ČESKÉ VYSOKÉ
UČENÍ TECHNICKÉ
V PRAZE

F3

Fakulta elektrotechnická
Katedra počítačů

Diplomová práce

Sofware pro analýzu signálů a inteligentní detekci událostí v optických senzorových sítích

Bc. Gleb Petrovichev
Obor: Softwarové inženýrství
Program: Otevřená informatika

Leden 2020
Vedoucí práce: doc. Ing. Leoš Boháč, Ph.D.

ZADÁNÍ DIPLOMOVÉ PRÁCE

I. OSOBNÍ A STUDIJNÍ ÚDAJE

Příjmení: **Petrovichev** Jméno: **Gleb** Osobní číslo: **453251**

Fakulta/ústav: **Fakulta elektrotechnická**

Zadávající katedra/ústav: **Katedra počítačů**

Studijní program: **Otevřená informatika**

Studijní obor: **Softwarové inženýrství**

II. ÚDAJE K DIPLOMOVÉ PRÁCI

Název diplomové práce:

Sofware pro analýzu signálů a inteligentní detekci událostí v optických senzorových sítích

Název diplomové práce anglicky:

Signal Analysis and Intelligent Event Detection System for Optical Sensing Networks

Pokyny pro vypracování:

Navrhněte a implementujte software analyzující signály z optických senzorových sítí v reálném čase a umožňující provádět inteligentní detekci různých typů událostí. K řešení se pokuste využít moderních principů analýzy velkých dat, vhodné signálové transformace, prvky strojového učení a softwarové řešení s využitím centralizace v cloudu. Součástí práce, kromě jiného, bude i návrh procesního řetězce a ověření systému experimentálně na syntetických, tak i reálných datech z optických senzorů.

Seznam doporučené literatury:

[1] GRATTAN, L.S. a B.T. MEGGITT. Optical Fiber Sensor Technology. 2010. ISBN 1441949836.
[2] SPILLMAN, William B. a edited by Eric UDD. Fiber optic sensors: an introduction for engineers and scientists. 2nd ed. Oxford: Wiley-Blackwell, 2010. ISBN 04-701-2684-1.
[3] ROBERTS, Michael J. Signals and systems: analysis using transform methods and MATLAB. Third edition. New York, NY: McGraw-Hill Education, [2018]. ISBN 978-0078028120.
[4] MULLER, Andreas C a Sarah GUIDO. Introduction to machine learning with Python: a guide for data scientists. Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1449369415.

Jméno a pracoviště vedoucí(ho) diplomové práce:

doc. Ing. Leoš Boháč, Ph.D., katedra telekomunikační techniky FEL

Jméno a pracoviště druhé(ho) vedoucí(ho) nebo konzultanta(ky) diplomové práce:

Datum zadání diplomové práce: **27.09.2018** Termín odevzdání diplomové práce: **07.01.2020**

Platnost zadání diplomové práce: **19.02.2020**

doc. Ing. Leoš Boháč, Ph.D. podpis vedoucí(ho) práce	podpis vedoucí(ho) ústavu/katedry	prof. Ing. Pavel Ripka, CSc. podpis děkana(ky)

III. PŘEVZETÍ ZADÁNÍ

Diplomant bere na vědomí, že je povinen vypracovat diplomovou práci samostatně, bez cizí pomoci, s výjimkou poskytnutých konzultací. Seznam použité literatury, jiných pramenů a jmen konzultantů je třeba uvést v diplomové práci.

Datum převzetí zadání Podpis studenta

Poděkování / Prohlášení

Tímto bych chtěl poděkovat vedoucímu této práce doc. Ing. Leoši Boháčovi, Ph.D., Ing. Radku Maříkovi, CSc. a Ing. Jakubu Maršálkovi za jejich rady, náměty, připomínky, přípravu laboratorních dat a trpělivost.

Dále děkuji přátelům za jejich pomoc a podporu jak při psání diplomové práce, tak v průběhu celého studia. Bez nich bych tuto práci nikdy nedokončil.

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.
V Praze dne 7. 1. 2020

.....

Abstrakt / Abstract

Tato diplomová práce se věnuje navržení systému pro sběr signálových dat ze senzorů v optických sítích, detekci v signálovém průběhu anomálií a jejich klasifikaci s možností případného spuštění určité akce. Budou prodiskutovány možností uplatnění tohoto postupu v praxi v různých oblastech. V rámci práce se projednají různé metodiky využívané pro klasifikaci signálů a zvuků. Budou porovnané a vyhodnocené možné architektonické návrhy systému představené v podobě 4+1 pohledů a implementuje se funkční prototyp klasifikátoru. Na tomto prototypu se vyzkouší různé přístupy z oblasti strojového učení a neuronových sítí a provede se experiment s rozdělením vstupního signálu na několik frekvenčních pásem. Část práce tvoří diskuze o nutnosti transformace vstupních dat před jejich odesláním na vstup klasifikátorů a implementují se několik transformátorů. V závěru se uvedou možností pro navázání na práci a rozšíření tématu.

This master's thesis focuses on designing a system for collecting signal data from sensors in optical networks, anomaly detection in a waveform, and their classification with an option of conditional action execution. Ways of applying this technique in practice in different fields are discussed. Various methodologies used for signal and sound classification are considered. Few possible architectures for the system in question are presented with the 4+1 views method, then compared and evaluated. Then a functional prototype of the classifier is implemented. This prototype is used to try different approaches in machine learning and neural networks domain, and the experiment with separation of an input signal into multiple frequency bands is conducted with its help. Part of the thesis is comprised of a discussion about the necessity of applying certain transformations to classifier's input data. Some of those transformers are implemented. In the thesis conclusion, ways of continuing and extending the topic of this thesis are presented.

Title translation: Signal Analysis and Intelligent Event Detection System for Optical Sensing Networks

Obsah /

1 Úvod ... 1
1.1 Cíl práce ... 1
1.2 Motivace k výběru tématu ... 1
1.3 Zasazení problému do kontextu ... 2
2 Teoretická část ... 4
2.1 Volba vhodného nástroje ... 4
2.2 Transformace vstupních dat ... 5
2.3 Metrika ... 5
2.4 Systém ... 5
2.4.1 Funkční požadavky ... 6
2.4.2 Nefunkční požadavky ... 6
2.4.3 Volba technologií ... 6
2.5 4+1 pohledy ... 7
2.5.1 Logický pohled ... 7
2.5.2 Procesní pohled ... 12
2.5.3 Vývojový pohled ... 15
2.5.4 Fyzický pohled ... 18
2.5.5 Scénáře ... 23
2.6 Specifika produkčního provozu systému ... 26
2.6.1 Možnosti paralelizace ... 26
2.7 Experiment s oddělením frekvenčních pásem ... 27
2.7.1 Srovnání podobnosti vzorků ... 27
3 Praktická část ... 29
3.1 Porovnání typů neuronové sítě ... 29
3.1.1 Plně propojený vícevrstvý perceptron ... 29
3.1.2 Konvoluční síť ... 30
3.2 Transformace vstupních dat ... 30
3.2.1 Přemapování tříd ... 32
3.2.2 FFT ... 32
3.2.3 Padding ... 32
3.2.4 Podvzorkování / decimace ... 32
3.2.5 Normalizace ... 33
3.3 Oddělení frekvenčních pásem ... 33
3.3.1 Pomocný experiment pro srovnání podobností vzorků ... 34
3.4 Experiment s ručně připravenými daty ... 35
4 Diskuze ... 36
4.1 Oddělení pásem ... 36
5 Závěr ... 40
Literatura ... 42
A Zdrojový kód ... 45
B Data s oddělenými frekvenčními pásmy ... 46
C Ručně označkovaná data ... 47

Kapitola 1
Úvod

1.1 Cíl práce

Cílem této diplomové práce je navržení systému pro automatickou klasifikaci anomálií detekovaných v signálu v optických senzorových sítích. Anomálií se v tomto kontextu rozumí neobvyklá nebo za normálních podmínek neočekávaná změna v odraženém signálu která může, ale nemusí svědčit o výskytu události z předem definovaných kategorií. Budou diskutovány otázky výběru vhodné technologie klasifikace v oblasti strojového učení, volby efektivní architektury výsledného softwaru jak pro účely prototypu, tak pro případnou produkční implementaci a nutnosti přípravy a transformace signálu před jeho klasifikací. V rámci praktické části se implementuje funkční prototyp a porovnávají se různé přístupy k reprezentaci vstupních dat.

Během experimentů se také ověří, zda je přínosné pro výslednou přesnost klasifikace oddělovat jednotlivá frekvenční pásma před detekcí anomálií. Předpokládají se dva možné závěry. Buď se najde charakteristické pásmo, které zvýší přesnost, nebo se původní data z celého rozsahu zaznamenaných hodnot ukážou být spolehlivější.

Další experiment ověří, zda bude úspěšnější klasifikace provedená na základě ručně připravených dat. Očekává se, že se podaří dosáhnout vyšší přesnosti predikce. Zároveň se vyzkouší sjednocení několika tříd za účelem zlepšení výsledků klasifikace.

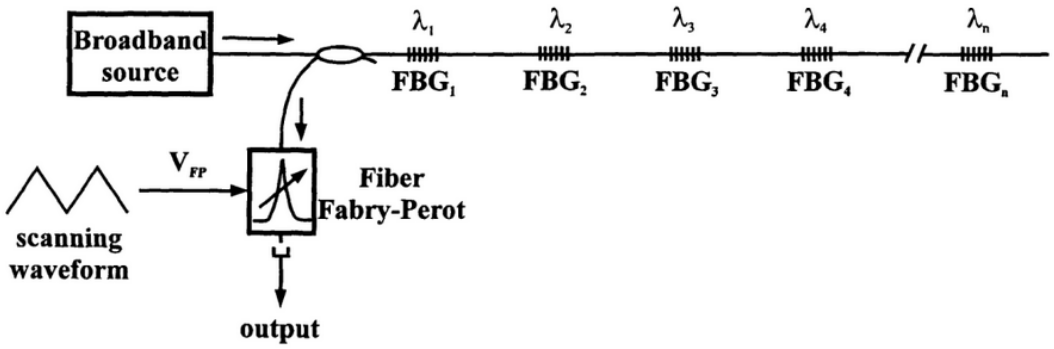
1.2 Motivace k výběru tématu

Důležitým faktorem, který činí téma zpracování a klasifikace signálu pozoruhodným, je všudypřítomnost objektu zájmu těchto oborů, tj. signálů. Zdroj signálů je jak sama Země, tak kosmická tělesa kolem ní. Vyzařují určité množství tepla s určitou intenzitou, která se neustále mění. Často se se signálovou reprezentací také pracuje ve sféře financí a podnikání. Mezi příklady takových využití lze zařadit vývoj ceny na burze, analýzu úspěšnosti prodeje, výroby nebo aktivity uživatelů webových stránek. Zdrojem různorodých signálů je i samotný člověk. Moderní medicínská praxe zahrnuje například práci se záznamy elektrického potenciálu způsobeného mozkovou aktivitou (elektroencefalografie, EEG), srdeční aktivitou (elektrokardiogram, EKG) nebo také svalovou aktivitou (elektromyografie, EMG). Všechny tyto nástroje nabízí výstup v podobě časové řády. Vzhledem k naznačené šířce uplatnění signálových technologií, univerzální klasifikátor signálů by mohl být přínosný hned v několika sférách. [1]

V případě úspěšného dosažení určité minimální potřebné míry přesnosti klasifikace, kterou vyžaduje oblast užití, se tento přístup dá aplikovat v systémech sledování zatížení hlukem, chytrých domech i kancelářích [2]. V ochranných systémech by se oproti tradičním metodám vyznačoval nejen větší mírou automatizace, ale i zrychlením celého procesu od pořízení záznamu do spuštění akce (např. oznámení uživatele). Za jinou výhodu v podmínkách moderní legislativy by se dala považovat také skutečnost, že záznamy pořízené v podobě odraženého signálu v optickém vlákně mohou, např.

v porovnání s kamerovými systémy, v mnohem menší míře a s menší pravděpodobností porušovat zákony o uchování osobních údajů bez souhlasu [3]. Analýza audiálních informací je zároveň také méně náročná na výpočetní zdroje v porovnání s videem [2].

Problém automatizace zpracování a klasifikace signálů je také zajímavý a přínosný i kvůli fyzickým a psychickým omezením lidského vnímání a rozlišování. Například u sluchu je prokázáno, že člověk je schopný rozlišovat frekvence zvuku v rozsahu 1000-2000 Hz, pokud je rozdíl mezi těmito dvěma zvuky aspoň 3,6 Hz [4]. Celé spektrum lidskému uchu přístupných frekvencí je rovněž omezen a činí rozsah 15-20000 Hz [5]. Stejně tak je člověk omezen i v intenzitě zvuků, které je schopen slyšet, aniž by pociťoval bolest. Práh bolesti nastává někde kolem 140 dB [6].



Obrázek 1.1. Ukázka skenovacího filtru s WDM multiplexováním založeného na FBG mřížkách odrážejících světlo s různou délkou vlny. [7]

Tato práce se ale bude zabývat hlavně signály v optických senzorových sítích. Konkrétně metodiky využívající k měření senzory. Tyto technologie se začaly vyvíjet již na začátku 70. let. Byla navržena široká řada různých metod nabízejících účinný způsob měření schopný konkurovat tradičním postupům v mnohých oblastech. Senzorové systémy postavené na optickém vlákně se vyznačují kompaktností, necitlivosti k elektromagnetické interferenci (EMI), jsou málo invazivní vůči prostředí a také mohou být vyrobeny s konkurenceschopnými náklady. Jednou z klíčových technologií v tomto oboru jsou Braggovy mřížky (FBG). FBG odráží pouze vlny určité délky a přenáší všechny ostatní. Tato vlastnost dovoluje multiplexování v jednom vlákně s využitím několika mřížek a senzorů s různou vlnovou délkou detekovaného světla, tj. barvou (wavelength-division multiplexing, WDM, viz obr. 1.1). Mřížky reagují na změnu teploty nebo roztažení vlákna a mění odražený signál. Oblasti využití senzorů v optickém vlákně zahrnují například hydrofonii (podvodní akustické snímání), sledování teploty, tlaku a zátěže, optický vláknový gyroskop a chemický a biomedicínský průmysly. [7]

1.3 Zasazení problému do kontextu

Tato práce se dotýká hned několika oborů. V první řadě je to analýza a zpracování signálu. Před nástupem strojového učení se pro analýzu signálu užívala řáda nástrojů. Mezi nimi lze vyjmenovat časovou analýzu, kdy se porovnává výška a šířka časového kroku, statisticky významných rysů a jiných vizuálně patrných charakteristik; frekvenční analýzu pomocí Fourierové a wavelet transformací vhodnou v případech, kdy důležitá je amplituda; autoregresní modely odvádějící hodnotu prvku časové řady jako funkci několika předchozích. [1]

Metodiky klasifikace zvuků nebo, obecněji, signálů lze rozdělit do dvou obsáhlých kategorií. První, diskriminativní klasifikátory, zahrnují k-means a polynomiální klasifikátory, vícevrstvý perceptron (neuronovou síť) a podpůrné vektorové stroje [2]. Princip takových klasifikátorů spočívá v určení hranic tříd v trénovacích datech. Pak se testovací data zařazují do konkrétních tříd na základě těchto hranic. Do druhé kategorie nediskriminativních klasifikátorů spadá Skrytý Markovův model (HMM). Klasifikátory této kategorie spoléhají na modelování rozdělení trénovacích dat pomocí Markovova řetězce. HMM dovoluje zohlednit rozvoj časové řady v podobě přechodů mezi stavy a zároveň zachovat určitou míru adaptivnosti (např. vzorek, který se mezi trénovacími daty vyskytoval pouze na začátku posloupnosti se správně detekuje a klasifikuje na konci posloupnosti z testovacích dat). Díky těmto kvalitám se HMM považuje za vhodný pro zpracování signálů nestacionární povahy [8] a je již po několik desetiletí uznáván za jednu z nejúspěšnějších metod pro rozpoznávání hlasu. Proto lze předpokládat, že se tento nástroj dá aplikovat i na podobnou úlohu, klasifikaci signálů [9]. Příklad využití HMM k řešení úlohy klasifikace zvuku lze najít v [8]. Pro daný projekt byla zvolena klasifikace pomocí neuronové sítě. Výhodou tohoto řešení je možnost čerpat z rozsáhlé praxe klasifikace zvuků, k níž se často využívají právě neuronové sítě. Za nevýhodu lze, v porovnání s jinými vyjmenovanými přístupy, označit vysokou náročnost na výpočetní zdroje. [10]

Kapitola 2
Teoretická část

2.1 Volba vhodného nástroje

Jako prostředek pro uskutečnění klasifikace se v rámci této práce nakonec zvolily neuronové sítě. Ačkoliv je využití strojového učení v mnoha oblastech na vzestupu (např. počítačové vidění, zpracování přirozených jazyků, analýza řeči a zvuků [10] či predikční analýza), v oblasti zpracování signálů ještě není tolik rozšířeno. Přitom, co se týče reprezentace dat a formulace úloh, rozpoznání zvuků představuje vůči klasifikaci signálu poměrně blízký problém [1] a řešení z příbuzných oborů by mohla být poměrně snadno aplikovatelná také v kontextu klasifikace signálů obecně a signálů v optických senzorových sítích, které je věnovaná tato práce.

Z různých typů úloh řešených prostřednictvím strojového učení, lze řešený problém v této práci zařadit do kategorie učení s učitelem (supervised learning), protože v momentě učení sítě jsou již známé třídy a jsou k nim přiřazené jednotlivé vzorky z kolekce trénovacích dat. Učení s učitelem je u klasifikačních úloh typickým přístupem. [11]

Z hlediska volby vhodného modelu neuronové sítě existuje několik možností. Na vstupu klasifikace data mají podobu digitálního signálu, tj. sekvence hodnot získaných vzorkováním měřeného analogového signálu s pravidelnou frekvencí. Délka těchto sekvencí se může v různých vzorcích značně lišit. V takovýchto případech se doporučuje použití rekurentního modelu neuronové sítě. Takový model uplatňuje princip sdílení parametrů mezi jednotlivými součástmi modelu tak, že každý člen výstupu je funkcí nad předchozími členy a každý takový člen se produkuje stejným způsobem jako předchozí. Sdílené parametry tedy nese samotná opakovaně aplikovaná funkce. Tento způsob dovoluje přijímat na vstup posloupnosti různé délky. [11]

Ačkoliv jsou vhodné pro detekci globálně se opakujících vzorků v celé posloupnosti, v případě delších sekvencí a proudových dat se prospěšnost jejich využití zpochybňuje [1]. Důvodem je zbytečně "nekonečná paměť" rekurentních sítí, která v praxi není jednoznačně přínosná [12][13]. Alternativu rekurentním sítím může nabídnout konvoluční nebo hybridní přístupy spojující prvky obou těchto návrhů [1].

Konvoluční sítě se používají ke zpracování dat, u kterých se očekává mřížková topologie. Příkladem takovýchto dat jsou obrázky tvořené dvourozměrnou mřížkou pixelů nebo také časová řada se vzorky pořízenými v pravidelných časových intervalech. Sítě tohoto druhu používají na alespoň jedné z vrstev matematickou operaci konvoluce místo obecného násobení celé matice vstupu celou maticí parametrů. Dalším charakteristickým rysem daného druhu je, že na rozdíl od klasického vícevrstvého perceptronu, kde dochází k interakci každého vstupního uzlu s každým výstupním, konvoluční sítě zpravidla implementují řídké interakce (sparse interactions). To znamená, že vstupní matice se násobí nikoliv maticí parametrů, jejíž velikost odpovídá počtu všech výstupních uzlů, ale jadrem konvoluce (kernel), které může být mnohonásobně menší. Tímto omezením počtu propojení lze dosáhnout nižší náročnosti na paměť. [11]

Jinou výhodou konvolučních sítí je jejich adaptivnost neboli prizpusobivost k rozdílům v umístění vzorku ve vstupních datech [2]. Tato vlastnost je činí vhodným ná-

strojem v situaci, kdy se charakteristický a podstatný rys může vyskytovat na různých místech časové osy vstupu.

