VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INFORMAČNÍCH SYSTÉMŮ
DEPARTMENT OF INFORMATION SYSTEMS

ROZPOZNÁVÁNÍ UDÁLOSTÍ VE FOTBALU Z PROSTOROČASOVÝCH DAT OBJEKTŮ VE HŘE

FOOTBALL EVENT RECOGNITION FOR SPATIOTEMPORAL DATA OF GAMING OBJECTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE **TOMÁŠ ČÍŽEK**
AUTHOR

VEDOUCÍ PRÁCE **RNDr. MAREK RYCHLÝ, Ph.D.**
SUPERVISOR

BRNO 2018

Zadání diplomové práce/20871/2017/xcizek12

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav informačních systémů Akademický rok 2017/2018

Zadání diplomové práce

Řešitel: **Čížek Tomáš, Bc.**
Obor: Inteligentní systémy
Téma: **Rozpoznávání událostí ve fotbalu z prostoročasových dat objektů ve hře**
Football Event Recognition for Spatiotemporal Data of Gaming Objects
Kategorie: Data mining

Pokyny:

1. Seznamte se s dostupnými prostoročasovými daty objektů ze záznamu fotbalového utkání (poloha hráčů a míče) od spol. Camvision. Dále prozkoumejte existující metody a nástroje pro automatickou analýzu fotbalových utkání (např. od spol. STAT).
2. Navrhněte či vyberte vhodné metody strojového učení pro rozpoznávání jednotlivých událostí v záznamu fotbalového utkání (např. klasifikace). U každé události specifikujte typ a rozsah potřebných vstupních dat.
3. Připravte trénovací a testovací množiny z navržených typů vstupních dat.
4. Po konzultaci s vedoucím implementujte řešení dle návrhu tak, aby bylo schopno automaticky rozpoznávat jednotlivé události (rohový kop, výkop brankaře, gól aj.).
5. Výsledek důkladně otestuje, zdokumentujte a navrhněte možná rozšíření.

Literatura:

- Guangyu Zhu, Qingming Huang, Changsheng Xu, Yong Rui, Shuqiang Jiang, Wen Gao, and Hongxun Yao (2007). Trajectory based event tactics analysis in broadcast sports video. In Proceedings of the 15th ACM international conference on Multimedia (MM '07). ACM, New York, NY, USA, 58-67. [https://doi.org/10.1145/1291233.1291250]
- Bialkowski, Alina, Lucey, Patrick J., Carr, Peter, Yue, Yisong, Sridharan, Sridha, & Matthews, Iain (2014) Large-scale analysis of soccer matches using spatiotemporal tracking data. In IEEE International Conference on Data Mining (ICDM 2014), 14-17 December 2014, Shenzhen, China. [https://eprints.qut.edu.au/78124/]

Při obhajobě semestrální části projektu je požadováno:

- Body 1 a 2.

Podrobné závazné pokyny pro vypracování diplomové práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva diplomové práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap, které byly vyřešeny v rámci dřívějších projektů (30 až 40% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Rychlý Marek, RNDr., Ph.D.**, UIFS FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 23. května 2018

L.S. VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ Fakulta informačních technologií Ústav informačních systémů 612 66 Brno, Božetěchova 2

doc. Dr. Ing. Dušan Kolář
vedoucí ústavu

Abstrakt

Tato diplomová práce se věnuje automatické detekci událostí ve fotbale. Jejím cílem je uvést čtenáře do této problematiky, diskutovat možná řešení této úlohy a poté detekci událostí realizovat. Práce je zaměřena především na detekci událostí z prostoročasových dat objektů ve hře. Představený způsob realizace tohoto problému spočívá v jeho převedení na úlohu typu značení sekvencí. Taková úloha je poté řešena pomocí rekurentních neuronových sítí LSTM. Výsledek značení sekvencí je nakonec interpretován jako detekované události. Jako produkt této práce vznikla knihovna pro realizaci detekce událostí a experimentování s různými variantami formulace této úlohy jako problém značení sekvencí.

Abstract

This diploma thesis deals with automatic soccer event detection. Its goal is to introduce reader to this issue, discuss possible ways of solution of this task and then implement event detection. This work aims at event recognition using spatio-temporal data of gaming objects. Introduced way of dealing with event detection lies in its converting to sequence labeling task. Then such task is solved using LSTM recurrent neural networks. Lastly, result of sequence labeling is interpreted as detected events. Library for event detection has been created as the output of this work. This library allow user to experiment with different variants how to formulate event detection as sequence labeling task.

Klíčová slova

rozpoznávání událostí, fotbal, prostoročasová data, strojové učení, značení sekvencí, umělé neuronové sítě

Keywords

event detection, soccer, spatio-temporal data, machine learning, sequence labeling, artifical neural network

Citace

ČÍŽEK, Tomáš. *Rozpoznávání událostí ve fotbalu z prostoročasových dat objektů ve hře.* Brno, 2018. Diplomová práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce RNDr. Marek Rychlý, Ph.D.

Rozpoznávání událostí ve fotbalu z prostoročasových dat objektů ve hře

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana RNDr. Marka Rychlého, Ph.D. Další informace mi poskytl pan Ing. Igor Potůček, Ph.D. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Tomáš Čížek
14. května 2018

Poděkování

Chtěl bych poděkovat panu RNDr. Marku Rychlému, Ph.D. za odborné vedení práce a cenné rady, které mi pomohly tuto práci zkompletovat. Mé poděkování patří také panu Ing. Igoru Potůčkovi, Ph.D. za poskytnutí dat pro tuto práci a vstřícnost při konzultacích, během kterých jsem nabyl spoustu informací a znalostí užitečných pro tvorbu této práce.

Obsah

Kapitola 1

Úvod

Informační technologie ulehčují lidskou práci a otevírají nové možnosti. To je obecně platný fakt, který už nějakou dobu platí také ve sportu. Sport je samozřejmě založen na dovednostech a fyzických výkonech sportovců, s čímž technologie nemá nic společného. A tak přirozený talent a pořádný tréning je pro sportovce stále jedinou cestou k dosažení lepších individuálních výkonů. Avšak ve sportu a okolo něj se najdou i oblasti, v nichž mají vyvíjející se technologie velký potenciál. V týmových sportech hrají velkou roli, kromě individuálních výkonů, také strategie a taktika zahrnující koordinaci hráčů na hřišti. Proto jsou, hlavně v týmových sportech, tak důležití trenéři, jejichž úkolem je mimo jiné právě příprava týmu po taktické stránce. Často je změna trenéra řešením, ke kterému se vedení týmu uchýlí, pokud se týmu dlouhodobě nedaří. V této práci se budeme soustředit na pravděpodobně nejpopulárnější týmový sport na světě – fotbal.

1.1 Využití technologie pro taktickou přípravu ve fotbale

Ve fotbale toho taktika zahrnuje spoustu. Mezi to hlavní patří samotný styl hry. Mezi základní herní styly patří např. pasivní bránění, aktivní napadání, protiútoky, držení míče, dlouhé přihrávky před bránu, úmyslné vystavování soupeře do ofsajdového postavení tzv. ofsajdové pasti a další. Tyto základní herní styly se dají zkombinovat do složitějšího stylu hry, který se v průběhu hry samozřejmě může také měnit na základě vývoje utkání. Příkladem komplexnějšího herního stylu může být třeba pasivní hra se zataženou obranou a využívání protiútoků k ohrožení soupeřovy branky nebo naopak snaha o co největší kontrolu míče a postupné vypracování šancí ke vstřelení gólu velkým množstvím kratších přihrávek. Trenér může buď zvolit vhodný herní styl na základě taktiky soupeře a nebo naopak zvolit vlastní styl a pokusit se tak donutit soupeře, aby se přizpůsobil. S volbou herního stylu úzce souvisí výběr samotných hráčů, kteří jsou do utkání nasazeni a jejich rozestavení nebo-li formace. Každý hráč má své silnější a slabší stránky a může být tak více či méně vhodný pro různé styly hry, proti různým soupeřům nebo pro hru na různých pozicích zvolené formace. Další součástí taktiky jsou reakce týmu na jednotlivé herní situace a události. Tím je myšleno chování týmu po zisku či ztrátě míče, způsob zahrávání a bránění standardních situací nebo taktické pokyny pro jednotlivé hráče jako je osobní obrana konkrétního hráče soupeře a podobně.

Jen z toho mála co bylo právě uvedeno je zřejmé, že po taktické stránce je fotbal poměrně složitý sport. Není tak divu, že je v dnešní době snaha do této oblasti zapojovat technologie, které by zejména trenérům jejich práci trochu usnadňovaly. Na obrázku 1.1 můžeme vidět

rozdíl, který přinesla technologie do taktické analýzy a předávání taktických instrukcí hráčům. Při pilování taktiky jsou velice užitečné videozáznamy utkání. Jejich analýza lze použít

(a) Kdysi

(b) Nyní

Obrázek 1.1: Předávání taktických instrukcí

k hledání důvodů neúspěchu týmu, různých týmových či individuálních chyb a způsobů, jak herní výkon mužstva vylepšit. Trenéři však většinou nechtějí sledovat znovu celé, 90 minut dlouhé utkání, kterého se obvykle sami zúčastnili a jehož průběh a výsledek velice dobře znají. Často se ale potřebují vracet ke konkrétním situacím, které chtějí detailněji analyzovat a nebo prezentovat týmu. Proto je zapotřebí videozáznamy utkání předzpracovat tak, aby se s nimi snadno a rychle pracovalo. To znamená vyhledat všechny události a situace utkání, které by mohly být významné, a zařadit je do příslušné kategorie. Příklady takových kategorií jsou gól, rohový kop, protiútok, faul, vhazování a podobně. Uživatel takto zpracovaného záznamu utkání si pak muže snadno a rychle vyfiltrovat jen konkrétní části utkání, které jej zrovna zajímají. Například všechny ofsajdové situace hráčů svého týmu. Takové předzpracování videozáznamu utkání se dnes dělá převážně ručně. Větší fotbalové kluby si mohou dovolit mít na tuto práci vlastní lidi a nebo existují i externí firmy, které tuto službu poskytují. Je to však pochopitelně časově náročná činnost, která není zadarmo a vzhledem k nedokonalosti člověka, nemusí být výsledek vždy bezchybný a stoprocentně přesný. Proto je v posledních letech snaha o nahrazování této lidské práce počítačem s cílem tuto činnost úplně zautomatizovat. Pouhou detekcí událostí však ambice automatického zpracování videozáznamu sportovních utkání nekončí. Potenciálně by šlo automaticky zjišťovat i složitější znalosti. Například detailnější analýzou situací, po kterých následuje gól nebo ohrožení branky, by mohlo být možné odhalit taktickou chybu bránícího týmu, která k tomu pravděpodobně vedla. Taková informace by pro trenéra fotbalového týmu zajisté měla velkou hodnotu.

1.2 Náplň a struktura práce

Jak již bylo výše naznačeno, tato práce se zabývá automatickou detekcí událostí fotbalového utkání, a to z prostoročasových (také časoprostorových) dat. Náplní práce je nejprve definovat a diskutovat problém, představit možné přístupy a metody použitelné k řešení, poté detekci událostí implementovat a nakonec vyhodnotit výsledky. Na této práci se podílela

firma CamVision s.r.o. poskytnutím vstupních dat pro detekci událostí. Více o této společnosti a vstupních datech se lze dozvědět hned v následující kapitole 2, která si klade za cíl seznámení s řešenou problematikou. V kapitole 3 jsou popsány možné přístupy k řešení detekce událostí. Další kapitola 4 se věnuje metodám pro řešení úloh typu značení sekvencí. Především se zaměřuje na použití rekurentních neuronových sítí v této oblasti. Zbytek textu se věnuje praktické části práce. Kapitola 5 tak popisuje implementaci a použití knihovny pro detekci událostí, která vznikla v rámci této práce. V 6. kapitole je pak provedeno pár experimentů jako ukázka použití vytvořené knihovny a také obsahuje sekci s návrhy pro její rozšíření.

Kapitola 2

Seznámení s problematikou detekce událostí ve fotbale

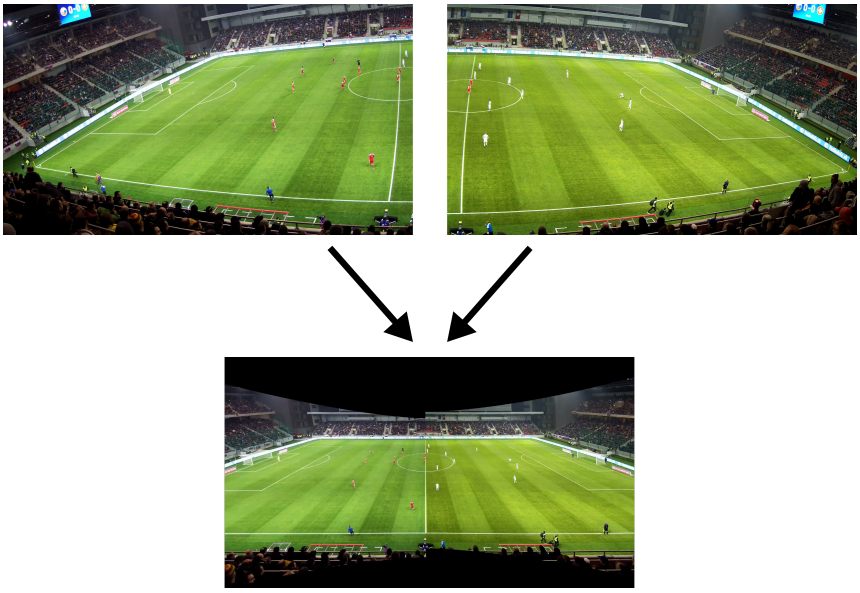
V této kapitole se detailněji seznámíme s problematikou automatické detekce událostí ve fotbale. Navážeme na úvod a povíme si o jejím využití v praxi, přičemž si představíme také systém od firmy CamVision s.r.o. Hlavním tématem této části jsou však vstup a výstup detekce událostí.

2.1 Motivace a využití

V úvodu práce bylo již diskutováno využití automatické detekce událostí v taktické přípravě týmu. Mohla by usnadnit nebo dokonce nahradit činnost lidí, kteří anotují obsah záznamů fotbalových utkání a vytvářejí statistiky zápasů. Jedna z firem, které se zabývají vývojem softvéru pro analýzu fotbalových utkání pochází z Česka a nazývá se CamVision s.r.o. [1]. Společnost poskytuje profesionálním fotbalovým klubům systém, který se skládá ze dvou částí. Tou první je část systému realizující automatické nahrávání utkání či tréninků. Druhou část tvoří aplikace pro taktickou analýzu utkání, v které lze buď sledovat živý přenos právě nahrávaného zápasu a nebo načíst již dříve pořízený záznam.

Obě poloviny hřiště zvlášť nahrává kamera ve vysokém rozlišení. Záběry z obou kamer jsou spojovány v jeden široký panoramatický obraz zabírající celé hřiště (viz obrázek 2.1). Nemůže se tak stát, že by nějaký moment utkání nebyl zaznamenán. Po naplánování časového rozvrhu je systém nahrávání plně automatický. Není tedy zapotřebí žádný kameraman nebo technik kvůli obsluze. Protože panoramatický obraz hřiště je příliš veliký pro zobrazení a praktické použití na běžných zařízeních, je součástí také inteligentní softvér, který z panoramatického obrazu vybírá jen určitou část hřiště. Většinou je to ta část, na které se zrovna hraje. Automaticky se tak vytváří záběr simulující přibližování, oddalování a natáčení kamery. Pokud bychom měli informaci o nastalé události na hřišti v reálném čase, umožnilo by to umělou inteligenci virtuální kamery značně zdokonalit. Příkladem může být vhodné přizpůsobení záběru při zahrávání pokutového kopu nebo poskytnutí přiblíženého záběru na hráče slavícího právě vstřelený gól.

Na obrázku 2.2 vidíme jak vypadá aplikace pro taktickou analýzu. V ní si může uživatel sám zvolit mezi použitím panoramatického nebo automaticky vytvářeného záběru. Má také možnost virtuální kameru ovládat manuálně a zobrazit si tedy kdykoli kteroukoli část panoramatického obrazu hřiště. Aplikace slouží pro rozbor herních situací a jeho prezentaci hráčům. Poskytuje proto různé vizualizační nástroje:



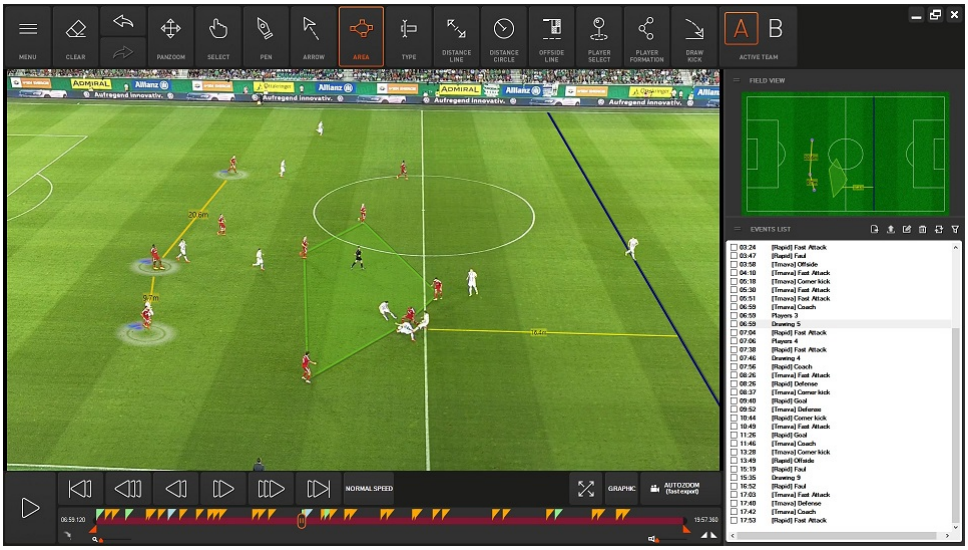
Obrázek 2.1: Složení panoramatického záběru

- kreslení šipek čar, kruhů a volné kreslení,
- zvýraznění vybrané oblasti,
- měření a zobrazení vzdáleností,
- zobrazení ofsajdové hranice,
- zobrazení zarovnání formace,
- označení hráčů a zobrazování jejich jmen, rychlosti pohybu, aktuálního zrychlení a uběhnuté vzdálenosti,
- zobrazení modelu hřiště a hráčů z ptačí perspektivy.

Pro lepší orientaci ve videu je možné manuálně označovat vybrané chvíle utkání a zařazovat je do zvolené kategorie událostí. Cílem je, aby právě tuto činnost dokázal takový systém provádět automaticky. Uživatel si poté může podle kategorií událostí vyfiltrovat jen konkrétní momenty utkání.

Další z velkých motivací pro výzkum na poli automatické detekce událostí je snaha zautomatizovat sumarizaci nebo-li vytváření sestřihu utkání. Zde je cílem vytvořit ze záznamu fotbalového utkání klip obsahující pouze důležité a pro diváka atraktivní momenty zápasu. Tyto sestřihy utkání jsou dnes velice populární. Není totiž reálné, aby běžný fotbalový fanoušek sledoval všechna utkání, které jej zajímají, celá.

Systém detekující události fotbalového utkání v reálném čase by bylo možné využít také třeba v aplikacích, které informují fanoušky o průběhu a klíčových momentech utkání prostřednictvím mobilních zařízení. Je tedy zřejmé, že využití technologií ve fotbale má značný potenciál.



Obrázek 2.2: Aplikace pro taktickou analýzu

2.2 Vstupní data

U detekce událostí ve fotbale lze rozlišovat dvě varianty: detekce událostí z prostoročasových dat nebo z audiovizuálního broadcastového vysílání. Rozdíl je právě v tom, co lze použít jako vstupní informace. Při použití prostoročasových dat jsou k dispozici informace o všech objektech, které se pohybují na hrací ploše a jsou součástí hry. Kdežto u detekce událostí z broadcastového vysílání lze využít režie přímého přenosu utkání. To je na jedné straně výhodou, protože obsah záběrů vybraných režisérem může být použit k rozpoznání událostí. Na druhou stranu nám informace o některých událostech může zcela uniknout, protože na rozdíl od prostoročasových dat totiž broadcastový přenos neposkytuje neustálý přehled o dění na celém hřišti. To je důvodem proč broadcastová vysílání nejsou vhodná pro taktickou analýzu a proč se tedy v této práci soustředíme spíše na prostoročasová data. Detekci událostí z broadcastových vysílání navíc nemůžeme označit za zcela automatickou, jelikož už samotné vysílání, které je vstupem detekce, je vytvářeno lidmi. Proč se broadcastovými vysíláními jako vstupem detekce událostí vůbec zabývat? Odpovědí je automatická sumarizace fotbalových utkání. V této sekci se tedy blíže podíváme jednak na detekci událostí z broadcastových vysílání a poté hlavně na prostoročasová data.

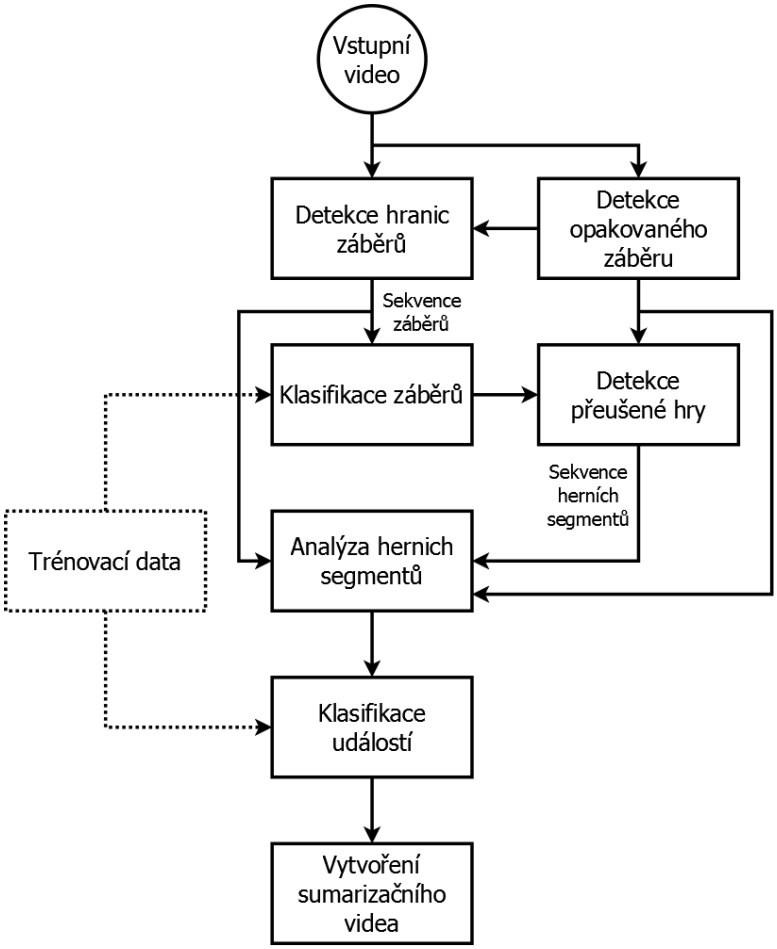
2.2.1 Broadcastová vysílání

Pro detekci událostí z broadcastových vysílání je klíčová videoanalýza. Článek [40] představuje rozdělení analýzy videa do tří úrovní: nízká, střední a vysoká. Do nízkoúrovňové analýzy patří detekce hřiště, jeho lajnování a brankoviště. Sledování hráčů, rozhodčích, míče a jejich trajektorií v obraze zahrnuje analýza střední úrovně. Vysokoúrovňová analýza obsahuje zjišťování více sémantických informací, jako je rozpoznání přerušení hry, herních taktik, sumarizace utkání nebo detekce jednotlivých událostí. Článek obsahuje přehled prací roztříděných podle toho, jakým problémem a kterou úrovní analýzy se zabývá.

Spousta prací o analýze broadcastových vysílání se zaměřuje na sumarizaci utkání. Článek [33] shrnuje výzkum právě v této oblasti. Obsahuje reference na práce zabývajících se nejen nízkoúrovňovou analýzou videa, ale také zvukovou analýzou nahrávky utkání. Dále

jsou zahrnuty články zabývající se metodami pro rozeznávání přerušené hry, opakovaného záběru a fotbalových událostí. Nakonec je obsažen seznam prací zabývajících se samotnou sumarizací utkání a extrakcí zajímavých momentů zápasu.

Práce [44] popisující kompletní systém pro automatické vytváření sestřihu utkání je dobrým příkladem varianty detekce událostí z audiovizuálního broadcastoveho vysílání. Blokový diagram navrhovaného systému je na obrázku 2.3. Jednotlivé bloky si nyní stručně popíšeme.

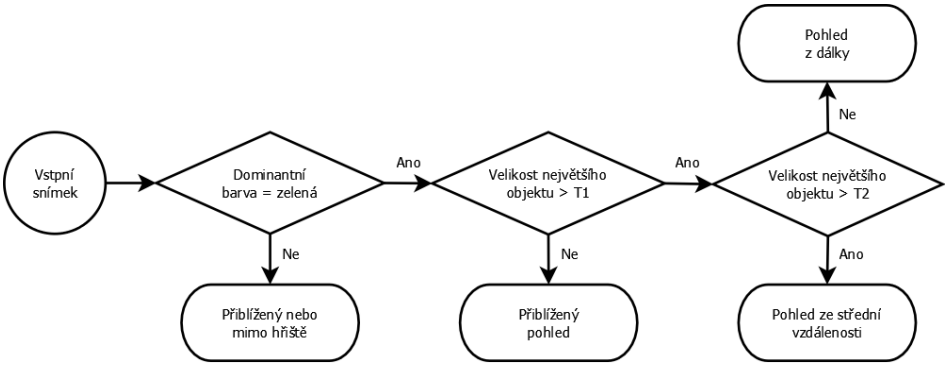


Obrázek 2.3: Blokový diagram systému pro automatické vytváření sestřihu utkání

Vstupní video je nejprve rozděleno na jednotlivé záběry kamery. Hranice po sobě jdoucích záběrů jsou detekovány pomocí metody support vector machine (SVM). Jejím vstupem je rozdíl barevných histogramu sousedních snímků a pohybový vektor snímku, který popisuje transformaci z jednoho snímku na následující. Hraniční snímek má ve srovnání s předchozím snímkem rozdílný histogram a krátký pohybový vektor. Naopak rozdíl histogramů dvou sousedních snímků stejného záběru je velice malý a pohybový vektor je velký.

