TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Simulace rychlostního profilu vozidla

Diplomová práce

Studijní program: N2612 – Elektrotechnika a informatika
Studijní obor: 1802T007 – Informační technologie

Autor práce: **Bc. Matěj Chumlen**
Vedoucí práce: Mgr. Jiří Vraný, Ph.D.

Liberec 2021

TECHNICKÁ UNIVERZITA V LIBERCI
Fakulta mechatroniky, informatiky
a mezioborových studií

Zadání diplomové práce

Simulace rychlostního profilu vozidla

Jméno a příjmení: **Bc. Matěj Chumlen**
Osobní číslo: M19000152
Studijní program: N2612 Elektrotechnika a informatika
Studijní obor: Informační technologie
Zadávající katedra: Ústav nových technologií a aplikované informatiky
Akademický rok: **2020/2021**

Zásady pro vypracování:

1. Seznamte se s rozhraním a datovými sadami Open Street Map, zaměřte se zejména na oblast dopravních informací ovlivňujících rychlost jízdy. Dále prostudujte metody strojového učení použitelné pro generování časových řad a signálů.
2. Proveďte exploratorní analýzu dat získaných v rámci testovacích jízd a zhodnoťte, které z dostupných atributů lze využít pro simulaci rychlosti jízdy po uměle vygenerované trase.
3. Navrhněte, proveďte a vyhodnoťte alespoň dva experimenty, ve kterých na základě uměle vygenerované trasy provedete pomocí zvolených metod strojového učení simulaci rychlosti jízdy vozidla po této trase tak, aby se vytvořený rychlostní profil co nejvíce blížil obvyklému chování řidiče v provozu.

Rozsah grafických prací:	dle potřeby dokumentace
Rozsah pracovní zprávy:	40 – 50 stran
Forma zpracování práce:	tištěná/elektronická
Jazyk práce:	Čeština

Seznam odborné literatury:

[1] GOODFELLOW, Ian, Yoshua BENGIO a Aaron COURVILLE. *Deep learning*. Cambridge, Massachusetts: The MIT Press, [2016]. ISBN 02-620-3561-8.
[2] *Scikit-Learn User Guide* [online]. Scikit-learn developers, 2020 [cit. 2020-10-05]. Dostupné z: https://scikit-learn.org/stable/user_guide.html.
[3] VANDERPLAS, Jacob T. *Python data science handbook: essential tools for working with data*. Sebastopol, CA: O'Reilly Media, 2016. ISBN 978-1491912058.

Vedoucí práce:	Mgr. Jiří Vraný, Ph.D. Ústav nových technologií a aplikované informatiky
Datum zadání práce:	19. října 2020
Předpokládaný termín odevzdání:	16. května 2022

L.S.

prof. Ing. Zdeněk Plíva, Ph.D.
děkan

Ing. Josef Novák, Ph.D.
vedoucí ústavu

V Liberci dne 19. října 2020

Prohlášení

Prohlašuji, že svou diplomovou práci jsem vypracoval samostatně jako původní dílo s použitím uvedené literatury a na základě konzultací s vedoucím mé diplomové práce a konzultantem.

Jsem si vědom toho, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb., o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu Technické univerzity v Liberci.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti Technickou univerzitu v Liberci; v tomto případě má Technická univerzita v Liberci právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Současně čestně prohlašuji, že text elektronické podoby práce vložený do IS STAG se shoduje s textem tištěné podoby práce.

Beru na vědomí, že má diplomová práce bude zveřejněna Technickou univerzitou v Liberci v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů.

Jsem si vědom následků, které podle zákona o vysokých školách mohou vyplývat z porušení tohoto prohlášení.

6. 1. 2022 Bc. Matěj Chumlen

Simulace rychlostního profilu vozidla

Abstrakt

Tato práce řeší problém simulace rychlostního profilu vozidla na libovolné trase. Zaměřil jsem se na možnosti simulace pomocí ML modelu natrénovaného na naměřených datech reálných jízd. Při tvorbě příznaků využívám zdroje otevřených geografických dat. Navržený systém libovolnou vstupní trasu obohatí o dostupná data a na jejich základě provede predikci rychlostního profilu. Na testovacích datech se podařilo dosáhnout střední absolutní chyby 0,8 m/s. Výsledkem této práce je ML model, který uživateli umožňuje vytváření syntetického rychlostního profilu podobného reálnému chování vozidla.

