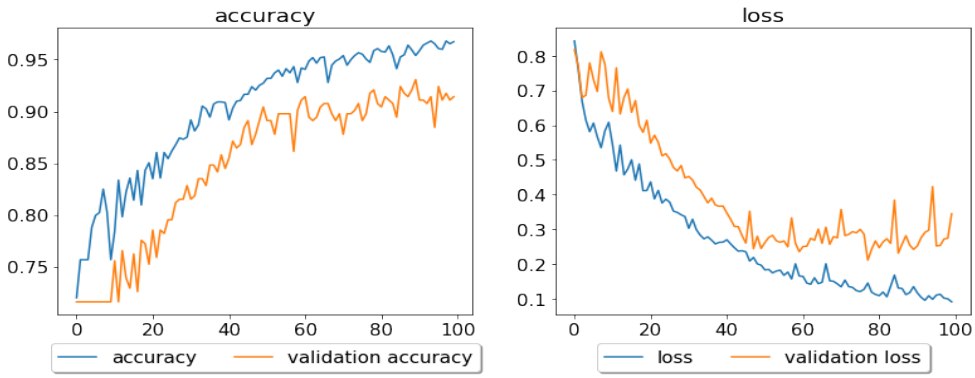
Jako další experiment provedeme rozšíření vstupní sekvence o hodnoty ramen, loktů a boků. Také se pokusím zjistit, jaké výsledky bude tento model mít pro sekvence kratší než 20 snímků a to konkrétně 15. Sekvence je tedy ve tvaru $(N \times 15 \times 18)$. Výsledek trénování je zobrazen na následujícím grafu 5.8.



Obrázek 5.8: Průběh trénování.

Z výsledků je vidět, že tento model se nedokázal dostatečně přesně naučit detekovat anomálie v sekvencích o délce 15 snímků. Výsledné hodnoty *loss* jsou příliš velké a během trénování se téměř nemění, nejspíše zde dochází k overfittingu, tento jev je popsán zde 2.3.1. Došlo k tomu asi díky tomu, že sekvence o 15 snímcích není dostatečná pro popsání vzoru pro dřep. Jak můžeme vidět zde 5.1, celkový průběh anomálie může trvat až cca 125 snímků.

Proto jsem se pokusil pomocí stejného nastavení provést tento experiment znovu, tentokrát s délkou sekvence 20 snímků a zjistit, jak se tyto dva modely budou lišit.



Obrázek 5.9: Průběh trénování.

Po provedení tohoto experimentu a porovnání výsledků trénování na dvou různých délkách sekvencí jsem zjistil, že pro detekci dřepu a skoku bude mnohem více vhodná sekvence o délce 20 snímků, tudíž tuto hodnotu budeme používat i během dalších experimentů.

5.1.3 Obousměrná neuronová síť

K dalšímu vylepšení neuronové sítě pro detekci anomálií v chůzi chodců budeme experimentovat s obousměrnou LSTM neuronovou sítí. Neuronová síť, použitá v následujících experimentech se bude skládat z celkem 4 vrstev a je zobrazena zde 4.1. První bude obousměrná LSTM vrstva, která bude obsahovat 26 neuronů. Toto množství neuronů bylo zvoleno podle počtu hodnot v jedné vstupní sekvenci pro 13 bodů lidského těla. Další dvě vrstvy jsou plně propojené vrstvy, které budou obě obsahovat 52 neuronů. U těchto vrstev bude stejně jako v předchozí architektuře použita aktivační funkce relu. Poslední výstupní vrstva bude v závislosti na počtu anomálií, které se snažíme detekovat buď, obsahovat 3 nebo 4 neurony. Aktivační funkce zde bude softmax.

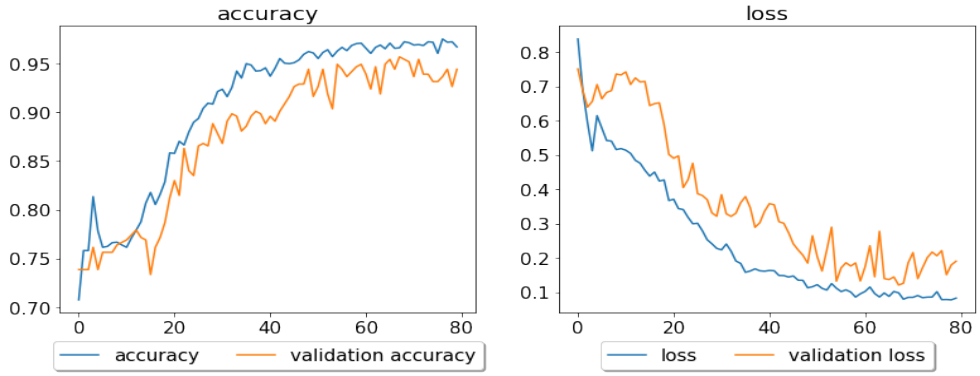
Detekce dřepu a skoku

V tomto experimentu se pokusím pomocí obousměrné LSTM neuronové sítě provést detekci skoku a dřepu, stejně jako v předchozích případech, nyní ale použijeme 13 souřadnic lidského těla. Z původních 9 provedeme rozšíření o hodnoty souřadnic dlaní a chodidel. V *OpenPose* modelu 3.2 se jedná o body 4, 7, 11 a 14.