Záběry jsou poté roztříděny do čtyř kategorií: pohled z dálky, pohled ze střední vzdálenosti, přiblížený pohled a pohled mimo hřiště. Rozhodovací proces rozdělení záběrů do kategorií je znázorněn vývojovým diagramem na obrázku 2.4. Jak můžeme vidět, používá pouze dvě kritéria: dominantní barvu záběru a velikost největšího objektu na hřišti.

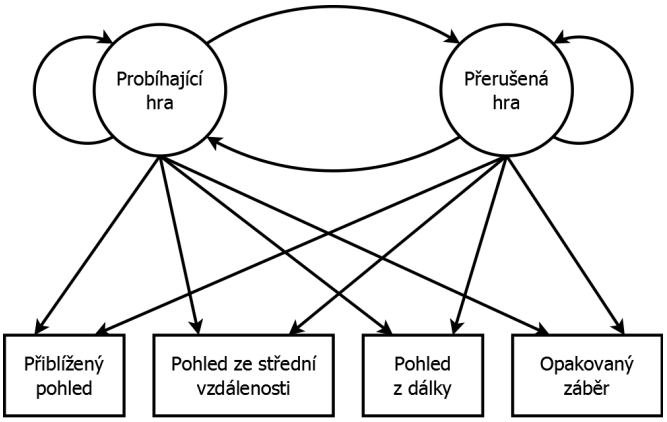
Rozpoznání opakovaného záběru je založeno na detekci loga, které se obvykle objevuje před a po opakovaném záběru. Logo je v obraze detekováno porovnáváním barevných his-



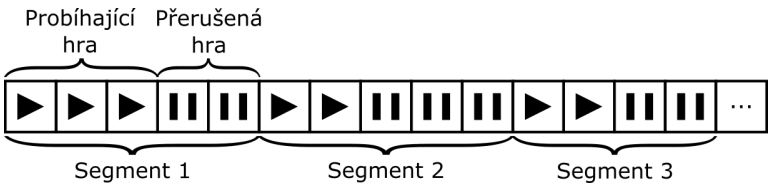
Obrázek 2.4: Vývojový diagram klasifikace záběrů

togramů jednotlivých snímků s histogramem snímku obsahující logo. Je však potřeba toto logo dopředu znát.

Pro rozlišení přerušené hry od probíhající je použit skrytý Markovův model se strukturou zobrazenou na obrázku 2.5. Model má dva skryté stavy (hra a pauza) a čtyři pozorovatelné výstupy, kterými jsou typy záběru (dlouhý, střední, přiblížený a záznam). Tento skrytý Markovův model je natrénován a použit pro rozhodnutí, zda jednotlivé záběry obsahují probíhající či přerušenou hru. Výsledkem je rozdělení videa utkání na herní segmenty. Každý segment obsahuje část probíhající hry a následující část přerušené hry (viz obrázek 2.6).



Obrázek 2.5: Struktura skrytého Markovova modelu pro rozpoznání probíhající a přerušené hry



Obrázek 2.6: Rozdělení utkání na herní segmenty (▶ – záběr probíhající hry, ❚❚ – záběr přerušené hry)

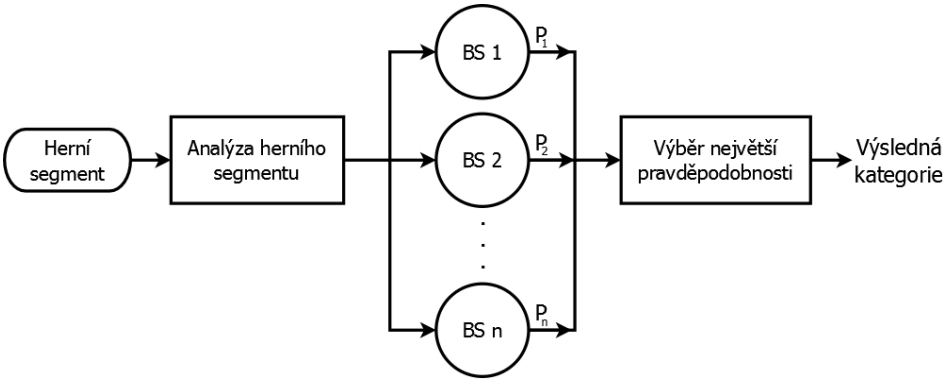
Každý herní segment je dále analyzován s cílem zjistit další jeho vlastnosti potřebné pro následnou klasifikaci herních segmentů do kategorií událostí. Potřebné jsou jednak podrobnější informace o herním segmentu: doba přerušené hry, doba opakovaných záběrů, počet přiblížených záběrů a také informace o obsahu záběrů v herním segmentu: detekce pokutového území, počet hráčů v pokutovém území, přítomnost rozhodčího, přítomnost brankáře nebo zobrazení grafických titulků.

Každý herní segment je považován za událost a je přiřazen do jedné z těchto kategorií: gól, udělení karetního trestu, střela na bránu, rohový kop, ofsajd, faul a nezajímavá událost. Pro klasifikaci herních segmentů jsou použity Bayesovské sítě. Ke každé kategorii událostí je vytvořena jedna Bayesovská síť, jejíž výstupem je pravděpodobnost, s kterou klasifikovaný herní segment obsahuje událost dané kategorie. Struktura sítí je stejná, avšak parametry jsou určeny pro každou síť zvlášť (podrobnější informace o konstrukci a trénování Bayesovských sítí lze nalézt ve zdrojovém článku [44]). Vstupními informacemi jsou vlastnosti klasifikovaného segmentu získané jeho analýzou:

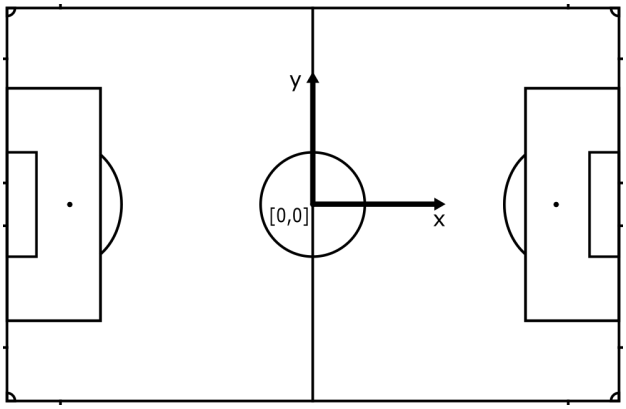
- počet záběrů přerušené hry,
- počet opakovaných záběrů,
- počet záběrů přerušené hry před opakovaným záběrem,
- počet přiblížených záběrů přerušené hry,
- počet záběrů obsahující rozhodčího,
- počet záběrů obsahující brankáře,
- doba trvání přerušené hry,
- doba trvání opakovaných záběrů,
- zda poslední záběr probíhající hry obsahuje pokutové území,
- zda se v pokutovém území nachází hodně hráčů,
- zda herní segment obsahuje grafické titulky,
- zda herní segment obsahuje pozastavený opakovaný záběr.

Výsledná kategorie, do které je herní segment přiřazen, je určena výběrem nejvyšší pravděpodobnosti z výstupů všech sítí (viz obrázek 2.7 obsahující blokový diagram části systému pro detekci událostí).

Poslední částí systému a je samotné vytvoření sestřihu utkání. Každému hernímu segmentu je přiřazená váha podle důležitosti události, kterou obsahuje. Cílem je vybrat do výsledného sumarizačního videa takové segmenty, aby součet jejich vah byl co nejvyšší a zároveň délka videa nepřekročila požadovanou hodnotu. Tento problém lze zredukovat na známou optimalizační úlohu zvanou problém batohu a vyřešit jej vhodným algoritmem založeným na metodě dynamického programování (např. [18]).



Obrázek 2.7: Blokový diagram části systému pro detekci událostí použitím Bayesovských sítí (BS)

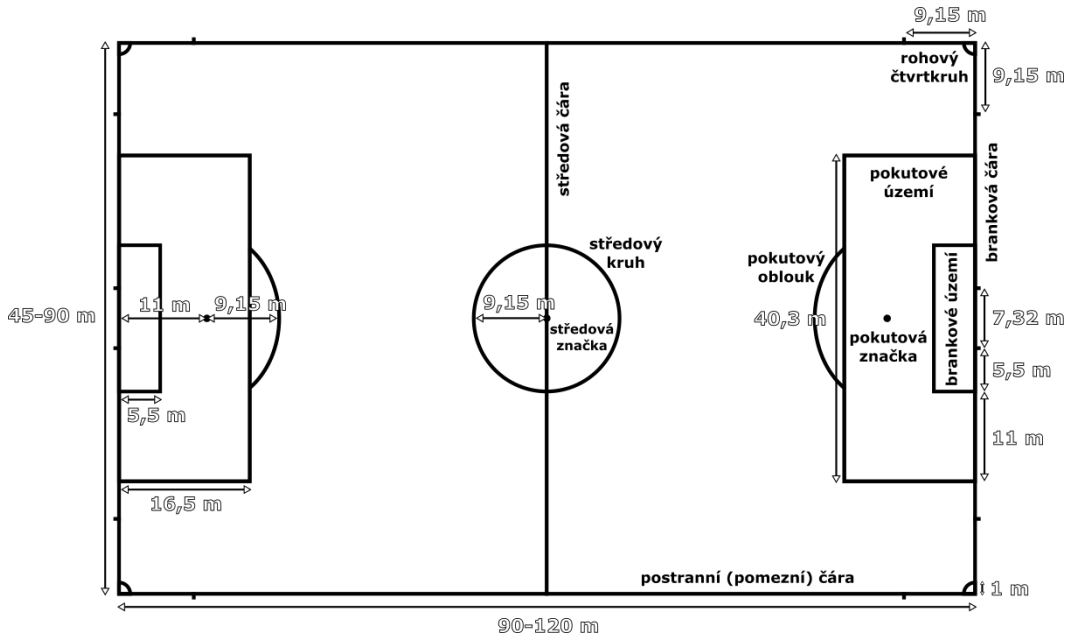


Obrázek 2.8: Souřadný systém hřiště

2.2.2 Prostoročasová data

Detekce událostí z prostoročasových dat znamená, že vstupními daty jsou pozice všech hráčů a míče z průběhu celého utkání. Ty jsou dány souřadnicemi. Souřadný systém má počátek ve středu hřiště přičemž osa x je vodorovná s pomezní čárou a osa y s brankovými čarami (viz obrázek 2.8). Pro definici pozic hráčů stačí dvoudimenzionální souřadnice. Avšak u polohy míče nás může zajímat i třetí rozměr, tedy vzdálenost od plochy hřiště.

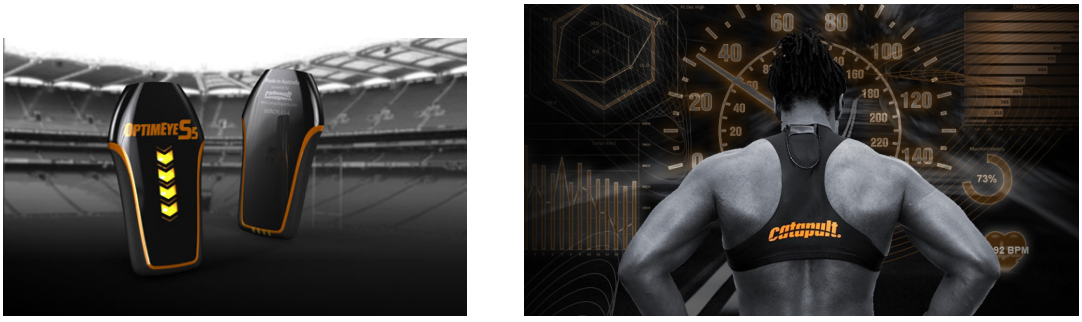
Součástí prostoročasových dat může být také identifikace hráčů. Tím jsou myšleny informace o hráči na dané pozici. Při nejmenším je užitečné vědět, ke kterému týmu hráč patří a zda se jedná o brankáře nebo hráče v poli. Ideálně bychom však mohli znát i hráčovo jméno a číslo dresu. Spolu s prostoročasovými daty mohou být součástí vstupu i souřadnice branek a lajnování hřiště. Většina vzdáleností definujících lajnování fotbalového hřiště jsou fixní, avšak jeho délka a šířka fotbalovými pravidly přesně stanovena není. Délka hřiště se může pohybovat v rozmezí 90–120 m a šířka je 45–90 m (viz obrázek 2.9). Pokud tedy známe rozměry hřiště a jsme schopni přepočítat vzdálenost v souřadném systému na reálnou vzdálenost, můžeme souřadnice veškerého jeho lajnování i odvodit. Vstupní prostoročasová data lze získat buďto analýzou videonahrávky utkání a nebo pomocí bezdrátových senzorů.



Obrázek 2.9: Rozměry a terminologie fotbalového hřiště

Získávaní prostoročasových dat pomocí bezdrátových senzorů

Pro získávání prostoročasových dat hráčů pomocí bezdrátových senzorů v prvé řadě potřebujeme, aby je během celého utkání měli všichni hráči na sobě připevněny. To už dnes nepředstavuje problém z hlediska velikosti senzorů nebo nepohodlí při jejich nošení (viz obrázky senzoru 2.10). Použití senzorů hráčů však přesto bývá při oficiálních utkáních zatím



Obrázek 2.10: Bezdrátový senzor OptimEye S5 od firmy Catapult Sport

zakázáno. Hlavním důvodem jsou pravděpodobně obavy, že by si hráč mohl při pádu kvůli senzoru přivodit zranění. K získávání dat míče však externí senzor použít nelze. Narazili bychom totiž na problém s jeho připevněním k míči tak, aby neovlivňoval hru. Je tedy třeba, aby byl senzor vyroben už jako součást míče a aby jeho přítomnost neovlivňovala vlastnosti míče. Díky tomu, že dnes už senzory dokáží být velice malé a lehké se takové míče vyrábějí. Příkladem je třeba Adidas miCoach smart soccer ball [7] na obrázku 2.11. Ale stejně jako senzory hráčů, použití míče se senzory zatím nebývá v oficiálních utkáních běžně povoleno. Přesto už v říjnu roku 2017 proběhlo v Německu první oficiální utkání na světě, v němž byli jak míč, tak všichni hráči vybaveni bezdrátovými senzory [26].

Obrázek 2.11: Fotbalový míč se senzory Adidas miCoach smart soccer ball

Pomocí senzorů lze získávat pozice hráčů, jejich rychlost a zrychlení nebo třeba i fyziologické údaje, jako rychlost srdečního tepu. Jelikož má každý hráč vlastní senzor, je také snadné získaná data přiřadit ke konkrétním hráčům. Použití senzorů tedy může poskytovat i identifikaci hráčů. Senzory míče dokáží detekovat nejen trojrozměrnou pozici, rychlost a zrychlení míče, ale také jeho rotaci.

Podle způsobu určování polohy lze rozdělit senzorové systémy na dva druhy. Buď jsou to lokální systémy s přijímači signálu rozmístěnými kolem hřiště nebo systémy využívající GPS. Lokální systémy tedy sice potřebují navíc základnové stanice kolem hřiště, ale za to jsou výrazně přesnější než GPS technologie. Použití bezdrátových senzorů je omezeno pouze výdrží baterie nebo dosahem přijímačů. Například senzor OptimEye S5 [41] od firmy Catapult Sport má výdrž baterie 5 h a používá se s přijímačem, který má dosah až 250 m. Pokud je tedy přijímač vhodně umístěn a baterie senzorů jsou nabity, neměl by být problém pokrýt plochu jak celého hřiště, tak i jeho blízkého okolí během celého utkání nebo běžného tréninku.

Při použití senzorů během utkání je třeba řešit rozlišení náhradníků od hrajících hráčů. Náhradníci musejí být připraveni se zapojit do hry a často se rozcvičují v těsné blízkosti hřiště. Jednou možností je aktivovat hráčův senzor až při střídání těsně před jeho nástupem na hřiště. Na to by se však mohlo snadno zapomenout a byla by to zbytečná starost pro hráče nebo jiného člena týmu, jenž by na to musel dohlížet. Jiné intuitivní řešení spočívá ve využití faktu, že náhradníci nemají během utkání povolen vstup na hřiště. Samotné použití bezdrátových senzorů nám ale neposkytuje žádné informace o velikosti hřiště nebo jeho hranicích. Informace o hřišti tedy musejí být získány jiným způsobem nebo je třeba se bez nich obejít.

Největším problémem bezdrátových senzorů je, že zatím nemohou být používány během oficiálních zápasů kvůli pravidlům. Tým navíc nemůže nutit hráče svého soupeře, aby senzory používali proti své vůli. Proto je hodně atraktivní získávání prostoročasových dat z obrazu videonahrávky utkání.

Získávání prostoročasových dat z videonahrávky

Abychom mohli získávat prostoročasová data z obrazu videonahrávky, potřebujeme záběr na celé hřiště nebo pokrýt celé hřiště pomocí záběrů z více kamer. V jednotlivých snímcích videa jsou detekováni hráči, jejichž pozice v obraze jsou pak převedeny na souřadnice

v souřadném systému hřiště. Detekce míče v obraze je také možná, avšak převod jeho pozice do souřadného systému je komplikovanější, protože na rozdíl od hráčů se míč během hry může pohybovat v libovolné výšce nad zemí. Informace o výšce letu míče je nejen důležitá z hlediska rozpoznávání událostí, ale také je nezbytná pro přesné určení zbylých souřadnic míče. Polohu míče lze z jednoho záběru vypočíst pomocí stereometrie [28]. Avšak abychom ji mohli úspěšně použít, potřebujeme vědět, zda míč letí vzduchem, odráží se od země nebo se kutálí. Získat tuto znalost je ale bohužel velice obtížné. Pro alespoň přibližné určení výšky polohy míče lze použít také jeho velikost v obraze. Pro výpočet polohy míče je však tato metoda příliš nepřesná. K přesnějšímu určení trojrozměrné polohy míče je proto potřeba více záběrů hřiště z různých pozic. Potom je možné polohu míče určit pomocí triangulace [20].

Použití více kamer je možným řešením i dalšího problému, který nastává při detekci hráčů a míče v obraze, a sice jejich vzájemné překrývání se. Často se stane, že míč nebo hráč není na některém záběru vidět kvůli tomu, že je zakrytý jinými hráči, případně brankovou konstrukcí nebo sítí. Další kamery zabírající hřiště z jiných stran či úhlů však jeho pozici pravděpodobně už detekovat dokážou.

Pokud úspěšně získáme pozice hráčů a míče z jednotlivých snímků videa, rychlost a zrychlení lze pak se znalostí frekvence snímků snadno dopočítat. S identifikací hráčů je to už složitější. Díky odlišným barvám dresů, lze poznat, kteří hráči hrají spolu a rozlišit také brankáře či rozhodčí. Abychom však získali přesnou identitu hráče, museli bychom dokázat přečíst číslo nebo jméno na jeho dresu, což za daných podmínek (vzdálenost a úhel záběru) často není reálné.

Protože v obraze lze detekovat i branky nebo lajnování hřiště, můžeme analýzou videa získat i hranice hřiště a tedy jeho velikost. Pokud se však hráč nebo míč vyskytuje za hranicemi hřiště, je jeho detekce velice obtížná. Při detekci objektů na hřišti se využívá hlavně toho, že kromě bílého lajnování má celé hřiště stejnou, zelenou barvu. Vše ostatní, co nemá barvu hřiště, je tedy s velkou pravděpodobností hráč, rozhodčí nebo míč. Za hranicí hřiště toto už ale neplatí. Naopak těsně kolem hřiště je spousta rušivých objektů (reklamní plachty či bannery, střídačka, trenéři, fotografové atd.), které detekci znemožňují.

Obě možnosti získávání prostoročasových dat, pomocí bezdrátových senzorů a zpracováním videozáznamu, mají své výhody, ale nesou sebou také jisté komplikace. Přehled diskutovaných kladů a záporů obou variant je shrnut tabulkou 2.1. Kvalita prostoročasových dat závisí na použitých technologiích a metodách jejich získávání. Zajímají nás tři kritéria. Prvním z nich je frekvence získávání údajů, tedy kolikrát za sekundu dostaneme nové údaje. Druhým kritériem je přesnost dat vyjádřená průměrnou odchylkou získané pozice od skutečné pozice objektu. Poslední kritérium se nevztahuje až tak k samotným časoprostorovým datům, ale určuje, zda je součástí vstupních dat identifikace hráčů a nebo informace o velikosti hřiště. Z hlediska kvality dat je ideální získávat prostoročasová data jak pomocí senzorů, tak ze záběrů kamer a poté data sloučit. Taková řešení jsou však nejkomplikovanější a také nejdražší.

2.3 Výstupní data

Výsledkem detekce událostí je pochopitelně seznam rozpoznaných událostí. U každé z nich potřebujeme zaznamenat přesný čas jejího nastání a případně dobu trvání události. Ne méně důležité je ale určit také kategorii rozpoznané události. Řekněme si tedy, jaké různé typy událostí je možné ve fotbale detekovat a jak je můžeme klasifikovat.

	Bezdrátové senzory	Videonahrávka
Potřebné vybavení	senzory hráčů a míč se senzory	pokrytí celého hřiště záběrem kamery nebo záběry více kamer
Výška polohy míče	stačí použít míč se senzory	potřeba více záběrů z různých úhlů
Identifikace hráčů	lze získat úplnou identifikaci hráčů	lze určit pouze, kteří hráči hrají spolu a rozpoznat brankáře
Lajnování a velikost hřiště	nelze získat přesné údaje	lze detekovat
Omezení použití	dosah senzorů i výdrž baterií jsou dostatečné	míč a hráče lze detekovat pouze uvnitř hřiště a to pokud se navzájem nepřekrývají
Použití v oficiálních utkáních	není běžně povoleno	je povoleno

Tabulka 2.1: Srovnání metod získávání prostoročasových dat

2.3.1 Fotbalové události a jejich kategorie

Téměř každý moment fotbalového utkání lze nějak popsat, pojmenovat a tedy i kategorizovat. Na fotbalové utkání by se tedy dalo pohlížet jako na sekvenci událostí. Na nejvyšší úrovni lze rozlišovat tři typy událostí. První z nich můžeme nazývat akce. Akcemi jsou myšleny události, které mají většinou krátkodobý charakter a vážou se jen na určitého hráče. Řadíme zde tyto kategorie událostí: střela, přihrávka, dribling, zákrok, zachycení přihrávky soupeře, odehrání míče do bezpečí, zablokování míče, vzdušný souboj, ztráta míče a zásah brankáře. Druhým typem jsou události spojené s přerušením hry. Patří zde jednak události, které přerušení hry zapříčiňují (faul, gól, ofsajd), události, kterými se míč znovu uvádí do hry (kop od branky, pokutový kop, rohový kop, vhazování, volný kop) a také střídání. Třetí typ událostí je možná lepší označovat jako herní úsek. Jedná se o části utkání, které jsou něčím charakteristické. Mezi události tohoto typu tedy patří třeba držení míče, protiútok, útok do zatažené obrany nebo ponechání výhody. Ve statistikách utkání se kromě držení míče tyto události běžně nevyskytují. Avšak pro taktickou analýzu mohou být také velmi významné.

Pro většinu zmíněných kategorií událostí je možné vytvářet i další podkategorie, které danou událost zápasu ještě více specifikují. Celkový přehled typů, kategorií a možných podkategorií událostí obsahuje tabulka 2.2. Za názvem každé kategorie je vždy uveden také zažitý anglický název pro danou událost. Tabulka 2.3 pak obsahuje popis jednotlivých kategorií. Pokud bychom měli záznam utkání anotovaný výskyty všech takových událostí, bylo by možné implementovat filtr, který by vyhledával v záznamu pomocí několika kritérií. Například bychom mohli vyhledat všechny protiútoky obsahující dlouhou přihrávku vzduchem a zakončené střelou na bránu.