Klíčová slova: syntetický rychlostní profil, simulace rychlostního profilu, geografická data, jízdní data, strojové učení, umělé neuronové sítě, map-matching, tvorba příznaků, geografické příznaky, OSM, OSRM

Vehicle Speed Profile Simulation

Abstract

The focus of this work is to design a solution of the speed profile simulation problem of a vehicle on an arbitrary route. I focus on the possibilities of the utilization of an ML model trained on data from real drives. I use open geographical data for feature engineering. The proposed system enriches input route with available data and makes a speed profile prediction based on the input. This approach achieves mean absolute error of 0.8 mps on test data. The outcome of this work is an ML model which allows user to create synthetic speed profile similar to the real behaviour of the vehicle.

Keywords: synthetic speed profile, speed profile simulation, geographical data, drive data, machine learning, artificial neural networks, map-matching, feature engineering, geographical features, OSM, OSRM

Poděkování

Rád bych poděkoval vedoucímu mé práce Mgr. Jiřímu Vranému Ph.D. za podporu, rychlé reakce na dotazy a vstřícné konzultace. Dále bych rád poděkoval Ing. Karlu Palečkovi Ph.D. za tipy a rady z oblasti strojového učení.

Obsah

Seznam zkratek

OSM	OpenStreetMap
API	rozhraní pro programování aplikací
OSRM	Open Source Routing Machine
ORS	OpenRouteService
MLP	vícevrstvý perceptron
ML	strojové učení
RNN	rekurentní neuronová síť
ESP	elektronický stabilizační program
GPU	grafický procesor

Seznam obrázků

1 Úvod

Zadání mé práce vychází z požadavků firmy Entry Engineering, která se zabývá testováním řídících jednotek automobilů a mobilních aplikací z oblasti automotive. Testování chování aplikací a řídících jednotek při scénářích závisejících na poloze a rychlosti jízdy automobilu (např. varování při vjezdu do zóny s omezenou rychlostí) nabízí dvojí přístup: Generování dat o poloze a rychlosti reálnou jízdou v automobilu za využití palubního počítače, nebo simulace jízdy takovým způsobem, aby data o poloze a rychlosti co nejlépe odpovídala reálnému chování vozidla na trase.

Simulovaný rychlostní profil skýtá oproti naměřenému množství výhod. S dostatečně obecným způsobem simulace je možné generovat rychlostní profily na libovolné trase a v libovolném množství bez nutnosti vydání dalších finančních a personálních prostředků na sběr dat.

V této práci prozkoumávám způsoby, jak by se k takovému účelu daly využít otevřené zdroje geografických dat v kombinaci s naměřenými daty reálných jízd. Tato naměřená data využívám v experimentech jako vstupní data modelů umělých neuronových sítí pro tvorbu syntetických rychlostních profilů. Řešený problém je součástí projektu "CK01000020 – Vývoj generátoru tras GNSS a signálu CANBUS pomocí strojového učení s využitím Software Defined Radio (2020 – 2022), fnancováno z programu Doprava2020 agentury TAČR".

V kapitole 2 popisuji a porovnávám různé zdroje geografických dat s důrazem na otevřenost dat. V další kapitole 3 stručně popisuji potenciálně využitelné architektury umělých neuronových sítí, jejich výhody a nevýhody a dostupné implementace v knihovnách pro strojové učení. Kapitola 4 dává vhled do faktorů ovlivňujících rychlost jízdy vozidla po trase a porovnává je z hlediska dostupnosti v otevřených datech a reprodukovatelnosti ve fázi nasazení systému pro tvorbu syntetického rychlostního profilu.

Pro trénování umělých neuronových sítí jsou důležitá naměřená data reálných jízd. V kapitole 5 popisuji, jaká data jsem pro práci získal. Zároveň popisuji implementaci algoritmu pro jejich zpracování. Následující kapitola 6 popisuje implementaci systému pro tvorbu syntetického rychlostního profilu. Kapitola 7 popisuje předzpracování datové sady pro zpracování modelem, iterativní způsob hledání vhodného modelu neuronové sítě a výsledky.

2 Zdroje geografických dat

2.1 Vymezení pojmů

Úvodem bych rád vymezil některé z pojmů, které jsou pro tuto práci důležité.

Trasou rozumím uspořádanou množinu zeměpisných souřadnic vyjádřených jako uspořádané dvojice hodnot [zeměpisná šířka, zeměpisná délka]. Trasa představuje způsob, jak po silniční síti dosáhnout pomocí vozidla z první (startovní) do poslední (cílové) zeměpisné souřadnice.

Rychlostní profil jsou taková data, která umožní určit průměrnou rychlost v bodě trasy (časová značka, průměrná rychlost). Každé trase lze tedy přiřadit rychlostní profil. Kombinace trasy a rychlostního profilu tvoří jízdu. Rychlostní profil může být naměřený (vzniklý měřením na reálné trase) a nebo syntetický (vzniklý modelováním).