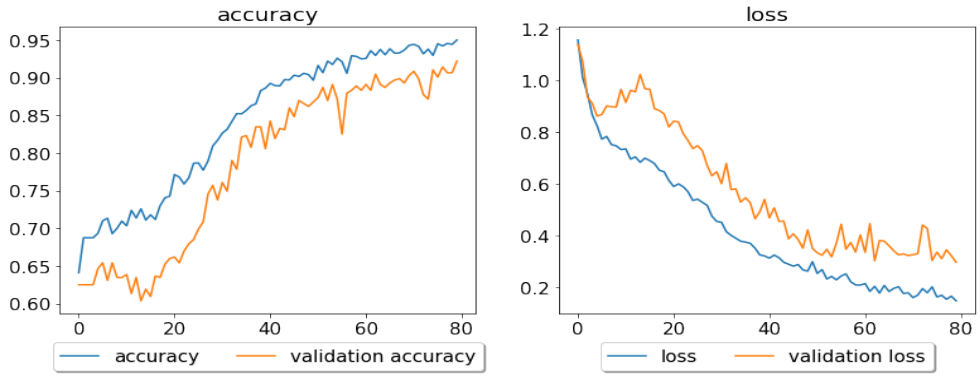
Tento nový model se stejně jako dva předchozí osvědčil při detekci dvou anomálií a během trénování dosahoval kvalitních výsledků.

Detekce dřepu, skoku a kliku

Nyní se pokusím vytvořit model, který dokáže detekovat tři různé anomálie najednou. Pomocí této architektury neuronové sítě se mi podařilo provést detekci dvou anomálií, proto se pomocí stejné neuronové sítě pokusím přidat další. Průběh kliku by měl být dostatečně odlišný od ostatních anomálií, zároveň by se měl ideálně vejít do sekvence o délce 20 snímků, jeho délka trvání bude hodně podobná dřepu.



Obrázek 5.10: Průběh trénování.



Obrázek 5.11: Průběh trénování.

Výsledky trénování detekce tří anomálií dosahují podobně vysokých hodnot jako trénování detekce pouze dvou. Tento model použiji v další části této kapitoly a to k testování celkové funkcionality aplikace na několika videích.

5.2 Testování výsledné aplikace

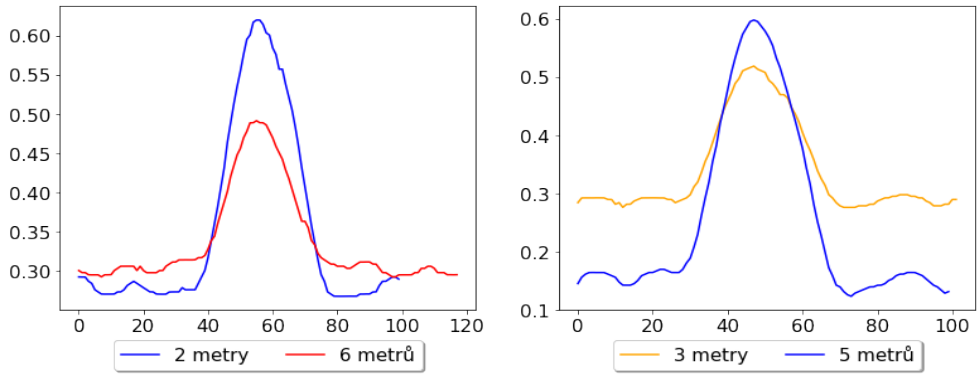
V této části experimentů se zaměříme na výslednou aplikaci, vyzkoušíme si pomocí ní detekci anomálií na různých možných scénářích a vyhodnotíme si její přesnost. První část této kapitoly budeme věnovat vlivu vzdálenosti chodce od kamery na výsledky detekce naší aplikací. Další experimenty se poté zaměří na různé směry pohybu chodce, například diagonálně, místo běžného přecházení ze strany na stranu.