2.2 Transformace vstupních dat

Liší se také názory na to, zda má být signál před vstupem do neuronové sítě zpracován nebo transformován. Na jednu stranu předběžná transformace vede k tomu, že výsledky už nezávisí pouze na správném nastavení modelu. Ovlivňuje je rovněž správnost návrhu těchto transformací [1]. Na druhou stranu existují studie, které se v podstatné míře spoléhají na zpracování dat jinými metodami než strojovým učením. Příkladem tohoto přístupu v praxi může sloužit projekt STORM, v rámci kterého zvuk před vstupem do klasifikační neuronové sítě nejdříve procházel odšumováním pomocí vlnek (wavelet), extrakcí určitých rysů vzorků a jejich následným modelováním prostřednictvím směsi gaussových rozložení (GMM) za účelem zachycení jejich změny v čase [10].

Předběžné zpracování signálu v podobě extrakce charakteristických rysů před jeho klasifikací se využívá také v oblasti rozpoznávání hlasu. Populární metodou je například extrakce Mel-kepstrálních koeficientů (MFCC). [8]

Mimoto, než se data dostanou na vstup neuronové sítě a to jak při učení, tak predikci, musí projít řadou dalších transformací spojených s nutností zpracování do podoby odpovídající požadavkům na vstupní data vyplývajícím z typu sítě (podvzorkování, padding, normalizace), změnou podmínek experimentu (přemapování událostí) nebo jejich převedením do zcela jiné formy (fourierova transformace).

2.3 Metrika

Všechny dosažené výsledky během jednotlivých experimentů s využitím neuronových sítí se budou hodnotit a porovnávat na základě přesnosti (accuracy[1]). Je to obvyklý způsob hodnocení výkonu strojového učení v klasifikačních úlohách. Přesnost v tomto významu je poměrem počtu příkladů se správným výstupem k počtu všech příkladů. Ekvivalentní informaci lze získat také měřením chybovosti (error rate), která udává naopak poměr chybných výstupů ke všem vstupům. [11]

V rámci této práce se ve stejné míře sleduje přesnost klasifikace událostí všech tříd. V případě reálného využití výsledků tohoto projektu však může nastat situace, kdy se v závislosti na tom co skutečně způsobilo anomálii a jak byla klasifikována, mění dopad takové chyby. Například v kontextu ochranných systému může mít špatná klasifikace nebezpečné události větší následky než opačná možnost. Při nasazení v podmínkách kdy jsou klasifikační chyby nerovnocenné, je u klasifikace tříd kterých se to týká přínosné, porovnávat taktéž jejich úplnost (recall). Jedná se o jinou metriku, která udává poměr počtu správně zařazených vzorků do jedné třídy ke všem do ní zařazeným vzorkům. [11]

2.4 Systém

Tato část práce je věnována popisu možných návrhů prototypu komponent pro učení a klasifikaci, který bude implementován v rámci této práce. Rovněž bude představen i návrh celého systému, do kterého klasifikátor může být zapojen v produkčních podmínkách. Budou definovány požadavky a ukázana konkrétní architektonická rozhodnutí v podobě 4+1 pohledů.

[1] Anglický pojem se zde uvádí pro to, aby nedocházelo k zaměňování s další možnou metrikou - precision, která také bývá překládaná jako „přesnost“ [14].

2.4.1 Funkční požadavky

Navrhovaný systém bude mít dva režimy:

1. V režimu učení dostane na vstupu předem označkované vzorky signálů s příklady anomálií různých tříd (trénovací data). Určité třídy uživatel označí za vyžadující akci a rovněž definuje konkrétní akce.
2. V provozním režimu detekce a klasifikace bude systém sbírat data z čidel propojených s optickým vláknem nebo jiným zdrojem signálového průběhu. Na základě těchto dat se určí, zda obsahují náznaky anomálií. Detekované anomálie se klasifikují a v případě jejich zařazení do jedné z tříd vyžadujících spuštění akce bude tato akce spuštěna.

2.4.2 Nefunkční požadavky

1. Navržený systém musí být spustitelný jak v cloudu[1], tak na hardwaru uživatele.
2. Systém musí po nastavení a načtení trénovacích dat na jejich základě poskytnout informace o dosažitelné přesnosti klasifikace anomálií jednotlivých tříd.
3. Akce vyžadované v případě detekce a klasifikace určitých tříd a třídy samotné budou definované uživatelem.

2.4.3 Volba technologií

Na základě uvedených požadavků a vybraného nástroje (viz 2.1 „Volba vhodného nástroje“) se rozhodlo o použití programovacího jazyka Python s knihovnou TensorFlow. TensorFlow se v průběhu práce na klasifikátoru jevil jako dobře zdokumentovaný a celosvětově populární nástroj pro vývoj softwaru využívajícího strojové učení [16]. TensorFlow je také kompatibilní s Google AI Platform.

Dalším důvodem pro použití této knihovny bylo to, že nabízí API pro dva programovací jazyky: Python a C++. Python díky své relativní jednoduchosti a rychlosti vývoje dovoluje flexibilní práci na prototypu. Druhý jazyk, C++, naopak nabízí možnosti pro optimalizaci a tudíž výhodnější a rychlejší běh aplikace v produkčních podmínkách. V rámci této práce se implementace omezí na první variantu, tj. prototyp připravený v jazyce Python. Nicméně použití společné oběma jazykům knihovny má dovolit jednodušší přechod z prototypu k plnohodnotnému systému pro detekci a klasifikaci události v signálu v porovnání s využitím jiných technologických prostředků.

Python slučuje univerzalitu všeobecného multiparadigmatického programovacího jazyka a jednoduchost řešení konkrétních úloh, která je vlastní spíše jazykům se specifickou oblastí použití. Tento jazyk poskytuje širokou řadu knihoven pro načtení a vizualizaci dat a statistik (NumPy, SciPy, matplotlib a jiné). Vývoj nástroje pro strojové učení je ve své podstatě iterativním procesem, kdy rozhodování o dalším postupu závisí na analýze výstupu z předchozí iterace. Python je pro tyto účely vhodný právě díky jednoduchosti zápisu a okamžité interakci s uživatelem prostřednictvím příkazové řádky. [17]

Jelikož na vyvíjený prototyp není kladen požadavek na kompatibilitu s žádnými staršími moduly a knihovnami, se pro projekt použil v souladu s postupem Müllera a Guidové [17] Python verze 3 (3.5.8 pro kompatibilitu s Google Cloud AI Platform viz [18]).

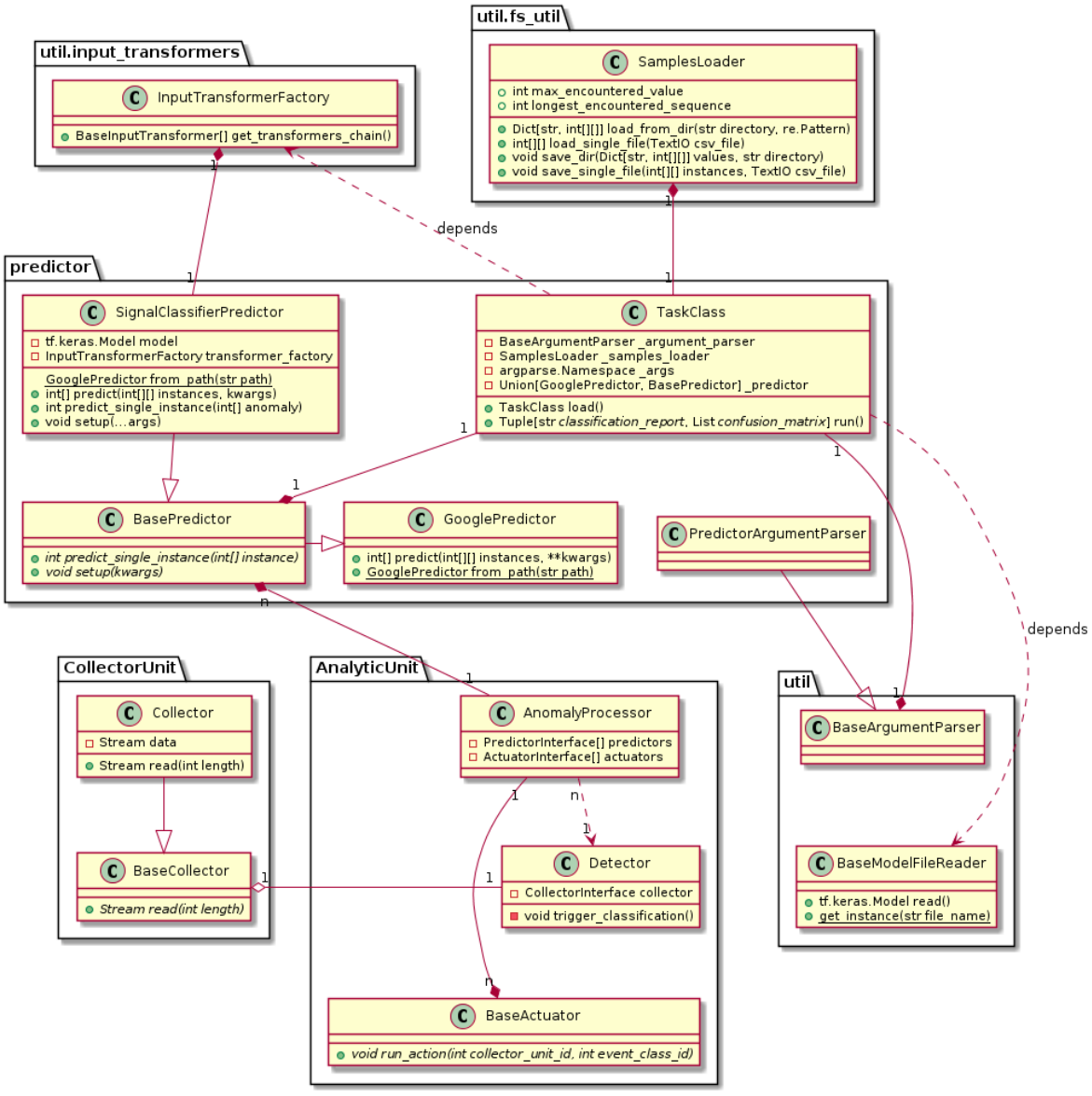
[1] Konkrétně je vyžadována podpora Google AI Platform (dříve Google ML Engine). Jedná se o cloudovou službu pro umístění trénovacích a predikčních algoritmů, zejména na základě TensorFlow [15]

2.5 4+1 pohledy

Softwarová architektura může být nahlížená z různých perspektiv, nazývaných též „pohledy.“ Model 4+1 pohledů byl zaveden Kruchtenem v roce 1995 [19]. Znázorňuje navrhovaný systém:

- logickým pohledem odrážejícím design objektového modelu,
- procesním, který zdůrazňuje dynamické aspekty systému,
- fyzickým představujícím mapování softwarových komponent na hardware
- vývojovým pohledem, který se zaobírá statickou organizaci software ve vývojovém prostředí.

2.5.1 Logický pohled



Obrázek 2.1. Hlavní diagram tříd celého systému pro predikci (zdroj: vlastní data).

Na obr. 2.1 je znazorněna základní struktura tříd nutných k provedení sběru dat, detekce a predikce. Zohledňuje tři možné způsoby, kterými může dojít k jejímu spuštění v závislosti na zvolené metodě použití:

- samostatné spuštění modulu `predictor.task` na lokálních datech přes řídící třídu `TaskClass`, která nastaví a spustí instanci `SignalClassifierPredictor`;
- spuštění přes API služby Google Cloud AI Platform přímými voláními metod třídy `SignalClassifierPredictor`;
- spuštění po aktivaci detektorem anomálií v rámci řídící třídy `AnomalyProcessor` při nasazení v rámci systému zpracování anomálií v reálném čase.

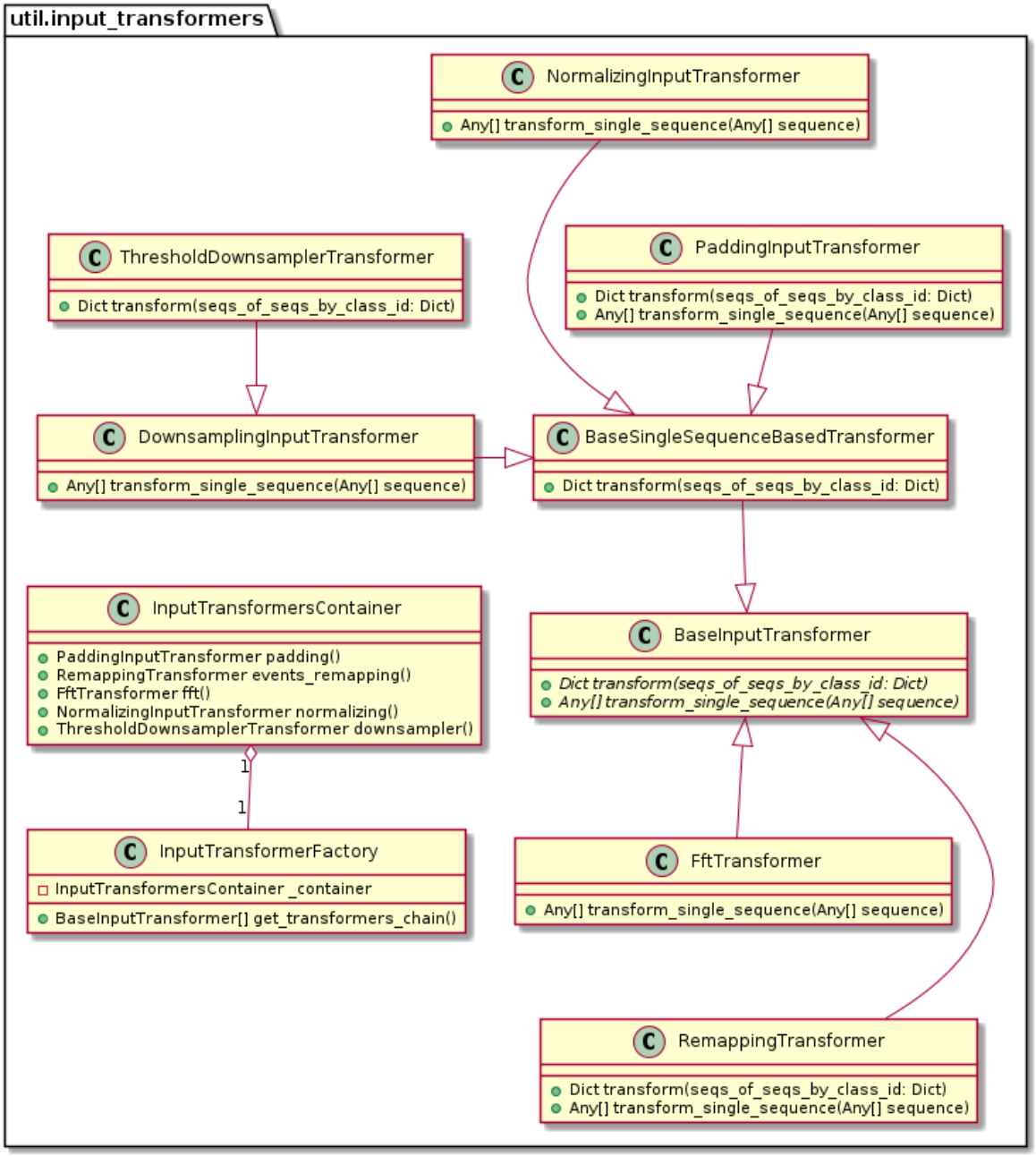
První z vyjmenovaných způsobů výše je vhodný hlavně při počátečním nastavení a testech. Tento postup se však dá použít také i v rámci systému zpracování anomálií a to za předpokladu, že modul predikce se použije jako samostatný proces. Je to například možné pokud se pro implementaci zvolí architektura microservices, kdy jednotlivé části systému budou mít podobu autonomních aplikací [20].

Kromě samotné třídy prediktoru řídící třída k svému běhu využívá instance následujících pomocných tříd modulu `util`:

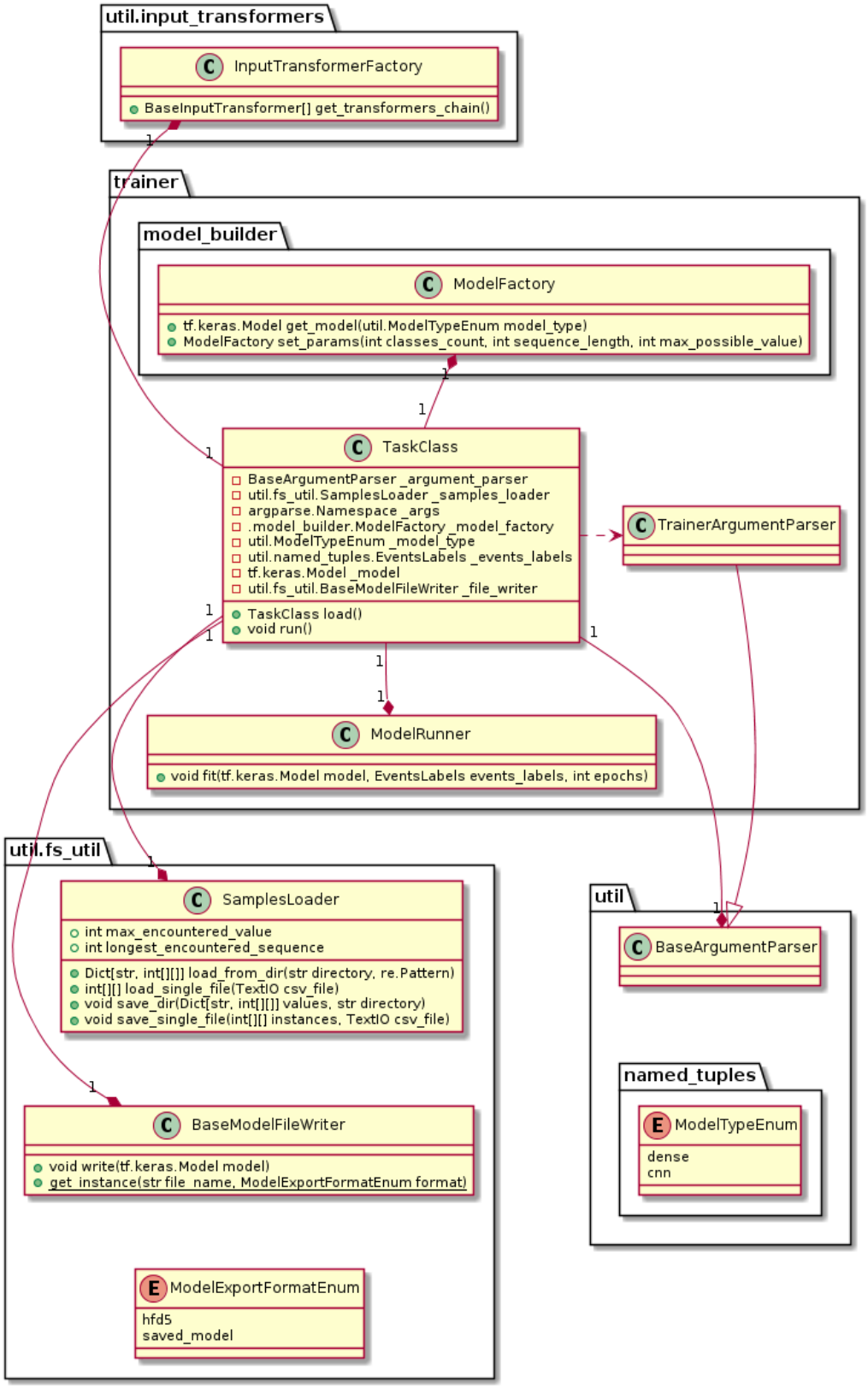
- `SamplesLoader` - zajišťuje načítání dat ze souborů a také stanovení základních údajů o načítaných datech - největší hodnoty a maximální délky posloupnosti. Největší hodnota je nutná pro normalizační transformátor (viz sekce 3.2.5) a maximální délka pro podvzorkovací (viz sekce 3.2.4).
- `BaseArgumentParser` a jeho konkrétní implementaci `PredictorArgumentParser` - slouží k získání a kontrole argumentů, se kterými byl spuštěn program.
- `InputTransformerFactory` se na tomto místě použije kvůli zjednodušení přípravy experimentů ve fázi prototypování a slouží k vytvoření řetězce s jedním transformátorem — `RemappingTransformer`, sloužícím k přemapování tříd (podrobněji viz 3.2.1). Je to jediný transformátor, který nelze spustit později v rámci běhu třídy samotného prediktoru, protože prediktor je navržen také pro predikci dosud neznámých dat a v takovém režimu nedostane na vstupu označkovaná data. Přiřazení vzorků k jednotlivým třídám se tady používá pouze pro vyhodnocení výsledků predikce na předem klasifikovaných datech.