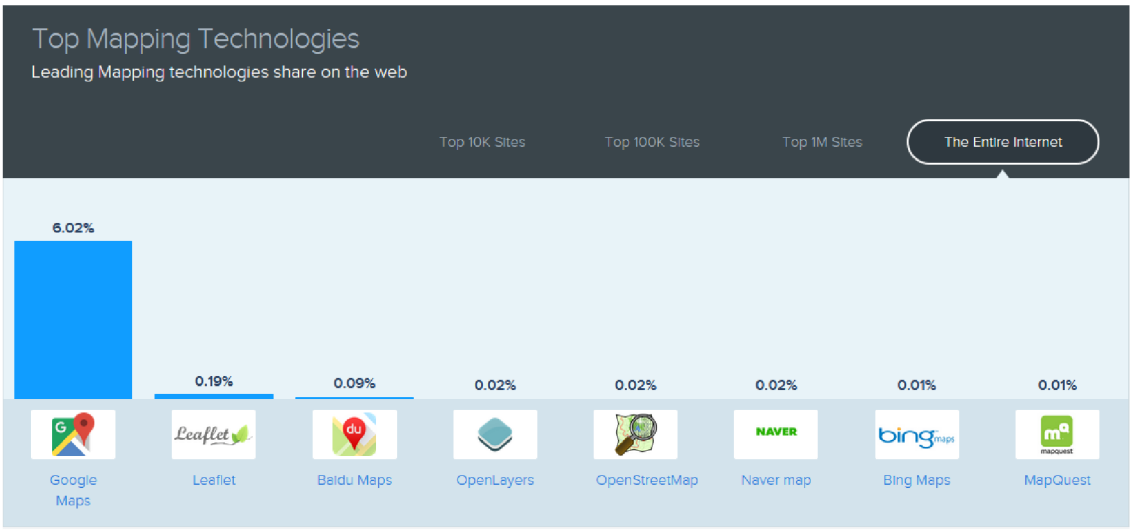
Jízda je trasa doplněná o rychlostní profil. Dvě jízdy mohou mít stejnou trasu, ale lišit se v rychlostním profilu. Dvě jízdy mohou míst stejný rychlostní profil, ale mít jinou trasu.

Vzdálenost na trase označuje kumulativní vzdálenost mezi souřadnicemi trasy. Vzdáleností mezi souřadnicemi rozumím vzdálenost na referenční elipsoidu WGS84 [1]. Ve výjimečných případech, kdy je porovnávána vzájemná vzdálenost dvou bodů je použita aproximace Eukleidovskou nebo Manhattanskou vzdáleností. Tyto případy jsou označeny.

2.2 OpenStreetMap

Základem pro modelování syntetického rychlostního profilu jsou kvalitní zdroje geografických dat. Na grafu 2.1 je vidět, že nejpoužívanějším zdrojem mapových podkladů na internetu je Google Maps. Anekdotickým porovnáním map Google Maps a OpenStreetMap lze dojít k závěru, že mapové podklady Google Maps jsou úplnější, zároveň Google poskytuje rozsáhlé API pro práci nad mapovými podklady včetně navigačních nástrojů. Vzhledem k tomu, že se jedná o proprietární software, který nabízí pouze omezené množství volných dotazů do API a v posledních letech došlo k výraznému zdražení služeb nad rámec volné kvóty [2], je potenciální využití Google Maps pro účely této práce vhodné jen jako doplňkové služby.

Pro otevřenost dat jsem pro tuto práci vybral OpenStreetMap. OpenStreetMap je projekt, který byl založen v roce 2004 Stevem Coastem ve Spojeném království.



Obrázek 2.1: Podíl služeb pro zobrazení mapových podkladů [3]

Projekt je založen na myšlence otevřeného software a motivací pro jeho vznik byly právě nedostatky komerčních mapových služeb poskytovaných společnostmi vyrábějícími satelitní navigace a nebo výše zmíněným Googlem. Mapové podklady OpenStreetMap jsou využívány mimo jiné společnostmi Apple, Amazon, Facebook, Microsoft nebo Wikipedia [4]. Data OpenStreetMap jsou dostupná pod licencí ODbL [5].

2.2.1 Struktura dat

Datová struktura map OpenStreetMap sestává ze čtyř základních prvků [6]. Těmi jsou uzel (Node), cesta (Way), relace (Relation) a značka (Tag). Podrobněji jsou popsány v následujících sekcích. Dalším podstatným prvkem OSM je sada změn (Changeset), což je záznam změn, které v mapových podkladech provedl přispěvatel v čase. Podstata tohoto prvku spočívá v možnosti sledovat změny na otevřených datech a v případě problémů nebo chyb se navrátit k původní verzi. Pro účely této práce je prvek nepodstatný.