Typ události	Kategorie	Podkategorie
akce (události typu 1)	dribling (dribble)	úspěšný
		neúspěšný
	odehrání (clearance)	hlavou
		nohou
	přihrávka (pass)	úspěšná
		neúspěšná
		krátká po zemi
		dlouhá vzduchem
		centrovaný míč
	střela (shot)	na bránu
		mimo bránu
		zblokovaná
	vzdušný souboj (aerial duel)	
	zablokování (block)	střely
		centrovaného míče
	zachycení (interception)	
	zákrok (tackle)	úspěšný
		neúspěšný
	zásah brankáře (save)	
	ztráta (loss of possession)	
události spojené s přerušením hry (události typu 2)	faul (foul)	hra rukou
		faul potrestaný kartou
	gól (goal)	
	ofsajd (offside)	
	kop od branky (goal kick)	rozehraný na krátko
		rozehraný dlouhou přihrávkou
	pokutový kop (penalty kick)	
	rohový kop (corner kick)	z levé strany
		z pravé strany
		rozehraný na krátko
		rozehraný dlouhou přihrávkou
	vhazování (throw-in)	na polovině soupeře
		na vlastní polovině
	volný kop (free kick)	přímý
		nepřímý
		rozehraný na krátko
		rozehraný dlouhou přihrávkou
		rozehraný střelou
		z místa vzdáleného do určité vzdálenosti od soupeřovy branky
	střídání (substitution)	
charakteristické úseky hry (události typu 3)	držení míče (possession)	držení míče konkrétním týmem
		držení míče konkrétním hráčem
	protiútok (counterattack)	
	útok do zatažené obrany	
	výhoda (advantage)	

Tabulka 2.2: Přehled fotbalových událostí

Název kategorie	**Popis kategorie**
dribling	vedení míče jediným hráčem kolem bránících protihráčů
odehrání	odehrání míče z bráněné oblasti do bezpečí
přihrávka	záměrné usměrnění míče k spoluhráči
střela	záměrné usměrnění míče směrem na bránu soupeře s cílem vstřelit gól
vzdušný souboj	soupeření dvou protihráčů o míč který padá k nim ze vzduchu
zablokování	překažení střely či prudké přihrávky protihráče vložením části svého těla do její trajektorie
zachycení	záměrné zachycení přihrávky soupeře
zákrok	pokus o odebrání míče soupeři, který provádí dribling
zásah brankáře	situace, kdy brankář zabrání soupeři vstřelit gól
ztráta	situace, kdy hráč ztratí míč a získá jej soupeř
faul	provinění se hráče proti pravidlům během aktivní hry (vyjma ofsajdu)
gól	překročení míče brankovou čáru mezi tyčemi branky celým svým objemem
ofsajd	situace, kdy se do hry zapojí hráč, který se při posledním kontaktu míče s jeho spoluhráčem nacházel v ofsajdovém postavení (hráč se nachází v ofsajdovém postavení, pokud je na soupeřově polovině blíže soupeřově brance než míč i než předposlední hráč soupeře)
kop od branky	opětovné uvedení míče do hry z malého vápna po tom, co míč odražený od protihráče překročil vlastní brankovou čáru mimo branku
pokutový kop	střela z penaltové značky na bránu bráněnou pouze brankářem (pokutový kop pro mužstvo soupeře je nařizován rozhodčím jako trest za faul ve vlastním pokutovém území)
rohový kop	opětovné uvedení míče do hry od rohového praporku potom, co míč odražený od spoluhráče překročil vlastní brankovou čáru mimo branku
vhazování	opětovné uvedení míče do hry od postranní čáry poblíž místa, kde míč odražený od protihráče postranní čáru překročil
volný kop	opětovné uvedení míče do hry z místa, kde se protihráč dopustil přestupku proti pravidlům
střídání	výměna hráče na hřišti za jiného
držení míče	úsek hry, kdy má konkrétní hráč nebo tým míč pod kontrolou
protiútok	úsek hry, kdy se tým bezprostředně po zisku míče snaží co nejrychleji dostat míč k soupeřově brance a ohrozit ji
útok do zatažené obrany	úsek hry, kdy mužstvo útočí a všichni hráči soupeře jsou v hlubokém defenzivním bloku
výhoda	úsek hry po tom, kdy rozhodčí ponechá mužstvu výhodu

Tabulka 2.3: Popis kategorií fotbalových událostí

Kapitola 3

Možnosti řešení detekce událostí z prostoročasových dat

Lze rozlišit dva hlavní přístupy k řešení automatické detekce událostí z prostoročasových dat. První z nich je založen na popisu jednotlivých událostí pomocí heuristických pravidel a vyžaduje velice kvalitní vstupní data. Druhý přístup představuje možnosti použití i méně kvalitních vstupních dat pomocí metod strojového učení. Tato kapitola detailně popisuje princip obou přístupů, metody pro jejich realizaci a zhodnocuje také jejich výhody, nevýhody či reálnost použití.

3.1 Přístup založen na heuristických pravidlech

Přístup popisovaný v této sekci je inspirován článkem [45] a dále rozvíjí v něm prezentovanou metodu. Tento způsob detekce událostí je založen na myšlence, že každou fotbalovou událost lze rozložit na několik elementárních, dílčích událostí, které je možné v prostoročasových datech detekovat. Hodně takových dílčích událostí představují určitou pozici míče a hráčů vzhledem k lajnování hřiště nebo vzhledem k sobě navzájem. Často také bývá pro danou událost charakteristický konkrétní směr pohybu míče nebo naopak to, že se nepohybuje. Nejdůležitější a také nejobtížněji detekovatelnou dílčí událostí je však dotek hráče s míčem. Detekci interakce hráčů s míčem je proto věnována speciální část této sekce (viz 3.1.1). Některé důležité dílčí události obsahuje tabulka 3.1.

Použití tohoto přístupu vyžaduje velice kvalitní vstupní data. Aby bylo možné správně detekovat všechny dílčí události, potřebujeme prostoročasová data s identifikací hráčů. Musíme být schopni odlišit brankáře od hráčů v poli a bezpodmínečně nutné je vědět, kteří hráči hrají v kterém týmu. Dále je třeba znát souřadnice branek a lajnování hřiště. Nezbytná je také informace o volbě stran týmů, tedy na kterou bránu týmy útočí. Tuto znalost lze jednoduše získat z počátečního rozestavení hráčů na začátku utkání. Detekce některých dílčích událostí si vyžaduje také trojrozměrné souřadnice míče. Výška polohy míče je potřeba zejména při určení směru jeho pohybu nebo rozlišení toho, zda se míč nachází v brance nebo nad ní.

Uveďme si nyní jednoduchý příklad detekce události na rozpoznání zásahu brankáře. Rozpoznání událostí této kategorie je realizováno detekcí následující sekvence dílčích událostí.

1. dotek hráče týmu B s míčem

Název	Popis
dotek hráče s míčem	jakýkoli pravidly povolený kontakt hráče s míčem na hřišti
vhození míče	hráč drží míč oběma rukama nad hlavou a hodí jej do hřiště
míč v brance	míč je celým svým objemem za brankovou čárou a mezi tyčemi branky
míč za brankovou čárou	míč je celým svým objemem za brankovou čárou a mimo branku
míč za postranní čárou	míč je celým svým objemem za postranní čárou
míč na hřišti	míč je na hřišti (není celým svým objemem za postranní ani brankovou čárou)
míč bez pohybu	rychlost míče je nulová
míč směřuje do brány	pohyb míče směřuje mezi tyče brány
míč směřuje do blízkosti brány	pohyb míče nesměřuje do brány, ale směřuje do její blízkosti za postranní čáru (za blízkost brány může být považováno třeba brankové území)
pozice hráčů při rozehrávce (týmu A) ze středu hřiště	uvnitř středového kruhu jsou jen hráči týmu A a ostatní hráči jsou každý na své polovině hřiště
pozice hráčů při pokutovém kopu (týmu A)	všichni hráči, kromě brankáře týmu B a maximálně jednoho hráče týmu A se nachází za hranicí pokutového území týmu B, za pokutovým obloukem týmu B a dále od brankové čáry týmu B než míč
hráč v ofsajdovém postavení	hráč je v ofsajdové pozici, je-li současně blíže k soupeřově brankové čáře než míč a než předposlední hráč soupeře
dotek dvou protihráčů	dva protihráči jsou dostatečně blízko, aby mezi nimi mohlo dojít k fyzickému kontaktu

Tabulka 3.1: Dílčí události

2. pohyb míče směřuje do brány týmu A

3. dotek brankáře týmu A s míčem

4. míč nesměřuje do brány týmu A

Aby byla definice procesu detekce událostí jednoznačná, potřebujeme ke každému bodu sekvence přiřadit také jakýsi typ návaznosti, který definuje vztah mezi dílčími událostmi z po sobě následujících bodů sekvence. Typ návaznosti řeší otázku, zda se mezi jednotlivými body sekvence mohou případně vyskytovat i další dílčí události nebo-li jak dlouho čekat na výskyt události z následujícího bodu sekvence. Tím je jednoznačně definováno, kdy proces detekce události končí neúspěchem. Rozlišujeme následující tři typy návaznosti dílčích událostí a označujeme je symboly *, ^, ~.

* Hvězdička znamená, že se mezi dílčími událostmi v současném a předchozím bodu sekvence nemůže vyskytovat dotek žádného hráče s míčem. Dílčí události tedy musejí následovat bezprostředně za sebou.

^ Symbol stříšky znamená, že mezi dílčími událostmi v současném a předchozím bodu sekvence může dojít pouze k tečování (v případě brankáře i vyražení) míče protihráči hráče, který se naposledy dotkl míče.

~ Vlnovka znamená, že se mezi dílčími událostmi v současném a předchozím bodu sekvence může vyskytovat neomezený počet doteků jakýchkoli hráčů s míčem, avšak nesmí být úspěšně dokončena detekce jiné události.

Protože typ návaznosti vyjadřuje vztah k dílčím událostem z předchozího bodu sekvence, první bod jej nikdy nemá. Výsledný popis detekce zásahu brankáře i s typy návaznosti tedy vypadá následovně.

1. dotek hráče týmu B s míčem

*2. pohyb míče směřuje do brány týmu A

^3. dotek brankáře týmu A s míčem

*4. míč nesměřuje do brány týmu A

Tabulka 3.2 obsahuje popis detekce událostí spojených s přerušením hry a také všech podkategorií události střely (použití symbolu + vyjadřuje současné nastání dílčích událostí).

Při popisu událostí zapřičiňujících přerušení hry sekvencí dílčích událostí musíme dbát na to, aby součástí procesu detekce bylo i potvrzení, že rozhodčí danou situaci zaznamenal a posoudil jako onu událost. Například pro rozpoznání platného gólu nestačí jen detekovat, že míč přešel brankovou čáru do branky, ale je třeba detekovat také rozehrávku ze středu hřiště, která podle pravidel po gólu musí následovat. Podobně u kategorií událostí, kterými hra pokračuje po uvedení míče zpět do hry, je užitečné, aby součástí procesu detekce byla situace, která k přerušení hry vedla. Příkladem je rozpoznávání rohového kopu, které začíná už detekcí doteku hráče s míčem a následným překročením míče brankovou čáru mimo branku. V některých případech, kdy jsou události na sobě závislé, tato metoda vede dokonce ke spojení detekce dvou událostí různých kategorií do jednoho rozhodovacího procesu. Třeba po odpískaném ofsajdu nebo faulu vždy následuje volný kop. Tímto způsobem můžeme předcházet chybné klasifikaci speciálních situací jako je třeba neplatný gól,

„ofsajd“ z kopu od branky nebo volný kop z místa těsně u rohového praporku, který by mohl být snadno klasifikován jako rohový kop. Použití této metody umožňuje také tolerovat chybná rozhodnutí rozhodčích.

Větší potíží je však případ, kdy rozhodčí odpíská událost buď chybně nebo správně, avšak nezaznamená ji náš systém automatické detekce událostí třeba v důsledku nepřesných vstupních prostoročasových dat. V takovém případě je možným řešením nejprve detekovat důsledek události a poté zpětně zkontrolovat příčinu. Pro vysvětlení můžeme použít opět příklad rozpoznávání gólu. Pokud systém detekuje dílčí událost rozehrávky ze středu hřiště, ale nezaznamenal předchozí výskyt míče v brance, můžeme dodatečně zkontrolovat, zda míč neměl k překročení brankové čáry mezi tyčemi branky alespoň blízko. Tento způsob navádí k myšlence detekovat dílčí události s určitou tolerancí. Než abychom striktně rozlišovali pouze to, zda dílčí událost nastala nebo ne, můžeme místo toho určit míru nebo pravděpodobnost, s kterou k ní došlo. Pro použití této metody je tedy třeba pro každou dílčí událost definovat vhodnou pravděpodobnostní funkci a práh detekce. Funkce by měly reflektovat skutečnosti jako, čím blíže je míč nebo hráč nějaké dané pozici, tím je pravděpodobnost, že se na této pozici nachází, větší nebo podobně čím menší je rychlost pohybu míče, tím je větší pravděpodobnost nastání dílčí události zastavení míče. Práh detekce pak určuje jakousi míru tolerance. Detekci dílčí události považujeme za úspěšnou, pouze pokud je pravděpodobnost její nastání vetší než daný práh. Nakonec totiž musíme být vždy schopni striktně rozhodnout, zda k události došlo nebo ne. Touto cestou je možné se lépe vypořádat nejen s nepřesností vstupních prostoročasových dat, ale i s chybnými rozhodnutími rozhodčích nebo s jejich benevolencí při lpění na přesném dodržování pravidel.

Úspěšnost tohoto přístupu stojí na vhodném popisu událostí a na úspěšnosti detekce jednotlivých dílčích událostí. Události je třeba popsat dostatečně obecně, aby byly pokryty všechny varianty události, ale zároveň dostatečně detailně, aby nedocházelo k chybné pozitivní klasifikaci. Někdy je velice těžké i pro člověka přiřadit určitou situaci do správné kategorie události, protože v definici některých událostí figuruje i úmysl hráče. Například mezi centrem do pokutového území a střelou na bránu může být v některých extrémních případech velice tenká hranice a tudíž jediným způsobem, jak správně klasifikovat událost, je znát úmysl hráče. Avšak třeba detekce událostí spojených s přerušením hry, jejíž popis je obsažen v tabulce 3.2, je docela reálná. Úspěšnost detekce dílčích událostí je ale hodně závislá na přesnosti a kvalitě vstupních dat, což je hlavním problémem.

Při rozpoznávání některých jednodušších událostí může způsobovat potíže i skutečnost, že nemáme informaci o tom, zda je hra přerušena nebo probíhá. Může se tak stávat, že detekujeme události i při přerušené hře. To by zejména při použití systému pro automatické počítání statistik utkání vadilo. Tento problém se týká hlavně událostí typu 1 (akcí), jako je přihrávka, zachycení nebo ztráta. Tyto akce jsou totiž součástí běžné manipulace s míčem i během přerušené hry.

3.1.1 Detekce interakce hráče s míčem

Interakce hráčů s míčem definuje průběh hry a je součástí téměř každé události. Dílčí událost doteku hráče s míčem je použita pro definici typů návaznosti a detekce většiny událostí jí právě začíná. Proto je správná a přesná detekce všech doteků základem úspěchu tohoto přístupu. Hráčův dotek s míčem se v prostoročasových datech projeví jako náhlá změna směru nebo také rychlosti pohybu míče. Důležitý je zde přívlastek náhlá, protože k pozvolným změnám směru a rychlosti pohybu míče dochází také přirozeně v důsledku gravitace a rotace míče. Náhlá změna pohybu míče však nastává i při jeho kontaktu se zemí nebo

brankovou konstrukcí. Úkolem je tedy rozpoznat, jestli jsou tyto změny způsobeny hráčem nebo jsou součástí přirozené trajektorie míče. Pokud bychom měli k dispozici ideálně přesná prostoročasová data s trojrozměrnými souřadnicemi míče, bylo by možné předpokládanou přirozenou trajektorii míče vypočítat. V momentě, kdy by se skutečná pozice míče náhle vychýlila z této trajektorie, zaznamenali bychom dotek hráče. Reálná prostoročasová data míče na to však nejsou dostatečně kvalitní. Proto je třeba použít nějaký více heuristický přístup.

Článek [16] popisuje systém automatické detekce ofsajdu, kde dotek s míčem hraje velkou roli. V tomto systému jsou při každé změně směru a rychlosti pohybu míče nové hodnoty porovnávány s těmi předchozími. Pokud se rychlost změní o více než 30% a nebo je změna směru větší něž 15°, systém vyhodnotí, že tato změna je způsobena hráčem nacházejícím se nejblíže míče. Tyto hodnoty autoři získali po dlouhém experimentálním pozorování změn rychlosti a směru míče během hry. Tato heuristika může být sice vyhovující pro systém automatické detekce ofsajdu, avšak je nepravděpodobné, že by tato metoda úspěšně rozpoznala všechny doteky hráčů s míčem. V článku [49] autoři detekují míč a sledují rychlost a trajektorii jeho pohybu v obraze broadcastového vysílání. Pozice míče je určena souřadnicemi středu míče v obraze, přičemž osa Y představuje výšku a osa X šířku snímku. Jsou vytvořeny čtyři množiny: $V = \{f \mid f$ je snímek, na kterém rychlost míče dosáhla lokálního minima nebo snímek, na němž začala rychlost míče narůstat$\}$, $S1 = \{f \mid f$ je snímek, na kterém trajektorie míče dosáhla lokálního maxima v ose Y$\}$, $S2 = \{f \mid f$ je snímek, na kterém trajektorie míče dosáhla lokálního minima v ose Y$\}$ a $S3 = \{f \mid f$ je snímek, na kterém trajektorie míče dosáhla lokálního maxima nebo minima v ose X$\}$. Z množiny $S = V \cup S2 \cup S3 \setminus S1$ jsou pak odebrány snímky, na kterých se v blízkosti míče nevyskytuje žádný hráč. Zbylé snímky jsou považovány za momenty kontaktu míče s nejbližším hráčem. Těmito metodami je inspirována následující heuristická metoda detekce doteku hráče s míčem.

Základní podmínkou tedy je, aby se v dostatečné blízkosti míče nacházel alespoň jeden hráč. Ten nejbližší je pak považovaný za toho, jenž dotek s míčem provedl. Zrychlení míče znamená dotek v každém případě. U záporného zrychlení, tedy zpomalení míče, tomu tak už není. K pozvolnému zpomalování míče totiž dochází i v důsledku tření míče se zemí. Zpomalení míče z tohoto důvodu je však oproti tomu, způsobném dotekem hráče, většinou značně menší. Je tedy třeba experimentálně stanovit práh, který bude tyto příčiny oddělovat. Zpomalení míče potom znamená dotek hráče pouze tehdy, pokud je větší než daný práh. Co se týče změn směru pohybu míče, rozlišujeme změny horizontální (v ose X a Y) a vertikální (osa Z). Horizontální změna směru pohybu míče znamená dotek hráče vždy, ale změna ve vertikálním směru může být způsobena i odrazem míče od země nebo v důsledku gravitace. Pokud se mění pohyb míče z klesání na stoupání, znamená to dotek hráče s míčem pouze tehdy, pokud je Z-souřadnice míče větší než minimální Z-souřadnice, respektive pokud se míč nedotýká země. Poslední variantou je změna směru míče ze stoupání na klesání. Ta znamená dotek hráče s míčem jen tehdy, pokud se rychlost pohybu míče ve směru osy Z před a po změně směru neblíží nule. Takový případ totiž popisuje situaci, kdy míč přestane stoupat v důsledku působení gravitace.

Při realizaci takové heuristické metody je třeba brát v potaz, že k zanedbatelným změnám směru a rychlosti míče může docházet i v důsledku rotace míče a odporu vzduchu nebo také kvůli větru. Tyto malé změny by tedy měly být rozeznány a zcela zanedbány. Největším nepřítelem detekce interakce hráče s míčem jsou však samozřejmě nekvalitní vstupní prostoročasová data. Pokud by byla třeba frekvence získávání aktualizace pozice míče příliš nízká, nebyli bychom schopni rozlišit náhlé změny pohybu míče od plynulých.

U interakce hráče s míčem je možné rozlišovat také různé typy doteku hráče s míčem. Z popisu způsobu detekce událostí výše v této sekci víme, že je důležité umět rozlišit tečování míče hráčem, protože je použito v definici typu návaznosti označeném symbolem $\widehat{\ }$, nebo vhození míče, jelikož je součástí detekce vhazování. Jednotlivé typy doteků identifikujeme na základě toho, zda míč v důsledku kontaktu s hráčem zrychlí či zpomalí, jak se změní jeho směr pohybu nebo jestli se stejný hráč dotkne míče vícekrát. Celkově můžeme rozlišovat následujících šest typů doteku hráče s míčem.

- zahrání – Míč zrychlí a poté následuje dotek jiného hráče nebo míč mimo hřiště.
- popostrčení – Míč zrychlí a poté následuje další dotek stejného hráče.
- tečování – Míč zpomalí, změní směr o méně než 90° a poté následuje dotek jiného hráče nebo míč mimo hřiště.
- odražení/vyražení (brankářem) – Míč zpomalí a poté následuje dotek jiného hráče nebo míč mimo hřiště.
- zpracování – Míč zpomalí a poté následuje další dotek stejného hráče.
- vhození/vyhození (brankářem) – Míč změní směr v důsledku interakce s jedním hráčem alespoň dvakrát po sobě, aniž by výška polohy míče klesla pod určitou hranici. U vhození by touto hranicí měla být výška hráče bez hlavy a u vyhození brankářem může být nižší (třeba výška pasu hráče). Poté následuje dotek jiného hráče nebo míč mimo hřiště.

Tyto definice typů doteků byly vytvořeny heuristickým postupem, avšak pro rozpoznávání doteků hráčů s míčem a jejich klasifikaci lze využít i metody strojového učení. Vhodným prostředkem jsou třeba umělé neuronové sítě, které lze natrénovat pro rozpoznání různých druhů doteků hráčů s míčem na základě rychlosti, zrychlení a změny směru. Například článek [35] popisuje využití umělé neuronové sítě pro rozpoznání kopu do míče a přijetí míče hráčem.

3.2 Využití metod strojového učení

Potenciál tohoto přístupu spočívá v tom, že algoritmy metod strojového učení se samy dokáží naučit vztah mezi vstupními a výstupními hodnotami řešeného problému. V našem případě to znamená, že nemusíme vymýšlet a definovat pravidla popisující jednotlivé události, jako tomu je u přístupu popisovaném v předchozí sekci 3.1. Místo toho necháme nějaký algoritmus, aby se sám naučil rozpoznávat události z daných vstupních dat. Metody strojového učení lze rozdělit do dvou hlavních kategorií: učení s učitelem a bez učitele. Metodám učení s učitelem je potřeba poskytnou příklady vstupů a požadovaných výstupů. Cílem je z takových trénovacích dat určit obecnou funkci, která mapuje vstupy na výstupy. Při učení bez učitele požadované výstupy známé nejsou a algoritmus musí sám nalézt strukturu vstupních dat. Protože se tyto algoritmy učí na vstupních datech, vypořádávají se s jejich nepřesnostmi implicitně a mohou se samy naučit také míru, s jakou rozhodčí tolerují nedodržování určitých fotbalových pravidel. Nevýhodou použití strojového učení je fakt, že zvolený algoritmus nemusí být schopen se naučit řešit zadaný úkol dostatečně přesně nebo může učení trvat příliš dlouho. To lze ovlivnit jednak výběrem vhodného algoritmu pro řešený problém a jednak výběrem vstupních dat, které mu poskytneme. Strojové učení lze

aplikovat na problémy spadající do různých oblastí: klasifikace, regresní analýza, shluková analýza atd. Daný specifický problém, který chceme řešit pomocí strojového učení, je tedy většinou potřeba převést na úlohu spadající do některé z takových oblastí.

3.2.1 Značení sekvencí

Problém rozpoznávání událostí lze formulovat jako klasifikační úlohu, konkrétně značení sekvencí (anglicky sequence labeling) [19]. Cílem značení sekvencí je přiřadit sekvenci návěští z konečné abecedy k sekvenci vstupních dat, přičemž uvažujeme, že výstupní sekvence je nanejvýš stejně dlouhá jako vstupní sekvence. Klasifikační úlohy se obvykle řeší metodami učení s učitelem. Konkrétním metodám vhodným pro řešení problémů spadajících do oblasti značení sekvencí se věnuje až další kapitola 4. Problém značení sekvencí můžeme formalizovat následovně:

Nechť $S \subset \mathcal{X} \times \mathcal{Z}$ je množina trénovacích vzorků. Vstupní prostor $\mathcal{X} = (\mathbb{R}^M)^*$ je množina všech sekvencí délky M vektorů reálných čísel. Výstupní prostor $\mathcal{Z} = L^*$ je množina všech sekvencí návěští z abecedy L. Každý prvek množiny trénovacích vzorků S je dvojice sekvencí (x, z). Výstupní sekvence $z = (z_1, z_2, ..., z_N)$ není delší než vstupní sekvence $x = (x_1, x_2, ..., x_M)$. Platí tedy $|z| = N \leq |x| = M$. Úkolem je použít trénovací množinu S k vytvoření algoritmu pro značení sekvencí $h : \mathcal{X} \mapsto \mathcal{Z}$ a poté s jeho použitím co nejpřesněji označit vstupní sekvence testovací množiny $S' \subset \mathcal{X} \times \mathcal{Z}$ přičemž platí $S \cup S' = \emptyset$.

Na výslednou sekvenci návěští můžeme klást ještě větší omezení a podle toho rozlišovat tři druhy úloh značení sekvencí: klasifikace sekvencí, klasifikace segmentů a časovou klasifikaci.

Klasifikace sekvencí

Jako klasifikace sekvencí označujeme úlohy, kdy je délka výstupní sekvence omezena na jedinou hodnotu. To znamená, že každá vstupní sekvence je přiřazena do jediné třídy. Pokud jsou vstupní sekvence stejně dlouhé nebo je lze doplnit tak, aby byly, můžeme hodnoty popisující jednotlivé prvky sekvence spojit do jediného vstupního vektoru. Pak je možné aplikovat i algoritmy pro klasifikaci klasických nesekvenčních dat (příklady takových metod jsou zmíněny na začátku kapitoly 4). Nicméně, použití metod navržených přímo pro zpracování sekvenčních dat může být výhodnější. Dokáží totiž lépe modelovat data sekvenčního charakteru a obecně dosahují lepších výsledků.

Klasifikace segmentů

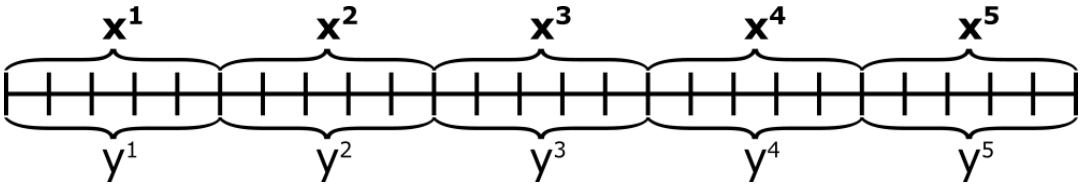
Klasifikace segmentů označuje úlohy, kdy se výsledná sekvence skládá z více návěští. Avšak pozice jednotlivých návěští, nebo-li části vstupní sekvence, ke kterým se jednotlivé návěští vztahují, jsou předem známy. Klíčovou složkou klasifikace segmentů, je využití kontextové informace z okolí klasifikovaného segmentu. Efektivní využití kontextu je pro úspěšnou klasifikaci segmentu velice důležité. To představuje problém pro standardní algoritmy pro klasifikaci nesekvenčních dat, protože nedokáží zpracovávat více vstupů naráz. Jednoduchým řešením je posbírat data z okolí segmentů do časových oken a jako vstup klasifikátoru použít data z celého okna najednou (viz sekce 4.1). Potíží tohoto přístupu je však fakt, že rozsah užitečného kontextu (tedy i velikost časového okna), je obecně neznámá a může se lišit segment od segmentu. Metody strojového učení pro zpracování sekvencí jsou tedy zde zapotřebí ještě více než u klasifikace sekvencí.