5.2.1 Přesnost detekce v závislosti vzdálenosti chodce od kamery

Při prvním experimentováním s výslednou aplikaci jsem testoval přesnost detekce v závislosti na vzdálenosti chodce od kamery. Pro tento účel jsem natočil několik videí v různých vzdálenostech. Vzdálenosti jsem zvolil 2, 3, a 4 metry. Hodnoty cca do 3 metrů by měli odpovídat hodnotám z videí získaných spíše ve vnitřních prostorách, také náš dataset byl vytvořen zhruba za těchto podmínek, takže se dá očekávat vyšší úspěšnost právě pro tyto hodnoty. Delší vzdálenosti by poté odpovídaly například nějakým kamerám umístěným ve větších chodbách nebo venkovních prostorech. Pro ukázku, jaký má vliv vzdálenost na prů-

běh anomálie jsem vytvořil graf 5.12, který zobrazuje Y hodnotu hlavy během dřepu v testovacích videích. Z tohoto grafu je jednoznačně vidět, že vzdálenost významně ovlivňuje velikosti hodnot souřadnic. Pokud je chodec ve vzdálenosti 2 metrů nebo 6 metrů, budou se hodnoty během anomálie lišit téměř dvojnásobně.

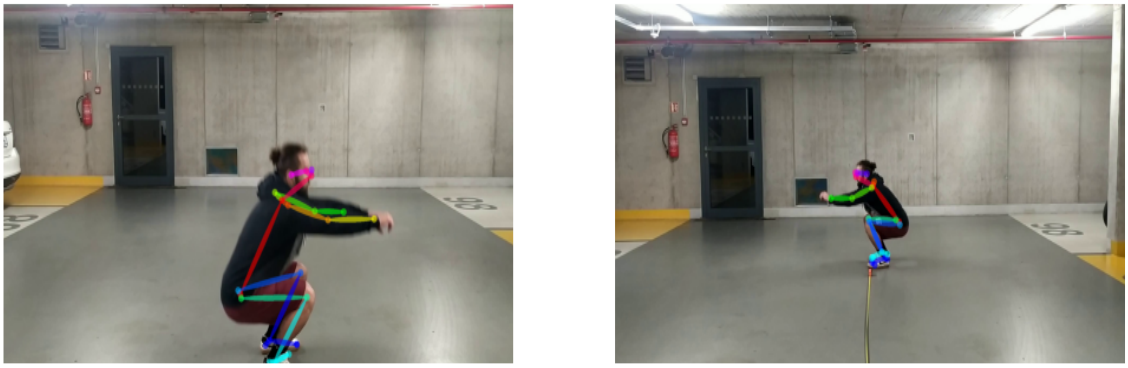
Nejdříve jsem tedy provedl experimentování se záběry ze vzdálenosti 2 metrů od kamery. Video, které jsem pro tyto účely vytvořil obsahuje každou ze tří anomálií přesně desetkrát. Pro validaci jsem nejdříve spustil samotná videa a vizuálně vyhodnocoval úspěšnost detekce na základě video výstupu. Při tomto experimentu si námi vytvořený model vedl velmi dobře, když detekoval anomálie s přesností až *93%* a během testování nedošlo k žádnému špatnému označení chůze, jako anomálie. Po zkontrolování, zda probíhá samotná detekce správně, jsem sledoval, zda jsou dané anomálie správně označovány do příslušných kategorií. Správná detekce anomálie dřepu proběhla v tomto případě vždy, její průměrná hodnota byla *60%*, maximální detekovaná hodnota byla *87%* a nejmenší *45%*. Nejlepších hodnot dosahovala detekce kliku, která proběhla správně v každém případě a dosahovala průměrně nejvyšších hodnot. Průměrná hodnota pravděpodobnosti, že se jedná o klik byla *88,3%*. Poslední anomálie je skok, pro který jako jediný došlo ke 2 selháním detekce. Jeho průměrná hodnota detekce byla byla také nejnižší a to *51%*, pokud započítáme i dvě selhání, bez nich by tato hodnota byla *63%*.



Obrázek 5.12: Srovnání průběhu hodnot jedné souřadnice lidského těla, v porovnání v různých vzdálenostech od kamery.

Při videu ze vzdálenosti 3 metrů si opět model nevedl špatně, jako anomálii označil 24 případů ze 30, což znamená přesnost *80%*. U tohoto videa se již vyskytnulo pár případů, kdy byla chůze označena chybně jako anomálie, to se stalo přesně ve dvou případech během celého 4 minutového videa. Při sledování, zda je anomálie zařazena do správné kategorie, jsem dostal pro dřep průměrnou hodnotu detekce *55%*, bez selhání detekce. Tato anomálie měla největší chybovost v tomto experimentu, kdy byla pětkrát neoznačena. Další anomálií je klik, který měl průměrnou detekci *85%* a ani jednou detekce neselhala. Detekce skoku byla správná pro 7 případů, průměrná hodnota detekce byla *58%* bez selhání.