Uzel

Uzel reprezentuje bod na Zemi údaji o zeměpisné šířce, zeměpisné délce a jednoznačným identifikátorem (id). Pokud cesta sdílí úrovňové křížení s jinou cestou, musí sdílet také uzel. V opačném případě naopak uzel sdílet nesmí.

Cesta

Cesta je uspořádaný seznam 2 až 2000 uzlů (limit OSM). Sémanticky neodpovídá českému slovu cesta. V OSM je cesta jakákoliv linie. Druhy cest jsou dvojí. Otevřená cesta (cesta, která začíná a končí na různých místech) představující například vozovku, chodník nebo železniční trať, nebo uzavřená cesta: kruhový objezd, oplocení.

Zároveň může uzavřená cesta reprezentovat plochu (area), například budovu, park atp.

Relace

Jedná se o dvojici jedné nebo více značek a uspořádaného seznamu jednoho nebo více uzlů nebo cest. Může představovat významné silniční tahy, přírodní rezervace nebo územně samosprávné celky. Relace může také obsahovat informaci o zákazu odbočení.

Značka

Značka je prvek, který lze přiřadit všem dříve zmíněným prvkům. Značky rozvíjí informace o prvcích. Jedná se o dvouprvková data typu klíč-hodnota. OSM nemá z podstaty standard nastavený autoritou pro hodnoty, které mohou značky obsahovat, avšak existují obecné konvence, které jsou do určité míry uznávány [7].

Soubor běžně používaných konvencí v používání značek je možné nalézt v mapové legendě dokumentace OSM [8]. Česká komunita OSM popisuje standard, který by měl sjednotit postupy při tvorbě mapových podkladů v České republice [9].

2.3 Směrovací knihovny

Pro vytvoření umělého rychlostního profilu mezi libovolnými body je třeba nejprve zjistit, jaká trasa body spojuje. K hledání trasy nad daty OSM existuje několik desítek směrovacích knihoven [10]. Dokumentace OSM poskytuje obsáhlé srovnání těchto služeb na základě podporovaných funkcionalit v kategoriích *pokrytí, způsob dopravy*, *pokročilé možnosti navigace*, *hromadná doprava*, *uživatelské rozhraní* a *informace o službách*. Jako podmínky nutné vzhledem k řešenému problému jsem při volbě zvolil následující vlastnosti: *Globální pokrytí routingové služby*, *hledání tras pro automobily* a *otevřenost zdrojového kódu*. Další podmínkou bylo dostupné online API pro testování.

Tímto sítem prošly čtyři routingové služby: *OpenRouteService*, *YOURS*, *OSRM* a *GraphHopper*.

2.3.1 OpenRouteService

OpenRouteService je projekt spravovaný týmem Institutu geoinformačních studií univerzity v Heidelbergu. Je vydáván pod licencí Apache 2.0. Projekt poskytuje API hledání trasy, *matrix API*, které pro soubor zeměpisných souřadnic vrací matici časů nebo vzdáleností mezi nimi, *isochrones API*, které pro zadané souřadnice a čas vrací oblast, které lze v zadaném čase dosáhnout, API *Geocode*, které pro vstupní souřadnice vrací adresu a obráceně, API *Pois*, které vrací body zájmu v okolí nebo oblasti a *Elevation* API, které pro zadané souřadnice vrací souřadnice obohacené o informaci o nadmořské výšce. Chybí *map-matching* API. Projekt má dobře zpracovanou dokumentaci. Na repozitáři projektu je dostupný Docker kontejner.

Výsledek volání do routing API *https://api.openrouteservice.org/v2/directions/driving-car/geojson* metodou POST s daty

```
{
  "coordinates": [[15.07133,50.77279], [15.06465,50.76297]],
  "attributes": ["avgspeed"], "elevation": "true",
  "extra_info": ["waytype", "surface", "countryinfo"],
  "maneuvers": "true"
}
```

obsahuje seznam zeměpisných souřadnic trasy včetně jejich nadmořské výšky, informace o jednotlivých manévrech na trase včetně informace o úhlu točení, informace o výškovém profilu trasy, průměrných rychlostech a vzdálenostech mezi jednotlivými manévry, povrchu a typu vozovky. Online API umožňuje denně 2000 volání.

Výpočet času trasy [11] je rozdělen na výpočet pro každý segment na základě informace o rychlostním limitu pro jednotlivé typy cest, informace o třídě vozovky a povrchu vozovky. Jako rychlost je zvolen nejnižší z limitů.

2.3.2 YOURS

Projekt YOURS poskytuje velice omezenou dokumentaci. Podle instance na *http://www.yournavigation.org/* využívá pro hledání trasy data z roku 2014 a wiki stránka na OSM zaznamenala poslední úpravu začátkem roku 2018. Projekt je vydáván pod licencí BSD. Dostupná API umožňují hledání trasy a export trasy do GPX nebo WPT.