Časová klasifikace

Nejobecnějším případem značení sekvencí je časová klasifikace, u níž žádné další omezení vyžadovány nejsou. Platí tedy pouze původní omezení o maximální délce výstupní sekvence návěští. Zásadním rozdílem mezi časovou klasifikací a klasifikací segmentů je, že předchozí přístup vyžaduje algoritmus určující, v kterých místech vstupní sekvence má být klasifikace provedena. Tento druh značení sekvence zase vyžaduje implicitní či explicitní model celkové struktury sekvence.

3.2.2 Formulace detekce událostí jako problém značení sekvencí

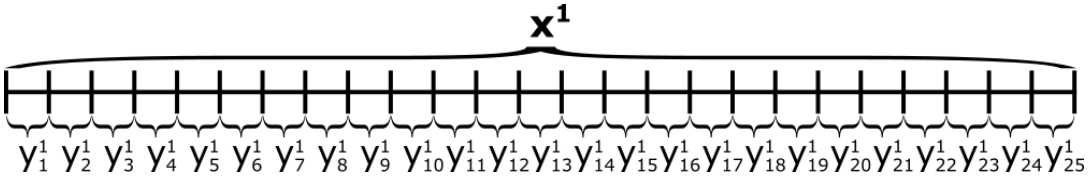
Prostoročasová data fotbalového utkání tvoří jednu dlouhou časovou sekvenci. Každý prvek této sekvence reprezentuje stav na hřišti v daném okamžiku. Můžeme tedy utkání rozdělit do více kratších sekvencí a pomocí klasifikátorů rozhodovat, zda daný úsek utkání obsahuje událost a jakou (viz obrázek 3.1). Tento přístup spadá pod klasifikaci sekvencí nebo-li značení sekvencí N:1. Abeceda návěští je tvořena identifikátory událostí, které chceme detekovat, plus jedním návěštím pro sekvence neobsahující žádnou událost. Otázkou však je, jak utkání rozdělit tak, aby každá ze sekvencí buď obsahovala pouze jednu celou událost nebo neobsahovala žádnou. To představuje velký problém vzhledem k tomu, že pozice událostí v časové sekvenci celého utkání je právě to, co se snažíme zjistit. Jednotlivé události ve fotbale mají navíc různou délku trvání.



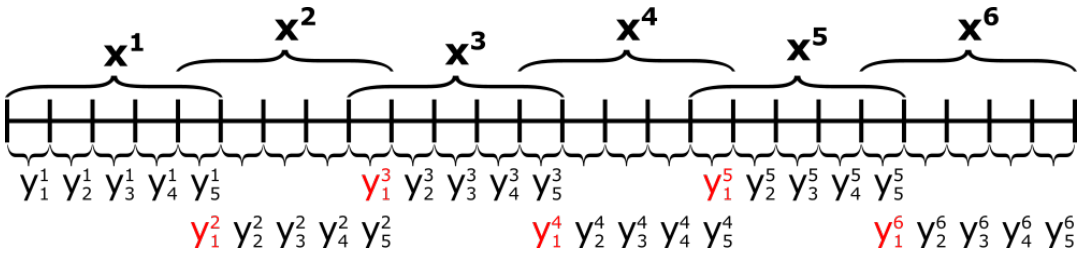
Obrázek 3.1: Detekce událostí jako klasifikace sekvencí ($\boldsymbol{x^i}$ – vstupní sekvence, y^i – výstupní návěští)

Další možností je formulovat úlohu jako klasifikaci segmentů, přičemž délka sekvence návěští je totožná s délkou vstupní sekvence. Tuto variantu úloh můžeme označovat také jako značení sekvencí N:N. Přístup je znázorněn obrázkem 3.2. Návěští pocházejí ze stejné abecedy jako v předchozím případě. Jsou však přiřazovány každému prvku vstupní sekvence. V podstatě se snažíme zjistit, zda jsou jednotlivé okamžiky utkání součástí nějaké události a určit, o kterou kategorii události se jedná. Zbavili jsme se tedy nutnosti rozdělovat utkání na sekvence obsahující pouze jednu událost, jelikož můžeme jako vstup použít celou sekvenci utkání. Přesto může být žádoucí utkání rozdělit do kratších sekvencí. Některé metody pro značení sekvencí, jako je třeba LSTM RNN, totiž mohou mít s příliš dlouhými vstupními sekvencemi problém (viz podsekce 4.3.2). Pokud tedy utkání do více sekvencí rozdělujeme, je vhodné zajistit, aby měl klasifikátor k dispozici dostatek kontextu pro klasifikaci každého prvku vstupní sekvence. Je zjevné, že nedostatkem kontextu mohou trpět hlavně prvky na začátku a na konci každé vstupní sekvence. Řešením může být použití vstupních sekvencí, které se navzájem částečně překrývají (viz obrázek 3.3). Díky tomu můžeme $N/2$ prvků ze začátku a z konce každé výstupní sekvence návěští zahodit, přičemž N je počet prvků, kterými se vstupní sekvence překrývají (v případě, že je N liché číslo, musíme zahodit z jedné strany o jeden prvek více). Výjimku tvoří první výstupní sekvence, jejíž počáteční prvky nezahazujeme. Stejně tak jsou ponechány prvky i z konce poslední sekvence. Pro příklad na obrázku 3.3 je $N = 1$ a prvky výstupních sekvencí, které jsou zahozeny, jsou

označeny červeně. Spojením výstupních sekvencí dostaneme sekvenci návěští odpovídající celému utkání. Velikost potřebného kontextu pro určení správného návěští však obecně není známa. Hodnotu N je proto třeba odhadnout či určit experimentálně.



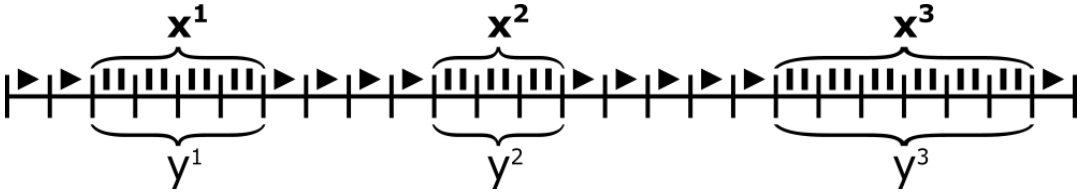
Obrázek 3.2: Detekce událostí jako klasifikace segmentů ($\boldsymbol{x^i}$ – vstupní sekvence, y^i_j – prvek j výstupní sekvence i)



Obrázek 3.3: Překrývající se s vstupní sekvence pro klasifikaci segmentů ($\boldsymbol{x^i}$ – vstupní sekvence, y^i_j – prvek j výstupní sekvence i)

Formulace úlohy jako klasifikace segmentů přináší komplikace s reprezentací výsledků. Pokud by přesnost klasifikátoru byla 100%, detekovaná událost by se ve výsledné sekvenci projevila jako souvislý úsek stejných návěští. Bylo by však naivní předpokládat, že klasifikátor bude bezchybný. Může se tedy například stát, že pár chybně určených návěští způsobí nesprávnou interpretaci daného úseku jako dvě po sobě následující události stejného typu. Možným způsobem, jak tolerovat nepřesnost klasifikátoru při reprezentaci výsledků, je místo souvislých sekvencí stejných návěští hledat jejich shluky. Běžnou metodou, jak se vyrovnat se zašumělými daty, je použít nějaký filtr pro jejich vyhlazení. Pro tento problém by se hodil například jednoduchý modus filtr [36], který nahrazuje každou hodnotu nejčastější hodnotou z okolí o fixní velikosti. Další možností je třeba heuristicky určit minimální mezeru mezi dvěma různými událostmi a nebo minimální délku události. Je-li však klasifikátor příliš nepřesný, nepomohou ani tyto úpravy.

Klasifikátory s menší abecedou návěští jsou obecně přesnější, protože mají menší možnost vybrat nesprávné návěští. Toho můžeme využít a zkombinovat obě předchozí varianty, klasifikaci sekvencí a segmentů. Zaměříme-li se pouze na detekci událostí spojených s přerušením hry (typ 2), můžeme nejprve hru rozdělit na úseky, kdy hra probíhá a kdy je přerušena, pomocí metody klasifikace segmentů a poté klasifikovat jednotlivé části přerušené hry metodou klasifikace sekvencí. Znázornění této metody můžeme vidět na obrázku 3.4. Rozdělením utkání do částí, kdy hra probíhá a kdy je přerušena, lokalizujeme sekvence obsahující vždy jen jednu událost typu 2. Navíc, protože klasifikace segmentů je v tomto případě pouze binární (abeceda návěští obsahuje pouze dva prvky), klasifikátor má potenciál být přesnější. Podobný přístup, kdy jsou části probíhajícího a přerušeného utkání analyzovány zvlášť, je popsán v článku [46].



Obrázek 3.4: Detekce událostí klasifikováním sekvencí přerušené hry ($\boldsymbol{x^i}$ – vstupní sekvence, y^i – výstupní návěští). Symboly ▶ (navěští probíhající hry) a ❙❙ (návěští přerušené hry) znázorňují výsledek rozdělení utkání na části obsahující probíhající hru a části s přerušenou hrou pomocí metody klasifikace segmentů.

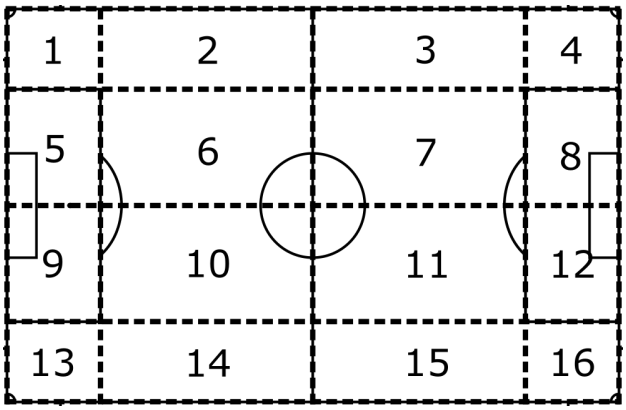
3.2.3 Předzpracování vstupních prostoročasových dat

Přímočarý přístup je použít vstupní sekvence, jejichž prvky tvoří vektory obsahující přímo pozice hráčů a míče v souřadnicovém systému hřiště. Může být však vhodnější z prostoročasových dat vypočítat i jiné hodnoty popisující stav na hřišti a použít je na vstupu společně s pozicemi nebo místo nich. Cílem takového předzpracování je pomoci zvolené metodě strojového učení vybrat z prostoročasových dat relevantní informace a tím zvýšit přesnost či urychlit učení klasifikátoru. Například můžeme předpokládat, že při rozhodování, zda hra probíhá nebo je pozastavena, bude mít velkou vypovídající hodnotu rychlost pohybu hráčů. Jedním z hlavních příznaků rohového kopu je zase pravděpodobně velký počet hráčů v pokutovém území.

V této podsekci si tedy představíme různé charakteristické hodnoty, které je možné použít k popisu stavu na hřišti a lze je vypočítat ze vstupních prostoročasových dat. Jejich výběr je inspirován články [22] [34] [23]. Charakteristické hodnoty můžeme rozdělit jednak podle toho, zda popisují pozice nebo pohyb, a také podle toho, zda se vztahují k hráčům nebo k míči (viz následující seznam).

- Pozice hráčů:
  - počet hráčů v jednotlivých buňkách mřížky, která rozděluje hřiště (lze použít mřížku o libovolném počtu buněk stejné velikosti nebo třeba mřížku představenou v článku [22], která rozděluje hřiště do šestnácti buněk podle obrázku 3.5)
  - počet hráčů na hřišti či mimo hřiště
  - aritmetický průměr souřadnic pozic hráčů (aritmetický průměr X souřadnic a Y souřadnic)
  - největší a nejmenší hodnota z X a Y souřadnic všech hráčů kromě brankářů (tyto čtyři hodnoty vymezují oblast hřiště, ve kterém se pohybují hráči)
  - délka a šířka oblasti, ve kterém se pohybují hráči (kromě brankářů), či poměr těchto dvou hodnot
  - Y/X souřadnice pozice hráče s největší a nejmenší X/Y souřadnicí jeho polohy (jinými slovy tyto hodnoty určují Y/X souřadnice hráčů, kteří jsou nejblíže postranním/brankovým čarám)
  - aritmetický průměr vzdáleností všech hráčů od středové čáry (aritmetický průměr absolutních hodnot Y souřadnic všech hráčů)

  - aritmetický průměr vzdáleností všech hráčů od pomyslné čáry kolmé na středovou čáru a procházející středem hřiště (aritmetický průměr absolutních hodnot X souřadnic všech hráčů)
  - aritmetický průměr vzdáleností mezi jednotlivými hráči
  - počet shluků hráčů při dané minimální vzdálenosti mezi shluky
  - průměrná vzdálenost mezi shluky hráčů při daném počtu shluků
- Pohyb hráčů:
  - průměrná rychlost a zrychlení pohybu všech hráčů (buď v 2D prostoru nebo zvlášť ve směru os X a Y souřadného systému hřiště)
  - součet směrových vektorů pohybu všech hráčů (výsledný vektor můžeme reprezentovat jeho velikostí a úhlem který svírá s osou X souřadného sytému hřiště)
- Pozice míče:
  - identifikátor buňky mřížky v níž se míč nachází (pro podrobnosti o mřížce viz první odrážka druhého stupně tohoto seznamu)
  - zda se míč nachází na hřišti nebo ne
  - vzdálenost od cíle (za cíl je považován střed branky, ke které se míč pohybuje)
- Pohyb míče:
  - rychlost a zrychlení pohybu míče(buď v 2D prostoru nebo zvlášť ve směru os X a Y souřadného systému hřiště)
  - směrový vektor pohybu míče (velikost vektoru a úhel jenž svírá s osou X souřadného systému hřiště)



Obrázek 3.5: Možné rozdělení hřiště do buněk

Existuje spousta hodnot, které lze vypočítat či odvodit ze vstupních prostoročasových dat. Uvedený seznam obsahuje relativně jednoduché hodnoty. Je však možné pracovat i se složitějšími. Například práce [23] představuje výpočet hodnoty určující, jak moc se momentální rozestavení týmu blíží standardní formaci týmu. Různé hodnoty mohou být užitečné při detekci různých událostí. Může být tedy prospěšné s použitím různých charakteristických hodnot experimentovat a zjistit tak, které z nich pomohou dosáhnout lepších výsledků.

Kategorie události	**Popis detekce**
gól	1. dotek jakéhokoli hráče s míčem *2. míč v brance týmu B ~3. míč bez pohybu uprostřed středového kruhu + pozice hráčů při rozehrávce ze středu hřiště (týmu B) *4. dotek hráče týmu B s míčem
ofsajd a následný volný kop týmu B	1. dotek hráče P1 týmu A s míčem + hráč P2 týmu A v ofsajdovém postevení ^2. dotek hráče P2 s míčem ~3. míč bez pohybu na pozici, kde se hráč P2 nacházel při doteku hráče P1 s míčem *4. dotek hráče týmu B s míčem
rohový kop	1. dotek hráče týmu B s míčem *2. míč za brankovou čárou týmu B ~3. míč bez pohybu v rohovém čtvrtkruhu týmu B *4. dotek hráče týmu A s míčem
faul a následný volný kop týmu B	1. dotek dvou protihráčů PA a PB ~2. míč bez pohybu na pozici doteku protihráčů PA a PB *3. dotek hráče týmu B s míčem nebo (hra rukou) 1. dotek hráče P týmu A s míčem ~2. míč bez pohybu na pozici doteku hráče P s míčem *3. dotek hráče týmu B s míčem
pokutový kop	1. míč bez pohybu na pokutové značce v pokutovém území týmu B + pozice hráčů při pokutovém kopu (týmu A) *2. dotek hráče týmu A s míče
vhazování	1. dotek hráče týmu B s míčem *2. míč za postranní čárou ~3. vhození míče hráčem týmu A *4. míč na hřišti
kop od branky	1. dotek hráče týmu B s míčem *2. míč za brankovou čárou týmu A ~3. míč bez pohybu v brankovém území týmu A *4. dotek hráče týmu A s míčem
střela na bránu	1. dotek hráče týmu A s míčem ^2. míč směřuje do brány týmu B ^3. gól nebo dotek posledního hráče týmu B s míčem (posledním hráčem je ten nejblíže vlastní brance)
střela mimo bránu	1. dotek hráče týmu A s míčem *2. míč směřuje do blískoti brány týmu B *3. míč za brankovou čárou v blízkosti brány týmu B
zblokovaná střela	1. dotek hráče týmu A s míčem *2. míč směřuje do brány týmu B nebo do její blízkosti ^3. dotek ne posledního hráče týmu B s míčem

Tabulka 3.2: Popis detekce událostí některých kategorií

Kapitola 4

Metody strojového učení pro značení sekvencí

Jak již bylo zmíněno v podsekci 3.2.1, pro klasifikaci sekvencí (nebo-li značení sekvencí N:1) mohou být použity i algoritmy strojového učení navrženy pro klasifikaci normálních dat nesekvenčního charakteru. Tyto metody jsou někdy označovány jako algoritmy pro klasifikaci vzorů či struktur. Patří zde dopředné neuronové sítě jako je třeba vícevrstvý perceptron, support vector machine (SVM), rozhodovací stromy, náhodné lesy, bayesovské klasifikátory, mnohočlenná logistická regrese, algoritmus k-nejbližších sousedů (k-NN) a další. Některé z těchto algoritmů byly v oblasti analýzy prostoročasových dat fotbalových utkání také použity. Například článek [34] popisuje a srovnává výsledky použití SVM, k-NN a metody náhodného lesu pro rozpoznávání přihrávky, převzetí míče a doteku s míčem. Obecně lze však dosáhnout lepších výsledků použitím metod, které sekvenční charakter vstupních dat berou v potaz.

V této kapitole se nejprve zaměříme na problém klasifikace segmentů, kdy počet segmentů je stejný jako délka vstupní sekvence (značení sekvencí N:N). Vstupní a výstupní sekvence jsou tedy stejně dlouhé. Pro řešení tohoto problému budou uvedeny metody posuvného okna a pravděpodobnostní modely.

Nejuniverzálnějším a nejsilnějším modelem pro značení sekvencí jsou rekurentní neuronové sítě. Ty mohou být teoreticky použity pro všechny úlohy typu značení sekvencí. Ve zbytku kapitoly se tedy zaměříme na neuronové sítě.

4.1 Metody posuvného okna

Metoda posuvného okna [15] je způsob jak převést značení sekvencí N:N na úlohu typu klasifikaci vzorů. Princip použití posuvného okna je následující. Mějme vstupní sekvenci $x = (x_1, ..., x_n)$, výstupní sekvenci $y = (y_1, ..., y_n)$ a velikost posuvného okna $w = 2d + 1$. Potom každý prvek výstupní sekvence y_i je stanoven na základě podmnožiny prvků vstupní sekvence $W_i = (x_{i-d}, x_{i-d+1}, ..., x_i, ..., x_{i+d-1}, x_{i+d})$. Kvůli klasifikaci segmentů ze začátku a z konce vstupní sekvence musí být vstupní sekvence rozšířeny na každé straně o d prvků nulové hodnoty. Metoda posuvného okna tedy umožňuje použít jakýkoli algoritmus pro klasifikaci vzorů. Ten je trénován na dvojicích (W_i, y_i). Nová sekvence x je označena nejprve její transformaci do podmnožin W_i, použitím klasifikátoru k stanovení každého návěští y_i a nakonec zřetězením návěští pro vytvoření výstupní sekvence y.

Jeden způsob, jak metodu posuvného okna dále vylepšit, je rozšířit vstup použitého klasifikátoru o hodnoty předchozích, již vypočtených návěští. Tato varianta se nazývá rekurentní posuvné okno. Konkrétně při použití okna o velikosti $2d + 1$, hodnoty návěští $y_{i-d}, y_{i-d+1}, ..., y_{i-1}$ jsou použity spolu s podmnožinou prvků vstupní sekvence W_i k stanovení y_i. Díky tomu je možné využít případné závislosti mezi jednotlivými prvky výstupní sekvence y_i, které nejsou předpověditelné pouze z prvků podmnožiny W_i.

4.2 Pravděpodobnostní modely

Pro značení sekvencí N:N lze použít také pravděpodobnostní modely jako je hidden Markov model (HMM) [42] [32], maximum entropy Markov model (MEMM) [29] nebo conditional random field (CRF) [43]. Příklad použití pravděpodobnostních modelů pro analýzu prostoročasových dat fotbalových utkání můžeme nalézt v práci [22], kde jsou HMM použity pro rozpoznávání čtyř fotbalových událostí: vhazování, přímého kopu, rohového kopu a kopu od branky.

V terminologii HMM označujeme vstupní sekvenci $x = (x_1, ..., x_n)$ jako sekvenci jednotlivých pozorování x_i a výstupní sekvenci návěští $y = (y_1, ..., y_n)$ jako sekvenci skrytých stavů y_i. HMM je generativní pravděpodobnostní model popisující generování jak sekvence skrytých stavů y tak sekvence pozorování x. HMM tedy reprezentuje sdruženou pravděpodobnost $P(x, y) = P(y)P(x|y)$. Pravděpodobnostní funkce $P(y) = \prod_{i=1}^{n} P(y_i|y_{i-1})$ definuje závislost mezi sousedními skrytými stavy a pravděpodobnostní funkce $P(x|y) = \prod_{i=1}^{n} P(x_i|y_i)$ popisuje závislost pozorování na hodnotách skrytých stavů. Proces generování sekvencí probíhá tak, že nejprve je vygenerován další skrytý stav (návěští) y_i s pravděpodobností $P(y_i|y_{i-1})$ a poté je vygenerováno další pozorování (prvek vstupní sekvence) x_i s pravděpodobností $P(x_i|y_i)$. Co však ve skutečnosti potřebujeme pro řešení úloh typu značení sekvencí N:N, je nalézt nejpravděpodobnější sekvenci skrytých stavů, která by mohla generovat danou sekvenci pozorování $\hat{y} = \text{argmax}_y P(y|x) = \text{argmax}_y P(x, y)$. Tento úkol se řeší dynamickým programováním pomocí Viterbiho algoritmu [17], který pracuje právě s HMM.

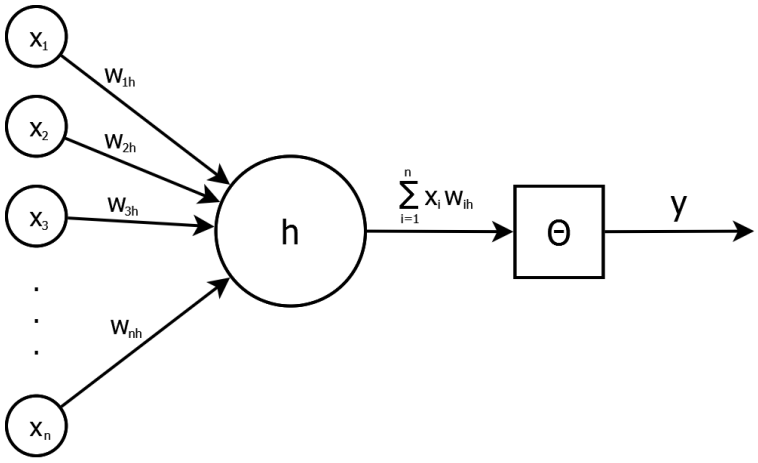
HMM však mají jistá omezení [15]. Při použití HMM se předpokládá, že jednotlivé hodnoty každého pozorování $x_i = (x_{i,1}, ..., x_{i,m})$ jsou navzájem nezávislé. To však většinou pro užitečné hodnoty popisující pozorování neplatí. Abychom pomocí HMM zachytili vztah mezi oddělenými skrytými stavy (např. y_1 a y_4), musel by tento vztah být zprostředkován skrz všechny stavy mezi nimi (y_2 a y_3). Klasický HMM, kde $P(y_i)$ závisí pouze na y_{i-1}, tedy obecně tyto vztahy zachytit nedokáže. HMM generuje každé pozorování x_i pouze na základě příslušného skrytého stavu y_i. Teoreticky bychom mohli nahradit distribuci pravděpodobnosti $P(x_i|y_i)$ složitější distribucí $P(x_i|y_{i-1}, y_i, y_{i+1})$, což by umožňovalo jednomu pozorování x_i ovlivňovat tři skryté stavy. Není však jasné, jak bychom tak složitou distribuci pravděpodobnosti reprezentovali.

Ve snaze vyhnout se těmto omezením HMM vnikly diskriminativní pravděpodobnostní modely pro značení sekvencí N:N MEMM a CRF. Diskriminativní modely vystihují lepé povahu úloh značení sekvencí, protože reprezentují místo sdružené pravděpodobnosti $P(x, y)$ přímo závislost výstupní sekvence na vstupní $P(y|x)$. To jim dovoluje použít posuvná okna a libovolné hodnoty k popisu pozorování včetně vlastností vztahujících se na celou vstupní sekvenci či hodnot popisujících vztahy mezi oddělenými (vzdálenými) prvky. Diskriminativní modely tedy poskytují větší svobodu ve volbě hodnot reprezentujících pozorování (prvky vstupní sekvence) a navíc, na rozdíl od HMM, nepředpokládají jejich vzájemnou nezávislost.

Mnohočlenná logistická regrese (angl. nultinomial logistic regression nebo maximum entropy classifier) [31] je model reprezentující pravděpodobnost $P(y|x)$. Je to však také metoda pro klasifikaci vzorů a tudíž nepracuje se sekvenčními daty. Jejím použitím můžeme tedy generovat každé návěští y_i pouze v závislosti na vstupní sekvenci x nehledě na kontext (ostatní návěští výstupní sekvence). MEMM kombinuje vlastnosti HMM a modelu pro mnohočlennou logistickou regresi. Reprezentuje tak pravděpodobnost $P(y|x) = \prod_{i=1}^{n} P(y_i|y_{i-1}, x)$ popisující závislost každého návěští na předchozím návěští a vstupní sekvenci. Nedostatkem MEMM je tzv. label bias problem jehož důsledkem je, že některá pozorování x_i mohou být ignorována. Modelem který odstraňuje tento problém a přitom si zachovává výhody MEMM je CRF [27].