Poslední video, které obsahuje záznam ze vzdálenosti 4 metrů mělo opět o něco menší přesnost něž předchozí videa. Přesnost se pohybovalo okolo *73%*. Během testování se ale zvýšil počet špatně označených sekvencí jako anomálie. Tento jev se zde objevil sedmkrát. Také se výsledné video s detekcí jevilo o něco méně přesné než ty předchozí. Přesnost detekce dřepu byla v tomto experimentů velmi nízká, detekce selhala pro 5 případů z 10, a průměrná hodnota byla bez selhání *42%*. Detekce kliku si opět vedla nejlépe, nedošlo



Obrázek 5.13: Rozdíl vzdálenosti na snímku z videa.

k žádnému selhání a průměr detekce byl *70,7%*. Během detekce skoku došlo ke 2 selhání, přesnost detekce se opět o snížila a to na *45,8%*.

Vzdálenost	Celková přesnost	Přesnost detekce dřepu	Přesnost detekce skoku	Přesnost detekce kliku
2 metry	93%	60%	63%	88,3%
3 metry	80%	55%	58%	85%
4 metry	73%	42%	45,8%	70,7%

Tabulka 5.1: Shrnutí experimentu pro jednotlivé vzdálenosti.

Experimenty jsou shrnuty v tabulce 5.1, kde jsou uvedeny získané hodnoty pro každou anomálii i celkovou přesnost pro danou vzdálenost. Hodnoty jsou procentuální detekce pro danou anomálii.

5.2.2 Přesnost detekce v závislosti na úhlu chůze vzhledem ke kameře

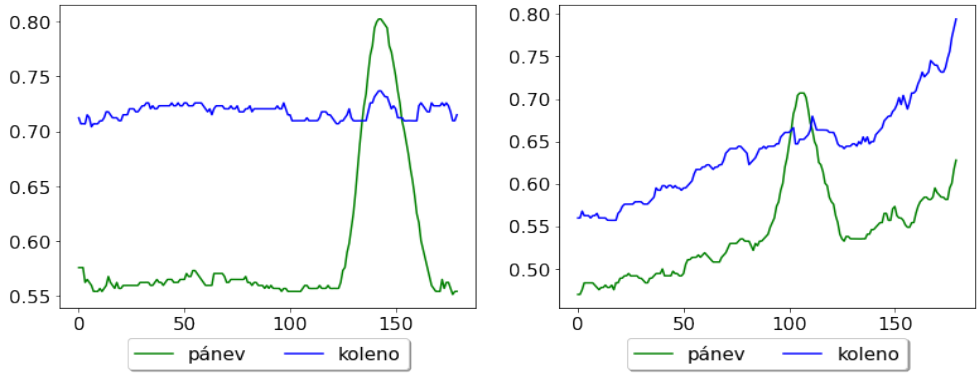
Chůze po diagonále

Během tohoto experimentu pracuji se scénářem, kdy se chodec pohybuje z jednoho rohu do druhého tzv. diagonálně a nikoliv pouze zleva do prava. Opět jsem vytvořil testovací videa a znázornil na grafu 5.14 průběh Y souřadnice pro koleno a pánev chodce. Je zde porovnání chůze po diagonále a chůze pouze ze strany na stranu. Na první pohled je vidět, že se tyto průběhy od sebe liší, Y souřadnice při chůzi po diagonále průběžně roste nebo klesá a nedrží si stabilně jednu hodnotu.