Zbývající práci odvádí třída `SignalClassifierPredictor` založená na abstraktní třídě `BasePredictor` definující rozhraní používané uvnitř aplikace. Tato třída dědí `GooglePredictor`, zastupující interface vymáhaný službou Google Cloud AI Platform. Tady probíhá zbývající transformace dat (třídy transformátorů jsou znázorněné na obr. 2.2, podrobněji se jednotlivé transformátory probírají v podkapitole 3.2), načtení (pomocí jednoho z potomku `BaseModelFileReader`) a spuštění TensorFlow modelu.

Výsledky predikce v podobě čísla predikované třídy pro každý vzorek ze vstupu se dostávají zpátky do `TaskClass`, kde se formátují do podoby zprávy o klasifikaci a matici záměn a následně se odešlou na výstup. V případě, že je nastavená cesta pro ukládání logu, zmíněná zpráva o klasifikaci se zde také uloží.



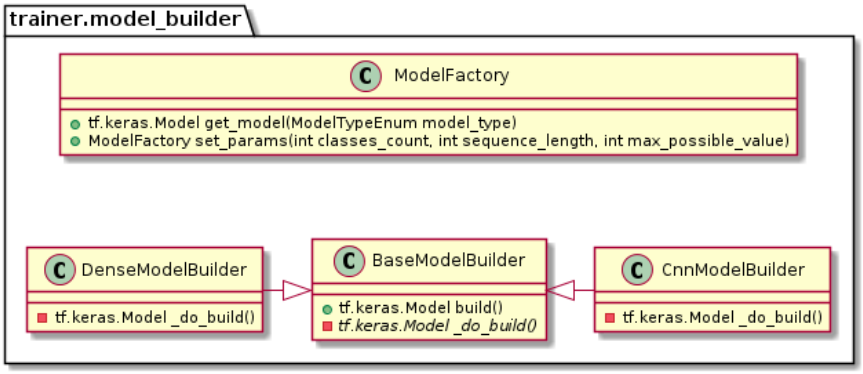
Obrázek 2.2. Diagram tříd znázorňující modul `util.intput_transfomers` obsahující třídy samotných transformátorů vstupu a také třídy spojené s vytvořením jejich instancí - `InputTransformerFactory` a `InputTransformersContainer` (zdroj: vlastní data).



Obrázek 2.3. Diagram tříd znázorňující závislosti `trainer.TaskClass`, řídící třídy pro učení (zdroj: vlastní data).

Na rozdíl od predikce, učení se spouští jedině pomocí řídící třídy `TaskClass` z modulu `trainer` (Obr. 2.3). Tento modul pomocí souboru `trainer/task.py` se spouští i při učení ve službě Google Cloud AI Platform. Stejně jako u predikce se používají transformátory a třída `InputTransformerFactory` s tím rozdílem, že se při učení do každého z transformátorů předávají všechna data včetně přiřazených tříd. Proto vstup ve stejném řetězci může být zpracován také transformátorem přemapování tříd (`RemappingTransformer`).

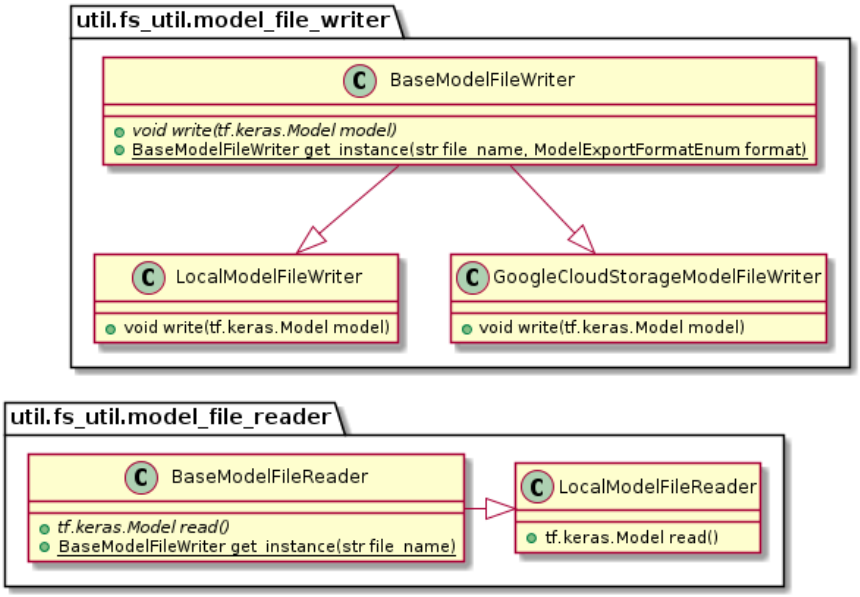
Jinou třídou, kterou využívá jak komponenta učení, tak komponenta predikce je `SamplesLoader`. Stejně jako předtím plní funkci načtení vzorků ze souborů, jejich přiřazení do jednotlivých klasifikačních tříd a zjištění základních atributů vstupu.



Obrázek 2.4. Diagram tříd znázorňující modul `trainer.model_builder` (zdroj: vlastní data).

Potom se pomocí třídy `ModelFactory` modulu `trainer.model_builder` vytvoří model jednoho ze dvou podporovaných typů - hustá plně propojená síť (vícevrstvý perceptron) pokud byla hodnota argumentu `--model-type` nastavena na `dense` a konvoluční síť v případě, že byla hodnota argumentu `--model-type` nastavena na `cnn` (pro porovnání a popis obou použitých typu sítí viz 3.1). Propojení a struktura tříd modulu `trainer.model_builder` je znázorněna na obr. 2.4.

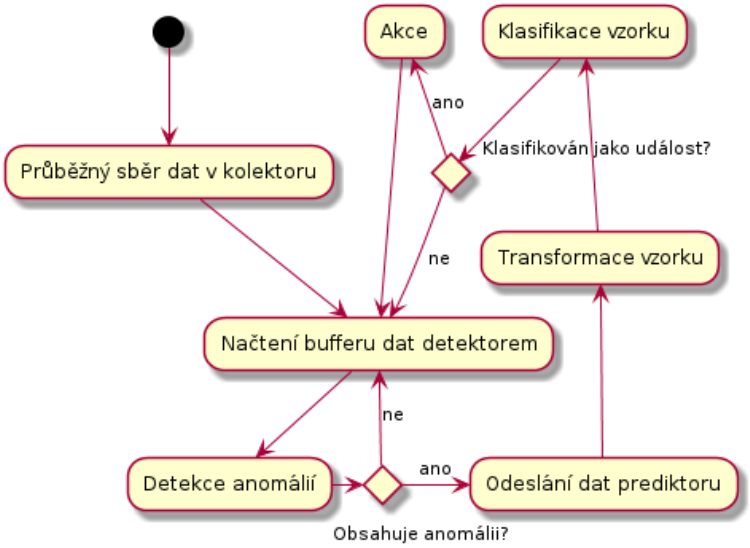
Na základě zpracovaných dat a připraveného modelu se spouští predikce řízena třídou `ModelRunner` odpovídající mimo jiné také za nastavení logování učení do TensorBoard formátu. Z `ModelRunner` se naučený model vrací do `TaskClass` a následně se ukládá do lokálního nebo vzdáleného souborového systému prostřednictvím potomka `BaseModelFileWriter`, implementujícího navrhový vzor `Factory method` [21] svojí statickou metodou `get_instance` (obr. 2.5).



Obrázek 2.5. Diagram tříd znázorňující části modulu `util.fs_util`: třídy pro ukládání do souborového systému a načítání modelů z něj (zdroj: vlastní data).

2.5.2 Procesní pohled

Tato podkapitola má poskytnout procesní pohled v podobě diagramů aktivit a popisu pro tři hlavní procesy systému: predikci, učení a transformaci vstupních dat. Vzhledem k podstatným odlišnostem v nutných krocích u predikce při spuštění přes řídící třídu `TaskClass` a v rámci potenciálního produkčního zasazení do systému sledování a detekce událostí v reálném čase se tyto dvě možnosti proberou zvlášť.

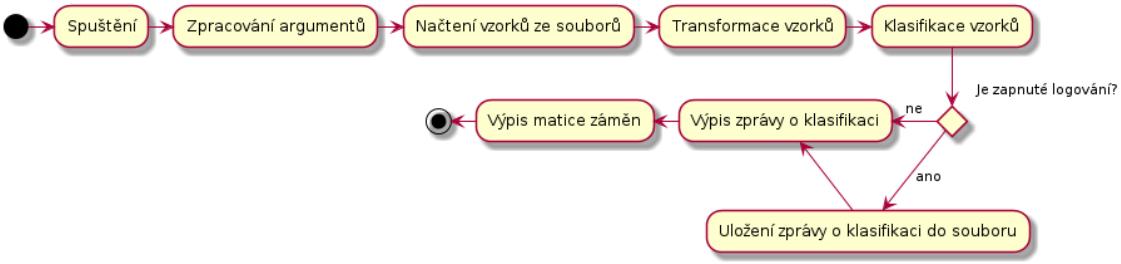


Obrázek 2.6. Diagram aktivit predikce při produkčním nasazení (zdroj: vlastní data).

U produkčního nasazení prediktoru (obr. 2.6) se předpokládá existence komponent pro sběr dat a detekci anomálií zajišťující první kroky iterativního procesu a také centrální analytické komponenty, odpovídající za předání vstupů a výstupů mezi jednot-

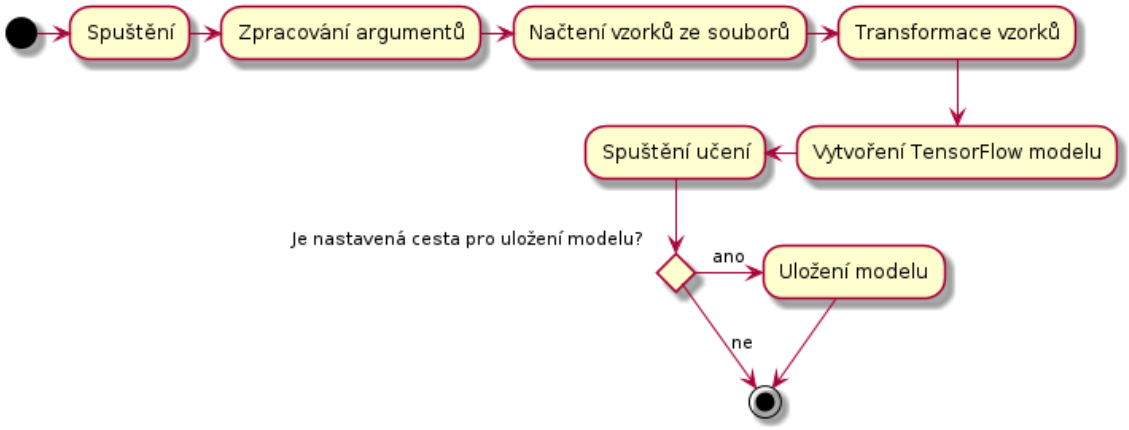
livými částmi systému. Kolektor průběžně sbírá data, která se proudově odesílají do detektoru, kde se analyzují po částech. Je důležité, aby se tyto části překrývaly. Cílem tohoto překryvu je zamezit situaci, kdy dojde k rozdělení jedné události, která by jinak měla být detekována, na dva úseky, které se zvlášť od sebe už nedetekují [10]. Tento proces se opakuje dokud se nesplní podmínka detekce anomálie.

V případě detekování anomálie tímto způsobem potenciální událost v signálovém průběhu se má vyjmout z celého vzorku a odeslat dál do prediktoru. V něm dochází ke zpracování vstupních dat tak, aby odpovídala vstupním parametrům neuronové sítě. Dalším krokem je samotná klasifikace. Na jejím základě se rozhodne, zda se jedná o událost, která má vyvolat následnou akci (např. odeslání notifikace zaujaté straně nebo spuštění alarmu, podle využití systému). Pokud byl vzorek zařazen do jedné z takových tříd, dojde ke spuštění odpovídající akce a opakování cyklu, jinak se rovnou pokračuje na načtení další dávky dat z proudu z kolektoru.



Obrázek 2.7. Diagram aktivit predikce při spuštění řídící třídou (zdroj: vlastní data).

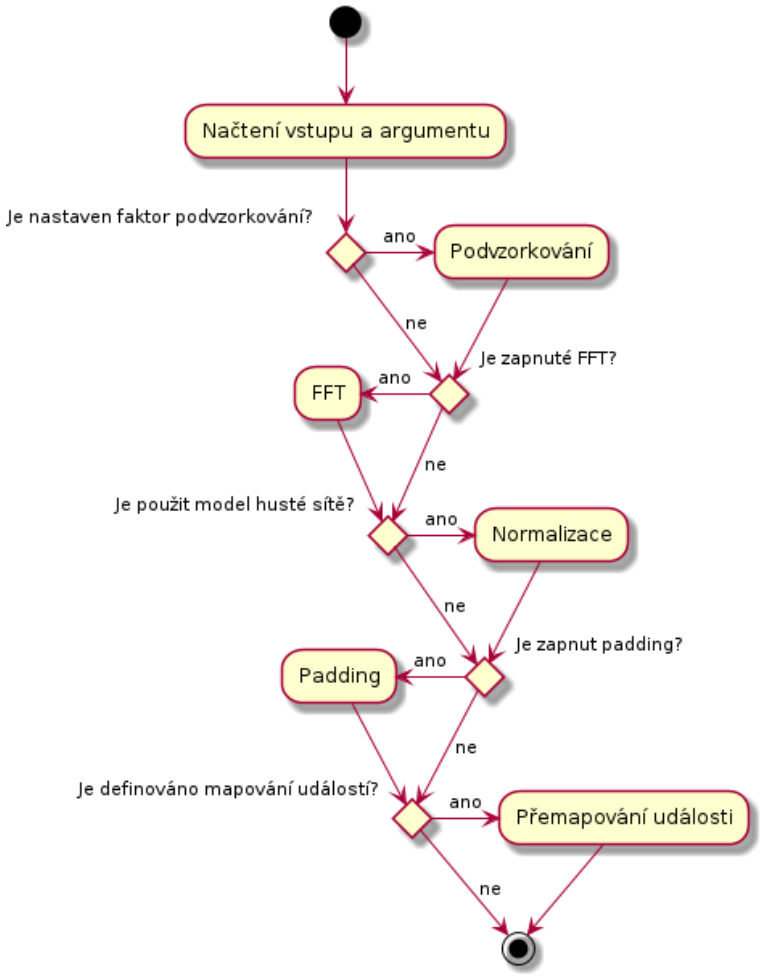
Při spuštění pomocí řídící třídy `TaskClass` (obr. 2.7), tj. primárně v případě predikce za účelem vyhodnocení přesnosti s předem označkovanými vzorky, se tento proces liší. Zdrojem dat je v této verzi lokální souborový systém. Ze souborů obsahujících vzorky a roztříděných podle událostí se načtou jak samy vzorky, tak jejich třídy pro vyhodnocení výsledků klasifikace. Stejně jako v případě začlenění klasifikace do širšího systému se zde provede transformace vstupních dat. Jediným rozdílem je zapojení do řetězce transformátorů také transformátoru přemapování tříd (viz 3.2.1). Následuje klasifikace a pokud mezi argumenty příkazu pro spuštění modulu byla uvedená cesta pro ukládání logů, práva o klasifikaci se uloží do souboru. Tato zpráva obsahuje údaje o přesností (precision), úplnosti (recall) a hodnotě kombinované metriky F1-score pro jednotlivé třídy, průměr a vážený průměr pro všechny třídy (podle počtu vzorků, kterými je každá třída zastoupená ve vstupních datech). Ještě poskytuje další souhrnný údaj, a to přesnost (accuracy), spočítanou pro všechny zastoupené třídy. Bez ohledu na to, zda byla zpráva uložena se vypíše na standardní výstup společně s maticí záměn.



Obrázek 2.8. Diagram aktivit učení (zdroj: vlastní data).

Proces učení (obr 2.8) začíná podobně jako predikce v případě spuštění přes řídící třídu `TaskClass`. Nejdříve se načítají argumenty, pak označkované vzorky ze souborového systému. Potom se na základě nastavení předaných přes argumenty a základních údajů o datech vytvoří model neuronové sítě na kterém se spouští učení.

Jiný proces, který by se dal zobrazit na diagramu aktivit je již zmíněná transformace vstupních dat, která probíhá jak při učení, tak při predikci. Na obr. 2.9 lze vidět v jaké posloupnosti a s jakými podmínkami se aplikují transformátory (popis jednotlivých transformaci viz 3.2).

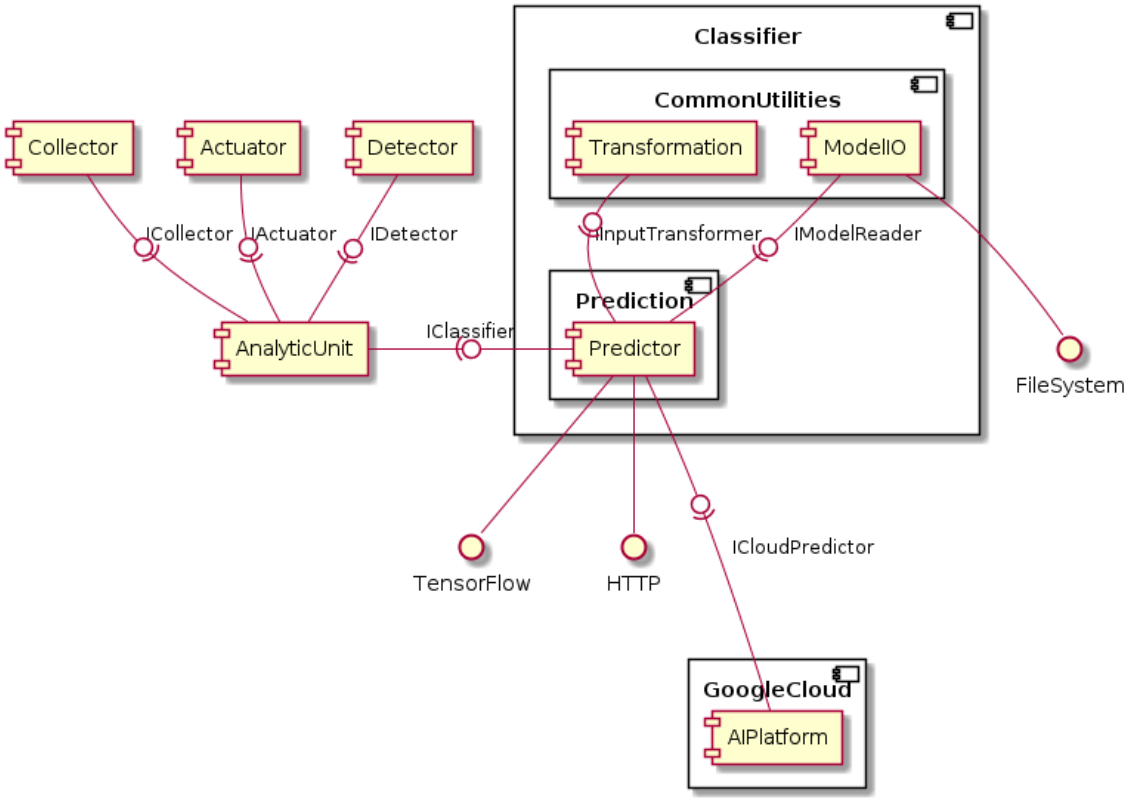


Obrázek 2.9. Diagram aktivit transformačního řetězce včetně přemapování událostí (zdroj: vlastní data).