Výsledek volání do routing API *http://www.yournavigation.org/api/1.0/gosmore.php* metodou GET s parametry *format=geojson*, *flat=50.77279*, *flon=15.07133*, *tlat=50.76297*, *tlon=15.06465*, *v=motorcar*, *fast=1*, *layer=mapnik* a *instructions=1* obsahuje jednoduchý seznam zeměpisných souřadnic trasy bez dalších informací, vzdálenost, odhad času a jednoduchý popis trasy.

Dokumentace neobsahuje informaci o způsobu výpočtu času jízdy.

2.3.3 GraphHopper

GraphHopper je vydáván pod licencí Apache 2.0. Jeho dokumentace je dobře zpracovaná. Projekt nabízí API pro hledání tras, optimalizaci tras (problém obchodního cestujícího), *Isochrones* API, API pro *map-matching*, pro hledání časů a vzdáleností nejrychlejších cest mezi polem zeměpisných souřadnic, *Geocode* API pro převod textové adresy na zeměpisné souřadnice a *Cluster* API, které pro vstupní soubor zeměpisných souřadnic provádí klastrování do požadovaného množství klastrů na základě zvolené vzdálenosti. Webové rozhraní umožňuje 15 000 volání do API zdarma a k dispozici jsou placené programy s vyššími kvótami. Repozitář obsahuje Docker kontejner.

Výsledek volání do routing API *https://graphhopper.com/api/1/route?key=552c7fe8-febd-48d0-95f7-46702000fd23* metodou POST s daty

```
{
  "elevation":true,
  "points_encoded":false,
  "points":[[15.06465,50.76297],[15.06745,50.76937]],
  "vehicle":"car",
  "details":[
    "time",
    "distance",
    "max_speed",
    "road_class",
    "surface"]
}
```

obsahuje seznam zeměpisných souřadnic trasy včetně informace o nadmořské výšce, informace o délce a času průjezdu úseků trati a textové informace o manévrech. Dále informace o povrchu, maximální rychlosti a typu vozovky jednotlivých úseků trasy. Také informace o celkově nastoupaných a sestoupaných výškových metrech.

Trvání průjezdu jednotlivých segmentů je vypočteno na základě informace o typu vozovky a povrchu vozovky. Pro každý povrch a typ vozovky je určena maximální rychlost [12], pokud jsou pro segment trasy určeny obě hodnoty, je zvolena nižší z nich. Data o rychlosti na jednotlivých typech vozovky a površích lze upravit v lokální instanci nastavením faktorů pomocí vlastního profilu.

2.3.4 OSRM

OSRM má obsáhlou a dobře strukturovanou dokumentaci. Je licencován pod licencí BSD. Nabízí API pro hledání trasy, hledání nejbližších uzlů OSM, časů a vzdáleností nejrychlejších cest mezi polem zeměpisných souřadnic, API pro *map-matching*. Dále nabízí *trip service*, což je služba řešící problém obchodního cestujícího a *tile service*, který vrací oblast mapy ve vektorové podobě včetně metadat o rychlostech, časech přejezdů segmentů a dalších [13].

Výsledek volání do routing API metodou GET *http://router.project-osrm.org/route/v1/driving/15.06465,50.76297;15.06745,50.76937* s parametry *steps=true*, *geometries=geojson*, *overview=full*, *annotations=true* obsahuje seznam zeměpisných souřadnic trasy, souřadnice manévrů a jejich vlastnosti, informace o vzdálenosti a době přejezdu mezi jednotlivými body trasy. Součástí výsledku je i seznam identifikátorů uzlů z OSM. Online API umožňuje neomezená volání. Je dostupný kontejner Docker.

Výpočet času trasy se řídí profilem, který je nastaven v souboru formátu LUA [14] a vychází ze značek a informací z OSM. Profily lze přidávat a upravovat v lokální instalaci OSRM.

2.3.5 Porovnání směrovacích knihoven

Projekt YOURS poskytuje velice omezenou dokumentaci a není dlouhodobě udržovaný, což ho s ohledem na budoucí vývoj vyřazuje. Ostatní tři služby, *ORS*, *OSRM* a

GraphHopper Poskytují dostatečně podrobnou dokumentaci, jsou aktuální a stojí za nimi vývojářská komunita.