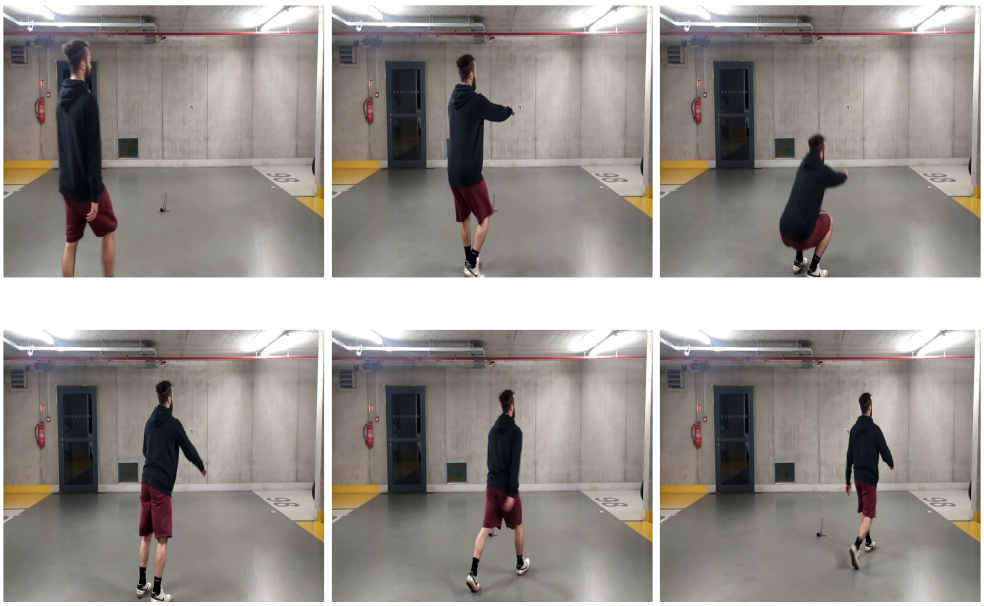
Průběh samotné anomálie, není až tak směrem pohybu ovlivněn, protože při anomáliích, které jsme zvolili k detekci dojde v podstatě vždy k zastavení. Tomu také odpovídají výsledky experimentování, kdy se získané hodnoty zase tolik neliší hodnotám získaným z předchozího experimentu a celkově jsou velmi dobré. Přesnost detekce a její úspěšnost je zde spíše znovu ovlivněna vzdáleností od kamery, než směrem, kterým se zrovna chodec pohyboval.

Chůze proti kameře

Během tohoto experimentu budeme uvažovat situaci, kdy se chodec pohybuje čelem nebo zády přímo ke kameře. Myslím si, že během tohoto experimentu nedostaneme příliš přesné



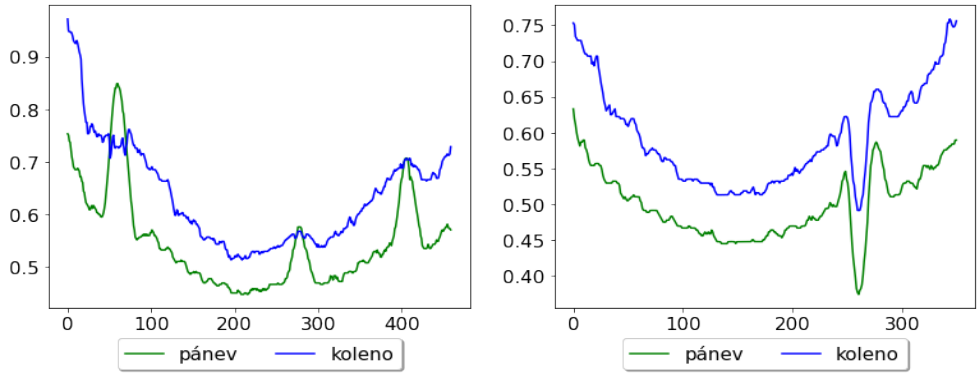
Obrázek 5.14: Vlevo se nachází průběh souřadnic lidského těla při chůzi z jedné strany na druhu. Vpravo jsou hodnoty pro chůzi po diagonále.



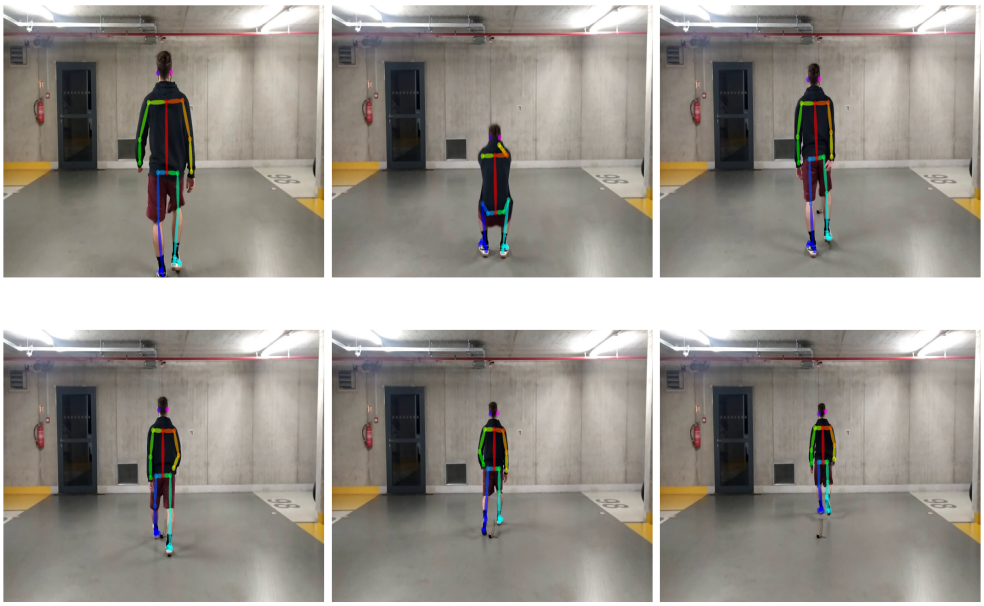
Obrázek 5.15: Ukázka chůze po diagonále ve videu.