Řetězec transformátorů se skládá na základě argumentů, se kterými byl spuštěn proces. Transformátory jsou seřazené způsobem, který dovoluje každému následujícímu transformátoru na vstupu využít výstup z předchozího.

2.5.3 Vývojový pohled

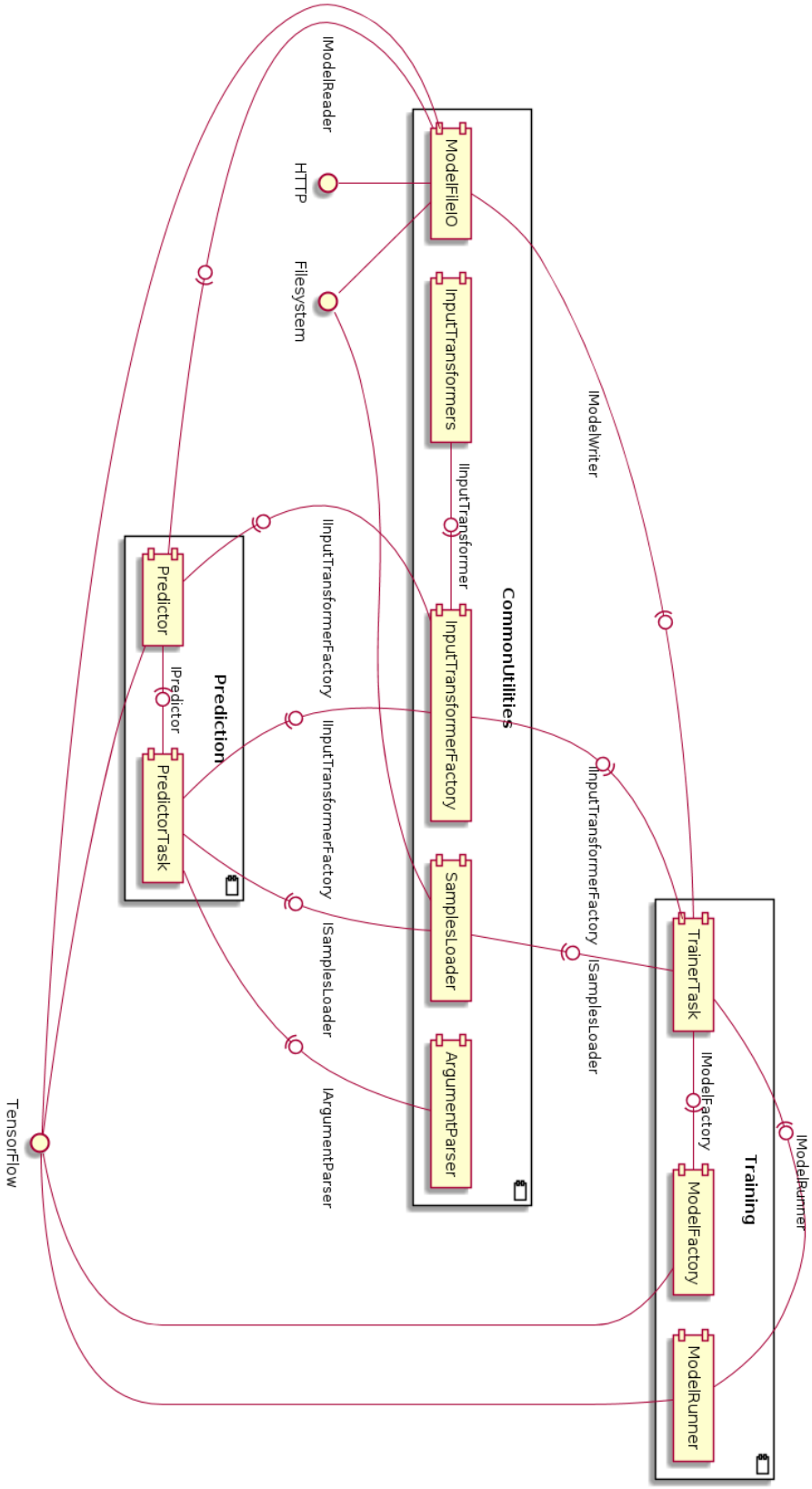
Z hlediska logického rozdělení systému detekce a klasifikace události v signálovém průběhu reálného času lze vyčlenit prvky označené na diagramu komponent na obr. 2.10. Hlavní komponentou, která řídí běh systému, je `AnalyticUnit`, která využívá kolektor, aktuator, detektor a klasifikátor, který je ve středu pozornosti této práce. Klasifikátor se z pohledu tohoto využití skládá ze dvou komponent: `Prediction` a `CommonUtilities`. První poskytuje interface klasifikátoru `IClassifier` a `ICloudPredictor` pro GoogleCloud AI Platform, zatímco druhá sjednocuje prvky využívané také učením. Jsou to transformátory a utility pro práci se souborovým systémem.



Obrázek 2.10. Základní diagram komponent z pohledu celého systému detekce a klasifikace událostí v signálovém průběhu. (zdroj: vlastní data).

Na dalším diagramu komponent (obr. 2.11) je znázorněn pouze klasifikátor včetně modulů pro predikci a učení. Mezi nimi se nachází komponenta `CommonUtilities` zmíněna v popisu předchozího diagramu, která tady obsahuje všechny části sdílené oběma komponentami: načtení a uložení TensorFlow modelu (`ModelFileIO`), transformátory (`InputTransformers` a `InputTranformerFactory`), načtení vzorků ze souborového systému (`SamplesLoader`) a načtení argumentů (`ArgumentParser`).

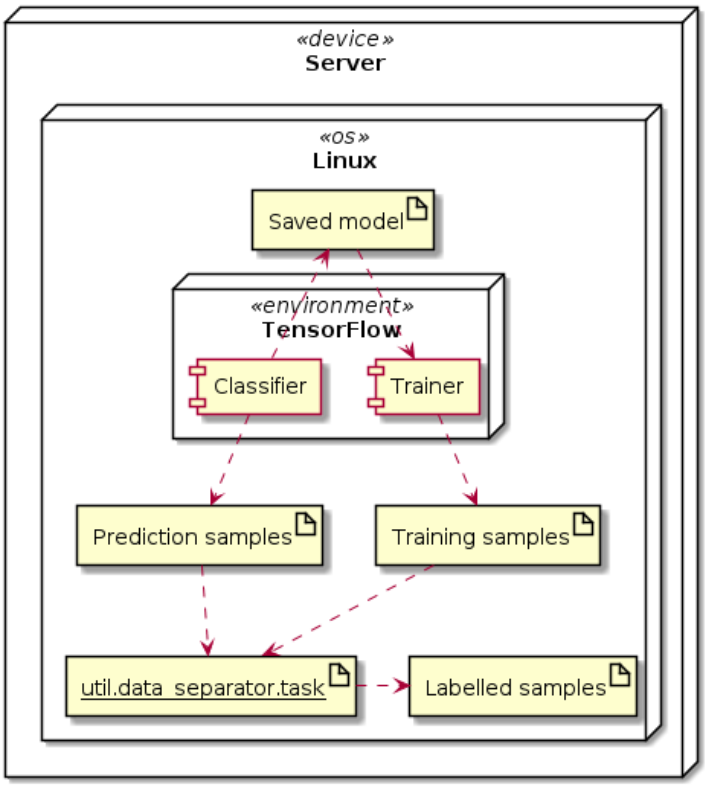
Na diagramu jsou rovněž vyznačené závislosti jednotlivých komponent na prostředí: spojení s internetem prostřednictvím HTTP protokolu (je nutné pro komunikaci s Google Cloud), souborový systém a TensorFlow pro práci s neuronovou sítí.



Obrázek 2.11. Diagram komponent klasifikátoru (zdroj: vlastní data).

2.5.4 Fyzický pohled

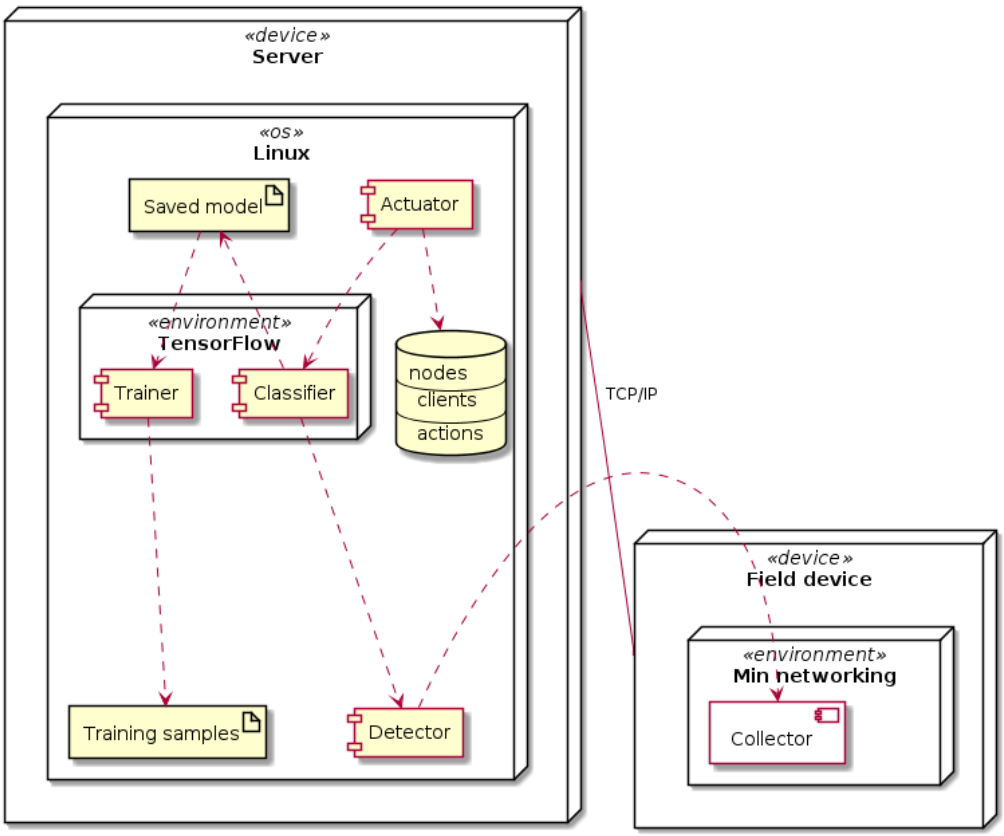
První fyzický pohled je věnován nasazení prototypu na jednom lokálním počítači (viz obr. 2.12). Je to nejjednodušší možný kontext, ve kterém se dá použít klasifikátor. Tento kontext pak poslouží jako základ pro nasazení a propojení prvků celého systému. Všechny prvky jsou umístěné na stejném serveru na kterém běží operační systém poskytující prostředí TensorFlow pro spuštění modulu systému. Předpokladem pro zprovoznění této konfigurace je, že vzorky označkované odpovídajícími kategoriemi jsou uložené na stejném počítači. Tyto vzorky slouží jako vstup do modulu `util.data_separator.task`, který vzorky rozdělí v zadaném poměru na trénovací a testovací. Trénovací vzorky dále pokračují na vstup do komponenty učení, která vytvoří model. Klasifikátor využívá tento model pro následné zpracování testovacích vzorků.



Obrázek 2.12. Diagram nasazení prototypu klasifikátoru na jednom počítači (zdroj: vlastní data).

Širší systém pro detekci a klasifikaci události v signálovém průběhu reálného času, do kterého může být zasazen klasifikátor, lze z fyzického pohledu umístění komponent nasadit na produkci několika základními způsoby. Každý z nich má odlišné výhody a nevýhody a je jinak náročný na zdroje. Na následujícím diagramu (obr. 2.13) je znázorněn návrh také využívající jeden server, ale s ohledem na produkční požadavky: k dispozici jsou pouze trénovací data, vzorky pro klasifikaci se berou z detektoru a po klasifikaci vzorku se má spustit definovaná akce.

V tomto případě dochází k následující posloupnosti procesů. Nejdříve se na serveru pomocí trénovací komponenty na základě trénovacích označkovaných vzorků vytváří model neuronové sítě. Z připojených zařízení, u nichž je vyžadováno minimální prostředí nutné pro implementaci komponenty kolektoru a odeslání sbíraných dat po síti



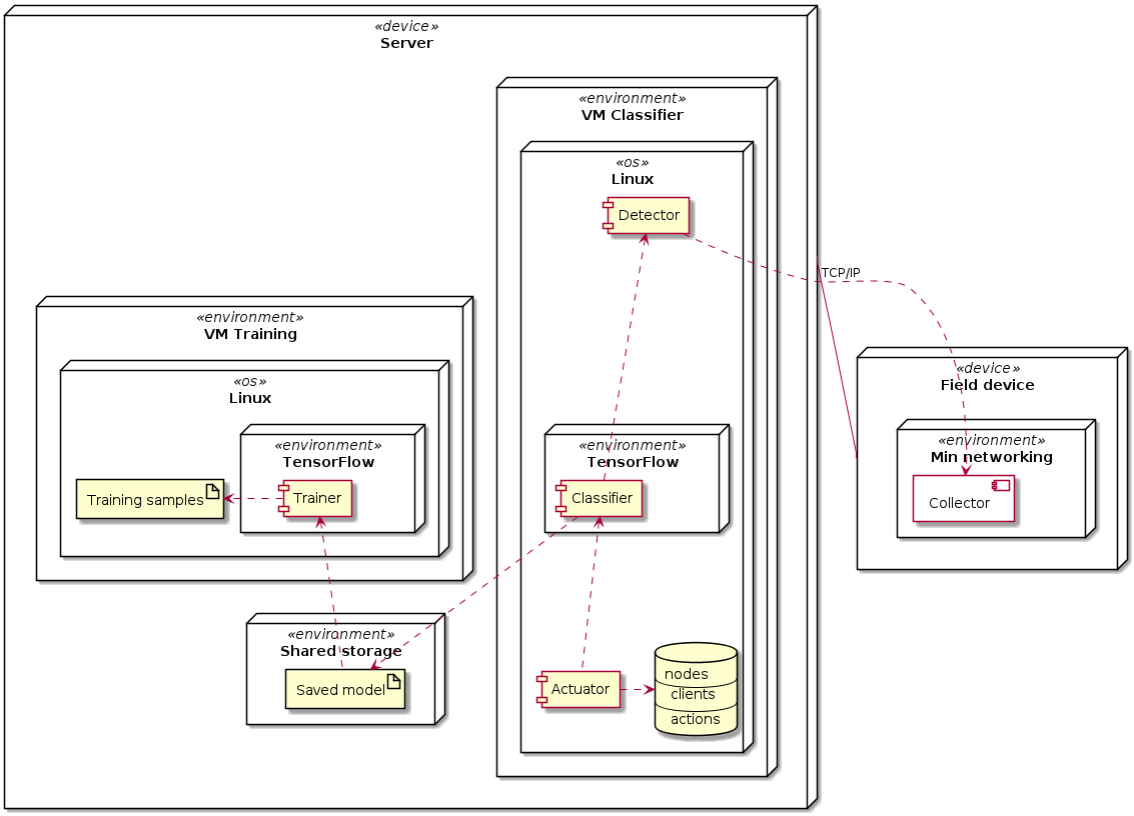
Obrázek 2.13. Diagram nasazení celého systému s jedním serverem (zdroj: vlastní data).

(nemusí tedy mít plnohodnotný operační systém), se na server dostává v podobě proudu dat signálový průběh. Pak komponenta detektoru zpracovává vstupní data a v případě detekce anomálií, ji vyjme z celého průběhu. Klasifikátor na základě hotového modelu a těchto vzorků určí třídu detekované události. Aktuátor s přístupem k databázi definovaných událostí, jednotek implementujících komponentu kolektoru a předepsaných akcí určí a spustí odpovídající určené třídě akci.

Z pohledu využívaných zdrojů za výhodu tohoto řešení lze považovat nízké požadavky na zařízení pro sběr dat. Jelikož není požadované spuštění žádného výpočetně významného algoritmu, stačí pro snímání hodnot ze senzoru a síťové rozhraní relativně jednoduchý programovatelný logický obvod napojený na analogově digitální převodník. Tento přístup však klade poměrně vysoké požadavky na síťové připojení k serveru. Jelikož ještě neproběhla detekce anomálií, musejí se poslat veškerá pořízená data. V případě stálého odesílání proudu dat se signálem s frekvencí v rozmezí od 0 do 7800 Hz (jak je tomu v případě dat, na kterých probíhá testování prototypu v této práci, viz příloha B) podle Shannonova teorému pro zachování kvalitního digitálního signálu s možností jeho obnovení je třeba využít vzorkovací frekvenci dvakrát vyšší, tj. 15600 Hz [22]. Pokud se data budou odesílat binárně v nekomprimované podobě a pro předání každé hodnoty se použije 8 bytů (např. pro typ „double“, číslo s plovoucí řádovou čárkou s dvojitou přesností), minimální vyžadovaná rychlost spojení činí přibližně 0,95 Mb/s. Tento údaj musí být při nasazení přepočítán s ohledem na skutečný možný rozsah frekvencí signálu.

Jiným „úzkým hrdlem“ je použití stejného serveru pro trénování sítě, detekci, klasifikaci a následné spuštění akce. Na jednu stranu to je nejjednodušší řešení pro předání modelu od trénovacího modulu ke klasifikačnímu – oba přistupují přímo k disku. Z

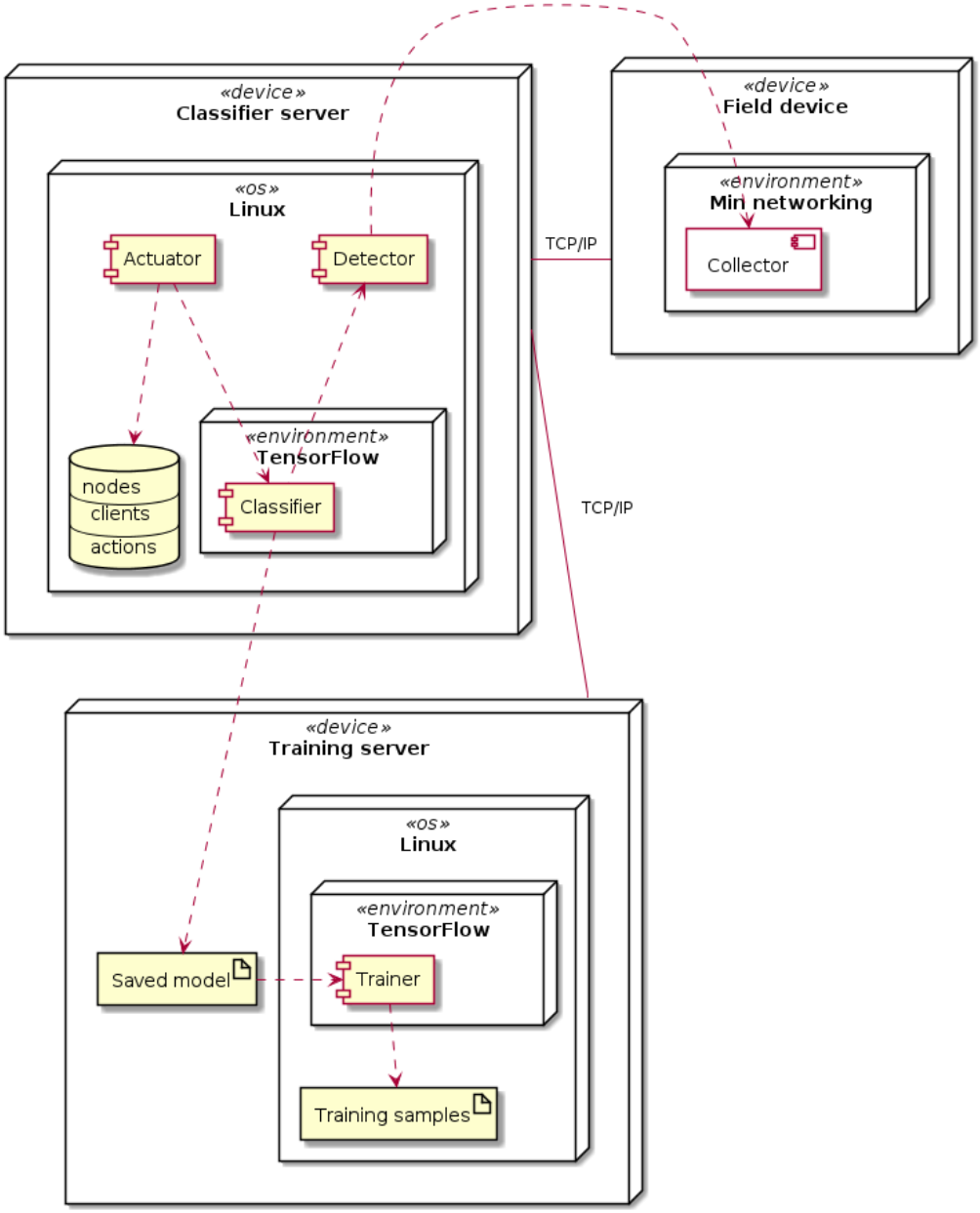
druhé strany ale takové umístění komponent a možnost spuštění zároveň několika procesů (např. aktualizace modelu na základě nových vzorků a zpracování dat z kolektoru) nevyhnutelně povede ke konkurenci procesů o systémové zdroje – paměť, procesor nebo přístup k disku. V případě nutnosti se takový server může škálovat pouze vertikálně, tj. nahrazením hardwaru například výkonnějšími analogy (CPU) nebo analogy většího objemu (RAM). Je také možné spuštění několika instancí detektorů a klasifikátorů, například pro každé připojené zařízení pro sběr dat.