4.3 Umělé neuronové sítě

Umělé neuronové sítě jsou velice silným modelem v oblasti strojového učení a velmi vhodnou volbou pro značení sekvencí [19]. Základní strukturu umělé neuronové sítě tvoří malé výpočetní jednotky zvané neurony, které jsou navzájem spojeny váženými propojeními. Neuron může mít libovolný počet vstupů, ale pouze jeden výstup. Navzájem propojené neurony si předávají signály a transformují je pomocí určitých přenosových funkcí. Model umělého neuronu můžeme vidět na obrázku 4.1. Vstupy neuronu jsou označeny jako x_i a w_{ih} jsou příslušné váhy propojení vstupů s neuronem h. Ten je charakterizován přenosovou (také aktivační) funkcí Θ. Výstup neuronu y je pak definovaný vztahem $y = \Theta(\sum_{i=1}^{n} w_{i,h}x_i)$.



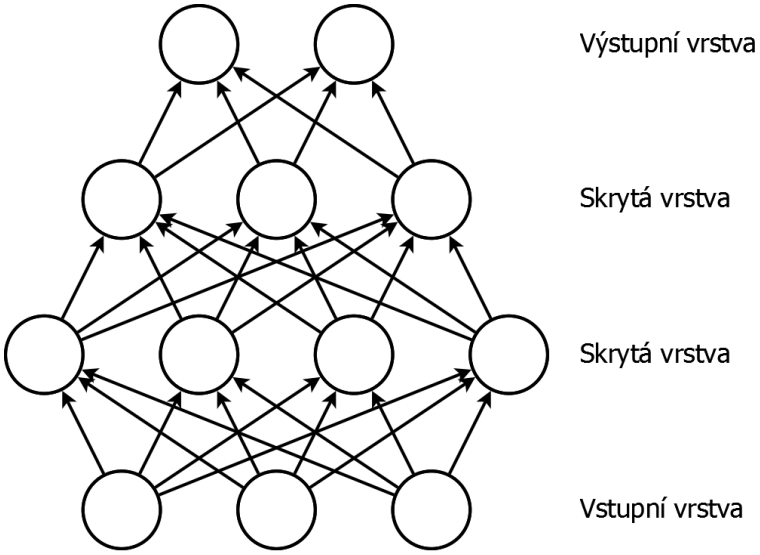
Obrázek 4.1: Model umělého neuronu

Neuronová síť s konkrétní množinou vah propojení definuje funkci mapující vstupní vektory na výstupní. Změnou vah je stejná neuronová síť schopna vytvořit mnoho různých funkcí. Trénování nebo-li učení sítě spočívá právě v modifikaci vah propojení sítě tak, abychom minimalizovali danou chybovou funkci. Chybová funkce říká, jak moc se skutečný výstup sítě liší od požadovaného výstupu.

Existuje mnoho variant neuronových sítí s různými vlastnostmi. Jedno z nejpodstatnějších rozdělení neuronových sítí závisí na tom, jestli propojení mezi neurony sítě tvoří cykly. Neuronové sítě s acyklickým propojením neuronů se nazývají dopředné a sítím jejichž propojení cykly tvoří se říká buď zpětnovazební, rekurzivní nebo rekurentní (angl. Recurrent Neural Network – RNN).

4.3.1 Vícevrstvý perceptron

Nejrozšířenější variantou dopředné neuronové sítě je vícevrstvý perceptron. Jak vidíme na obrázku 4.2, neurony vícevrstvého perceptronu jsou uspořádány do vrstev a propojení vedou pouze do neuronů následující vrstvy. Vstupní hodnoty jsou posílány na vstupní vrstvu a poté se síří přes skryté vrstvy do výstupní. Tomuto procesu se říká dopředné šíření sítí. Metoda učení vícevrstvého perceptronu se nazývá algoritmus zpětného šíření chyby (angl. backpropagation) [37]. Nejprve je spočítána chyba na neuronech výstupní vrstvy a tato chyba se poté šíří zpět strukturou sítě na všechny vrstvy předchozí. Každému neuronu sítě je tak dopravena chyba, která odpovídá jeho příspěvku k celkové chybě na výstupní vrstvě. Podle toho jsou pak upraveny jednotlivé váhy propojení sítě. Trénování sítě se proto označuje jako zpětné šíření sítí.

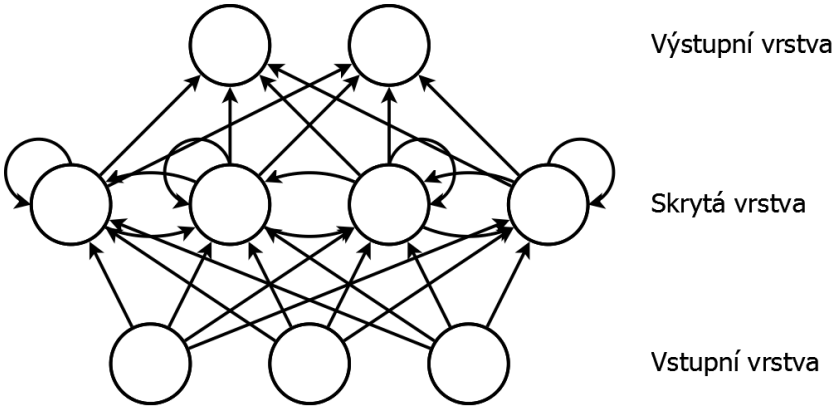


Obrázek 4.2: Vícevrstvý perceptron

Protože výstup vícevrstvého perceptronu závisí pouze na současném vstupu (a ne na minulých či budoucích vstupech), jsou vícevrstvé perceptrony vhodné pro klasifikaci pouze obyčejných nesekvenčních dat. Povolení zpětných vazeb (smyček v propojení sítě) však umožňuje neuronové síti uchovávat informaci o předchozích vstupech a tím pádem produkovat výstup závislý na v podstatě celé historii předešlých vstupů. To je pro zpracování sekvencí klíčové a proto se dále budeme zabývat rekurentními neuronovými sítěmi. Příklad architektury rekurentní neuronové sítě s jednou skrytou vrstvou můžeme vidět na obrázku 4.3.

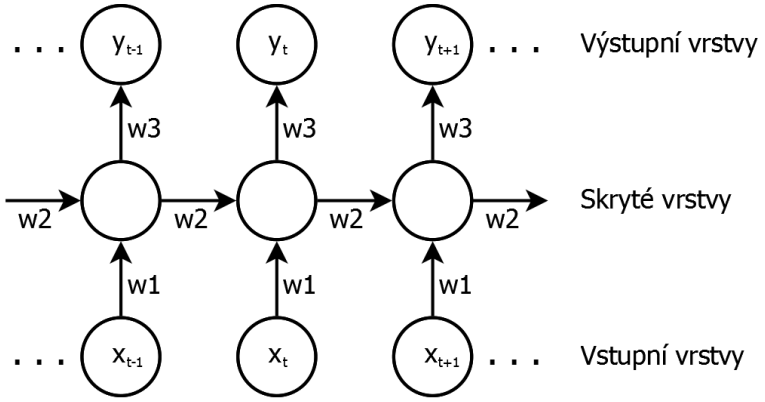
4.3.2 Rekurentní neuronové sítě

RNN si také můžeme představit jako více kopií stejné sítě, z nichž každá předává informaci následující kopii (viz obrázek 4.4). Tomuto zobrazení se říká rozvinutá RNN. Každý uzel zde reprezentuje vrstvu neuronů v konkrétním časovém kroku t, kdy sít zpracovává daný prvek x_t vstupní sekvence a produkuje příslušný prvek výstupní sekvence y_t. Propojení ze vstupní vrstvy do skryté jsou označeny $w1$, propojení skryté vrstvy tvořící smyčky jsou označeny jako $w2$ a propojení skryté vrstvy s výstupní je označeno $w3$. Všimněme si, že v každém časovém kroku jsou použity stejné váhy, protože jednotlivé sloupce uzlů tohoto



Obrázek 4.3: Rekurentní neuronová síť

grafu představují stejnou sít. RNN tedy umožňují pracovat se sekvencemi libovolné délky jak na jejich vstupu tak na výstupu. Obrázek 4.5 pomocí rozvinutí sítě ukazuje různé varianty, jak může RNN pracovat se sekvencemi [25]. Varianta c) představuje klasifikaci sekvencí a varianta e) zase odpovídá úlohám typu značení sekvencí N:N.

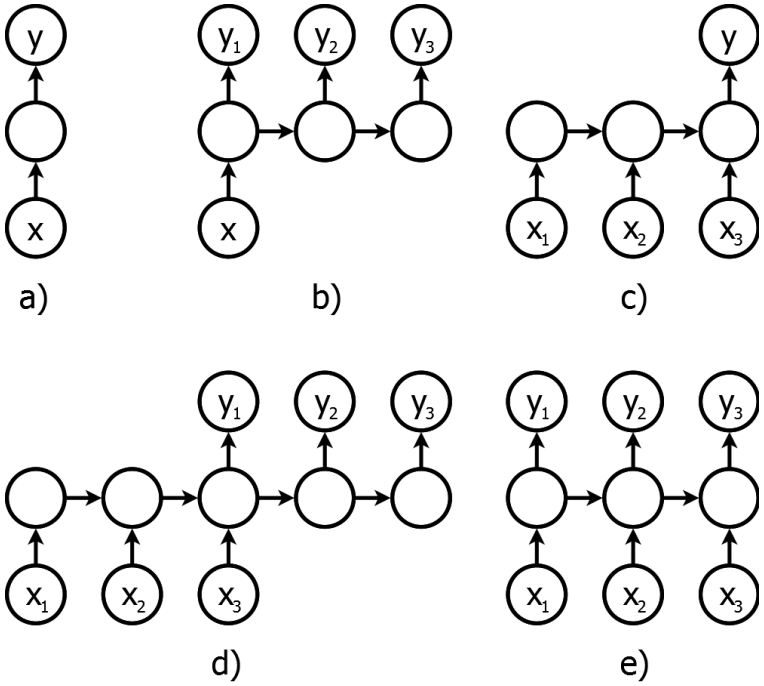


Obrázek 4.4: Rozvinutá rekurentní neuronová síť

Princip dopředného šíření se u RNN nijak nemění. Neurony skryté vrstvy akorát přijímají na svém vstupu navíc výstupy neuronů skryté vrstvy z předchozího kroku. Algoritmus pro učení RNN se nazývá backpropagation through time (BPTT) [47] [48] a je obdobou algoritmu zpětného šíření chyby pro učení dopředných neuronových sítí. Liší se v tom, že chyba je propagována zpět rozvinutou sítí skrz jednotlivé časové kroky.

Bylo již vysvětleno, že RNN těží ze schopnosti uchovávat informaci o předcházejících vstupech. Pro spoustu úloh typu značení sekvencí (typicky pro klasifikaci segmentů) je však výhodné mít informaci také o následujících prvcích vstupní sekvence. Tento nedostatek elegantně řeší tzv. obousměrné RNN [39]. Základní myšlenka obousměrných RNN je poskytnout síti vstupní sekvenci navíc také pozpátku prostřednictvím další rekurentní skryté vrstvy. Rozvinutá obousměrná RNN je zobrazena na obrázku 4.6. Tato struktura poskytuje síti při výpočtu každého prvku výstupní sekvence už kompletní kontext. To znamená informaci jak o předešlých tak o následujících prvcích vstupní sekvence.

Největší výhodou RNN je tedy jejich schopnost využít informaci o kontextu při výpočtu momentálního výstupu. Naneštěstí velikost dostupného kontextu je pro obyčejné architek-



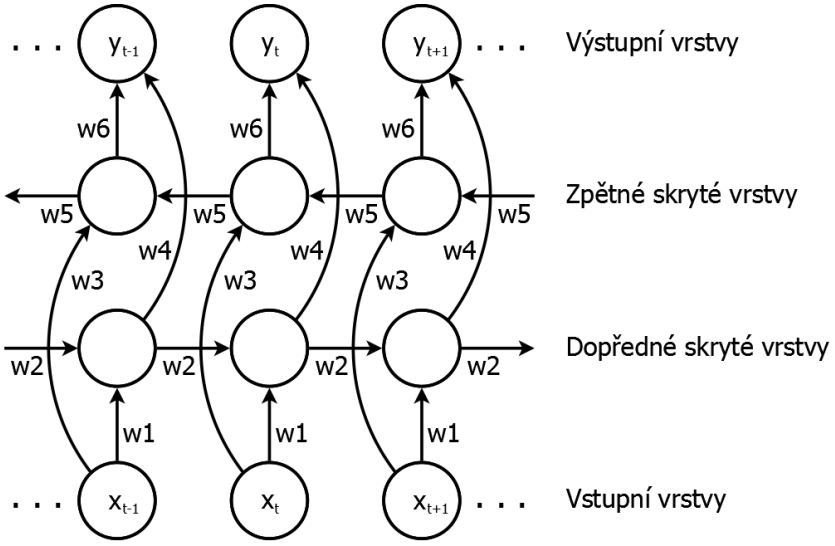
Obrázek 4.5: Varianty zpracování sekvencí pomocí RNN

tury RNN poměrně omezená. Problém spočívá v tom, že vliv daného vstupu na skrytou vrstvu (a tudíž i na výstup sítě) časem buďto rychle mizí nebo naopak roste exponenciálně. Tento jev se označuje jako problém mizejícího gradientu [8]. Pro tento problém však naštěstí existuje řešení. Long short-term memory (LSTM) [21] je název zvláštního druhu RNN, která je speciálně navržena pro učení dlouhodobých závislostí.

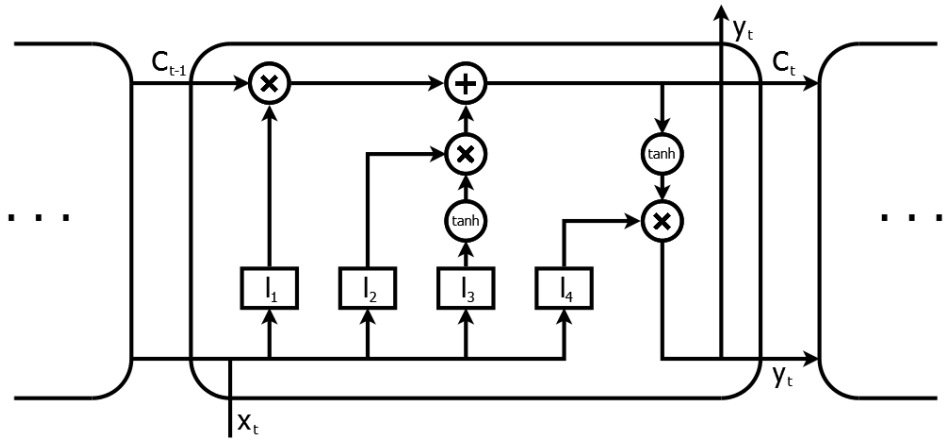
Long short-term memory (LSTM)

LSTM dokáže přechovat informaci po mnohem delší dobu než obyčejné RNN. Článek [30] velmi pěkně vysvětluje princip, jakým LSTM sítě fungují. V podstatě se dokáže naučit, jaké informace o předchozích vstupech si pamatovat a které naopak zapomenout. Skrytá vrstva LSTM není tvořena pouze jednou řadou neuronů, ale ve skutečnosti je složena ze čtyř vrstev, které jsou navzájem propojeny velmi speciálním způsobem. Toto propojení můžeme vidět na obrázku 4.7. C_t je stav LSTM, který drží informaci o předchozím kontextu. Vrstva l_1 rozhoduje o tom, které části informace starého stavu C_{t-1} mají být zapomenuty. Další vrstva l_2 zase určuje, jaké nové informace budou do nového stavu C_t přidány. Tyto nové informace jsou vybírány z výstupu třetí vrstvy l_3, která vytváří vektor možných hodnot pro přidání do nového stavu C_t. Nakonec vrstva l_4 rozhoduje o tom, které informace ze stavu C_t mají být poslány na výstup.

Přesto, že LSTM dokáže přechovat informaci o předchozím kontextu mnohem delší dobu než obyčejné RNN, má také své limity. Jsou-li vstupní sekvence přespříliš dlouhé (více než 500 prvků), muže mít i LSTM s jejich efektivním zpracováním potíže [12]. Trénování sítě tak může trvat buď příliš dlouho a nebo se projeví problém mizejícího gradientu, což může vést k neschopnosti sítě se zadaný model naučit. Pokud tedy řešíme úlohu, kdy vstupní sekvence jsou příliš dlouhé, máme na výběr ze tří řešení. Prvním z nich je vstupní sekvence jednoduše zkrátit. To lze zjevně provést mnoha způsoby, avšak zkrácením sekvence



Obrázek 4.6: Rozvinutá obousměrná rekurentní neuronová síť (dopředná skrytá vrstva zpracovává vstupní sekvenci od začátku a zpětná skrytá vrstva od konce)



Obrázek 4.7: Rozvinutá LSTM vrstva

vždy riskujeme ztrátu dat hodnotných pro přesný výstup sítě. Druhou variantou je rozdělit vstupní sekvence do většího množství kratších sekvencí. Tímto způsobem však ztratíme možnost zachytit případný vztah mezi blízkými prvky, které v důsledku rozdělení připadnou do různých vstupních sekvencí. Třetím řešením je použit pro učení LSTM sítě speciální verzi algoritmu pro učení RNN zvanou truncated backpropagation through time (TBPTT) [13]. Tato varianta omezuje počet časových kroků použitých při dopředném i zpětném šíření sítí. Důsledkem je zrychlení učení sítě na velmi dlouhých vstupních sekvencích a vyhneme se také problému mizejícího gradientu. Cenou však je, že sít v podstatě bere při výpočtu výstupu v daném časovém kroku v potaz pouze určitý počet předchozích časových kroků.

4.3.3 Parametry neuronové sítě pro značení sekvencí

Již bylo uvedeno, že pro značení sekvencí je nejvhodnější neuronová síť typu LSTM. Jak ale vytvořit správnou architekturu sítě pro konkrétní úlohu? Jakou zvolit ztrátovou funkci a které funkce použít jako přenosové [19]?

Velikost vstupní vrstvy odpovídá vždy počtu hodnot popisující každý prvek vstupní sekvence. Počet neuronů výstupní vrstvy a jejich aktivační funkce závisí na počtu možných hodnot návěští. Nechť a označuje součet všech n vážených vstupů výstupního neuronu h, tedy $a = \sum_{i=1}^{n} w_{i,h}x_i$. Je-li úloha typu binární klasifikace, standardní konfigurace výstupní vrstvy je jediný neuron s logistickou sigmoidní aktivační funkcí. Výstup neuronu tedy vyjadřuje rovnice $y = \frac{1}{1+e^{-a}}$. Protože obor hodnot logistické sigmoidní funkce je interval (0,1), může být výstup neuronu interpretován jako pravděpodobnost, že výsledné návěští nabývá jedné hodnoty. Naopak jedna mínus výstup neuronu udává pravděpodobnost, že návěští je rovno druhé hodnotě. Pro klasifikační úlohy, kdy máme více než dvě třídy, se používá výstupní vrstva obsahující právě jeden neuron pro každou třídu. Jako aktivační funkce je vhodná softmax. Výstup každého neuronu h je proto dán vztahem $y_h = \frac{e^{a_h}}{\sum_{k=1}^{K} e^{a_h}}$, kde K je počet tříd (tedy počet neuronů). Výstup každého neuronu je opět hodnota z intervalu (0,1) a odpovídá pravděpodobnosti, že výsledné návěští patří do třídy, kterou neuron reprezentuje. Stačí tedy vybrat neuron s největší výstupní hodnotou. Co se týče skrytých vrstev neuronové sítě, přenosovou funkcí běžně bývá hyperbolický tangens. Avšak jak určit počet skrytých vrstev sítě a jejich velikost je otázka, na kterou není jednoznačná odpověď. Obvykle se tyto parametry odhadují na základě již existujících řešení podobných úloh nebo určují experimentálně.

Jako ztrátová funkce během učení sítě se pro klasifikační úlohy používá cross entropy. Cross entropy je způsob porovnání vektoru pravděpodobností y, který máme na výstupu naší sítě, s binárním vektorem z, jenž reprezentuje správný výsledek klasifikace tzv. kódem 1 z n (angl. one-hot encoding). Chyba L na výstupu je pomocí cross entropy dána rovnicí $L = -\sum_{k=1}^{K} z_k \log y_k$. Konkrétně pro binární klasifikaci vypadá výpočet chyby takto: $L = (z-1)\log(y-1) - z\log y$.

Trénování sítě lze ovlivnit dvěma parametry: počtem epoch a velikostí tzv. dávky. Počet epoch určuje, kolikrát je celá trénovací sada použita pro trénování. Jedné dvojici vstupních a jim odpovídajících výstupních sekvencí trénovacích dat se říká vzorek. Použití každého vzorku trénovacích dat pro dopředný a zpětný průchod sítí je tedy jedna epocha. Počet epoch přirozeně určuje dobu trénování sítě. Čím déle je síť trénována, tím přesnější výstupy by měla produkovat. Příliš velký počet epoch však může vést k tzv. přeučení sítě. To znamená, že síť bude klasifikovat vzorky z trénovací sady velice dobře, ale pro jiná vstupní data bude výstup špatný. Druhý parametr, velikost dávky, zase ovlivňuje, kolik vzorků dat je použito pro dopředné a zpětné šíření sítí před tím, než jsou váhy propojení sítě aktualizovány. Vlivy, které mají jednotlivé vzorky dávky na váhy propojení, jsou tedy sčítány a model sítě je aktualizován až na konci každé dávky. Velikost dávky může ovlivnit jak rychle se síť učí. Obecně bychom mohli říct, že menší dávky urychlují učení, protože jsou váhy upravovány častěji. Avšak čím menší velikost dávky je, tím jsou také úpravy vah méně přesné. Hodnoty počtu epoch a velikosti jedné dávky tedy mohou být předmětem experimentů s cílem naučit síť co nejlépe.

Kapitola 5

Realizace detekce událostí

Náplní této kapitoly je popis implementace a použití vytvořené knihovny pro detekci událostí. V kapitole 3 jsou uvedeny dva hlavní přístupy k detekci událostí. Pro realizaci byl vybrán přístup založen na použití metod strojového učení (viz sekce 3.2). K této volbě vedou dva důvody. První důvod plyne z faktu, že úspěch přístupu založeného na pravidlech je silně spjat s kvalitou a přesností vstupních prostoročasových dat. Efektivní získávání dostatečně přesných prostoročasových dat je v dnešní době totiž ještě pořád výzvou. Proto není jednoduché se k takovým datům dostat. Hlavním problémem je získávání přesných trojrozměrných souřadnic míče. Druhým důvodem je, že využití metod strojového učení umožňuje použít i jiná vstupní data než ta prostoročasová, popsaná v podsekci 2.2.2. Teoreticky stačí jakákoli data reprezentující stav na hřišti v jednotlivých okamžicích během celého utkání.

Knihovna je zaměřená na detekci událostí spojených s přerušením hry (viz tabulka fotbalových událostí 2.2). V podsekci 3.2.2 jsou řešeny možné formulace detekce událostí jako problém značení sekvencí. Knihovna podporuje realizaci všech tří navržených variant: detekci událostí formulovanou jako značení sekvencí N:1, značení sekvencí N:N i kombinovaný přístup. Pro realizaci značení sekvencí jsou použity neuronové sítě LSTM. Výsledná knihovna je nástroj umožňující experimentování s různými variantami, jak přesně detekci událostí jako problém značení sekvencí formulovat. Umožňuje nám definovat podobu sekvencí vytvořených z prostoročasových dat utkání. Můžeme zvolit různé architektury sítí pro značení sekvencí a ovlivnit způsob jejich učení. Dále lze vybrat návěští, kterými chceme sekvence označit a tudíž vybrat detekované události. A nakonec můžeme ovlivnit i způsob interpretace výsledku značení sekvencí.

Knihovna je implementovaná v jazyce Python a skládá se ze dvou modulů nazvaných *Data* a *ANN*. Modul *Data* implementuje načtení vstupních dat, přípravu vstupních a výstupních sekvencí a také interpretaci výsledku značení sekvencí. Druhý modul, *ANN*, poskytuje vytvoření neuronových sítí, jejich učení a použití pro značení sekvencí. Pro implementaci neuronových sítí je použita knihovna Keras [3] využívající TensorFlow knihovnu [6] s podporou výpočtů na grafické kartě.

V první sekci této kapitoly se budeme zabývat vstupními prostoročasovými daty použitými pro tuto práci a jejich přípravou. Druhá sekce se zabývá implementací a popisem použití modulu *Data* pro vytváření vstupních a výstupních sekvencí. Použití a implementaci druhého modulu *ANN* s neuronovými sítěmi pro značení sekvencí se věnuje třetí sekce. V poslední části se vrátíme k modulu *Data* a popíšeme si jim poskytovanou funkci pro interpretaci výstupu značení sekvencí.

Název videozáznamu	Délka videozáznamu
DenBosch_2015-09-26-13-19-27_3	12:32
Rapid_2016-07-16-18-02-39	47:03
Salzburg_2016-02-20-19-18-00	48:22
Terneuzen_2017-11-11-12-55-00	10:00
Trnava_2015-11-13-20-46-17_3	45:53
Celkem:	**2:43:50**

Tabulka 5.1: Názvy videozáznamů a jejich délky

5.1 Vstupní data a jejich příprava

Společnost CamVision pro tuto práci poskytla prostoročasová data celkem pěti panoramatických videozáznamů z různých utkání. Dohromady videa obsahují téměř dvě a tři čtvrtě hodiny hry. Názvy videozáznamů a jejich délky můžeme vidět v tabulce 5.1.