hodnoty detekce, protože souřadnice bodů, které používáme k detekci budou dosahovat velmi odlišných hodnot než při běžné chůzi ze strany na stranu. Tento typ pohybu je zobrazen na obrázku 5.17, který obsahuje několik snímků takového videa. Pro ukázku jsem také vytvořil graf 5.16, který zobrazuje průběh dvou souřadnic během tohoto pohybu. Na obrázku vlevo je chůze narušená dřepem, na druhém je chůze se skokem. Z grafu je vidět, že se průběh anomálie už znatelně liší od normálu a to především během jeho začátku a konce. Toto je nejlépe pozorovatelné u skoku, kdy se pohyb chodce nemusí zastavit, tudíž se jeho konečná hodnota už nevrátí do původní.

Během experimentování se naplnila moje očekávání a přesnost detekce se pohybovala okolo *55%*, kdy natrénovaný model dokázal detekovat některé dřepy a kliky, ale nedokázal detekovat ani jeden skok. Navíc se zde objevilo velké množství špatných označení sekvencí, které neobsahovaly žádnou anomálii a to především ke konci videa, když se člověk nacházel čelem v blízkosti kamery. V takové situaci náš model detekoval anomálii skoku.



Obrázek 5.16: Srovnání průběhu hodnot souřadnice bodu lidského těla při chůzi přímo naproti kameře. Vlevo je chůze narušená dřepem, vpravo je chůze se skokem.



Obrázek 5.17: Ukázka chůze chodce přímo naproti kameře.

5.2.3 Přesnost detekce během experimentů

V této části se zaměřím na video výstup, který získáváme z naší aplikace. Konkrétně na jeho přesnost označení dané abnormální sekvence. Během testování aplikace jsem si nemohl nevšimnout určité nepřesnosti ve výstupu. Například pokud se daný chodec pohybuje a těsně ještě před tím, než začne měnit svůj pohyb z chůze na danou anomálii, zobrazí se na videu nápis značící nějaký abnormální jev. Myslím si, že tato nepřesnost je způsobena několika faktory. Jeden z nich je délka sekvencí, které vytváříme. Pokud bychom nastavili její hodnotu například na 30 snímků, tedy 1 sekundu videa, byla by tato nepřesnost ještě o něco více znatelná. Bohužel zde není už více možností pro zlepšení, protože během trénování a experimentování s neuronovými sítěmi jsem dospěl k tomu, že délka 20 snímků je nejmenší možná, při které stále dochází k přesné detekci. Další faktor je vytváření sekvencí ze vstupu. Pokud zrovna nastane situace, kdy je na konci sekvence nějaký útržek začátku anomálie, dochází naopak ke zpožděnému zobrazení, kdy se zbytek promítne až v další sekvenci, tudíž

může dojít ke zpoždění až cca 20 snímků, což ve výsledku je 0,66 sekundy. Během tohoto času se může stát, že celý děj anomálie skončí, například skok má podobnou dobu trvání. Výsledek se bude tedy jevit nepřesně i když detekce proběhla v pořádku.

Kapitola 6

Závěr

Cílem práce bylo vytvořit aplikaci, která dokáže na vstupních videích detekovat anomálie v chůzi chodců.

Během práce jsem nejdříve nastudoval potřebnou teorii týkající se neuronových sítí a strojového učení, poté jsem se zaměřil na již existující podobná řešení. Veškeré tyto získané znalosti a informace jsem následně sepsal v první části mé bakalářské práce. Následně jsem vytvořil návrh, jak by mohla výsledná aplikace pracovat a tento vytvořený návrh jsem se snažil implementovat, během tohoto procesu jsem získával další rozšiřující znalosti a průběžně původní návrh upravoval a měnil až do výsledné podoby naší aplikace, která dokáže detekovat 3 anomálie a to skok, dřep a klik. K naimplementovanému programu jsem následně vytvořil i grafické uživatelské prostředí, které umožňuje jednoduché zacházení s aplikací. Když byla tato práce hotova, přešel jsem k experimentování s výsledným produktem. Během experimentů jsem objevil i slabší stránky mého řešení, jako je například detekce anomálií v chůzi chodců, kteří se nachází ve větší vzdálenosti od kamery nebo selhání detekce, když se chodec pohybuje čelem nebo zády ke kameře. Mezi silnější stránky patří detekce na kratší vzdálenost, cca do 3 metrů. Natrénovaný model při videích chodců, kteří byli vzdáleni 2 metry od kamery dosahoval přesnosti detekce anomálií *93%* a u videí ve vzdálenosti 3 metrů přesnosti *80%*. Během experimentů jsem také ověřil funkčnost řešení při pohybu chodců ve videu po diagonále, kdy model dokázal detekovat anomálie s dostatečnou přesností pro jeho použití při tomto scénáři. Myslím si proto, že by bylo moje řešení použitelné pro kamerový systém ve vnitřních prostorách budov, například chodby.