Obrázek 2.14. Diagram nasazení systému s příkladem oddělení procesu do separátních virtuálních strojů (zdroj: vlastní data).

Rozdělení zdrojů mezi jednotlivými procesy systému lze ovlivnit nastavením priorit konkrétním procesům nebo omezit pomocí oddělení procesů do zvláštních kontejnerů nebo virtuálních strojů (viz obr. 2.14). Zachovávají se zde stejná „úzká hrdla“ jako v předchozím modelu: stále je to stejný stroj, který se spolehá na sdílené fyzické zdroje a stále se musí předávat veškerá data ze zařízení pro sběr dat. Znázorněný přístup však nejenom dovolí ohraničit možný vliv učení sítě na ostatní procesy, ale také v případě použití kontejnerů otevírá cestu horizontální škálovatelnosti, protože kontejnery s predikční částí lze replikovat v clusteru. Nicméně oddělení kontejnerů vyvolá také nutnost vzdáleného přístupu k aktuálnímu modelu neuronové sítě, se kterým tento návrh nepočítá.

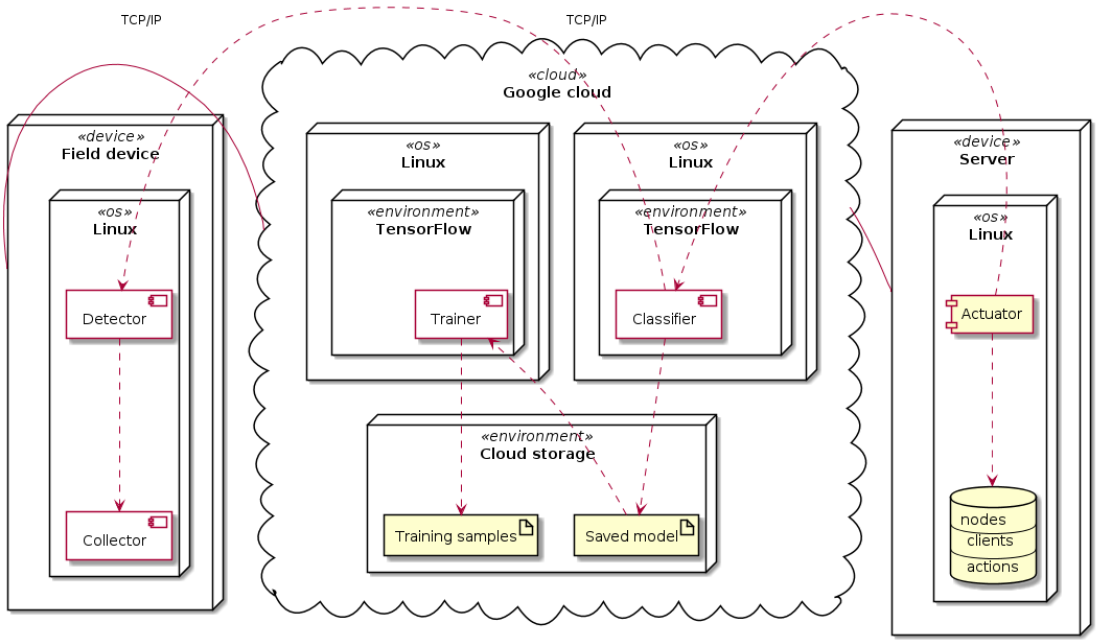
Vzdálený přístup k uloženému modelu řeší další možná konfigurace s oddělením učení a predikce na dva (nebo více) různých fyzických strojů (viz obr. 2.15). Tím je vyřešená jakákoliv konkurence o zdroje mezi predikci a učením a vzniká možnost neomezeného horizontálního škálování klasifikačních serveru (např. v podobě přidávání fyzických nebo dalších virtuálních strojů do clusteru). Optimalizovaný model se posílá ze serveru pro



Obrázek 2.15. Diagram nasazení systému s příkladem oddělení procesu do separátních fyzických strojů (zdroj: vlastní data).

učení po síti. Případné zahlcení serveru pro učení v momentě aktualizace modelu větším počtem klasifikačních serverů lze řešit vytvářením zrcadel nebo replik s možností čtení uloženého modelu. Tento přístup rovněž poskytuje cestu pro horizontální škálování.

Předchozí návrhy nechávaly v procesu zpracování vstupních dat jen nepatrnou roli samotným zařízením pro sběr dat. Kvůli tomu docházelo k jisté nutné míře centralizace procesů na serveru a vyšším požadavkům na spojení mezi ním a koncovým zařízením. Další navrhovaná konfigurace má naopak za cíl odlehčit roli serveru a maximálně přenechat zpracování službě Google Cloud AI Platform (viz obr. 2.16).



Obrázek 2.16. Diagram nasazení systému s detekcí na zařízení pro sběr dat, učením a klasifikací v cloudu (zdroj: vlastní data).

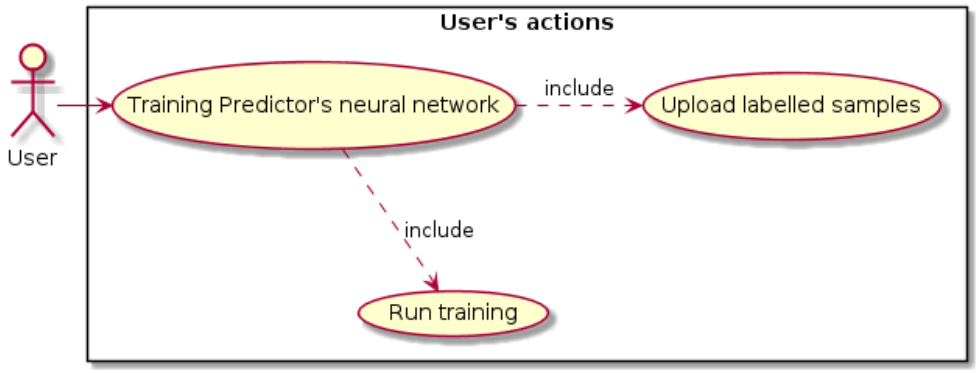
Zde se komponenta Detektor přenáší na stranu koncového zařízení pro pořízení dat. Takový přesun je možný s podmínkou, že na zařízení může běžet program pro detekci anomálií, což v závislosti na konkrétní implementaci může vyžadovat instalaci operačního systému na zařízení[1]. To znamená, že zařízení bude odesílat pouze detekované anomálie. Konkrétní požadavky na spojení tedy budou vyplývat z četnosti a velikosti detekovaných vzorků, ale dokud mezi detekovanými událostmi bude aspoň nějaký časový rozestup, objem dat pro přenos bude nižší než v případě stálého odeslání celého proudu.

Zbývající komponenty se v rámci daného návrhu přesouvají do cloudu. Přístup komponent pro učení a pro predikci k modelu řeší využití úložiště umístěného v cloudu (v případě Google AI Platform to může být Google Cloud Storage). Výsledek klasifikace může být v souladu s diagramem výše odeslán buď na server pro další rozhodování o nutné akci a její případné spuštění, nebo do další aplikace spuštěné v cloudu. První způsob je vhodný například v případě, kdy se data o klientech nesmějí dostat na servery jiných stran podle smluvních podmínek. Druhý je přínosný, když je cílem maximálně přenést zátěž na cloud.

[1] Např. detektor použitý při přípravě vstupních vzorku pro prototyp implementovaný v rámci práce na prototypu klasifikátoru je realizován pomocí jazyku Python. Jelikož se jedná o interpretovaný jazyk, nabízejí se dvě možnosti: instalace operačního systému a Python překladatele nebo implementace pro prostředí virtuálního stroje s podporou omezeného množství funkcí nabízených standardními knihovnami tohoto jazyka [23]. Druhá varianta v závislosti na využitých v detektoru knihovnách nemusí být realizovatelná.

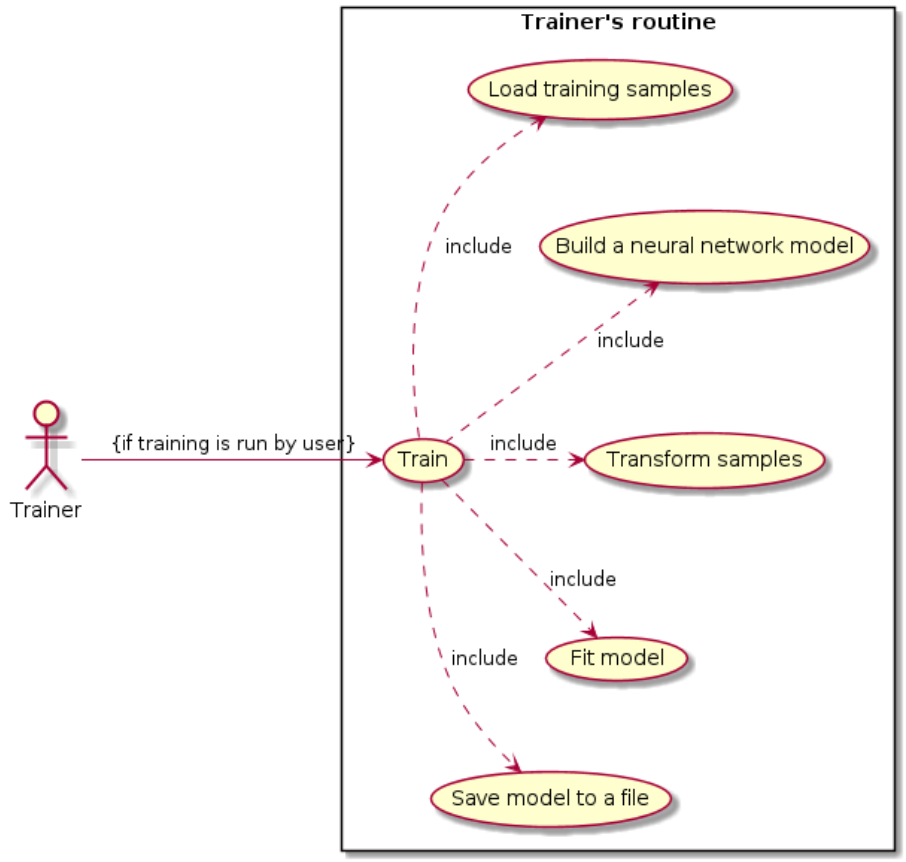
2.5.5 Scénáře

Z pohledu scénářů a možných případů užití lze popsat dva základní procesy probíhající v navrhovaném systému: učení a predikci. Vzhledem k automatické povaze funkčnosti systému jsou všichni aktéři případů užití s výjimkou uživatelem zahájeného učení zastoupení komponentami systému.



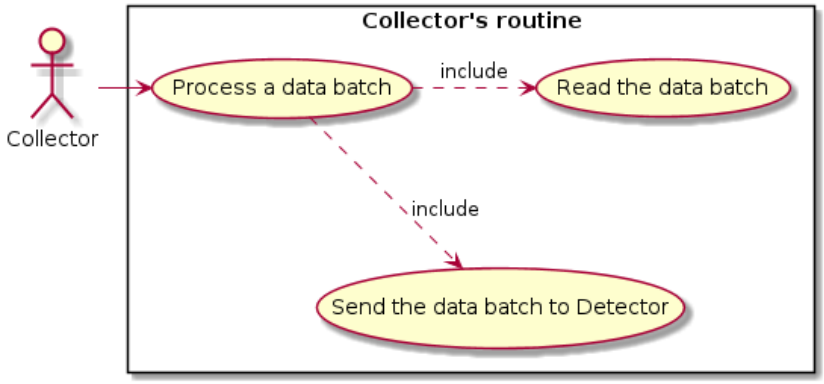
Obrázek 2.17. Diagram případů užití učení pro uživatelé (zdroj: vlastní data).

Proces učení je znázorněn na obr. 2.17 a začíná akcí uživatele. Spuštění učení sítě ležící v základu popisovaného systému vyžaduje nahrání trénovacích vzorků rozdělených do patřičných kategorií a bezprostřední spuštění procesu učení.



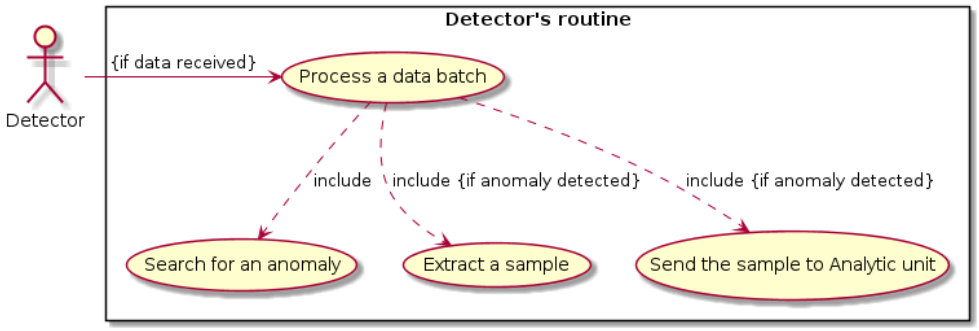
Obrázek 2.18. Diagram případů užití učení pro trénovací komponentu (zdroj: vlastní data).

V dalších krocích učení je hlavním aktérem samotná komponenta pro učení (viz obr 2.18). Učení z pohledu této komponenty zahrnuje načtení připravených a označkovaných vzorků, vytvoření modelu neuronové sítě v souladu s uživatelem předanými argumenty a vlastnostmi trénovacích dat. Dále do ní spadají transformace vzorků, nastavení modelu během epoch učení a následné uložení modelu do souborového systému.



Obrázek 2.19. Diagram případů užití detekce a klasifikace pro komponentu kolektoru (zdroj: vlastní data).

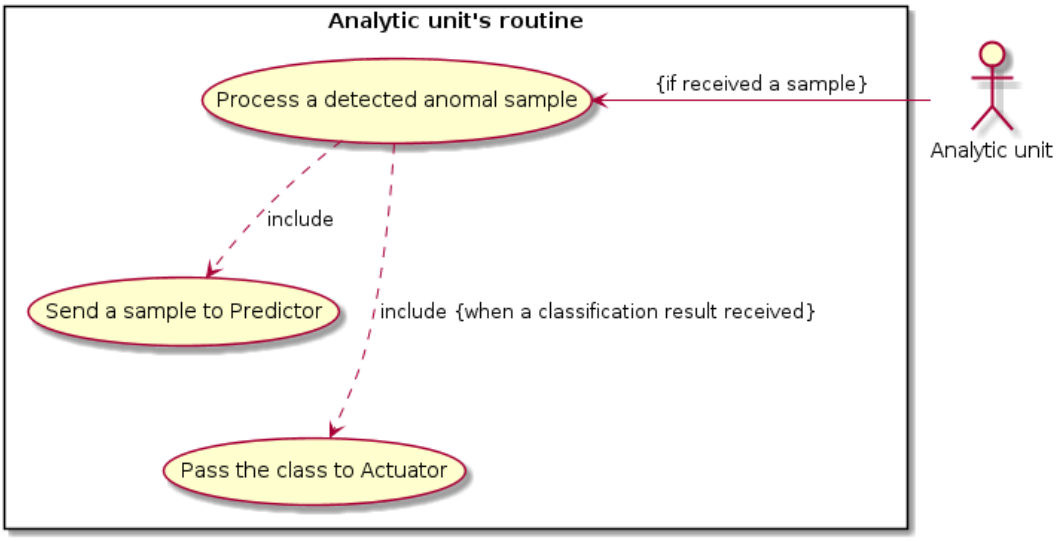
Procesy probíhající v rámci zde popisovaného systému sledování a detekce událostí v reálném čase, do kterého může být klasifikátor integrován, začínají sběrem dat na koncovém zařízení (obr. 2.19). Tento proces se spouští stále v iteracích a obsahuje dva podprocesy: načtení dávky dat ze senzoru a jejich následné odeslání do komponenty detektoru.



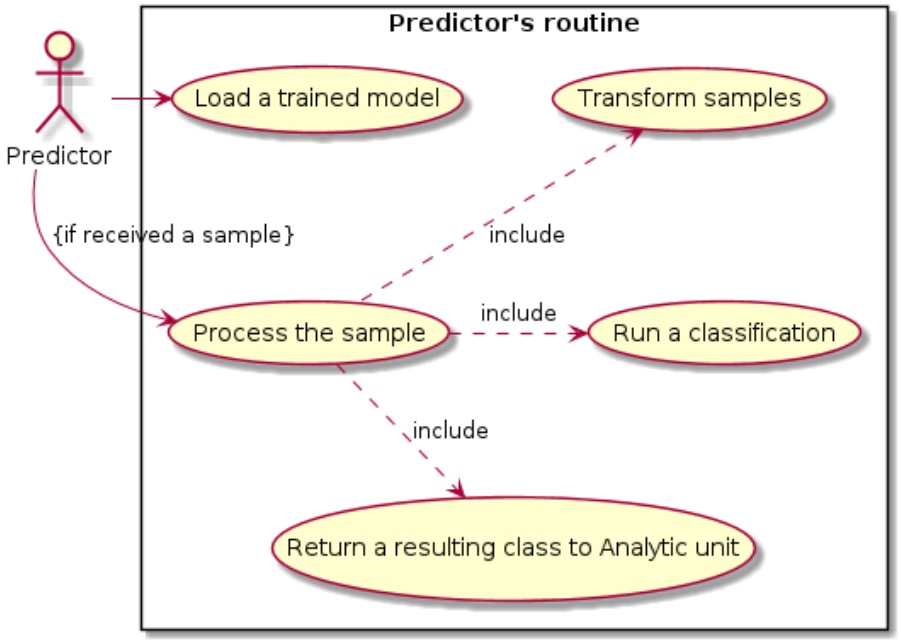
Obrázek 2.20. Diagram případů užití detekce a klasifikace pro komponentu detektoru (zdroj: vlastní data).

V okamžiku, kdy detektor dostává dávku dat pro analýzu, ji zpracovává ve třech krocích: hledá anomálii, pak v případě její úspěšné detekce extrahuje odpovídající úsek vzorku a odesílá jej do analytické jednotky (obr. 2.20).