Poskytnutá časoprostorová data jsou poměrně odlišná od těch „ideálních“ popsaných v podsekci 2.2.2. V první řadě nemáme k dispozici žádné informace o míči a v druhé řadě, spíše než pozice hráčů, máme informace o objektech na hřišti. Takovými objekty jsou sice většinou jednotliví hráči, avšak ne vždy. Časoprostorová data jsou totiž získávána zpracováním pouze panoramatického videozáznamu. Hráči nejsou rozlišování od sebe navzájem ani od rozhodčího nebo případně jiných lidí, kteří by se mohli během utkání ocitnout na hřišti (např. zdravotníků, fanoušků nebo členů ochranky). Navíc, pokud se osoby na hřišti nachází těsně u sebe nebo se v záběru překrývají, mohou být detekovány jako jeden objekt. V důsledku toho může být počet detekovaných objektů v každém okamžiku utkání jiný. Frekvence získávání údajů je 25, což je také hodnota frekvence snímků videozáznamů. Data jsou tedy získávána z každého snímku. To znamená, že každé 4 setiny sekundy máme aktuální informace o objektech na hřišti.

Prostoročasová data ke každému videozáznamu jsou uložena ve formě CSV souboru. Každý řádek takového souboru obsahuje 8 hodnot a popisuje jeden objekt v daném čase (snímku). Položkami řádků jsou čas, ID, X souřadnice pozice, Y souřadnice pozice, X souřadnice vektoru pohybu, Y souřadnice vektoru pohybu, rychlost a výška. První položka, čas, udává pořadové číslo snímku, ve kterém byl objekt detekován (číslovaní začíná nulou). Hodnota ID je celé číslo v rozsahu 0 až $N - 1$, kde N je počet detekovaných objektů v daném čase t. Tato hodnota identifikuje objekt pouze v rámci času t. V dalším čase $t + 1$ už mohou být stejné objekty reprezentovány jinými hodnotami. Nelze tedy sledovat jejich trajektorie. Ostatní hodnoty, souřadnice pozice, souřadnice pohybového vektoru, rychlost a výška objektu, jsou reálná čísla. Začátek CSV souboru s časoprostorovými daty může vypadat takto:

```
Time;ObjectID;ModelX;ModelY;ModelVecX;ModelVecY;ModelSpeed;ModelHeight
0;0;19.31;31.89;0.000;0.000;0.000;2.10
0;1;17.91;31.63;0.000;0.000;0.000;2.02
...
0;20;47.00;0.14;0.000;0.000;0.000;2.94
0;21;10.84;-0.83;0.000;0.000;0.000;2.80
1;0;19.32;32.02;0.000;0.000;0.000;1.82
1;1;-10.38;31.89;0.000;0.000;0.000;1.89
...
```

5.1.1 Manuální vytváření anotací

Protože neuronové sítě patří do kategorie učení s učitelem, nestačí nám pouze prostoročasová data, ze kterých vytvoříme vstupní sekvence. Potřebujeme připravit také odpovídající výstupní sekvence. Proto bylo nejprve potřeba zhlédnout všechna videa a manuálně poznačit časy všech událostí. Pro vytvoření takových anotací ke každému panoramatickému videozáznamu byla použita beta verze aplikace Telestrátor, což je aplikace pro taktickou analýzu od společnosti CamVision, popsaná v sekci 2.1. Beta verze Telestrátoru je zobrazena na obrázku 5.1. Tato aplikace mimo jiné umožňuje přehrávat video, označit jeho libovolné úseky jako události zvoleného názvu a vybraného týmu a nakonec vygenerovat XML soubor s takto vytvořenými anotacemi. Příklad výstupního XML souboru vypadá následovně:

```
<?xml version="1.0" encoding="UTF-8"?>
<Telestrator TeamA="Team A" TeamB="Team B" Start="2016-07-16-18-02-43-337"
Lenght="47:03">
    <Event Type="ManualAnnotation" BeginTime="101240" EndTime="101240"
        Text="Begin" />
    <Event Type="ManualAnnotation" BeginTime="116200" EndTime="144789"
        Text="Foul" Team="A" />
    <Event Type="ManualAnnotation" BeginTime="224320" EndTime="238584"
        Text="Goal kick" Team="A" />
    ...
    <Event Type="ManualAnnotation" BeginTime="2736720" EndTime="2749563"
        Text="Throw-in" Team="A" />
    <Event Type="ManualAnnotation" BeginTime="2774880" EndTime="2804562"
        Text="Foul" Team="B" />
    <Event Type="ManualAnnotation" BeginTime="2809320" EndTime="2809320"
        Text="End" />
</Telestrator>
```

Jak můžeme vidět, pro každou událost je vytvořen element *Event* s několika atributy obsahujícími podrobnosti o dané události: *BeginTime* (počáteční čas), *EndTime* (koncový čas), *Text* (název kategorie události) a *Team* (tým, ke kterému se událost vztahuje). Časy *BeginTime* a *EndTime* mohou být vyjádřeny v milisekundách nebo případně ve formátu MM:SS. Hodnota atributu *Team* nabývá buď hodnoty A označující tým hrající zleva doprava (z pohledu kamery) nebo hodnoty B pro opačný tým.

Vytvořené anotace obsahují události celkem deseti kategorií: faul, gól, ofsajd, rohový kop, kop od branky, vhazování, střela, zásah brankáře, začátek a konec. Tabulka 5.2 obsahuje pro každou z těchto kategorií definici začátku a konce události. Nejpodstatnější jsou pro tuto práci události prvních šesti jmenovaných kategorií. Každá totiž označuje vždy konkrétní úsek přerušené hry. Všimněme si, že události kategorie faul a ofsajd obsahují i následný volný kop opačného týmu. V tabulce 5.3 můžeme vidět, kolik těchto událostí je obsaženo v jednotlivých videozáznamech. Kategorie střela a zásah brankáře jsou příklady událostí typu 1 (akcí). Jejich počty v jednotlivých videozáznamech obsahuje tabulka 5.4. Tyto události mají krátkodobý charakter. Nejsou proto považovány za úsek utkání o nějaké délce, ale za akci, jenž nastala v konkrétním čase. Atributy *BeginTime* a *EndTime* událostí těchto kategorií proto nesou stejnou hodnotu. Stejně tomu je u událostí kategorií začátek a konec, což jsou speciální události, které jsou použity, pokud videozáznam začíná a nebo končí přerušenou hrou. Událost kategorie začátek označuje čas konce přerušení hry, jehož

Obrázek 5.1: Beta Telestrator použitý pro tvorbu anotací

začátek ještě není ve videozáznamu obsažen. Podobně událost kategorie konec zaznamenává čas posledního přerušení hry, pokud videozáznam už dále neobsahuje probíhající hru. Všimněme si tedy, že každý videozáznam obsahuje maximálně jednu událost kategorie začátek a jednu událost kategorie konec. Události těchto dvou kategorií jsou neutrální, takže neobsahují atribut *Team*.

5.2 Vytváření sekvencí ze vstupních dat

Každý videozáznam odpovídá jedné datové sekvenci a každý snímek videozáznamu pak odpovídá jednomu prvku datové sekvence. Prvek datové sekvence je reprezentován vektorem, který popisuje stav na hřišti zachycený odpovídajícím snímkem. Hodnoty tohoto vektoru jsou vypočteny právě z prostoročasových dat popsaných v předchozí sekci. Z datových sekvencí jsou pak formovány vstupní sekvence. Začněme však popisem tvorby datových sekvencí.

Neuronové sítě stejně jako většina ostatních metod pro značení sekvencí vyžadují, aby vektory všech prvků vstupních sekvencí měly stejný počet dimenzí. Tuto podmínku tedy musí splňovat i naše datové sekvence. Pokud by měli být vektory prvků tvořeny přímo hodnotami popisujícími objekty detekované v příslušném snímku, počet dimenzí výsledného vektoru by pro všechny prvky sekvence stejný nebyl kvůli různému množství detekovaných objektů v jednotlivých snímcích. Bylo by potřeba některé vektory zkrátit nebo naopak doplnit tak, abychom docílili jejich stejné délky pro všechny prvky. I z tohoto důvodu bylo použito předzpracování prostoročasových dat již diskutované v podsekci 3.2.3, které převádí časoprostorová data každého snímku na pevný počet hodnot. Toto předzpracování dat tak problém se zarovnáváním vektorů odstraňuje. Ve výsledném řešení je každý prvek datové sekvence reprezentován devatenácti různými hodnotami vypočítanými z dat objektů příslušného snímku. Použité hodnoty byly již uvedeny v sekci 3.2.3. Zde si následujícím seznamem popíšeme jejich výpočet.

1. počet detekovaných objektů
2. aritmetický průměr X souřadnic pozice všech objektů

	Začátek události	**Konec události**
faul	kontakt hráče s míčem nebo s jiným hráčem způsobující faul	rozehrávka volného či pokutového kopu
gól	překročení míče brankovou čáru	rozehrávka ze středu hřiště
ofssajd	začátek přihrávky způsobující ofsajd	rozehrávka volného kopu
rohový kop	překročení míče brankovou čáru	rozehrávka od rohového praporku
kop od branky	překročení míče brankovou čáru	rozehrávka z brankového území
vhazování	překročení míče postranní čáru	vhození míče do hřiště
střela	dotek střílejícího hráče s míčem	
zásah brankáře	dotek brankáře s míčem	
začátek	první uvedení míče do hry	
konec	poslední přerušení hry	

Tabulka 5.2: Specifikace momentů začátku a konce události

	faul	**gól**	**ofsajd**	**rohový kop**	**kop od branky**	**vhazování**
DenBosch_ 2015-09-26-13-19-27_3	2	0	2	0	2	4
Rapid_ 2016-07-16-18-02-39	12	1	1	7	6	18
Salzburg_ 2016-02-20-19-18-00	16	0	1	2	4	33
Terneuzen_ 2017-11-11-12-55-00	0	1	0	1	2	1
Trnava_ 2015-11-13-20-46-17_3	12	1	3	3	5	22
Celkem:	**42**	**3**	**7**	**13**	**19**	**78**

Tabulka 5.3: Počty událostí typu 2 ve videozáznamech

	střela	**zásah brankáře**
DenBosch_ 2015-09-26-13-19-27_3	3	2
Rapid_ 2016-07-16-18-02-39	6	2
Salzburg_ 2016-02-20-19-18-00	5	5
Terneuzen_ 2017-11-11-12-55-00	3	1
Trnava_ 2015-11-13-20-46-17_3	11	2
Celkem:	**28**	**12**

Tabulka 5.4: Počty událostí typu 1 ve videozáznamech

3. aritmetický průměr Y souřadnic pozice všech objektů
4. největší X souřadnice pozice ze všech objektů
5. nejmenší X souřadnice pozice ze všech objektů
6. největší Y souřadnice pozice ze všech objektů
7. nejmenší Y souřadnice pozice ze všech objektů
8. absolutní hodnota rozdílu hodnot z bodů 4. a 5.
9. absolutní hodnota rozdílu hodnot z bodů 6. a 7.
10. X souřadnice pozice objektu s největší Y souřadnicí pozice
11. X souřadnice pozice objektu s nejmenší Y souřadnicí pozice
12. Y souřadnice pozice objektu s největší X souřadnicí pozice
13. Y souřadnice pozice objektu s nejmenší X souřadnicí pozice
14. aritmetický průměr absolutních hodnot X souřadnic pozice všech objektů
15. aritmetický průměr absolutních hodnot Y souřadnic pozice všech objektů
16. aritmetický průměr vzdáleností mezi pozicemi všech objektů navzájem
17. aritmetický průměr X souřadnic pohybového vektoru všech objektů
18. aritmetický průměr Y souřadnic pohybového vektoru všech objektů
19. aritmetický průměr rychlostí všech objektů

O načítání vstupních dat a vytváření datových, vstupních i výstupních sekvencí se stará modul *Data*. Nejprve je třeba vytvořit objekt třídy *Database*. Konstruktor této třídy vyžaduje cestu k adresáři obsahující databázi vstupních dat s konkrétní strukturou. Pro každý videozáznam musí databáze obsahovat podadresář pojmenovaný názvem videa. V něm je

očekávaný jednak soubor *ObjectTracking.csv* s prostoročasovými daty tohoto videozáznamu a také příslušný soubor *Events.xml* s anotacemi. Vytvořením instance třídy *Database* je z každého souboru s časoprostorovými daty v databázi vyrobena jedna datová sekvence. Výpočet datové sekvence odpovídající celému poločasu utkání však může trvat klidně i několik minut. Proto je každá vytvořená datová sekvence následně uložena ve formě NPY souboru, a to do stejného adresáře databáze, kde je umístěn odpovídající CSV soubor s časoprostorovými daty. Soubor datové sekvence je také stejně pojmenován. Nese tedy název *ObjectTracking.npy*. Další použití databáze je pak mnohem rychlejší, protože datové sekvence jsou pouze načteny ze souborů.

5.2.1 Způsob vytváření vstupních sekvencí

Z každé datové sekvence můžeme vytvořit sadu vstupních sekvencí dle libosti. Výslednou podobu vstupních sekvencí ovlivňují tři parametry: *seqLenght*, *overlapping* a *scalingMethod*.

První z nich udává délku sekvencí. Je možné zvolit jakoukoli délku vstupních sekvencí, avšak je třeba mít na paměti, že i LSTM sítě mají v tomto ohledu limity (jak již bylo zmíněno v 4.3.2). Rozumná délka každé sekvence je tedy v intervalu $\langle 2, 400 \rangle$.

Druhým parametrem, *overlapping*, lze určit, jestli a o kolik prvků se mají vstupní sekvence překrývat nebo od sebe vzdalovat. Konkrétně tato hodnota překrytí p udává rozdíl mezi začátky dvou sousedních sekvencí. Označíme-li prvky datové sekvence a z nich vytvořené vstupní sekvence indexy i a j začínajícími od nuly, pak každá vstupní sekvence j začíná prvkem datové sekvence $i = pj$. Pokud je hodnota překrytí p menší než délka vstupních sekvencí d, tak se každá z nich překrývá na obou svých koncích se sousedními sekvencemi o $d - p$ prvků. Platí-li však $p > d$, sousední sekvence jsou od sebe naopak odděleny $p - d$ prvky. Při rovnosti hodnot p a d jsou tedy vytvořeny navazující vstupní sekvence. Použitá varianta knihovny Keras vyžaduje, aby byly všechny vstupní sekvence stejně dlouhé. Pokud tedy délka datové sekvence není dělitelná hodnotou překrytí p, je poslední, zkrácená vstupní sekvence zahozena. Alternativní řešení by mohlo být tuto vstupní sekvenci obsahující poslední prvky datové sekvence doplnit do požadované délky (třeba nulovými vektory).

Posledním parametrem, kterým můžeme ovlivnit podobu vstupních sekvencí, je metoda pro transformaci dat *scalingMethod*. Tímto parametrem rozhodujeme, zda chceme data vstupních sekvencí normalizovat, standardizovat či nechat v jejich původním rozsahu. Normalizací je myšlena transformace dat do intervalu $\langle 0, 1 \rangle$. Standardizace zase znamená transformaci distribuce hodnot tak, že střední hodnota je rovna 0 a směrodatná odchylka je 1. Při použití neuronových sítí se transformace vstupních hodnot pomocí jedné z těchto metod doporučuje, protože má pozitivní vliv na rychlost učení sítě a snižuje pravděpodobnost uváznutí v lokálním minimu [38].

5.2.2 Způsob vytváření výstupních sekvencí

Ke každé vstupní sekvenci je potřeba vytvořit také odpovídající výstupní sekvenci návěští, jejichž podobu ovlivňují dva parametry: *eventsToDetect* a *labelingMode*.

První z nich určuje, jaké události chceme rozpoznávat. Tento parametr může nabývat jednak celočíselné hodnoty z intervalu $\langle 1, 3 \rangle$ nebo to může být seznam textových řetězců. V prvním případě celočíselná hodnota vyjadřuje typ událostí (dle tabulky 2.2). Zatím však knihovna podporuje rozpoznávání pouze událostí spojených s přerušením hry (typ 2). Tato varianta představuje prakticky detekci přerušené hry. Návěští proto nabývají pouze dvou hodnot: 0 pro probíhající hru a 1 pro přerušenou. Jako přerušená hra jsou označeny i úseky

hry před událostí začátek a za událostí konec. V druhém případě seznam textových řetězců obsahuje přímo názvy kategorií událostí. Pokud seznam obsahuje pouze jednu položku, tak výstupní sekvence odpovídají opět binární klasifikaci (návěští 1 pro danou událost a 0 pro „žádnou událost“). Je-li však položek v seznamu $N > 1$, jsou všem kategoriím přiřazena celočíselná návěští (1,2,...,N) odpovídající jejich pořadí v seznamu. Návěští hodnoty 0 značí opět „žádnou událost“. Tato celočíselná návěští jsou pak zakódována kódem 1 z n. To znamená, že každé návěští je ve výsledku vektor o $N+1$ složkách, přičemž jen jedna složka vektoru obsahuje hodnotu 1 a ostatní jsou 0. Například návěští 0 při $N = 3$ odpovídá vektoru $[1, 0, 0, 0]$.

Druhým parametrem, *labelingMode*, volíme druh značení sekvencí. Na výběr je buď značení sekvencí N:1 (klasifikace sekvencí) nebo N:N (klasifikace segmentů, kdy segment odpovídá jednomu prvku sekvence). K datové sekvenci jsou z příslušného souboru *Events.xml* načteny události a jejich časy jsou přepočítány na odpovídající elementy datové sekvence. Podle toho a již popsaného parametru *eventsToDetect* je vytvořena nejprve sekvence návěští k celé datové sekvenci (N:N). Poté jsou z této sekvence návěští pomocí parametrů *seqLenght* a *overlapping* vytvořeny výstupní sekvence pro značení sekvencí N:N úplně stejným způsobem, jako jsou tvořeny vstupní sekvence. Je-li zvolen druh značení sekvencí N:1, jsou nakonec všechny výstupní sekvence nahrazeny jediným návěštím, které je vypočteno jako modus odpovídající výstupní sekvence pro značení sekvencí N:N.

5.2.3 Rozhraní modulu Data pro získávání vstupních a výstupních sekvencí

Pro získání vstupních a výstupních sekvencí z celé databáze, ze všech datových sekvencí najednou, poskytuje třída *Database* metodu *getIOData*. Cílem použití této metody je získání dat, které poskytneme neuronové síti k trénování a testování při jejím vytvoření. Metodě *getIOData* musíme předat hodnoty všech pěti diskutovaných parametrů: *seqLenght*, *overlapping*, *scalingMethod*, *labelingMode* a *eventsToDetect*. Tabulka 5.5 obsahuje přehled možných hodnot, kterých mohou jednotlivé parametry nabývat. Návratová hodnota metody *getIOData* je dvojice seznamů stejné délky. První z nich obsahuje vstupní sekvence pro každou datovou sekvenci (nebo-li videozáznam) databáze a druhý seznam obsahuje odpovídající výstupní sekvence. Položkami seznamu se vstupními sekvencemi jsou tří-dimenzionální pole o rozměrech (počet sekvencí; délka sekvence; velikost vektoru prvku). Počet sekvencí každého pole závisí na délce datové sekvence a hodnotách parametrů *seqLenght* a *overlapping*. Velikost druhé dimenze je dána přímo hodnotou *seqLenght* a velikost vektoru každého prvku je 19 (počet hodnot popisující každý prvek sekvence). Počet dimenzí a rozměry pole s odpovídajícími výstupními sekvencemi se liší v závislosti na typu značení sekvencí. Pro N:N má pole tvar (počet sekvencí; délka sekvence; velikost vektoru návěští) a pro N:1 (počet sekvencí; velikost vektoru návěští). Velikost vektoru návěští závisí na počtu možných hodnot výstupních návěští a tedy na hodnotě parametru *eventsToDetect*.

V případě, že chceme vytvořit vstupní a výstupní sekvence pouze z jednoho konkrétního videozáznamu, můžeme použít také funkci *getExtIOData* modulu *Data*. Tato funkce nám umožňuje zadat soubor s prostoročasovými daty a anotacemi externě pomocí jejich parametrů. Při jejím použití máme možnost vytvořit i pouze vstupní sekvence bez odpovídajících výstupních sekvencí. To se hodí při používání již natrénovaných neuronových sítí pro značení vstupních sekvencí. K videozáznamu, ve kterém chceme detekovat události, totiž typicky nemáme anotace a výstupní sekvence tedy pochopitelně nelze vytvořit. Účelem použití funkce *getExtIOData* je tedy nejčastěji získání vstupních sekvencí pro jejich

Název par.	Možné hodnoty par. a jejich význam
seqLenght	a) $1 <$ *seqLenght* - délka každé vstupní sekvence
overlapping	a) $0 <$ *overlapping* $<$ *seqLenght* - překrývání sekvencí o $\lvert seqLenght - overlapping \rvert$ prvků b) *seqLenght* $<$ *overlapping* - oddělení sekvencí $\lvert seqLenght - overlapping \rvert$ prvky c) None - vytvoření navazujících sekvencí (*overlapping* = *seqLenght*)
scalingMethod	a) "std"- standardizace vstupních hodnot b) "nrm"- normalizace vstupních hodnot c) None - původní rozsah vstupních hodnot
labelingMode	a) SEQ_CLASSIFICATION - značení sekvencí N:1 b) SEQ_LABELING - značení sekvencí N:N
eventsToDetect	a) 2 - rozpoznávání událostí typu 2 b) seznam textových řetězců - rozpoznávání událostí daných kategorií (kategore: "Foul", "Goal", "Offside", "Goal kick", "Corner kick", "Throw-in")

Tabulka 5.5: Přehled možných hodnot parametrů metody *getIOData*

označení již natrénovanou sítí. Jako její první parametr je očekávána cesta k CSV souboru s prostoročasovými daty. Pokud chceme vytvořit i odpovídající výstupní sekvence, druhým parametrem je třeba zadat cestu k XML souboru s anotacemi. V opačném případě musíme předat prázdnou hodnotu *None*. Zbytek parametrů je ekvivalentní s parametry metody *getIOData* popsanými tabulkou 5.5. Výstupem funkce je dvojice hodnot. Tou první je vždy pole se vstupními sekvencemi. Druhou hodnotou je pole s odpovídajícími výstupními sekvencemi nebo prázdná hodnota *None*, pokud cesta k souboru s anotacemi nebyla zadána.

Data pro klasifikaci úseků přerušené hry

Kromě metody *getIOData* poskytuje třída *Database* pro vytváření vstupních a výstupních sekvencí také metodu *getIODataForBreakClass*. Tato metoda má však více konkrétní účel. Slouží k získání pouze částí datových sekvencí obsahující události typu 2 a jim odpovídajících výstupních návěští pro klasifikaci sekvenci (N:1). Účelem této metody je tedy získání trénovacích dat pro učení neuronových sítí klasifikovat pouze úseky přerušené hry. Metoda vyžaduje jen parametry *seqLenght*, *scalingMethod* a *eventsToDetect*. Povolené hodnoty parametru *seqLenght* a *scalingMethod* se oproti metodě *getIOData* nijak nemění (viz tabulka 5.5). Parametr *eventsToDetect* však v tomto případě musí být vždy seznam s názvy kategorií událostí typu 2. Pro každou událost je vytvořen jeden pár vstupní a výstupní sekvence. Délka vstupních sekvencí daná parametrem *seqLenght* je potřeba kvůli různé délce jednotlivých událostí. Kratší sekvence jsou doplněny nulovými vektory a sekvence s délkou větší než hodnota *seqLenght* jsou zkráceny z prostředku. To znamená, že vstupní sekvence je vytvořena spojením *seqLenght/2* prvních a *seqLenght/2* posledních prvků úseku obsahující událost přerušené hry. Návratová hodnota metody *getIODataForBreakClass* má stejnou formu, která je popsaná výše u metody *getIOData* (seznam polí se vstupními sekvencemi a odpovídající seznam polí s výstupními návěštími).

Metoda *getIODataForBreakClass* má také svůj ekvivalent pro vytvoření vstupních a výstupních sekvencí pouze z jednoho videozáznamu – funkci *getExtIODataForBreakClass*

modulu *Data*. Její první parametr je cesta k CSV souboru s prostoročasovými daty, druhý parametr udává cestu k XML souboru s anotacemi a zbylé tři parametry odpovídají parametrům metody *getIODataForBreakClass*. Pokud však soubor s anotacemi nemáme, nemůžeme místo něj jednoduše předat prázdnou hodnotu *None*. Bez obdržení anotací funkce není schopna identifikovat části utkání obsahující události přerušené hry a tedy vytvořit požadované vstupní sekvence. Pokud tedy anotace k daným prostoročasovým datům nemáme, můžeme události zadat přímo ve formě seznamu objektů třídy *Event* modulu *Data*. Takový seznam lze buďto vytvořit manuálně nebo získat použitím funkce *outputSeqsToEvents* (viz sekce 5.4). Každý objekt třídy *Event* reprezentuje jednu událost. Funkce *getExtIODataForBreakClass* pak vytvoří vstupní sekvence z částí videozáznamu odpovídajícím událostem typu 2. Návratovou hodnotou je stejně jako u funkce *getExtIOData* jednak pole se vstupními sekvencemi a případně i pole s výstupními sekvencemi. Pokud je místo souboru s anotacemi funkci předán seznam událostí, tak výstupní sekvence vytvořeny nejsou a místo nich je vrácena prázdná hodnota *None*.

5.3 Implementace neuronových sítí pro značení sekvencí

Vytváření, učení a použití neuronových sítí pro značení sekvencí realizuje modul *ANN*, který obsahuje pět tříd: *SeqClassifier*, *SeqLabeler*, *StatefulSeqClassifier*, *StatefulSeqLabeler* a *PatternClassifier*. Každá z nich představuje jiný typ neuronové sítě. *SeqClassifier* je LSTM sít pro značení sekvencí N:1 a *SeqLabeler* je LSTM sít určena pro značení sekvencí N:N. Obě pak mají svojí „stateful“ variantu. *PatternClassifier* je obyčejná dopředná síť, pomocí níž můžeme klasifikovat pouze nesekvenční data. Rozdíly mezi jednotlivými typy sítí budou vysvětleny později. Nejprve se zaměříme na to, co mají všechny typy sítí společného. Všechny jsou podtřídou společné rodičovské třídy *Net* a díky tomu mají stejné rozhraní.