V práci by se dalo pokračovat například rozšířením pro sledování daných osob a detekci anomálií v jejich chůzi. Moje řešení dokáže provádět detekci anomálií chodců, ovšem v rámci detekčního okna musí vždy být jeden chodec. Toto by šlo odstranit přidáním sledovacích algoritmů, který by dokázali od sebe odlišit jednotlivé osoby. Myslím si, že toto rozšíření by nemuselo být těžké naimplementovat, ale celkovou aplikaci by mohlo značně zpomalit, proto jsem se rozhodl tuto funkcionalitu nepodporovat. Jako další by se určitě nabízelo rozšíření o detekce dalších anomálií, k dosažení tohoto cíle by bylo především nutné rozšířit dataset, který byl použit k učení neuronové sítě. K rozšíření datasetu se rovněž nabízí vytvoření dalších videí z větší vzdálenosti, aby se vylepšila přesnost detekce i u tohoto scénáře.

Literatura

[1] Chandola, V., Banerjee, A. a Kumar, V. Anomaly Detection: A Survey. New York, NY, USA: Association for Computing Machinery. jul 2009, sv. 41, č. 3. DOI: 10.1145/1541880.1541882. ISSN 0360-0300. Dostupné z: https://doi.org/10.1145/1541880.1541882.

[2] Tschuchnig, M. E. a Gadermayr, M. Anomaly Detection in Medical Imaging - A Mini Review. In: Haber, P., Lampoltshammer, T. J., Leopold, H. a Mayr, M., ed. *Data Science – Analytics and Applications*. Wiesbaden: Springer Fachmedien Wiesbaden, 2022, s. 33–38. ISBN 978-3-658-36295-9.

[3] Morris, B. a Trivedi, M. A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance. *Circuits and Systems for Video Technology, IEEE Transactions on*. Září 2008, sv. 18, s. 1114 – 1127. DOI: 10.1109/TCSVT.2008.927109.

[4] Omar, S., Ngadi, M., Jebur, H. a Benqdara, S. Machine Learning Techniques for Anomaly Detection: An Overview. *International Journal of Computer Applications*. Říjen 2013, sv. 79. DOI: 10.5120/13715-1478.

[5] Likas, A., Vlassis, N. a J. Verbeek, J. The global k-means clustering algorithm. *Pattern Recognition*. 2003, sv. 36, č. 2, s. 451–461. DOI: https://doi.org/10.1016/S0031-3203(02)00060-2. ISSN 0031-3203. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0031320302000602.

[6] Su, M.-Y. Real-time anomaly detection systems for Denial-of-Service attacks by weighted k-nearest-neighbor classifiers. *Expert Systems with Applications*. 2011, sv. 38, č. 4, s. 3492–3498. DOI: https://doi.org/10.1016/j.eswa.2010.08.137. ISSN 0957-4174.

[7] Leszek J. Chmielewski, B.-S. S. a Wojciechowski, K. *Computer vision and graphics: international conference, iccvg 2014*. Warsaw, Poland: Springer, 2014. ISBN 978-3-319-11330-2.

[8] Ogale, N. *A survey of techniques for humandetection from video*. Disertační práce.

[9] Zhang, L. a Liang, Y. Motion Human Detection Based on Background Subtraction. In: *2010 Second International Workshop on Education Technology and Computer Science*. 2010, sv. 1, s. 284–287. DOI: 10.1109/ETCS.2010.440. Dostupné z: https://ieeexplore.ieee.org/document/5459067.

[10] Utsumi, A. a Tetsutani, N. Human detection using geometrical pixel value structures. In: *Proceedings of Fifth IEEE International Conference on Automatic*

Face Gesture Recognition. 2002, s. 39–44. DOI: 10.1109/AFGR.2002.1004128. Dostupné z: https://ieeexplore.ieee.org/document/1004128.