Analytická jednotka po obdržení relevantního úseku vzorku reaguje přeposláním vzorku do komponenty prediktoru (obr. 2.21). Možné případy užití z pohledu prediktoru jsou načtení naučeného modelu neuronové sítě a v případě inicializace analytickou jednotkou, zpracování vzorku (obr. 2.22). Poslední případ zahrnuje transformaci vzorku do podoby vhodné pro vstup do modelu neuronové sítě, samotnou klasifikaci a návrat



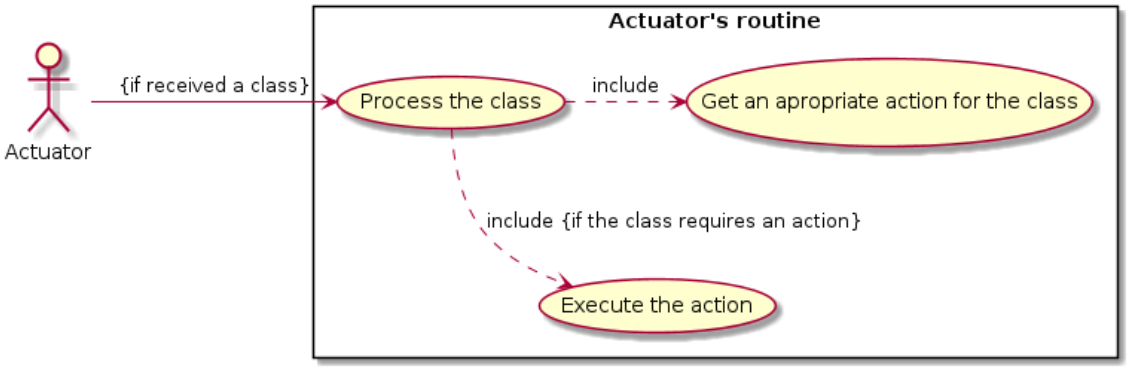
Obrázek 2.21. Diagram případů užití detekce a klasifikace pro komponentu analytické jednotky (zdroj: vlastní data).



Obrázek 2.22. Diagram případů užití detekce a klasifikace pro komponentu prediktoru (zdroj: vlastní data).

výsledné třídy do analytické jednotky. Analytická jednotka tuto třídu pak přeposílá komponentě aktuátoru.

Aktuátor po obdržení třídy prostřednictvím databáze akcí a tříd, k níž má přístup, určí, zda detekce události této třídy vyžaduje spuštění akce, a případně ji spustí v souladu s načtenou definicí (obr. 2.23).



Obrázek 2.23. Diagram případů užití detekce a klasifikace pro komponentu aktuátoru (zdroj: vlastní data).

2.6 Specifika produkčního provozu systému

Významnou část běhu predikce při experimentech tvořilo načtení a transformace vstupních dat a uloženého modelu ze souborového systému (při načtení s paddingem a podvzorkováním transformace dat trvala v průměru 94,73 % času běhu procesu prediktoru). Jelikož způsob využití prediktoru předpokládá, že se data budou pokaždé lišit, nenabízí se žádný zřejmý přístup, který by dovolil zkrátit jejich načtení. Zlepšení po této stránce lze ovšem dosáhnout zajištěním rychlého komunikačního kanálu mezi emitorem dat (např. měřící a detekční jednotkou fyzicky umístěnou na místě shromažďování dat) a platformou, na níž je nasazen prediktor.

Poměr času potřebného k transformaci vstupních dat k celkovému času běhu predikce může být snížen pomocí implementace modulů (například jednotlivých transformátorů) prostřednictvím jiných technologií.

Třetí faktor, načtení uloženého modelu, může být pozitivně ovlivněn využitím programu na principu serveru. Program v tomto případě stále běží ve své hlavní smyčce a model je po celou dobu v paměti, čímž se eliminuje nutnost opakovaného načtení z disku. Proces se pak spouští znovu až při načtení nového modelu.

2.6.1 Možnosti paralelizace

Dalším možným způsobem zrychlení zpracování je paralelizace. Paralelizovat se dá jak transformace vstupních dat, tak samotný proces predikce.

Prvním nutným krokem pro paralelizaci je rozložení neboli dekompozice celé úlohy na několik menších. Dekompozice úlohy při jejím paralelizování může být jednoho z následujících typů:

- Rekurzivní dekompozice (recursive decomposition) – vhodná pro případy, kdy úlohu lze představit v podobě rekurzivního algoritmu. Oddělit se dají například dvě instance algoritmů, spuštěné na jedné úrovní rekurze;
- Datová dekompozice (data decomposition) – instance se oddělují podle dat, které zpracovávají. Rozdělení lze provést na základě vstupních, výstupních nebo průběžných dat;
- Explorátorní dekompozice (exploratory decomposition) – dekompozice spojená s hledáním správného řešení metodou vyzkoušení několika možných větví v prostoru řešení. Každá instance v tomto případě se zabývá například výpočtem své větve potenciálního rozvoje modelu;

- Spekulativní dekompozice (speculative decomposition) – spuštění předem možných výpočtů, které mohou být vyžádané dále v průběhu programu. [24]

Pro některé z implementovaných transformátorů je nejjednodušší a nejúčinnější volbou datová dekompozice na základě vstupních dat. Tímto způsobem se dá paralelizovat běh padding transformátoru (viz 3.2.3), decimačního 3.2.4 a normalizačního (viz 3.2.5) transformátorů.

FFT transformátor (viz 3.2.2) může využít jednoho ze dvou přístupů pro paralelizaci Fourierovy transformace: algoritmu binární výměny (binary exchange) nebo algoritmu transpozice (transpose). Volba algoritmu záleží na šířce pásma spojení mezi uzly, náročnější na ni je algoritmus binární výměny. [24]

Paralelizace predikce je možná díky TensorFlow API `tf.distribute.Strategy` pro distribuovaný provoz. Využití tohoto API je kompatibilní s knihovnou `keras`, pomocí které bude implementován funkční prototyp klasifikátoru v průběhu této práce. Je podporovaná distribuce výpočtu na několik GPU, několik strojů sjednocených do jedné sítě nebo v cloudu na takzvaných TPU (Tensor processing unit). [25]

2.7 Experiment s oddělením frekvenčních pásem

Jednou z otázek, na kterou se má pokusit odpovědět tato práce je, zda filtrování vstupních hodnot ještě před detekováním anomálií má pozitivní vliv na výslednou přesnost (accuracy) predikce a existuje tedy charakteristické frekvenční pásmo. Za tímto účelem byla navržená série 10 experimentů na datech 15 vydělených z celého spektra pásem. Motivací pro určení užšího rozsahu frekvenci ke zpracování je kromě případného zvýšení přesnosti detekce a klasifikace anomálií především potenciální možnost ušetřit zdroje při produkčním nasazení, na které má vliv velikost dat: doba uložení, načtení a předávání informace z místa pořízení záznamu na server (v případě klient-server konfigurace).

Další výhodou z hlediska výpočetních zdrojů se může stát menší počet možných hodnot, kterých může nabývat zaznamenaný signál. Tento parametr má vliv například na výsledný objem dat v paměti po průchodu dat embedding vrstvou, kdy se každá unikátní skalární hodnota nahrazuje charakteristickým vektorem m-dimenzionálního prostoru.

Zprávu o rozdělení frekvenčních pásem a jejich následné klasifikaci pomocí rekurentní neuronové sítě lze najít v [26]. Tam se však data využívají jinak: klasifikovaný vzorek zastupuje 21-rozměrného pole, kde každá z dimenzí představuje jedno z 21 pásem, se kterými se v rámci studie pracuje.

2.7.1 Srovnání podobnosti vzorků

V průběhu práce na části věnované klasifikaci na datech s vyfiltrovanými odlišnými frekvenčními rozsahy (viz příloha B) vyvstala nutnost nezávislého ověření podobnosti vzorků dvou tříd v různých frekvenčních pásmech. Jako prostředek pro zjištění podobnosti signálu se zvolil Pearsonův korelační koeficient. Tento koeficient udává míru lineární závislosti mezi veličinami a nabývá hodnot z intervalu $\langle -1, 1 \rangle$, kde 0 znamená absenci lineární závislosti, +1 pozitivní a -1 negativní dokonalou lineární závislost [27].

Jelikož se výsledky porovnávají s výstupem z neuronové sítě, není okamžitě jasné, zda se při vyhodnocení podobnosti pomoci Pearsonova koeficientu musí brát ohled i na negativní závislost. Z tohoto důvodu vznikne pomocný experiment se stejnou sítí (konvoluční, viz 3.1.2) a daty, která se použila v podkapitole 3.3 (celé spektrum, 0-7800 Hz). Cílem tohoto pokusu je zjistit, zda výsledek klasifikace bude podobný u původního vzorku a jeho opačné hodnoty.

Dále, aby se zohlednily všechny vzorky porovnávaných tříd, koeficient se spočítá pro každý prvek kartézského součinu mezi množinami vzorku obou tříd. Pro porovnání podobnosti vzorku dvou tříd v různých frekvenčních pásmech se použije průměr absolutní hodnoty jednotlivých koeficientů (viz výsledky pomocného experimentu navrženého výše v praktické části této práce 3.3.1).

Pro to, aby se dosáhla stejná délka vzorků se použije padding neboli doplnění nulami (viz 3.2.3). Pearsonův korelační koeficient pro pár vzorků doplněných nulami do stejné délky se pak spočítá takto:

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x}) \cdot (y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2} \cdot \sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}, \tag{1}$$

kde x_i a y_i jsou hodnoty vzorků z první a druhé třídy, $\overline{x}$ a $\overline{y}$ - průměrné hodnoty těchto vzorků. Průměr absolutních hodnot těchto koeficientů pro každý z párů pak považujeme za průměrný Pearsonův korelační koeficient mezi vzorky dvou tříd.

Kapitola 3
Praktická část

3.1 Porovnání typů neuronové sítě

V průběhu práce na projektu se použily dva základní vícevrstvé návrhy neuronové sítě.

3.1.1 Plně propojený vícevrstvý perceptron

První síť je postavena na principu vícevrstvého perceptronu neboli plně propojené husté sítě a použitá pro první prototyp má následující vrstvy:

1. Vstupní vrstva s počtem neuronů odpovídajícím délce vstupu
2. Plně propojená vrstva (`keras.layers.Dense`):
   a) Počet neuronů: 512;
   b) Aktivační funkce: usměrněná lineární (ReLU).
3. Plně propojená vrstva (`keras.layers.Dense`):
   a) Počet neuronů: 256;
   b) Aktivační funkce: usměrněná lineární (ReLU).
4. Plně propojená vrstva (`keras.layers.Dense`):
   a) Počet neuronů odpovídá počtů možných výstupních tříd;
   b) Aktivační funkce: normalizovaná exponenciální (softmax).

Ztrátová funkce: `tf.keras.losses.sparse_categorical_crossentropy`.

Z hlediska typů neuronových sítí spadá tento návrh pod kategorii takzvaných dopředných (feedforward) sítí: vstupy prochází prostředními vrstvami, které aproximují klasifikační funkci, k výstupní vrstvě. Nedochází tedy ke zpětné vazbě - z jedné vrstvy se výstupy používají jako vstup pro další. [11]

Jakožto aktivační funkce pro všechny skryté vrstvy byla zvolená usměrněná lineární funkce (rectified linear function). Podle [11] je to doporučený typ aktivační funkce pro hluboké dopředné sítě.

Pro aktivaci výstupní vrstvy je použitá funkce softmax, která představuje zobecnění sigmoidy pro rozložení pravděpodobnosti u nebinární proměnné. Softmax je tedy typickou volbou v případě klasifikátorů s n výstupními třídami, kdy $n > 2$. [11] Počet neuronů ve skrytých vrstvách je dán převážně omezeními ze strany dostupných výpočetních zdrojů, primárně paměti.

3.1.2 Konvoluční síť

Druhá, konvoluční síť má vrstvy nastavené následovně:

1. Embedding (`keras.layers.Embedding`):
   a) Délka slovníku nastavená na maximální hodnotu vyskytující se v trénovacích a testovacích datech;
   b) Počet dimenzí embedding vektoru: 2;
   c) Délka vstupu nastavená na nejdelší vstup v trénovacích datech.
2. 1D konvoluční vrstva (`keras.layers.Convolution1D`):
   a) Počet konvolučních jader: 2;
   b) Velikost konvolučního jádra: 2;
   c) Padding (doplnění výstupu nulami) pro dosažení stejné velikosti, jakou měl vstup.
3. 1D pooling vrstva (`keras.layers.MaxPooling1D`)
   a) Velikost okénka: 2.
4. Zplošťující (flatten) vrstva (`keras.layers.Flatten`)
5. Hustá vrstva (`keras.layers.Dense`):
   a) Počet neuronů odpovídá počtů možných výstupních tříd;
   b) Aktivační funkce: normalizovaná exponenciální (softmax).

   Ztrátová funkce: `tf.keras.losses.sparse_categorical_crossentropy`.

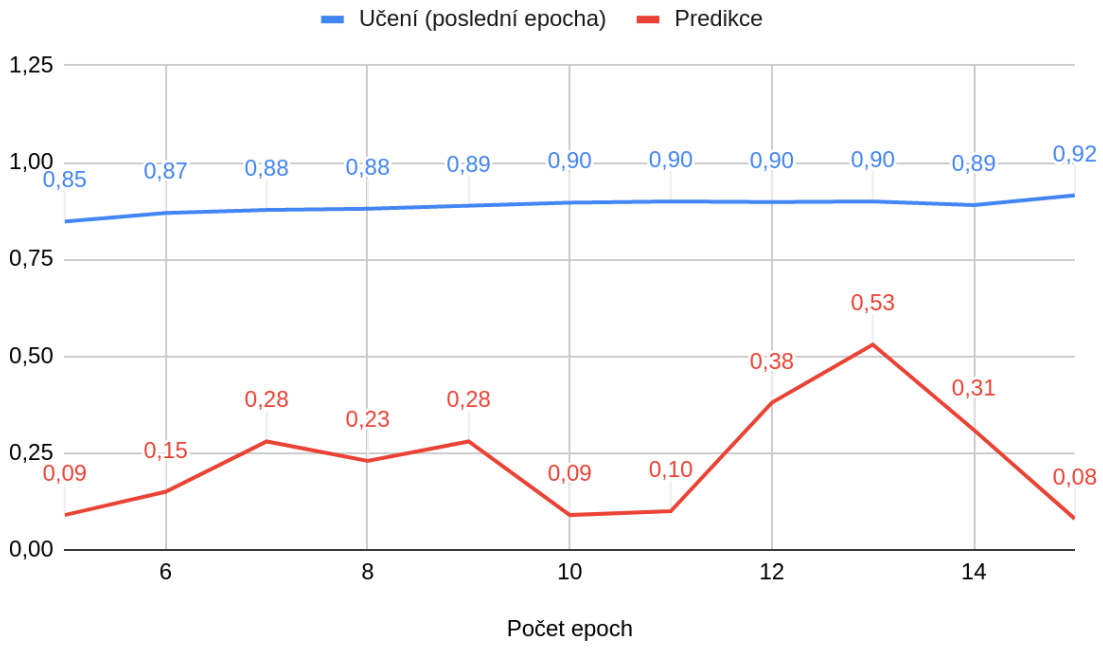
Pro porovnání výkonnosti a využití zdrojů u dvou implementovaných návrhů neuronové sítě se použila data s vyfiltrovaným pásmem 3400-7800 Hz (podrobněji viz příloha B) a náhodně se rozdělila na trénovací a testovací v poměru 4 ku 1 pomocí utility `util.data_separator`. Počet epoch učení se nastavill v rozmezí od 5 do 15. Přesnost klasifikace poslední epochy učení a predikce pro obě sítě je uvedena v grafech na obr. 3.1 a 3.2.

Na grafech lze vidět, že se učení vyzkoušelo v dostatečném počtu epoch, protože maximální přesnost predikce po dosažení optimálního počtu epoch začíná klesat, když začíná docházet k overfittingu [11].

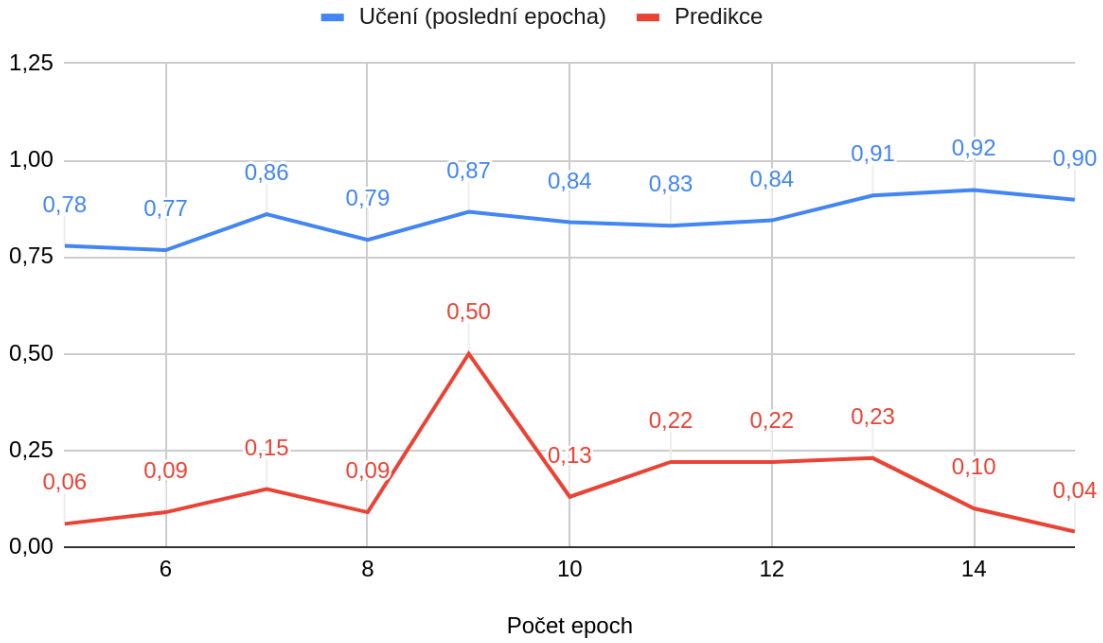
Své nejvyšší přesnosti, 0,53, dosáhla hustá síť po 13 epochách učení. Predikce pomocí konvoluční sítě dosáhla maximální přesnosti 0,5 po 9 epochách. Ačkoliv tedy plně propojená síť se ukázala jako přesnější, konvoluční síť pro svůj nejlepší výsledek vyžádala menší počet epoch učení. Další rozdíl spočívá ve objemu modelu: optimalizovaný po 15 epochách model husté sítě zabírá na disku 34,42 MB, zatímco konvoluční pouze 1 MB.

3.2 Transformace vstupních dat

V průběhu práce na tomto projektu vzniklo několik transformátorů vstupních dat v podobě tříd umístěných v modulu `util.input_transformers` a dědicích ze základní třídy s pouze abstraktními metodami `BaseInputTransformer` (zastupuje interface pro transformátor). Z těchto tříd se na základě typu sítě, aktuální úlohy (učení nebo predikce) a parametrů, se kterými byl spuštěn proces, vytváří instance `InputTransformerFactory` z modulu `util.input_transformers.input_transformer_factory`. `InputTransformerFactory` vytváří řetěz transformátorů, kterým projdou data. Zatímco `InputTransformerFactory` určuje pořadí



Obrázek 3.1. Přesnost (accuracy) dosažená v poslední epoše učení a při predikci - hustá síť (zdroj: vlastní data).



Obrázek 3.2. Přesnost (accuracy) dosažená v poslední epoše učení a při predikci - konvoluční síť (zdroj: vlastní data).

a nutnost využití jednotlivých transformátorů, deleguje samotné vytvoření instancí kontejneru služeb `InputTransformersContainer`. Tímto se uplatňuje princip oddělení zodpovědnosti. Řetězec instancí transformátorů se pak prochází v cyklu se zachováním určeného pořadí. Výstup z jednoho transformátoru se stává vstupem do dalšího, čímž se

realizuje návrhový vzor `pipelines and filters` vhodný pro zpracování proudových dat [28]. Níže následuje podrobnější popis funkcí každého transformátoru.

Vzhledem k tomu, že transformátory mají deterministické výstupy a chyba v jejich logice může značně ovlivnit výsledky celého systému, je vhodné pokrýt třídy transformátorů jednotkovými testy.