5.3.1 Rozhraní modulu ANN pro vytváření a používání neuronových sítí

Při vytváření instance sítě musíme do konstruktoru předat seznamy polí se vstupními a výstupními sekvencemi. Požadovaná podoba (počet dimenzí) polí je popsána v podsekci 5.2.3 (návratová hodnota metody *getIOData*). Tato data jsou pak rozdělena na trénovací a testovací sadu v poměru daném třídní proměnnou *Net.TEST_RATE*. Třetí parametr konstruktoru, *archType*, umožňuje volbu jedné z předpřipravených architektur sítě. Tabulka 5.6 obsahuje pro všech pět typů sítí možné hodnoty tohoto parametru a jim odpovídající popis architektury sítě. Jak můžeme vidět, tak na výběr je pouze u sítí pro značení sekvencí N:N. Kromě klasické a obousměrné LSTM sítě je implementována LSTM sít s CRF výstupní vrstvou [14] (implementováno s použitím knihovny keras-contrib [2]) a také encoder-decoder LSTM architektura [9]. Ještě před vytvořením sítě je možné její architekturu ovlivnit také změnou třídních proměnných *Net.LAYERS* a *Net.UNITS_NUMBER*, které určují počet a velikost skrytých vrstev.

Po vytvoření instance sítě můžeme spustit její učení na trénovacích datech metodou *fitModel*. Počet epoch a velikost dávky můžeme předem nastavit změnou hodnot proměnných *Net.EPOCHS* a *Net.BATCH_SIZE*. Váhy naučené sítě je možné uložit do souboru pomocí metody *saveModel* a později je z něj zase načíst metodou *loadModel*. Tyto soubory jsou ukládány do složky *models* v aktuálním pracovním adresáři uživatele. Metodě *saveModel* je třeba parametrem předat název, kterým má být soubor s uloženými váhami sítě pojmenován. Pro opětovné načtení sítě musíme stejný název předat metodě *loadModel*. Při načítání

uložených vah do nově vytvořené sítě si musíme akorát dát pozor aby její architektura odpovídala síti, z níž váhy původně pochází.

Další metoda *evaluateModel* provádí vyhodnocení výsledků klasifikace sekvencí z testovací sady dat. Jejím výstupem je čtveřice hodnot:

1. přesnost klasifikace vyjádřená v procentech
2. chybová matice [4]
3. metriky precision, recall, f1-score a support pro každou třídu klasifikace [5]
4. textová reprezentace metrik z předchozích bodů

Dále síť můžeme samozřejmě použít k tomu, k čemu byla vytvořena – značení sekvencí. To je realizováno metodou *predictData*. Jejím první parametrem jsou vstupní sekvence, ke kterým chceme predikovat odpovídající výstupní sekvence. Očekávaná forma vstupních sekvencí je jedno pole (ne seznam, jak tomu je u konstruktoru). Jedním voláním metody tak můžeme označit sekvence jen jednoho videozáznamu. Je třeba dbát na to, aby délka vstupních sekvencí odpovídala délce těch, které byly použity pro trénování sítě. Druhý parametr je volitelný a může obsahovat očekávané (správné) výstupní sekvence (opět ve formě jednoho pole), které pak mohou být pro srovnání zobrazeny společně s predikovanými výsledky ve výstupním souboru. Pokud tento výstupní soubor chceme vytvořit, musíme zadat i třetí parametr, a sice jeho název. Tyto soubory jsou ukládány do složky *predictions* v aktuálním pracovním adresáři uživatele a mají následující strukturu. Každé tři řádky odpovídají jedné výstupní sekvenci. První obsahuje pořadí výstupní sekvence. Druhý řádek je uvozen řetězcem „Exp:“ a obsahuje očekávanou výstupní sekvenci. V třetím řádku uvozeném řetězcem „Pre:“ je už výsledná sekvence návěští předpovězená sítí. Pokud referenční výstupní sekvence nejsou metodě předány, tak je druhý řádek jednoduše vynechán. Příklad začátku výstupního souboru může tedy vypadat třeba následovně.

```
1.
Exp: 00000000000000000000000000
Pre: 00000000000000000000000000
2.
Exp: 00000000000111111111111111
Pre: 00000000000000000011111111
3.
Exp: 11111111111111111111111111
Pre: 11111111111111111111111111
...
```

Návratová hodnota metody *predictData* je pole s výstupními sekvencemi predikovanými sítí. Pro značení sekvencí N:N má pole rozměry (počet sekvencí; délka sekvence). V případě značení sekvencí N:1 je výstupem jednodimenzionální pole o délce počtu sekvencí jelikož každá výstupní „sekvence“ obsahuje pouze jeden prvek.

5.3.2 Rozdíly mezi implementovanými typy neuronových sítí

Nyní, když máme za sebou společný popis použití všech pěti implementovaných typů sítí, můžeme přejít k tomu, v čem se liší. Jak již bylo zmíněno, třídy *SeqClassifier* a *StatefulSeqClassifier* realizují značení sekvencí typu N:1 a *SeqLabeler* a *StatefulSeqLabeler* jsou zase

Typ sítě	Hodnota archType	Popis architektury
SeqLabeler, StatefulSeqLabeler	0	klasická LSTM
	1	obousměrná LSTM
	2	obousměrná LSTM s CRF výstupní vrstvou
	3	encoder-decoder LSTM
SeqClassifier, StatefulSeqClassifier	0	klasická LSTM
PatternClassifier	0	vícevrstvý perceptron

Tabulka 5.6: Implementované architektury sítí

sítě pro značení sekvencí N:N. Je třeba dbát na to, abychom data předávané sítím vytvářeli pro správný typ značení (parametr *labelingMode*). Pokud bychom třeba při instanciaci objektu třídy *SeqClassifier* konstruktoru předali výstupní sekvence pro značení sekvencí typu N:N, skončilo by to chybou.

Obyčejné varianty LSTM sítí (*SeqClassifier* a *SeqLabeler*) považují všechny vstupní sekvence ze všech polí seznamu za na sobě nezávislé. Proto jsou spojeny do jednoho pole a před učením navíc promíchány. „Stateful“ LSTM sítě [11] však interpretují vstupní data trochu odlišně. Považují totiž jednotlivé vstupní sekvence každého pole za stejně dlouhé, navazující části jedné dlouhé datové sekvence. Je tedy vhodné data pro tyto sítě vytvářet tak, aby jednotlivé vstupní sekvence na sebe navazovaly (rovnost parametrů *seqLenght* a *overlapping*). Způsob, jakým je síť pak trénována odpovídá variantě algoritmu pro učení rekurentních sítí TBPTT (viz 4.3.2) [10]. Počet časových kroků při dopředném i zpětném šíření je určen právě délkou vstupních sekvencí a vnitřní stav LSTM sítě je resetován až po zpracování celé datové sekvence (všech vstupních sekvencí jednoho pole). Aby to takto opravdu fungovalo, je při učení třeba použít dávky o velikosti 1. Knihovna Keras totiž implementuje učení sítí tak, že vlivy jednotlivých vzorků konkrétní dávky jsou počítány paralelně. Každá vstupní sekvence jedné dávky tudíž vytváří vlastní stav LSTM sítě. Tyto stavy jsou pak použity jako počáteční stavy sítí pro zpracování sekvencí další dávky. To prakticky znamená že, vstupní sekvence i jedné dané dávky je považována za pokračování sekvence i předchozí dávky, přičemž i je pořadí sekvence v dávce. Použití velikosti dávky větší než 1 pro „stateful“ sítě možné je, avšak v tom případě je každé pole vstupních sekvencí považováno za d stejně velkých, nezávislých datových sekvencí, kde d je velikost dávky. Každé pole vstupních i výstupních sekvencí v tom případě musí být nejprve zarovnáno, aby obsahovalo počet sekvencí dělitelný velikostí dávky, a poté jsou vstupní sekvence v poli přeuspořádány tak, aby byly při trénování sítě rozděleny do správných dávek ve správném pořadí. Pro ujasnění si uveďme příklad. Mějme pole se vstupními sekvencemi [a,b,c,d,e,f,g,h,i,j,k] a velikost dávky 3. Potom po zarovnání a přeuspořádání bude pole vypadat takto: [a,d,g,b,e,h,c,f,i]. Díky tomu budou vstupní sekvence rozděleny při učení do dávek následovně: [a,d,g], [b,e,h], [c,f,i].

Posledním z implementovaných typů sítí je *PatternClassifier*. Jak již bylo zmíněno a jak také napovídá její název, tato třída realizuje pouze obyčejnou dopřednou síť typu vícevrstvého perceptronu. Tato síť řeší dvě různé úlohy podle toho, jaké data ji poskytneme. Pokud konstruktoru této sítě předáme data pro značení sekvencí N:N, tak sít ignoruje rozdělení vstupních dat do sekvencí a jednoduše klasifikuje každý prvek každé vstupní sekvence

samostatně. V případě, že sít dostane data pro značení sekvencí N:1, jsou vektory prvků každé vstupní sekvence spojeny do jednoho dlouhého vektoru. Takto reprezentované vstupní sekvence pak mohou být klasifikovány vícevrstvým perceptronem.

5.4 Interpretace výsledku značení sekvencí

Nakonec, po natrénování neuronových sítí a jejich použití k označení vytvořených vstupních sekvencí, je třeba vypočíst opravdový výstup detekce událostí. To znamená interpretovat výstupní sekvence jako seznam událostí, u nichž známe kategorii, počáteční čas a čas konce události. Detekování událostí zvlášť pro oba týmy knihovna zatím nepodporuje, takže identifikátor týmu součástí výstupu není.

Modul *Data* poskytuje pro interpretaci výstupních sekvencí funkci *outputSeqsToEvents*. Prvním vstupním parametrem jsou této funkci předány výstupní sekvence návěští, které chceme interpretovat. Tyto sekvence musejí mít stejnou podobu jako návratová hodnota metody *predictData* pro použití sítí k označení vstupních sekvencí (viz. 5.3.1). To znamená 2-D pole pro znační sekvencí N:N a 1-D pro N:1. Funkce *outputSeqsToEvents* nejdříve z obdržených výstupních sekvencí rekonstruuje sekvenci návěští odpovídající celému videozáznamu (označme ji pojmem super-sekvence). Super-sekvence tedy obsahuje jedno návěští pro každý snímek videozáznamu. Abychom ji dokázali správně rekonstruovat, musíme funkci předat hodnoty parametrů *seqLenght*, *overlapping* a *labelingMode*, které byly použity pro vytvoření vstupních sekvencí, jejichž označením interpretované sekvence návěští vznikly. Z podobného důvodu potřebujeme předat i hodnotu parametru *eventsToDetect*, abychom věděli, kterým událostem odpovídají jednotlivé hodnoty návěští. Je třeba předat stejnou hodnotu, která byla použita při vytváření výstupních sekvencí pro trénování použité sítě.

V super-sekvenci jsou poté detekovány části obsahující pouze nenulová návěští, protože návěští 0 vždy značí „žádnou událost“. Výsledek detekce událostí můžeme na této úrovni ovlivnit pomocí hodnot šestého, sedmého a osmého parametru: *preciseDetection*, *minEventLen* a *minEventDist*. První zmíněný může nabývat pouze pravdivostní hodnoty (*True* nebo *False*). Pokud je hodnotou *True*, tak za událost je považována pouze podsekvence nenulových návěští stejné hodnoty (např. "0111111110"). Je-li hodnotou parametru *preciseDetection False*, je detekce událostí méně striktní. Za událost je v tomto případě považována každá podsekvence nenulových návěští a kategorie události je určena podle nejčastější hodnoty (např. podsekvence "0223333440"je vyhodnocena jako událost kategorie, která přísluší návěští 3). Hodnota parametru *minEventLen* je číselná hodnota, která určuje minimální délku detekované události v sekundách. Můžeme tak s její pomocí odfiltrovat chybně označené podsekvence nenulových návěští, které jsou zjevně příliš krátké na to, aby představovali událost spojenou s přerušením hry. Poslední parametr *minEventDist* je také počet sekund, který tentokrát definuje minimální mezeru mezi dvěma různými událostmi stejné kategorie.

Výstupní hodnotou funkce *outputSeqsToEvents* je seznam objektů třídy *Event* modulu *Data*. Každý objekt reprezentuje jednu detekovanou událost. Třída *Event* obsahuje atributy: *beginTimestep* (číslo snímku videa se začátkem události), *endTimestep* (číslo snímku videa s koncem události), *category* (název kategorie události), *type* (typ události) a také metodu *printEvent*, která realizuje vytisknutí informací o události na standardní výstup. Součástí těchto informací jsou také vypočtené časy začátku a konce události i její délka. Výpis jedné události tedy vypadá například takto:

```
Type: 2
Category: Goal kick
BeginTimestep: 68325
EndTimestep: 68825
BeginTime: 2733000 ms (45:33)
EndTime: 2753000 ms (45:53)
Lenght: 20.0 s
--------------------
```

Pro interpretaci výstupních návěští, které vznikly klasifikací úseků přerušené hry je k dispozici funkce *labelBreakEvents*. Tyto výstupní návěští musíme funkci předat jejím prvním parametrem. Jako druhý parametr je třeba zadat stejnou hodnotu, která byla předána jako druhý parametr funkci *getExtIODataForBreakClass* použité pro vytvoření vstupních sekvencí jejichž klasifikací interpretovaná návěští vznikla. To znamená buď seznam událostí (objektů třídy Event) nebo cestu k souboru s anotacemi. Poslední, třetí parametr musí obsahovat hodnotu parametru *eventsToDetect*, jenž byla použita při vytváření vstupních sekvencí pro trénování použité sítě. Díky tomu funkce pozná, jaké hodnoty návěští přísluší kterým kategoriím. Funkce *labelBreakEvents* pak prakticky provádí pouze to, že všem událostem typu 2 přiřadí kategorii podle daných návěští. Výstupní hodnotou jsou tedy výsledné události opět ve formě seznamu objektů třídy *Event*.

5.5 Použití vytvořené knihovny pro detekci událostí

Na začátku této kapitoly bylo řečeno, že vytvořená knihovna umožňuje realizovat detekci událostí formulovanou všemi třemi způsoby navrženými v podsekci 3.2.2. To znamená detekci událostí jako značení sekvencí N:1 (F_1), značení sekvencí N:N (F_2) a kombinovaný přístup (F_3). V této sekci si povíme jak tedy knihovnu použít.

V případě prvních dvou zmíněných formulací, F_1 a F_2, je proces použití knihovny pro detekci událostí teoreticky stejný. Můžeme jej rozdělit do několika kroků:

1. načtení databáze se vstupními prostoročasovými daty videozáznamů a jejich anotacemi – vytvoření instance třídy *Data.Database* (sekce 5.2)

2. vytvoření vstupních a výstupních sekvencí z databáze pro trénování sítě – metoda *getIOData* třídy *Data.Database* (sekce 5.2.3)

3. vytvoření neuronové sítě pro značení sekvencí – vytvoření instance třídy *ANN.Net* (sekce 5.3)

4. naučení neuronové sítě – metoda *fitModel* třídy *ANN.Net* (sekce 5.3.1)

5. vytvoření vstupních sekvencí z prostoročasových dat videozáznamu, ve kterém chceme detekovat události – funkce *Data.getExtIOData* (sekce 5.2.3)

6. predikce výstupních sekvencí odpovídajících vstupním sekvencím vytvořeným v předchozím kroku – metoda *predictData* třídy *ANN.Net* (sekce 5.3.1)

7. interpretace výstupních sekvencí predikovaných v předchozím kroku – funkce *Data.outputSeqsToEvents* (sekce 5.4)

V praxi spočívá rozdíl mezi formulacemi F_1 a F_2 pouze v realizaci kroku 2 a 3. Při použití metody *getIOData* pro vytvoření trénovacích dat musíme zvolit tvorbu výstupních sekvencí pro správný druh značení sekvencí (N:1 nebo N:N) pomocí parametru *labelingMode* (viz tabulka 5.5). Podobně také pro vytvoření neuronové sítě je třeba zvolit podtřídu třídy *ANN.Net* pro správný typ značení sekvencí. Sítě pro značení sekvencí N:1 jsou realizovány třídami *SeqClassifier* a *StatefulSeqClassifier* a značení sekvencí N:N je implementováno zase třídami *SeqLabeler* a *StatefulSeqLabeler* (viz 5.3.2).

Při používaní knihovny musíme také dbát na to, aby vstupní sekvence z kroku 2 a 5 byly vytvořeny stejným způsobem (s použitím stejných parametrů). Stejně tak parametry vytvoření trénovacích dat z kroku 2 musejí být použity také v kroku 7 při interpretaci výsledku. V opačném případě by se jednoduše řečeno mohlo stát, že neuronovou síť naučíme něco jiného než to, k čemu ji nakonec použijeme. V důsledku toho by detekce událostí mohla skončit chybou nebo by mohl být výsledek znehodnocen.

Kombinovaný přístup detekce událostí můžeme rozdělit na dvě části. V první části je značení sekvencí N:N použito pro detekci úseků přerušené hry. Druhá část potom těmto úsekům přiřadí konkrétní kategorie událostí pomocí značení sekvencí N:1 (viz obrázek 3.4). Realizace detekce události tímto způsobem pomocí vytvořené knihovny je znázorněna diagramem 5.2. Hodnota parametru *eventsToDetect* metody *getIOData* pro vytváření trénovacích dat musí být číslo 2. Tím zajistíme, že výstupem první části budou úseky přerušené hry. Tento výstup je reprezentován seznamem událostí typu 2 avšak neznámé kategorie. Aby v druhé části mohly být kategorie přiřazeny právě těmto úsekům přerušené hry, musíme výstup první části předat funkcím *getExtIODataForBreakClass* a *labelBreakEvents*.

5.5.1 Možnosti experimentování

Když uvážíme proces použití knihovny pro detekci událostí popsaný seznamem na začátku této sekce, tak v krocích 2, 3, 4 a 7 máme na výběr z řady možností, které mohou ovlivnit výsledek detekce událostí. To nám umožňuje provádět experimenty pouze modifikací parametrů ovlivňující jednotlivé kroky a porovnáváním výsledků. Následující seznam obsahuje přehled všech parametrů, které můžeme modifikovat a ovlivnit tak způsob a tudíž i výsledek detekce událostí.

- krok 2 – vytvoření vstupních a výstupních sekvencí pro učení sítě:
  - podoba vstupních sekvencí:
    - délka (parametr metody *seqLenght*)
    - překrývání/oddělení (parametr metody *overlapping*)
    - transformace vstupních hodnot (parametr metody *scalingMethod*)
  - podoba výstupních sekvencí
    - druh značení (parametr metody *labelingMode*)
    - kategorie detekovaných událostí (parametr metody *eventsToDetect*)
- krok 3 – vytvoření neuronové sítě:
  - typ neuronové sítě (volba podtřídy třídy *ANN.Net*)
  - architektura neuronové sítě:
    - typ architektury (*parametr konstruktoru archType*)
    - počet skrytých vrstev (třídní proměnná *Net.LAYERS*)

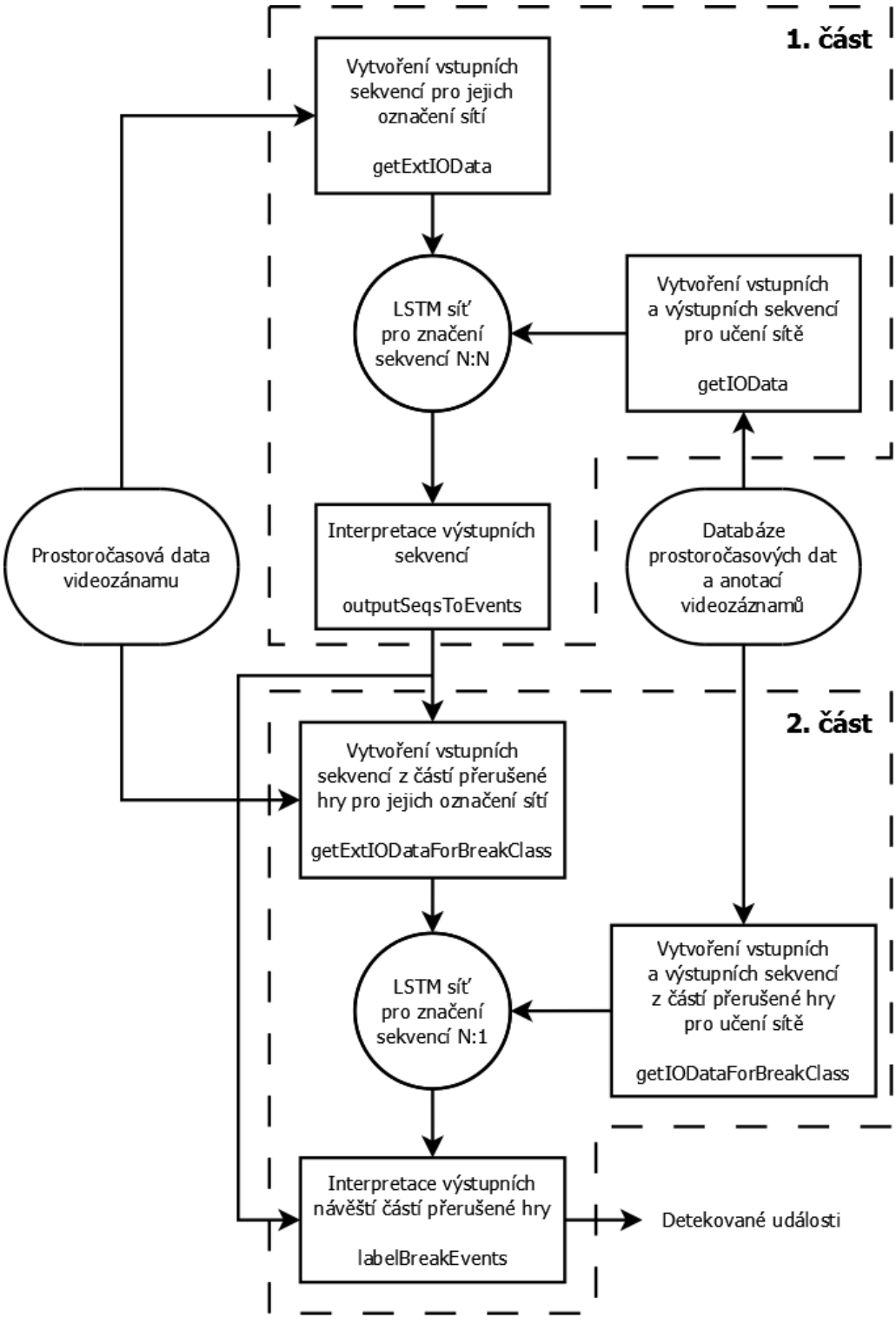
    - počet neuronů jedné skryté vrstvy (třídní proměnná *Net.UNITS_NUMBER*)

- krok 4 – naučení neuronové sítě:
  - počet epoch (třídní proměnná *Net.EPOCHS*)
  - velikost jedné dávky (třídní proměnná *Net.BATCH_SIZE*)

- krok 7 – interpretace výstupních sekvencí:
  - druh interpretace (parametr funkce *preciseDetection*)
  - minimální délka události (parametr funkce *minEventLen*)
  - minimální mezera mezi událostmi stejné kategorie (parametr funkce *minEventDist*)

Pokud experimentujeme pouze s modifikací parametrů kroku 3 a 4, můžeme porovnávat výsledky už na úrovni přesnosti klasifikace neuronové sítě. To znamená porovnáváním hodnot, které nám poskytuje metoda *evaluateModel* (viz 5.3.1). Čím přesněji bude síť značit sekvence, tím přesnější bude i celkový výsledek detekce událostí. Experimentujeme-li však s hodnotami parametrů kroku 2 a 7, je třeba vyhodnotit až finální výsledek detekce. Pro tento účel poskytuje modul *Data* funkci *evaluateEventDetection* očekávající tři parametry. Prvním je seznam detekovaných událostí, jako druhý parametr je očekávána cesta k souboru s anotacemi a třetím parametrem je seznam kategorií nebo typ událostí, které jsme detekovali. Při vyhodnocování výsledku je detekce událostí považována opět za klasifikační úlohu, kdy predikujeme ke každému snímku videozáznamu událost (nebo „ne-událost“), kterou snímek obsahuje. Ze souboru s anotacemi jsou načteny všechny události kategorií daných třetím parametrem. Vůči nim jsou pak detekované události porovnávány. Pro vyhodnocení detekce tak můžeme použít klasické metriky pro vyhodnocování výsledku klasifikace [24]. Návratovou hodnotou funkce *evaluateEventDetection* je tedy stejně jako u vyhodnocování sítí čtveřice hodnot:

1. přesnost detekce vyjádřená v procentech
2. chybová matice [4]
3. metriky precision, recall, f1-score a support pro každou událost [5]
4. textová reprezentace metrik z předchozích bodů



Obrázek 5.2: Proces použití vytvořené knihovny pro detekci událostí kombinovaným přístupem

Kapitola 6

Experimenty a možná rozšíření

Jak vyplývá z podsekce 5.5.1 o možnostech využití vytvořené knihovny pro experimentování, prostor pro experimenty v oblasti detekce událostí je poměrně velký. Zatím však nemáme k dispozici dostatečné množství prostoročasových dat na to, abychom mohli uskutečnit experimenty, na jejichž výsledky by se dalo spolehnout. Dvě a tři čtvrtě hodiny hry celkem pěti videozáznamů, které jsou pro tuto práci k dispozici, jednoduše neobsahuje dostatečné množství událostí. Proto si v této práci popíšeme pouze dva experimenty, jako ukázku použití vytvořené knihovny.

Volbu a popis provedených experimentů obsahuje první sekce této kapitoly. Druhá sekce se zabývá možnostmi pokračování v práci a rozšíření knihovny pro detekci událostí.