Tabulka 2.1: Porovnání směrovacích knihoven

	Map-matching API	**Nadmořská výška**	**Docker kontejner**	**Dokumentace**
ORS	NE	ANO	ANO	dobrá
YOURS	NE	NE	NE	špatná
GraphHopper	ANO	ANO	ANO	dobrá
OSRM	ANO	NE	ANO	výborná

Služby *ORS* a *GraphHopper* umožňují rozsáhlejší specifikaci dotazu. Zároveň poskytují informaci o nadmořské výšce. *ORS* využívá pro routování upravenou starší verzi *GraphHopper*. Pro tvorbu rychlostního profilu je důležitá informace o kříženích na trase, jako jsou přechody pro chodce, železniční přejezdy a podobně. Tato inoformace je v OSM uložena jako uzel. *OSRM* jako jediný poskytuje informace o OSM identifikátorech uzlů na trase. Pro dosažení stejného výsledku u zbývajících dvou služeb by bylo možné použít vhodný dotaz do Overpass API.

Výpočet rychlosti ve třech zmíněných službách probíhá obdobným způsobem na základě profilu vozidla definujícího maximální rychlost na různých površích a typech vozovky. Zároveň je možné definováním nového nebo úpravou stávajícího profilu na lokální instanci služby změnit výsledný odhad rychlosti.

Důležitým zjištěním této části rešerše je dostupnost map-matchingových algoritmů pro přichycení vstupních zeměpisných souřadnic k silniční síti OSM. Tato funkcionalita představuje důležitou součást celého systému. Je možné ji využít pro párování naměřených dat s metadaty OSM. Tuto funkcionalitu poskytuje *OSRM* a *GraphHopper*.

Na obrázku 2.2 jsou znázorněny body stejné trasy z vybraných routingových služeb. *ORS*, *GraphHopper* a *YOURS* vrací totožnou množinu zeměpisných souřadnic. *OSRM* (žlutou barvou) vrací jemnější soubor zeměpisných souřadnic včetně souřadnic, které korespondují s umístěním OSM uzlů na trase.

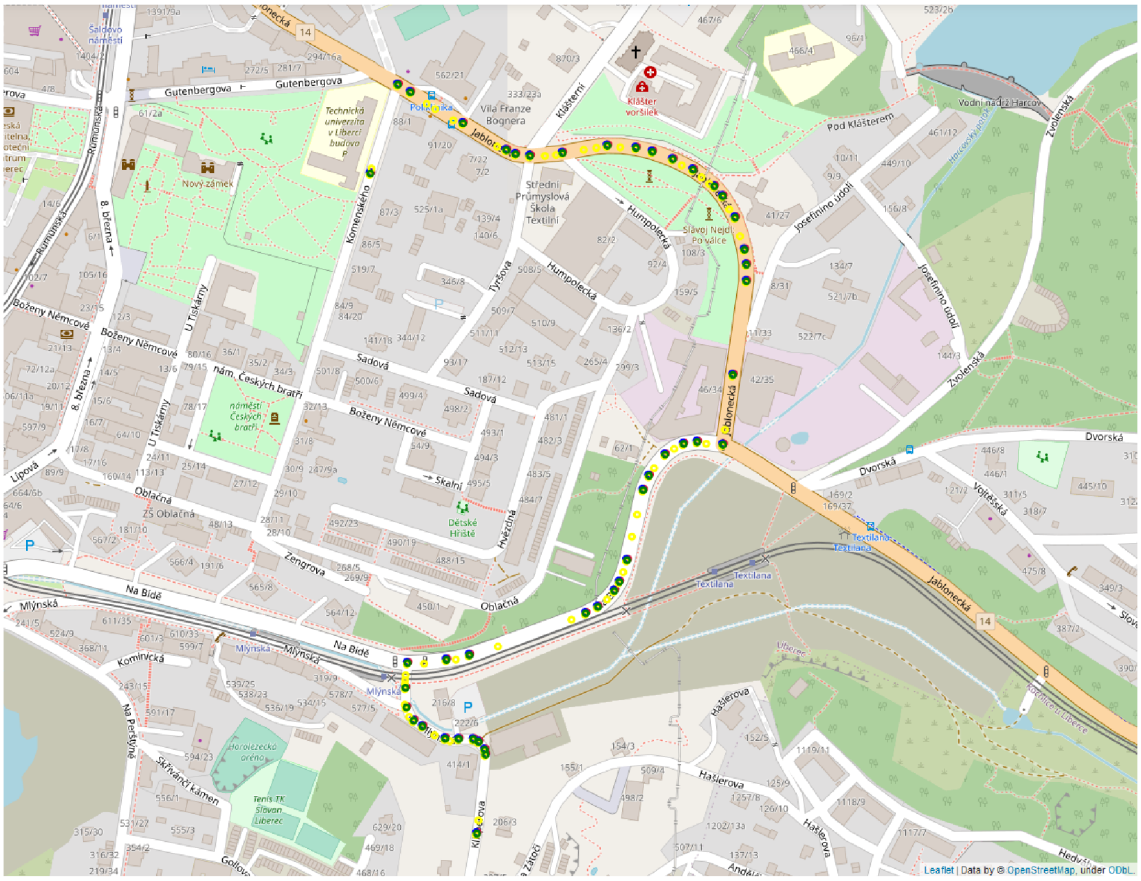
Tabulka 2.2: Celková vzdálenost a čas podle porovnávaných služeb

	vzdálenost [m]	čas [s]
GraphHopper	1142,8	163,5
ORS	1140,2	152,4
OSRM	1139,3	135,9
YOURS	1205.9	131,0