3.2.1 Přemapování tříd

Tento transformátor je implementován třídou `RemappingTransformer`. Slouží ke spuštění učení a následně také predikce s jiným rozložením tříd, aniž by se musela upravovat vstupní data. Přidání instance této třídy do řetězce transformátorů je podmíněno využitím argumentu `map` při spuštění procesu. Tento argument je nadefinován v `BaseArgumentParser` modulu `util` a je tedy dostupný jak pro učení, tak pro predikci. Zápis argumentu má následující formát: `--map x1.y2 x2.y2 … xn.yn`, kde x_i představují původní číslo třídy a y_i - číslo třídy, do které události označené jako x_i mají být přemapovany.

Funkčnost nabízená transformátorem přemapování událostí může být užitečná například v případě, kdy se pomocí matice záměn zjistí přílišná podobnost vzorků dvou tříd. Poté, když se rozhodne, že rozlišení těchto dvou událostí zbytečně snižuje celkovou přesnost predikce, stačí přemapovat obě tyto třídy do jedné společné.

3.2.2 FFT

`FftTransformer` aplikuje na data Fourierovu transformaci prostřednictvím funkce `fft` modulu `scipy.fftpack`, která implementuje jednodimenzionální diskrétní transformaci algoritmem rychlé Fourierovy transformace [29]. Instance `FftTransformer` se přidává do řetězce transformátorů pouze v případě explicitního přidání argumentu `--fft` při spuštění učení nebo predikce.

3.2.3 Padding

Model neuronové sítě implementovaný na základě frameworku TensorFlow přijímá na vstupu tzv. tensor - zobecnění n-dimenzionální matice nebo vektoru [30]. Z toho plyne, že jednotlivé serie hodnot přestavující průběhy signálů musí mít na vstupu do sítě stejnou délku. Padding neboli doplnění konstantou hodnot do dosažení minimální nutné délky je vedle podvzorkování jedním ze způsobů, kterým se řeší požadavek konstantní délky vstupních sekvencí. Z principu této transformace je zřejmé, že se její pomocí dají vyřešit pouze případy, kdy vstupní posloupnost je kratší než délka, s níž pracuje první vrstva sítě. Je realizován pomocí třídy `PaddingInputTransformer`, která ve svém jádru využívá funkci `pad` modulu z knihovny TensorFlow.

Konkrétně se jedná o přidání stejného (± 1) počtu nul z levé a pravé strany ke každé posloupnosti hodnot pro dosažení požadované délky, která se nastavuje pomocí argumentu `input-length` nebo se zjišťuje z modelu v případě predikce. Tento argument je definován v `BaseArgumentParser` modulu `util` a je tedy dostupný jak pro učení, tak pro predikci.

3.2.4 Podvzorkování / decimace

V případě kratších vstupů lze použít metodu paddingu pro doplnění vektoru hodnot nulami do dosažení nutné délky. V opačné situaci, kdy je vektor vstupních dat delší než vektory, na nichž se síť původně trénovala, jedním z řešení je podvzorkování.

V projektu je využitá knihovna Scipy a její modul `signal` nabízí k tomuto účelu dvě možnosti implementované metodami “resample” a “decimate”. V základě první leží

Faktor	Odchýlka
1	0,069298
2	0,079413
3	0,058998
4	0,055337
5	0,023223
6	0,025437
7	0,042326
8	0,068264
9	0,09225

Tabulka 3.1. Závislost vnější odchylky mezních hodnot od zvoleného faktoru při decimaci metodou `scipy.signal.signaltools.decimate` na 948 vzorcích (zdroj: vlastní data).

Fourierova metoda, u níž se předpokládá, že signál je periodický. Proto vzhledem k charakteru dostupných dat byla zvolena druhá využívající IIR filtr. [29]

Součástí metody je také aplikace anti-aliasing filtru. V důsledku při převzorkování může docházet k odchylce v mezních hodnotách signálů (tab. 3.1). Kvůli tomu se faktor decimace vypočítává s ohledem na tuto odchylku:

$$M = \frac{l_{max} \cdot (1 + k \cdot \sigma)}{n_1}, \tag{2}$$

kde l_{max} je maximální délka série hodnot na vstupu, σ je výše popsaná odchylka, k – koeficient, kterým se tato odchylka vynásobí pro případ, že skutečná odchylka přesáhne naměřenou při experimentech a n_1 je počet uzlů na první vstupní vrstvě sítě. Jakožto se vzala nejvyšší naměřená odchylka 0,09225. Koeficient k byl stanoven na 2, aby se vytvořila nutná rezerva pro možné zvětšení odchylky na jiných datech.

Dále k decimovaným hodnotám musí být přičtena δ, která se rovná rozdílu mezi nulou a nejmenší hodnotou ve výstupu podvzorkovacího transformátoru v případě, že je záporná. Důvodem k tomu je, že výčet možných hodnot i po zohlednění možné odchylky začíná od nuly.

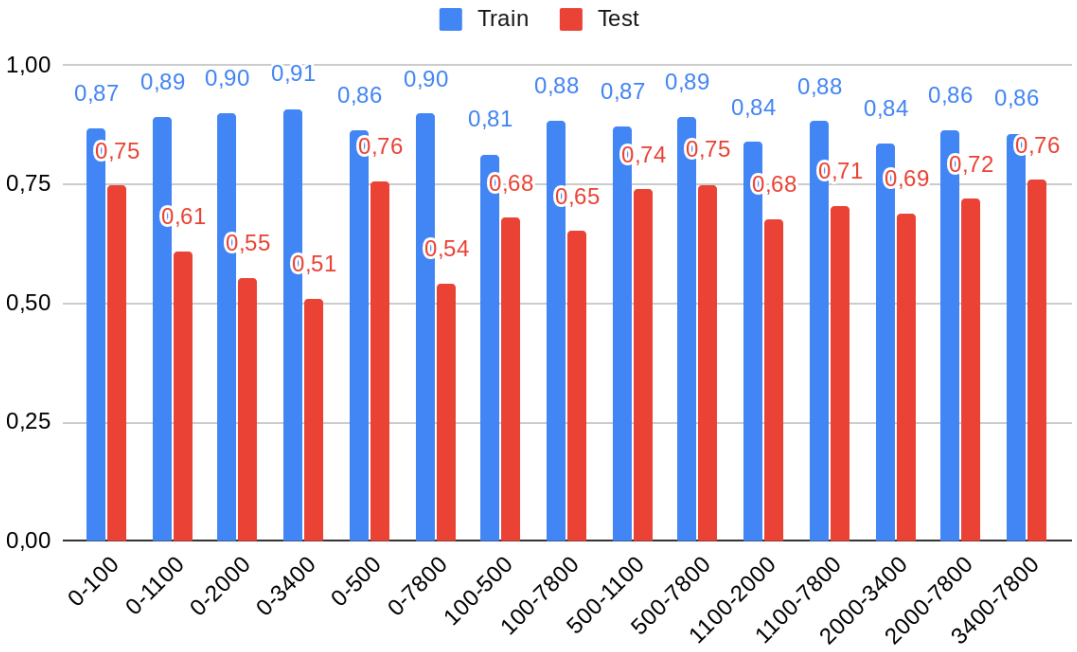
3.2.5 Normalizace

`NormalizingInputTransformer` realizuje min-max normalizaci. Je přínosná v případech, kdy se nepoužívá embedding vrstva. Hodnoty vstupu jsou po aplikaci tohoto transformátoru v intervalu mezi 0 a 1.

3.3 Oddělení frekvenčních pásem

Dále se uvádějí výsledky experimentu popsaného v teoretické části. Použila se konvoluční síť popsaná jako druhá v podkapitole „Vyzkoušené návrhy neuronové sítě“ (3.1). Učení probíhalo v deseti epochách.

Z výsledků znázorněných na obr. 3.3 je patrné, že nejvyšší přesnosti klasifikace (0,76) se podařilo dosáhnout při využití horního polovičního pásma (3400-7800 Hz) a naopak užšího rozsahu v dolní části sledovaného spektra (0-500 Hz). V porovnání s pásmem obsahujícím celý rozsah (0-7800 Hz) se tyto dvě pásma také vyznačují nižší přesností v poslední epoše učení (0,86 oproti 0,9). Zároveň je nutno poznamenat, že přesnost predikce na datech s celým spektrem nebyla nejvyšší v žádném z 10 experimentů. Největší přesnost predikce, které se dosáhlo byla 0,6 a to ve dvou experimentech, kdy nejvyšší přesnost mezi všemi rozsahy byla 0,82 a 0,85.



Obrázek 3.3. Porovnání průměrné přesnosti učení dosažené v poslední epoše (Train) a predikce (Test) podle využitého frekvenčního pásma (zdroj: vlastní data).

3.3.1 Pomocný experiment pro srovnání podobností vzorků

Experiment popsaný v teoretické části (sekce 2.7.1) se provedl se stejnou sítí (konvoluční, viz sekci 3.1.2) a daty, která se použila v 3.3 (celé spektrum, 0-7800 Hz). Vyzkoušely se vzorky ze 4 tříd. Data se rozdělila na trénovací a testovací v poměru 4:1. Z testovacích dat se náhodně vybralo po jednom vzorku z každé třídy a spočítaly se k nim opačné hodnoty. Není to sice z hlediska počtu reprezentativní vzorek, ale ilustruje schopnost této sítě správně klasifikovat průběh anomálie i potom, co byl nahrazen opačným vektorem. Níže se v tab. 3.2 uvádějí výsledky tohoto experimentu. Jak je vidět z v ní uvedených pravděpodobností, použití opačného vektoru hodnot nemá výrazný vliv na klasifikaci. Největší rozdíl je kolem 15,73 %. Proto se musí zohlednit ve stejné míře jak kladné, tak záporné korelační koeficienty.

skutečná třída	detekovaná třída	1	2	3	4
1	původní vzorek	0,35	0,11	0,09	0,44
1	opačná hodnota	0,35	0,11	0,09	0,45
2	původní vzorek	0,21	0,06	0,07	0,66
2	opačná hodnota	0,17	0,06	0,07	0,70
3	původní vzorek	0,37	0,11	0,09	0,43
3	opačná hodnota	0,37	0,11	0,09	0,43
4	původní vzorek	0,19	0,07	0,07	0,68
4	opačná hodnota	0,21	0,06	0,06	0,66

Tabulka 3.2. Pravděpodobnost příslušnosti k jednotlivým třídám původního vzorku a druhého tvořeného opačnými k němu hodnotami (zdroj: vlastní data).

3.4 Experiment s ručně připravenými daty

Na rozdíl od jiných experimentu provedených v rámci této diplomové práce, v průběhu tohoto experimentu se použijí ručně připravená data (viz příloha C). Ačkoliv obsahují méně vzorků, jsou z většího počtu tříd. Tyto třídy jsou také rozdělené s větší citlivostí (např. jsou od sebe oddělené záznamy pádu různých objektů).

V sérii pokusu se s konvoluční sítí (viz 3.1.2) a všemi 7 třídami podařilo dosáhnout průměrné přesnosti 0,14. Z analýzy matic záměn vyplynulo, že se nejvíce zaměňují vzorky ze tříd „Pád velkého míče“, „Stoupnutí na plochu“, „Sestoupení z plochy“ a „Skartovač“. Po sjednocení těchto tříd do jedné pomocí transformátoru přemapování tříd přesnost dosáhla až 0,71.

Kapitola 4
Diskuze

4.1 Oddělení pásem

Na základě pozorování ze sekce 3.3 se lze pokusit diskutovat o důvodech, proč přesnost klasifikace na neznámých datech nebyla nejvyšší u celého spektru. První možnost, která se nabízí z hlediska posloupnosti zpracování dat je odlišná detekce. Z počtů tab. B.1 (viz příloha B) plyne, že třída se záznamy pádu kuličky mezi 0 Hz a 500 Hz nemá žádné detekované výskyty. To, že zrovna přesnost klasifikace vzorků této třídy mohlo v podstatné míře ovlivnit celkovou výslednou přesnost pokusů na celém rozsahu je vidět, když se porovná průměrná přesnost klasifikace pouze této třídy. Činí 0,19 pro pásmo 0-7800 Hz a 0,631 pro 3400-7800 Hz.

Z údajů z matice záměn pro 1. třídu (tab. 4.1) lze odvodit, že se v pásmu 0-7800 Hz vzorky této třídy nejvíce zaměňují s třídou 4 (v průměru 76,57 % všech vzorků třídy 1 se chybně označí jako třída 4). Když to porovnáme se stejnými údaji pro pásmo 3400-7800 Hz (tab. 4.2), můžeme se přesvědčit, že v případě tohoto pásma takové výskyt takových chyb je o něco nižší (v průměru 66,97 %).

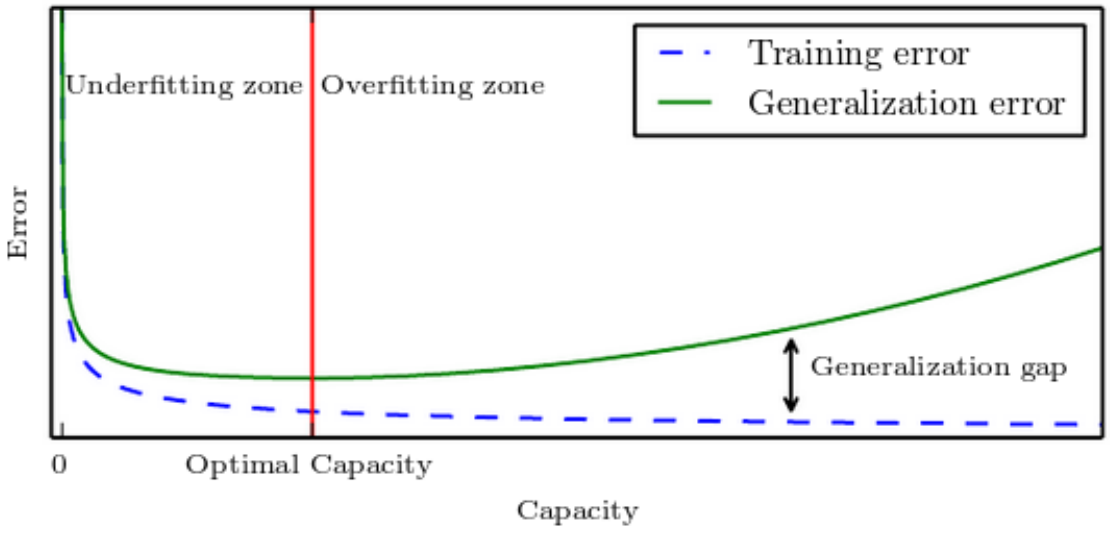
Číslo experimentu \ třída	0	1	2	3
1	1	0	0	34
2	0	0	6	29
3	0	1	2	32
4	0	0	4	31
5	0	0	0	35
6	0	0	4	31
7	25	0	2	8
8	5	0	4	26
9	0	0	5	30
10	22	1	0	12

Tabulka 4.1. Řádky z matice záměn popisující predikované třídy pro události ve skutečnosti spadající pod 1. třídu (pouze pásmo 0-7800 Hz) (zdroj: vlastní data)

Číslo experimentu \ třída	0	1	2	3
1	0	1	3	29
2	0	0	8	25
3	1	4	1	18
4	0	7	3	23
5	0	1	4	17
6	6	5	2	20
7	3	8	1	21
8	4	1	7	21
9	2	2	2	27
10	1	8	4	20

Tabulka 4.2. Řádky z matice záměn popisující predikované třídy pro události ve skutečnosti spadající pod 1. třídu (pouze pásmo 3400-7800 Hz) (zdroj: vlastní data).

Nejdříve zkusíme ověřit jinou metodou, zda se opravdu vzorky ze tříd 1 a 4 více podobají v celém spektru, než v rozsahu mezi 3400 Hz a 7800 Hz. Za tímto účelem spočítáme průměrný Pearsonův korelační koeficient pro vzorky 1. a 4. třídy pro tyto pásma (viz sekci 2.7.1 Teoretické části). Pro pásmo 0-7800 Hz je toto číslo 0,75204, zatímco u rozsahu 3400-7800 Hz je o něco nižší, 0,6170. Tímto zjištěním se dá částečně vysvětlit rozdíl v dosažené přesnosti.



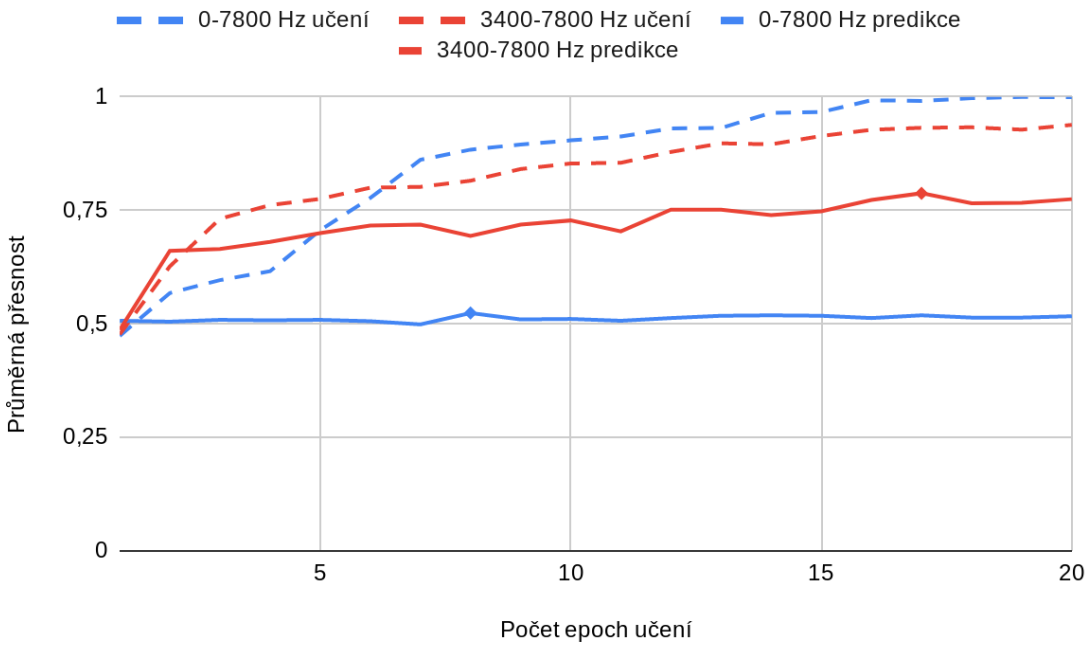
Obrázek 4.1. Vztah mezi objemem učení a chybovostí (zdroj [11]).

Dalším faktorem se mohl stát takzvaný underfitting nebo overfitting. S delším průběhem fáze učení a větším počtem epoch, ve kterých probíhá, se stále zvětšuje přesnost na trénovacích datech. To však neplatí o testovacím vzorku. Nejdříve je jak přesnost dosahovaná v průběhu učení, tak přesnost následné predikce nízká a to kvůli nedostatečnému počtu opakování při učení. Jedná se o underfitting. Po určitém počtu trénovacích epoch, který záleží na konkrétních datech, jejich objemu, míře jednotnosti a návrhu sítě, nastává přelomový okamžik, po kterém se setkáváme s opačným problémem - overfittingem neboli přílišným přizpůsobením sítě trénovacím datům na úkor kvality predikce na datech testovacích [11]. Souvztažnost objemu učení a chybovosti (tj. metriky opačné přesnosti) je ukázána na obr. 4.1 a charakterizuje se stálým snížením chybovosti při

trénování a zvýšením chybovosti predikce (na grafu - generalizace, generalization) po dosažení optimálního objemu učení. Se zvýšením chybovosti predikce se mezi nimi zvětšuje také generalizační rozdíl (generalization gap). Pro ověření se může uspořádat další pomocný experiment.

Podstatou experimentu bude porovnání vývoje průměrné přesnosti v průběhu učení a pak při testu na nových datech se změnou počtu epoch u obou pásem, které se dosud sledovaly. Použije se stejná konvoluční síť (viz 3.1.2), u níž se bude měnit počet epoch učení. Celkem se pro každé pásmo nezávisle udělá 10 experimentu s odlišným rozdělením dat na trénovací a testovací v poměru 4:1. Pro každé takové rozdělení se učení a predikce spustí dvacetkrát s počtem epoch v intervalu $\langle 1, 20 \rangle$.