6.1 Experimenty

Zajímavé by bylo porovnat všechny tři formulace detekce událostí (viz 3.2.2) a zjistit, která je nejlepší. To však nelze udělat zrovna přímočaře, protože nevíme, jakými parametry dosáhnout největší přesnosti detekce událostí při použití jednotlivých formulací. Nejprve bychom tedy museli provést řadu experimentů a zjistit nejlepší kombinaci parametrů pro každou formulaci a až poté bychom mohli vzájemně porovnat jimi dosažené výsledky detekce událostí. Pro ukázku je tedy vhodné zvolit jiné, méně ambiciózní experimenty.

Kvůli malé databázi prostoročasových dat jsme také docela omezeni při volbě událostí, které chceme detekovat. Většina událostí spojených s přerušením hry se ve videozáznamech naší databáze nevyskytuje ani dvacetkrát. Zaměříme se proto buďto na detekci všech úseků přerušené hry dohromady nebo na detekci faulů a vhazování, které jsou nejčastější (viz tabulka 5.3).

6.1.1 Experiment č. 1

V prvním experimentu budeme zkoumat, jaký vliv má na detekci událostí délka vstupních sekvencí. Jediný parametr, který tedy budeme modifikovat, je parametr *seqLenght* při vytváření vstupních sekvencí (metoda *getIOData* a funkce *getExtIOData*). Začneme na hodnotě 25, což odpovídá jedné sekundě hry (frekvence prostoročasových dat je stejně jako frekvence snímků videozáznamu rovna hodnotě 25). Poté budeme hodnotu parametru *seqLenght* zvyšovat o 75 (tři sekundy) dokud nedosáhneme délky 400.

Pro značení sekvencí je použita obyčejná síť pro značení sekvencí N:1 realizovaná třídou *ANN.SeqClassifier*. Ostatní parametry jsou nastaveny na hodnoty podle tabulky 6.1 a během experimentu se nemění. Ze zvolených hodnot parametrů můžeme mimo jiné vyčíst, že

Název parametru	Hodnota parametru
seqLenght =	?
overlapping =	None
scalingMethod =	"std"
labelingMode =	Data.SEQ_CLASSIFICATION
eventsToDetect =	["Foul", "Throw-in"]
archType	0
ANN.Net.LAYERS =	1
ANN.Net.UNITS_NUMBER =	128
ANN.Net.EPOCHS =	30
ANN.Net.BATCH_SIZE =	32
preciseDetection =	False
minEventLen =	2
minEventDist =	2

Tabulka 6.1: Hodnoty parametrů použity pro první experiment

pro tento experiment byla tedy zvolena formulace detekce událostí jako klasifikace sekvencí N:1 a že detekujeme pouze fauly a vhazování.

Protože v tomto experimentu porovnáváme až závěrečný výsledek detekce a ne výsledek značení sekvencí neuronovou sítí, použijeme všechna data databáze pro trénování sítě a testovací sadu dat můžeme nechat prázdnou. Toho docílíme nastavením třídní proměnné *Net.TEST_RATE* na hodnotu 0. Běžným postupem je natrénovat neuronovou síť na datech z databáze a poté detekovat události z úplně jiného videozáznamu, na jehož prostoročasových datech se síť neučila. Vzhledem k nedostatku vstupních prostoročasových dat však v tomto ukázkovém experimentu budeme detekovat události v jednom z videozáznamů obsažených v databázi (konkrétně ve videozáznamu *Trnava_2015-11-13-20-46-17_3*).

Výsledky experimentu jsou obsaženy v tabulce 6.2. Pro každou hodnotu parametru *seqLenght* obsahuje tabulka kvalitu detekovaných událostí zvlášť pro obě kategorie (fauly a vhazování). Kvalita je vyjádřena metrikami precision, recall, f1-score a také počtem detekovaných událostí. Počty detekovaných událostí mají vypovídající hodnotu až poté, co je dáme do poměru se skutečným počtem událostí videozáznamu. V tabulce 5.3 můžeme vidět že videozáznam *Trnava_2015-11-13-20-46-17_3* obsahuje 12 faulů a 22 vhazování.

Nejkvalitnějšího výsledku detekce událostí bylo dosaženo při použití sekvencí o délce 25. Počty detekovaných faulů i vhazování přesně odpovídají realitě a také hodnota f1-score je velmi vysoká – 0,98. Se zvětšující se hodnotou parametru *seqLenght* se počet detekovaných událostí snižuje a také jejich přesnost je horší. Můžeme tak říci, že při použití formulace detekce událostí jako značení sekvencí N:1 je lepší použít kratší vstupní sekvence. Jako pokračování toho experimentu bychom mohli délku vstupních sekvencí dále snižovat a sledovat, jestli přesnost detekovaných událostí ještě dále poroste.

6.1.2 Experiment č. 2

V rámci druhého experimentu se pokusíme zjistit jak volba typu architektury neuronové sítě ovlivňuje výsledek značení sekvencí. Experimentovat tedy budeme s hodnotou parametru *archType* při vytváření instance neuronové sítě pro značení sekvencí. Detekci událostí budeme tentokrát formulovat jako značení sekvencí N:N a použijeme obyčejnou variantu sítě realizovanou třídou *ANN.SeqLabeler*. Pro tuto síť poskytuje knihovna čtyři architektury

seqLenght		precision	recall	f1-score	počet
25	**faul**	0,99	0,97	0,98	12
	vhazování	0,98	0,97	0,98	22
100	**faul**	0,86	0,78	0,82	18
	vhazování	0,85	0,85	0,85	21
175	**faul**	0,88	0,88	0,88	15
	vhazování	0,88	0,80	0,84	15
250	**faul**	0,92	0,55	0,69	9
	vhazování	0,89	0,76	0,82	13
325	**faul**	0,90	0,67	0,77	6
	vhazování	0,78	0,61	0,68	8
400	**faul**	0,84	0,77	0,80	7
	vhazování	0,77	0,61	0,68	8

Tabulka 6.2: Výsledky prvního experimentu

Název parametru	**Hodnota parametru**
seqLenght =	200
overlapping =	50
scalingMethod =	"std"
labelingMode =	Data.SEQ_LABELING
eventsToDetect =	2
archType	?
ANN.Net.LAYERS =	1
ANN.Net.UNITS_NUMBER =	128
ANN.Net.EPOCHS =	30
ANN.Net.BATCH_SIZE =	32

Tabulka 6.3: Hodnoty parametrů použity pro druhý experiment

(viz tabulka 5.6). Postupně natrénujeme síť s použitím všech architektur a vyhodnotíme výsledky označení sekvencí z testovací sady dat. Ze získaných výsledků se pokusíme zjistit, která architektura je nejlepší.

Při vytváření sítí rozdělíme vstupní data na trénovací a testovací sadu náhodně v poměru 4:1 (nastavíme hodnotu třídní proměnné *Net.TEST_RATE* na hodnotu 0,2). Hodnoty ostatních parametrů použité pro tento experiment můžeme vidět v tabulce 6.3. Všimněme si, že sítě realizují binární klasifikaci, protože označujeme všechny události spojené s přerušenou hrou jedním návěštím (parametr *eventsToDetect = 2*). V tomto experimentu porovnáváme výsledky značení sekvencí a ne až konečný výsledek detekce událostí. Proto hodnoty parametrů interpretace výstupních sekvencí (krok 7), *preciseDetection*, *minEventLen* a *minEventDist*, znát nepotřebujeme.

Výsledky vyhodnocení sítí s jednotlivými testovanými architekturami obsahuje tabulka 6.4. Můžeme v ní nalézt hodnoty všech hlavních klasifikačních metrik (precision, recall, f1-score a support) pro obě klasifikované třídy a také celkovou přesnost klasifikace vyjádřenou v procentech. Návěští 0 odpovídá probíhající hře a návěští 1 značí přerušenou hru.

Z výsledků vyplývá, že první tři architektury odpovídající hodnotám 0, 1, a 2 parametru *archType* (klasická LSTM, obousměrná LSTM a obousměrná LSTM s CRF výstupní vrstvou) jsou si víceméně rovnocenné. Celková přesnost se sice o trochu liší, ale nejpod-

archType	přesnost		precision	recall	f1-score	support
0	93.99%	**0** (▶)	0,95	0,95	0,95	120961
		1 (❙❙)	0,92	0,93	0,92	75039
		celkem:	0,94	0,94	0,94	196000
1	94.52%	**0** (▶)	0,95	0,96	0,96	127155
		1 (❙❙)	0,93	0,92	0,92	68845
		celkem:	0,95	0,95	0,95	196000
2	93.77%	**0** (▶)	0,96	0,94	0,95	125158
		1 (❙❙)	0,90	0,94	0,92	70842
		celkem:	0,94	0,94	0,94	196000
3	84.72%	**0** (▶)	0,86	0,89	0,88	120218
		1 (❙❙)	0,82	0,77	0,80	75782
		celkem:	0,85	0,85	0,85	196000

Tabulka 6.4: Výsledky druhého experimentu

statnější metrikou je pro nás f1-score třídy 1, která je pro tyto tři architektury stejná – 0,92. Tato hodnota totiž v podstatě vyjadřuje, jak dobře byly označeny úseky s přerušenou hrou. Poslední implementovaná architektura, encoder-decoder LSTM, vykazuje znatelně horší hodnoty všech metrik.

6.1.3 Shrnutí experimentů

Jak již bylo naznačeno v úvodu této kapitoly, účelem těchto dvou experimentů je především demonstrovat použití vytvořené knihovny pro detekci událostí. Experimentování s cílem získat nové znalosti bude možné, až při získání vetší databáze časoprostorových dat. Zatím se kvůli malému množství dat použitých pro trénování sítí na dosažené výsledky nelze spolehnout.

S tímto vědomím jsme si mohli dovolit v experimentu č. 1 uchýlit k poněkud nestandardnímu řešení s cílem maximalizovat množství dat pro učení neuronových sítí. Porovnáváme v něm totiž výsledky detekce událostí z videozáznamu, který je součástí databáze, na níž se neuronová síť učila. Díky tomu vypadají výsledky docela slibně. Síť se však pravděpodobně naučila rozpoznávat pouze konkrétní události, které byly součástí trénovacích dat, a ne obecnou strukturu detekovaných událostí. Pro náš účel to však nevadí.

Druhý experiment je už korektní po všech stránkách. V něm porovnáváme přímo výsledek značení sekvencí sítěmi a detekce událostí zde neprobíhá. Nemusíme tak pro vyhodnocení výsledku vyčlenit celý videozáznam, ale stačí rozdělit vstupní data na trénovací a testovací sadu.

6.2 Další možnosti dosažení přesnější detekce událostí a možná rozšíření knihovny

Ve snaze dosáhnout přesnější detekce událostí lze experimentovat i s dalšími možnostmi, které vytvořená knihovna zatím neposkytuje. Jednou z nich je upravit předzpracování časoprostorových dat. Kromě možnosti reprezentovat data různým počtem různých hodnot diskutované v podsekci 3.2.3 můžeme uvážit i snížení frekvence získávání údajů. To by v praxi znamenalo vynechání každého n-tého prvku datové sekvence a tedy zrychlení hry.

Pokud je frekvence získávání údajů příliš velká, jsou hodnoty sousedních prvků datové sekvence velice podobné a tudíž nemají tak velkou vypovídající hodnotu. Naopak, je-li tato frekvence příliš malá, mohou nám důležité informace zcela uniknout.

Dále bychom se mohli pokusit vylepšit proces učení neuronových sítí volbou poměru počtů vzorků jednotlivých tříd v trénovací sadě dat. Za zvážení by stálo také implementovat další architektury neuronových sítí nebo i úplně jiné metody strojového učení pro značení sekvencí (např. HMM).

Vytvořená knihovna zatím neumožňuje detekovat události typu 1 ani typu 3 (viz přehled fotbalových událostí 2.2). Nelze ani rozlišovat mezi událostmi jednoho týmu a událostmi druhého týmu. To jsou další předměty možného rozšíření knihovny.

Další možnosti v oblasti detekce událostí otevírají kvalitnější vstupní prostoročasová data. Pokud bychom měli k dispozici prostoročasová data míče, mohli bychom se pokusit realizovat i přístup k detekci událostí založený na heuristických pravidlech (viz sekce 3.1). Tento přístup je sice poměrně naivní, avšak bylo by možné jej zkombinovat s využitím metod strojového učení. Ty bychom použili pro detekci jednotlivých dílčích událostí a tím odstranili hlavní nevýhodu tohoto přístupu, kterou je velká závislost úspěchu detekce událostí na přesnosti a kvalitě prostoročasových dat.

Zajímavou myšlenkou také je při detekci událostí využit rozpoznávání gest rozhodčího. Abychom se však mohli pokusit něco takového vůbec realizovat, bylo by třeba umět mezi detekovanými objekty v obraze rozpoznat rozhodčího podle barvy jeho dresu.

Kapitola 7

Závěr

Automatická detekce událostí je v dnešní době velkou výzvou a tato práce může sloužit jako odrazový můstek při skoku a ponoření se do této problematiky. Práce obsahuje kompletní rozbor problematiky detekce událostí ve fotbale, teorii potřebnou k řešení tohoto problému a také návrh a realizaci detekce událostí z prostoročasových dat. Jako výsledek vznikla knihovna, jenž umožňuje převést problém detekce událostí na úlohu typu značení sekvencí, tuto úlohu poté vyřešit pomoci umělých neuronových sítí a nakonec interpretovat výsledek značení sekvencí jako detekované události. Tento proces má v každém svém kroku několik parametrů, jejichž hodnoty ovlivňují jeho průběh a také výsledek. Je tak možné vytvářet různé formulace detekce událostí jako problém značení sekvencí, používat různé architektury neuronových sítí, ovlivňovat průběh jejich učení a také interpretaci jejich výstupů. Cílem použití této knihovny je experimentování s těmito variantami a zjistit, jak dosáhnout co nejpřesnějších výsledků detekce událostí.

Funkčnost knihovny byla ověřena v rámci provedení dvou ukázkových experimentů. Jejich výsledky však nejsou příliš relevantní, protože byly realizovány s použitím pouze velice omezené databáze prostoročasových dat. Práce navazující na tuto by se tedy měla zaměřit nejdříve na vytvoření mnohem větší databáze a poté na provádění experimentů s cílem získat nové znalosti. Díky této práci je tedy možné získat nové informace, které nám mohou napovědět, jakým směrem se dále vydat ve vývoji reálného systému pro automatickou detekci událostí ve fotbale.

Literatura

[1] *Camvision*. [Online; navštíveno 26.4.2018].
URL https://www.camvision.cz/

[2] *keras-contrib: Keras community contributions*. [Online; navštíveno 26.4.2018].
URL https://github.com/keras-team/keras-contrib

[3] *Keras Documentation*. [Online; navštíveno 26.4.2018].
URL https://keras.io/

[4] *sklearn.metrics.confusion_matrix*. [Online; navštíveno 2.5.2018].
URL http://scikit-learn.org/stable/modules/generated/sklearn.metrics.confusion_matrix.html

[5] *sklearn.metrics.precision_recall_fscore_support*. [Online; navštíveno 2.5.2018].
URL http://scikit-learn.org/stable/modules/generated/sklearn.metrics.precision_recall_fscore_support.html

[6] *TensorFlow*. [Online; navštíveno 26.4.2018].
URL https://www.tensorflow.org/

[7] Adidas: *Adidas miCoach Smart Soccer Ball*. [Online; navštíveno 18.12.2017].
URL https://www.adidas.com/us/micoach-smart-soccer-ball/G83963.html

[8] Bengio, Y.; Simard, P.; Frasconi, P.: *Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks*, ročník 5, č. 2, Březen 1994: s. 157–166, doi:10.1109/72.279181.

[9] Brownlee, J.: *Encoder-Decoder Long Short-Term Memory Networks*. [Online; navštíveno 26.4.2018].
URL https://machinelearningmastery.com/encoder-decoder-long-short-term-memory-networks/

[10] Brownlee, J.: *How to Prepare Sequence Prediction for Truncated Backpropagation Through Time in Keras*. [Online; navštíveno 26.4.2018].
URL https://machinelearningmastery.com/truncated-backpropagation-through-time-in-keras/

[11] Brownlee, J.: *Understanding Stateful LSTM Recurrent Neural Networks in Python with Keras*. [Online; navštíveno 26.4.2018].
URL https://machinelearningmastery.com/understanding-stateful-lstm-recurrent-neural-networks-python-keras/

[12] Brownlee, J.: *How to Handle Very Long Sequences with Long Short-Term Memory Recurrent Neural Networks*. Červen 2017, [Online; navštíveno 11.4.2018].
URL https://machinelearningmastery.com/handle-long-sequences-long-short-term-memory-recurrent-neural-networks/

[13] Brownlee, J.: *How to Prepare Sequence Prediction for Truncated Backpropagation Through Time in Keras*. Červen 2017, [Online; navštíveno 11.4.2018].
URL https://machinelearningmastery.com/truncated-backpropagation-through-time-in-keras/

[14] CreateMoMo: *CRF Layer on the Top of BiLSTM - 1*. [Online; navštíveno 26.4.2018].
URL https://createmomo.github.io/2017/09/12/CRF_Layer_on_the_Top_of_BiLSTM_1/

[15] Dietterich, T. G.: *Machine Learning for Sequential Data: A Review*. In *Structural, Syntactic, and Statistical Pattern Recognition*, Springer, Srpen 2002, s. 15–30, doi:10.1007/3-540-70659-3_2.

[16] D'Orazio, T.; Leo, M.; Spagnolo, P.; aj.: *An Investigation Into the Feasibility of Real-Time Soccer Offside Detection From a Multiple Camera System. IEEE Transactions on Circuits and Systems for Video Technology*, ročník 19, č. 12, Červenec 2009: s. 1804–1818, doi:10.1109/TCSVT.2009.2026817.

[17] Forney, G.: *The viterbi algorithm. Proceedings of the IEEE*, ročník 61, č. 3, Březen 1973: s. 268–278, doi:10.1109/PROC.1973.9030.

[18] Geeksforgeeks: *Dynamic Programming | Set 10 (0-1 Knapsack Problem)*. [Online; navštíveno 15.12.2017].
URL http://www.geeksforgeeks.org/knapsack-problem/

[19] Graves, A.: *Supervised Sequence Labelling with Recurrent Neural Networks*. Springer, Leden 2012, doi:10.1007/978-3-642-24797-2.

[20] Hashimoto, S.; Ozawa, S.: *A System for Automatic Judgment of Offsides in Soccer Games*. In *2006 IEEE International Conference on Multimedia and Expo*, IEEE, Červenec 2006, s. 1889–1892, doi:10.1109/ICME.2006.262924.

[21] Hochreiter, S.; Schmidhuber, J.: *Long Short-Term Memory. Neural Computation*, ročník 9, č. 8, Listopad 1997: s. 1735–1780, doi:10.1162/neco.1997.9.8.1735.

[22] Itoh, H.; Takiguchi, T.; Ariki, Y.: *Event Detection and Recognition Using HMM with Whistle Sounds*. In *2013 International Conference on Signal-Image Technology & Internet-Based Systems*, IEEE, Prosinec 2013, s. 14–21, doi:10.1109/SITIS.2013.14.

[23] Jensen, J. C. C.: *Event detection in soccer using spatio-temporal data*. diplomová práce, Aarhus University, Listopad 2015.
URL http://www.cs.au.dk/~gerth/advising/thesis/jens-christian-christensen-jensen.pdf

[24] Joshi, R.: *Accuracy, Precision, Recall & F1 Score: Interpretation of Performance Measures*. [Online; navštíveno 3.5.2018].
URL http://blog.exsilio.com/all/accuracy-precision-recall-f1-score-interpretation-of-performance-measures/

[25] Karpathy, A.: *The Unreasonable Effectiveness of Recurrent Neural Networks*. Květen 2015, [Online; navštíveno 11.4.2018].
URL http://karpathy.github.io/2015/05/21/rnn-effectiveness/

[26] Kinexon: *World premiere: KINEXON presents sensor in ball at live-TV soccer match*. [Online; navštíveno 12.1.2018].
URL http://kinexon-sports.com/world-premiere-kinexon-presents-sensor-in-ball-at-live-tv-soccer-match/

[27] Lafferty, J. D.; McCallum, A.; Pereira, F. C. N.: *Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data*. In *ICML '01 Proceedings of the Eighteenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, Červen 2001, s. 282–289.

[28] Liu, Y.; Liang, D.; Huang, Q.; aj.: *Extracting 3D information from broadcast soccer video*. *Image and Vision Computing*, ročník 24, č. 10, Říjen 2006: s. 1146–1162, doi:10.1016/j.imavis.2006.04.001.

[29] McCallum, A.; Freitag, D.; Pereira, F. C. N.: *Maximum Entropy Markov Models for Information Extraction and Segmentation*. In *ICML '00 Proceedings of the Seventeenth International Conference on Machine Learning*, Morgan Kaufmann Publishers, Červen 2000, s. 591–598.

[30] Olah, C.: *Understanding LSTM Networks*. Srpen 2015, [Online; navštíveno 11.4.2018].
URL http://colah.github.io/posts/2015-08-Understanding-LSTMs/

[31] Potts, C.: *Sentiment Symposium Tutorial: Classifiers*. 2011, [Online; navštíveno 11.4.2018].
URL http://sentiment.christopherpotts.net/classifiers.html#nb

[32] Rabiner, L.; Juang, B.: *An introduction to hidden Markov models*. *IEEE ASSP Magazine*, ročník 3, č. 1, Leden 1986: s. 4–16, doi:10.1109/MASSP.1986.1165342.

[33] Rehman, A.; Saba, T.: *Features extraction for soccer video semantic analysis: current achievements and remaining issues*. *Artificial Intelligence Review*, ročník 41, č. 3, Březen 2014: str. 451–461, doi:10.1007/s10462-012-9319-1, (publikováno online 6. Března 2012).

[34] Richly, K.; Bothe, M.; Rohloff, T.; aj.: *Recognizing Compound Events in Spatio-Temporal Football Data*. In *International Conference on Internet of Things and Big Data (IoTBD 2016)*, SCITEPRESS, 2016, s. 27–35, doi:10.5220/0005877600270035.

[35] Richly, K.; Moritz, F.; Schwarz, C.: *Utilizing Artificial Neural Networks to Detect Compound Events in Spatio-Temporal Soccer Data*. In *SIGKDD Workshop on Mining and Learning from Time Series*, ACM, Srpen 2017.

[36] RoboRealm: *Mode pixel value filter*. [Online; navštíveno 11.4.2018].
URL http://www.roborealm.com/help/Mode.php

[37] Rumelhart, D. E.; Hinton, G. E.; Williams, R. J.: *Learning internal representations by error propagation*. In *Parallel distributed processing: explorations in the microstructure of cognition, vol. 1*, MIT Press, Leden 1986, s. 318–362.

[38] Sarle, W. S.: *Neural Network FAQ, part 2 of 7: Learning*. [Online; navštíveno 26.4.2018].
URL ftp://ftp.sas.com/pub/neural/FAQ2.html#A_std

[39] Schuster, M.; Paliwal, K.: *Bidirectional recurrent neural networks. IEEE Transactions on Signal Processing*, ročník 45, č. 11, Listopad 1997: s. 2673–2681, doi:10.1109/78.650093.

[40] de Sousa, S. F.; de A. Araújo, A.; Menotti, D.: *An overview of automatic event detection in soccer matches*. In *2011 IEEE Workshop on Applications of Computer Vision (WACV)*, IEEE, Leden 2011, s. 31–38, doi:10.1109/WACV.2011.5711480.

[41] Sport, C.: *OptimEye S5*. [Online; navštíveno 18.12.2017].
URL https://www.catapultsports.com/products/optimeye-s5

[42] Staff, C. T.: *Sequence Labeling and HMMs*. Březen 2016, [Online; navštíveno 11.4.2018].
URL https://cs.brown.edu/courses/csci1460/assets/files/hmm.pdf

[43] Sutton, C.; McCallum, A.: *An Introduction to Conditional Random Fields. Foundations and Trends in Machine Learning*, ročník 4, č. 4, Duben 2004: s. 267–373, doi:10.1561/2200000013.

[44] Tavassolipour, M.; Karimian, M.; Kasaei, S.: *Event Detection and Summarization in Soccer Videos Using Bayesian Network and Copula. IEEE Transactions on Circuits and Systems for Video Technology*, ročník 24, č. 2, Leden 2013: s. 291–304, doi:10.1109/TCSVT.2013.2243640.

[45] Tovinkere, V.; Qian, R. J.: *Detecting Semantic Events in Soccer Games: Towards A Complete Solution*. In *IEEE International Conference on Multimedia and Expo, 2001. ICME 2001.*, IEEE, Srpen 2001, s. 1040–1043, doi:10.1109/ICME.2001.1237851.

[46] Wei, X.; Sha, L.; Lucey, P.; aj.: *Large-Scale Analysis of Formations in Soccer*. In *2013 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, IEEE, Prosinec 2013, s. 1–8, doi:10.1109/DICTA.2013.6691503.

[47] Werbos, P.: *Backpropagation through time: what it does and how to do it. Proceedings of the IEEE*, ročník 78, č. 10, Říjen 1990: s. 1550–1560, doi:10.1109/5.58337.

[48] Williams, R. J.; Zipser, D.: *Gradient-based learning algorithms for recurrent networks and their computational complexity*. In *Backpropagation*, L. Erlbaum Associates, Leden 1995, s. 433–486.

[49] Yu, X.; Xu, C.; Leong, H. W.; aj.: *Trajectory-based ball detection and tracking with applications to semantic analysis of broadcast soccer video*. In *ACM international conference on Multimedia*, ACM, Listopad 2003, s. 11–20, doi:10.1145/957013.957018.

Příloha A

Obsah CD

- **databáze/** – adresář databáze s prostoročasovými daty a anotacemi videozáznamů
- **technická zpráva.pdf** – technická zpráva ve formátu PDF
- **zdrojové soubory/** – adresář obsahující zdrojové soubory knihovny pro detekci událostí a soubor s příklady použití této knihovny
- **zdrojové soubory technické zprávy/** – adresář se zdrojovými soubory technické zprávy