[11] Cao, Z., Simon, T., Wei, S.-E. a Sheikh, Y. Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[12] Li, D. Human Skeleton Detection and Extraction in Dance Video Based on PSO-Enabled LSTM Neural Network. *Computational Intelligence and Neuroscience*. Září 2021, sv. 2021, s. 1–10. DOI: 10.1155/2021/2545151.

[13] S.S. Cross, R. K. Introduction to neural networks. *The Lancet*. 1995, 346, Issue 8982, s. 1075–1079. ISSN 0140-6736. Dostupné z: https://doi.org/10.1016/S0140-6736(95)91746-2.(https://www.sciencedirect.com/science/article/pii/S0140673695917462).

[14] Novikov, A., Podoprikhin, D., Osokin, A. a Vetrov, D. P. Tensorizing Neural Networks. In: Cortes, C., Lawrence, N., Lee, D., Sugiyama, M. a Garnett, R., ed. *Advances in Neural Information Processing Systems*. Curran Associates, Inc., 2015, sv. 28. Dostupné z: https://proceedings.neurips.cc/paper/2015/file/6855456e2fe46a9d49d3d3af4f57443d-Paper.pdf.

[15] Brownlee, J. *A gentle introduction to the rectified linear unit (ReLU)*. Aug 2020. Dostupné z: https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/.

[16] Sharma, S. a Sharma, S. ACTIVATION FUNCTIONS IN NEURAL NETWORKS. *International Journal of Engineering Applied Sciences and Technology*. Apr 2020, sv. 4, č. 12, s. 310–316. DOI: https://ijeast.com/papers/310-316,Tesma412,IJEAST.pdf.

[17] Pratiwi, H., Windarto, A. P., Susliansyah, S., Aria, R. R., Susilowati, S. et al. Sigmoid Activation Function in Selecting the Best Model of Artificial Neural Networks. *Journal of Physics: Conference Series*. IOP Publishing. feb 2020, sv. 1471, č. 1, s. 012010. DOI: 10.1088/1742-6596/1471/1/012010. Dostupné z: https://doi.org/10.1088/1742-6596/1471/1/012010.

[18] Szandała, T. Review and Comparison of Commonly Used Activation Functions for Deep Neural Networks. 2020. Dostupné z: https://arxiv.org/pdf/2010.09458.pdf.

[19] Toth, D. a Aach, T. Detection and recognition of moving objects using statistical motion detection and Fourier descriptors. In: *12th International Conference on Image Analysis and Processing, 2003.Proceedings*. 2003, s. 430–435. DOI: 10.1109/ICIAP.2003.1234088. Dostupné z: https://ieeexplore.ieee.org/document/1234088.

[20] Brownlee, J. *Overfitting and underfitting with machine learning algorithms*. Aug 2019. Dostupné z: https://machinelearningmastery.com/overfitting-and-underfitting-with-machine-learning-algorithms/.

[21] Paul, M., Haque, S.M.E. a Chakraborty, S. Human detection in surveillance videos and its applications. *EURASIP Journal on Advances in Signal Processing*.

Springer Science+Business Media. 2013, sv. 176. Dostupné z: https://asp-eurasipjournals.springeropen.com/articles/10.1186/1687-6180-2013-176#citeas.

[22] Kishan G. Mehrotra, H. H. *Anomaly Detection Principles and Algorithms*. Cham, Switzerland: Springer, 2017. ISBN 978-3-319-67524-4.

[23] Morais, R., Le, V., Tran, T., Saha, B., Mansour, M. et al. Learning Regularity in Skeleton Trajectories for Anomaly Detection in Videos. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, s. 11988–11996. DOI: 10.1109/CVPR.2019.01227.

[24] Kepski, M. a Kwolek, B. Person Detection and Head Tracking to Detect Falls in Depth Maps. In: Chmielewski, L. J., Kozera, R., Shin, B.-S. a Wojciechowski, K., ed. *Computer Vision and Graphics*. Cham: Springer International Publishing, 2014, s. 324–331. ISBN 978-3-319-11331-9.

[25] Medsker, L. *Recurrent Neural Networks*. únor 2000. ISBN 9780849371813.

[26] Yu, Y., Si, X., Hu, C. a Zhang, J. A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*. Červenec 2019, sv. 31, č. 7, s. 1235–1270. ISSN 0899-7667.

[27] Schuster, M. a Paliwal, K. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*. 1997, sv. 45, č. 11, s. 2673–2681. DOI: 10.1109/78.650093.

[28] Lubanovic, B. *Introducing python*. Sebastopol, CA: O'Reilly Media, jul 2014. ISBN 9781492051367.

[29] Ginés Hildago, Y. R. *OpenPose: OpenPose Doc.* 2022. Accessed 6 May 2022. Dostupné z: https://cmu-perceptual-computing-lab.github.io/openpose/web/html/doc/index.html.