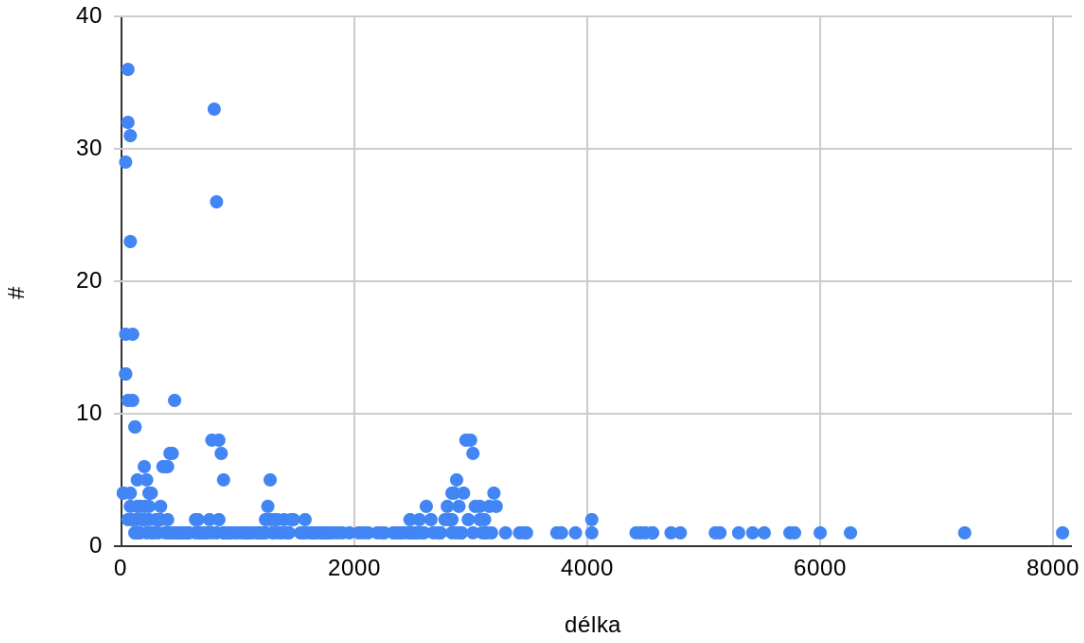
Výsledky jsou uvedeny na grafu 4.2. Na grafu v případě pásma 0-7800 Hz lze pozorovat některé známky, které mohou svědčit o overfittingu: přesnost v poslední epoše učení se přibližuje k maximální v 16 epoše (průměr 0,9929), zatímco přesnost predikce dosahuje svého vrcholu již v 8 epoše. Však na rozdíl od vzorového příkladu na grafu 4.1, po 8. epoše nedochází ke stálému klesání přesnosti predikce - kolísá kolem 0,5 stejně jako před dosažením maxima. Přesnost pro data v pásmu 3400-7800 Hz se vyvíjela tak, jak se předpokládalo a dosáhla optima v 17. epoše, po níž začala mírně klesat.



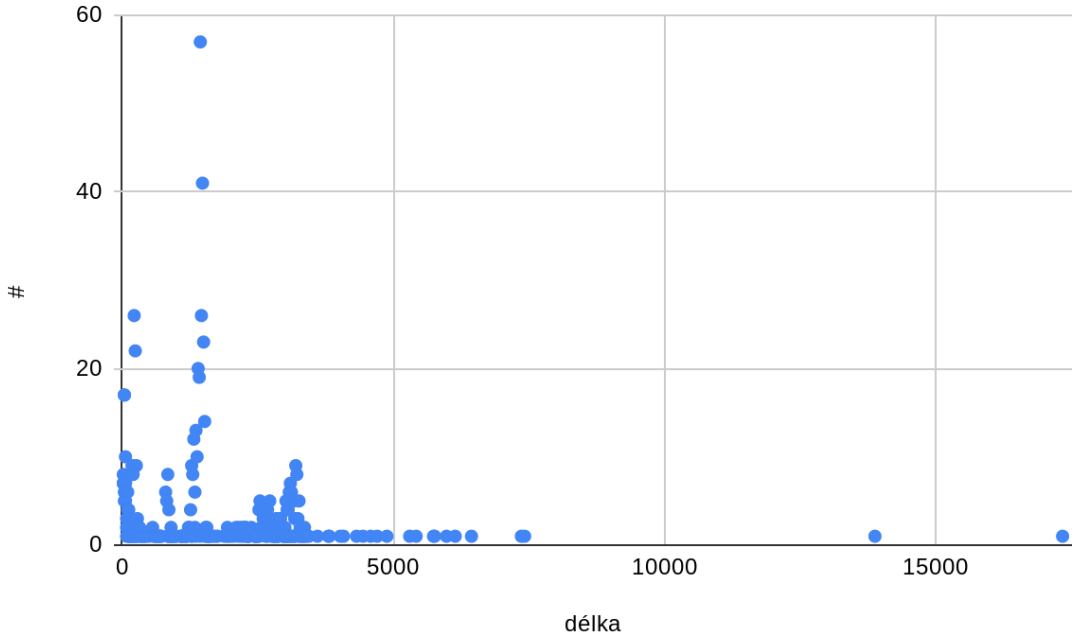
Obrázek 4.2. Vztah mezi počtem epoch učení a průměrnou dosaženou přesností v poslední epoše učení a při predikci v pásmech 0-7800 Hz a 3400-7800 Hz. Maximální hodnoty označené na grafu jsou: 0,524 pro pásmo 0-7800 Hz a 0,788 pro pásmo 3400-7800 Hz (zdroj: vlastní data).

Z tohoto chování lze usoudit, že po zpracování vzorků (hlavně max pooling, k převzorkování nedocházelo) si jsou vzorky příliš podobné v celém spektru k tomu, aby byly podle charakteristických příznaků správně klasifikovány neuronovou sítí. Nerovnoměrná délka vzorků mohla rovněž vést k tomu, že kratší vzorky byly po paddingu doplněny nulami do takové délky, že se charakteristická informace „ztratila“. V tomto případě by se mělo značně lišit rozložení délek události obou pásem. Konkrétně by z něj mělo plynout, že více zastoupené jsou kratší události, zatímco existuje několik mnohem del-

ších. Skutečná čísla ale ukazují, že k této nesrovnalosti délek je náchylnější spíše pásmo 3400-7800 Hz (srov. grafy na obr. 4.3 a 4.4). Události detekované v pásmu 0-7800 Hz se rozdělují podle svých délek mnohem rovnoměrněji a nejdelší událost má 8081 hodnot, zatímco v pásmu 3400-7800 Hz je to 17341.



Obrázek 4.3. Rozložení délek události pro data z pásma 0-7800 Hz.



Obrázek 4.4. Rozložení délek události pro data z pásma 3400-7800 Hz.

Kapitola 5
Závěr

V průběhu této diplomové práce se v několika variantách navrhl systém pro snímání signálu v optických senzorových sítích, detekci anomálií v naměřených datech, jejich klasifikaci jako událostí a spuštění akce na základě výsledků klasifikace. Byly probrány otázky volby technologií a nástrojů pro tento systém, nutnosti transformace vstupních dat. V rámci přístupů nabízených zvolenými nástroji (Python a TensorFlow) byly implementované v podobě funkčního prototypu klíčové komponenty navrhovaného systému (komponenta pro učení neuronové sítě a komponenta pro klasifikaci s využitím této sítě) s důrazem na využití se službou Google Cloud AI Platform.

Na základě zvolené metriky se porovnaly dva typy neuronové sítě (vícevrstvý perceptron a konvoluční síť) a také využití celého spektra naměřených hodnot oproti dílčím frekvenčním pásmům. V prvním z těchto experimentů se ukázalo, že vícevrstvý perceptron je o něco přesnější, ale má své nevýhody - v porovnání s konvoluční sítí větší objem naučeného modelu a vyšší počet nutných epoch učení. Volba typu sítě tudíž záleží na zdrojích, které jsou k dispozici pro nasazení systému. Samozřejmě, vyzkoušené modely sítí mohou být dále optimalizované a pravděpodobně existují konfigurace jejich parametrů, které povedou k lepším výsledkům na stejných datech.

Výsledky druhého zmíněného experimentu svědčí, že lepší přesnosti (accuracy) se podařilo dosáhnout nikoliv s využitím celého spektra, ale jen jednoho z vydělených pásem. Rozdíly mezi pásmem obsahujícím data ze všech frekvencí a dílčími frekvenčními rozsahy, u nichž byla klasifikace na testovacích datech nejpřesnější, se dají vysvětlit několika potenciálními faktory. Mohly mít vliv odlišnosti ještě během detekce a to hlavně v počtu detekovaných události. Tak by se dalo objasnit větší průměrnou přesnost pásma 0-500 Hz, ve kterém se nedetekoval žádný výskyt události jedné ze tříd. Dále se vyzkoušela jiná metoda než neuronové sítě pro vyhodnocení podobnosti signálů ve dvou nejvíce při klasifikaci zaměňovaných třídách pro data z celého spektru. Podobnost se opravdu ukázala být vyšší než v pásmu 3400-7800 Hz. Navíc byla ověřena možnost overfittingu a vady při zpracování dat. Po vyloučení těchto dvou faktorů lze tvrdit s větší jistotou, že pro zvolené události lze najít charakteristické pásmo. To znamená, že při využití detekce na datech vyfiltrovaných pouze pro tento rozsah frekvencí lze dosáhnout vyšší přesnosti, než pomocí celého spektru. Vzhledem k tomu, že proces detekce anomálií leží mimo rámec této práce, nelze však s jistotou prohlásit, že hlavní rozdíl je důsledkem právě klasifikační etapy zpracování.

Na příkladě experimentu s ručně připravenými daty se ukázalo, že pomocí zkvalitnění trénovacích dat a sjednocení tříd pro podobné třídy se dá dosáhnout poměrně vysoká přesnost klasifikace.

Vzhledem k šířce zvoleného tématu a důrazu na implementační stránku projektu, tato diplomová práce neaspiruje na to být všestranným a plným bádáním v otázce klasifikace signálů v optickém vlákně, které si tato oblast bezpochyby zaslouží. Lze na ni však navázat a případně prohloubit v několika směrech.

Podle McLoughlin et al. [2] klasifikace zvukových událostí, aby se dala považovat za robustní a v praxi uplatnitelnou, vyžaduje vytvoření systému schopného správně dete-

kovat a klasifikovat události obsažené v neznámém zvukovém záznamu a to nehledě na obsažený akustický šum a překrývající se zvuky. Požadavky z této definice by se daly rozšířit i na případ signálů získaných ze senzorů optického vlákna. Přitom v průběhu práce na tomto projektu nebyl brán patřičný ohled na přítomnost šumu v klasifikovaných vzorcích, ani na možnost několika signálů z různých tříd. Vylepšení systému s přihlédnutím k podmínkám provozu v praxi tvoří jeden z možných směrů pro navázání.

Dalším možným krokem pro zkvalitnění výsledků je využití systému klasifikace signálu pro vyhodnocení jeho výkonnosti na jedné ze standardních databází s klasifikovanými vzorky zvuků. To by mělo dovolit posoudit úspěšnost vyvinutého řešení v porovnání s jinými pracemi v tomto oboru, které využívají stejná data. [2]

V rámci této práce také nebyl vyzkoušen další způsob využití dat s oddělenými frekvenčními pásmy. Pokud se anomálie detekují ještě před rozdělením pásem vstupních průběhů, data z různých pásem vztahující se ke stejné události by mohly být použita společně podobně jako u Sawhney [26] .

Literatura

[1] Honchar, Alexandr. Deep learning: the final frontier for signal processing and time series analysis?. *Medium: Data Science*. 2018.
https: / / medium . com / @alexrachnog / deep-learning-the-final-frontier-for-signal-processing-and-time-series-analysis-734307167ad6.

[2] McLoughlin, Ian, Haomin Zhang, Zhipeng Xie, Yan Song, Wei Xiao, Huy Phan a Juan A. Anel. Continuous robust sound event classification using time-frequency features and deep learning. *PLOS ONE*. 2017-9-11, ročník 12, č. 9. ISSN 1932-6203. Dostupné na DOI 10.1371/journal.pone.0182309.
https://dx.plos.org/10.1371/journal.pone.0182309.

[3] Šidlauskas, Aurimas. Video surveillance and the GDPR. *Social Transformations in Contemporary Society*. 2019, č. 7. ISSN 2345-0126.
https://repository.mruni.eu/handle/007/15840.

[4] Olson, Harry. *Music, Physics and Engineering*. 2 ed. vyd. Dover: Dover Publications, 1967. ISBN 978-0486217697.

[5] Kulkarni, Sanjeev. *Information Signals*.
https://www.princeton.edu/~cuff/ele201/kulkarni.html.

[6] Smith, Steven W. *The Scientist and Engineer's Guide to Digital Signal Processing*. Second Edition vyd. San Diego: California Technical Publishing, 1999. ISBN 0-9660176-6-8.

[7] T.Meggit b: a L. S. Grattan. *Optical Fiber Sensor Technology*. 1st ed. reprint vyd. New York: Springer Science+Business Media, 2000. ISBN 978-1-4419-4983-7.

[8] Ma, Ling, Dan Smith a Ben Milner. Environmental Noise Classification for Context-Aware Applications. *Database and Expert Systems Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2003, s. 360-370. Dostupné na DOI 10.1007/978-3-540-45227-0_36.
http://link.springer.com/10.1007/978-3-540-45227-0_36.

[9] Couvreur, Christophe. *Environmental Sound Recognition: A Statistical Approach*.

[10] Mitilineos, Stelios A., Stelios M. Potirakis, Nicolas-Alexander Tatlas a Maria Rangoussi. A Two-Level Sound Classification Platform for Environmental Monitoring. *Journal of Sensors*. 2018-06-03, ročník 2018, s. 1-13. ISSN 1687-725X. Dostupné na DOI 10.1155/2018/5828074.
https://www.hindawi.com/journals/js/2018/5828074/.

[11] Goodfellow, Ian, Yoshua Bengio a Aaron Courville. *Deep learning*. The MIT Press vyd. Cambridge, Massachusetts: The MIT Press, [2016. ISBN 978-0262035613.
http://www.deeplearningbook.org/.

[12] Miller, John. When Recurrent Models Don't Need to be Recurrent. *BAIR: Berkley Artificial Intelligence Research*. 2018.
https://bair.berkeley.edu/blog/2018/08/06/recurrent/.

[13] Bai, Shaojie, J. Zico Kolter a Vladlen Koltun. An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling.
https://arxiv.org/abs/1803.01271.

[14] Janda, Miloš. *Detekce hran pomocí neuronové sítě*.

[15] *Cloud Machine Learning Engine (CMLE)*.
https://cloud.google.com/ml-engine/.

[16] *Google Trends - TensorFlow*.
https://trends.google.com/trends/explore?date=2014-10-20%202019-11-20&q=TensorFlow.

[17] Muller, Andreas C. a Sarah Guido. *Introduction to machine learning with Python*. 1st edition vyd. Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1-449-36941-5.

[18] *Development environment*.
https://cloud.google.com/ml-engine/docs/environment-overview.

[19] Kruchten, P.B. The 4 1 View Model of architecture. *IEEE Software*. 1995, ročník 12, č. 6, s. 42-50. ISSN 07407459. Dostupné na DOI 10.1109/52.469759.
http://ieeexplore.ieee.org/document/469759/.

[20] Nadareishvili, Irakli. *Microservice architecture*. Sebastopol, CA: O'Reilly Media, 2016. ISBN 14-919-5625-9.

[21] Shvets, Alexander. *Dive Into Design Patterns*. v2019-2.0 vyd. 2019.

[22] Roberts, Michael J. *Signals and systems*. 2nd ed vyd. New York: McGraw-Hill, c2012. ISBN 978-0-07-338068-1.

[23] *Python on a chip*.
http://www.pythononachip.org.

[24] Grama, Ananth. *Introduction to parallel computing*. 2nd ed vyd. New York: Addison-Wesley, 2003. ISBN 978-0201648652.

[25] *Distributed training with TensorFlow*.
https://www.tensorflow.org/guide/distributed_training.

[26] Sawhney, Nitin. *Situational Awareness from Environmental Sounds*.

[27] *Pearson Correlation - SPSS Tutorials - LibGuides at Kent State University*.
https://libguides.library.kent.edu/SPSS/PearsonCorr.

[28] Schmidt, Douglas, Michael Stal, Hans Rohnert a Frank Buschmann. *Pattern-oriented software architecture volume 1*. Chichester: John'Wiley & Sons Ltd, 1996. ISBN 0471958897.

[29] *Numpy and Scipy Documentation (NSD)*.
https://docs.scipy.org/doc/.

[30] *TensorFlow tensors*.
https://www.tensorflow.org/guide/tensor.

Příloha A
Zdrojový kód

GIT repozitář obsahující výsledný zdrojový kód implementovaného prototypu je dostupný z `https://gitlab.fel.cvut.cz/petrogle/signalclassifiergcloud.git`.

Příloha B
Data s oddělenými frekvenčními pásmy

Využila se data pořízená v laboratorních podmínkách. Jako podklad pro klasifikaci se dodala v podobě již automaticky kategorizovaných vzorků. Data zahrnují záznamy ze 4 odlišných tříd:

1. stokrát zaznamenaný pád kuličky na koš,
2. stokrát zaznamenané puštění vrtačky naprázdno,
3. stokrát zaznamenané spuštění kompresoru,
4. audio soubor spuštěný ve smyčce.

Veškeré záznamy byly pořízeny v laboratorních podmínkách pomocí senzorů v optické síti. Je důležité poznamenat, že vzorky jednotlivých tříd se projevily v různých dílčích frekvenčních pásmech v různé míře. Z tohoto důvodu se výrazně liší počty vzorků, jimiž jsou třídy zastoupené v různých rozsazích frekvencí. Počet vzorků v každé třídě pro každé z pásem je uvedeno v tab. B.1. Z počtů v tabulce je patrné, že v několika případech v průběhu signálu nebyl detekován žádný vzorek některé ze tříd. Rozdíly v počtech detekovaných události v různých frekvenčních pásmech znemožnily podobné použití těchto dat způsobem popsaným u Sawhney [26].

GIT repozitář s daty je dostupný z `https://gitlab.fel.cvut.cz/petrogle/signalclassifierdata.git`, adresář v projektu `separated_frequencies`.

Hz třída	1	2	3	4	Celkem
0-100	0	58	120	336	514
0-1100	144	44	148	365	701
0-2000	171	47	149	334	701
0-3400	173	58	145	331	707
0-500	0	44	142	336	522
0-7800	176	60	148	334	718
100-500	125	58	132	423	738
100-7800	151	93	129	382	755
500-1100	146	96	134	405	781
500-7800	157	105	0	389	651
1100-2000	153	123	134	402	812
1100-7800	155	110	139	379	783
2000-3400	143	111	129	334	717
2000-7800	154	108	141	384	787
3400-7800	165	116	146	368	795

Tabulka B.1. Počet detekovaných vzorků každé třídy události pro jednotlivá frekvenční pásma (zdroj: vlastní data).

Příloha C
Ručně označkovaná data

Jiná dávka vzorků byla získána ručním vynětím z celého průběhu a klasifikací na základě videozáznamu měření experimentů s plochou, ve které se nacházelo optické vlákno, reagující na vibrace. V porovnání s daty představenými v příloze B se jedná o relativně malý počet vzorků. V průběhu experimentů se s těmito daty zacházelo podobně jako s daty z přílohy B, tj. vzorky se náhodně rozdělily v rámci třídy na trénovací a testovací v poměru 4:1.

GIT repozitář s daty je dostupný z `https://gitlab.fel.cvut.cz/petrogle/signalclassifierdata.git`, adresář v projektu `dataset_1`.

Číslo třídy	Popis	Počet vzorků
0	Šum	103
1	Pád míčku	52
2	Pád velkého míče	6
3	Pád víčka	21
4	Stoupnutí na plochu	158
5	Sestoupení z plochy	160
6	Kompresor	50
7	Skartovač	55

Tabulka C.2. Počet vzorků každé třídy (zdroj: vlastní data).