Ačkoliv se nalezená trasa překrývá, celková délka trasy vypočtená jednotlivými službami se liší (viz. tabulka 2.2). To zřejmě souvisí s různými způsoby výpočtu vzdálenosti mezi zeměpisnými souřadnicemi.

V grafu 2.3 jsou zobrazeny průběhy rychlosti na trase z obrázku 2.2 vypočtené routingovými službami.

GraphHopper a *OSRM* vycházejí ze srovnání podobně. Výhodou GraphHopperu jsou poskytovaná data o nadmořské výšce. OSRM má naopak subjektivně lepší dokumentaci a podle GitHubu větší vývojářskou komunitu. Zároveň poskytuje o něco více dat o trase. Proto jsem se rozhodl ho využít pro další vývoj.



Obrázek 2.2: Porovnání nalezené trasy

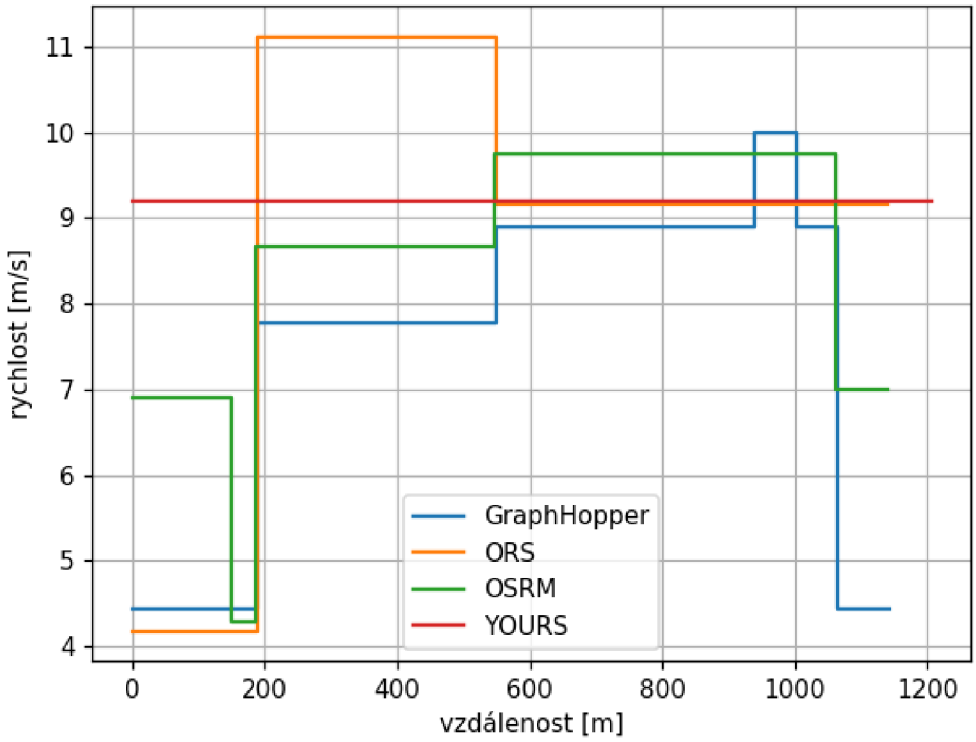
2.4 Overpass API

OSM nabízí pro účely získávání dat Overpass API [15], což je read-only služba, která funguje jako webová databáze s vlastním dotazovacím jazykem Overpass QL. Pro dotazování je možné využít veřejné instance Overpass API a pro testování dotazů webovou aplikaci overpass turbo [16]. Overpass API je také volně dostupná pro lokální instalaci [17] pod licencí Affero GPL v3. Na GitHubu je dostupný projekt, který vytváří z aktuální verze Overpass API Docker kontejner, který je možné využít pro lokální instanci [18].

Aktuální geodata OSM pro OSRM i Overpass API jsou dostupná v komprimované podobě z webu geofabrik.de.

2.5 Open Elevation

Projekt Open-Elevation vznikl jako open-source protiváha Google Elevation API. Jedná se o jednoduchou službu, která po vložení libovolných geodat o nadmořské výšce (očekávaným formátem dat je GeoTIFF) poskytuje jednoduché API, které pracuje nad těmito geodaty a uživateli pro zadané zeměpisné souřadnice vrací je-



Obrázek 2.3: Průběh rychlosti na trase

jich nadmořskou výšku. Součástí projektu je Docker kontejner. Projekt poskytuje předpřipravené skripty pro stažení geodat SRTM, která mají rozlišení 90 metrů na obrazový bod. Od roku 2018 jsou dostupná data s vyšším rozlišením 30 metrů na obrazový bod. Tato data jsem v práci použil [19].

3 Metody strojového učení pro sekvenční data

Problém predikce rychlostního profilu na trase jsem v rámci této práce formuloval jako problém regrese nad sekvenčními daty. Jedná se tedy o podproblém učení s učitelem. Formulace nabízí řadu řešení využívajících strojové učení (ML). Například lineární nebo polynomickou regresi, metodu podpůrných vektorů a mnoho dalších [20]. Pro zpracování této práce jsem si zvolil umělé neuronové sítě. Pro předchozí zkušenost s knihovnou PyTorch a její subjektivně přehlednější API oproti alternativě TensorFlow (Keras) jsem ji zvolil pro další práci.

3.1 Vícevrstvý perceptron

Vícevrstvý perceptron (MLP) je typ dopředného modelu umělé neuronové sítě. Název dopředný vychází z faktu, že pro vstup x poskytuje výstup y, aniž by výsledek závisel na zpětné vazbě výstupu zpět do modelu. Pokud model obsahuje zpětnou vazbu, nazývá se rekurentní (viz 3.3). MLP představuje důležitý základ velkého množství komerčních aplikací neuronových sítí [21]. Mějme pro každý vstup x očekávaný výstup y. Pak MLP představuje funkci $\hat{y} = f(x, \theta)$, kterou procesem učení (změnou parametrů θ) optimalizujeme tak, aby rozdíl, mezi očekávaným výstupem y a skutečným výstupem $\hat{y}$ byl co nejmenší. Jako kritérium rozdílu pro regresní úlohy se používá střední kvadratická chyba nebo střední absolutní chyba [21]. Klasicky model obsahuje několik plně propojených (lineárních) vrstev, výstup každého neuronu vrstvy pak prochází aktivační funkcí (nelinearitou). Doporučenou aktivační funkcí je ReLU [21, 22]. V knihovně PyTorch jsou tyto prvky implementovány pod názvem *nn.Linear* a *nn.ReLU* [23].

3.2 Jednodimenzionální konvoluční neuronové sítě

MLP model nebere v úvahu prostorové uspořádání vstupních příznaků [24], což znamená vzhledem k sekvenční povaze vstupních dat ztrátu potenciálně důležitého kontextu. V doméně počítačového vidění je úspěšně využíván koncept 2D konvolučních vrstev pro práci s obrazovými daty (3D data tvaru [*počet kanálů* × *šířka* × *výška*]) [25]. Analogicky je možné pro zpracování sekvenčních dat (2D data tvaru [*počet kanálů* × *délka sekvence*]) využít 1D konvolučních vrstev. Tohoto přístupu využívá firma Google ve svém modelu pro generování řeči WaveNet [26]. Přístup také využívá *Time-delay neural networks* [21]. Knihovna PyTorch obsahuje implementaci 1D konvoluční

vrstvy pod názvem *nn.Conv1d* [23].

3.3 Rekurentní neuronové sítě

Rekurentní neuronová síť (RNN) je neuronová síť specializovaná na zpracovávání sekvence hodnot $x_1, x_2, \ldots, x_n$. Většina rekurentních sítí je schopna zpracovávat vstupní sekvence proměnlivé délky. Rekurentní neuronová síť má obecně skrytý stav h, který nabývá v čase t hodnoty daných přechodovou funkcí $f(h_{t-1}; \theta; x_t)$ kde h_{t-1} je předchozí hodnota skrytého stavu a θ parametry modelu. Typicky obsahuje model využívající RNN nějakou další vrstvu, která na základě skrytého stavu h produkuje požadovanou výstupní predikci. Skrytý stav h v čase t v podstatě představuje ztrátové shrnutí dosavadní sekvence do času t. Čas t nemusí představovat ubíhání času jako takového. Jedná se obecně o označení pozice v sekvenci [21].

Jednou z nejběžněji používaných RNN architektur je LSTM (Long short-term memory). LSTM dva skryté stavy, cell state a hidden state. Architektura řeší problém toku gradientů v původní implementaci (explodující a mizející gradienty), podobně jako například architektura Resnet (skip connections). LSTM se dají vrstvit pro dosažení lepších výsledků. Běžně ne více, než 5 vrstev [27].

Pro vstupní sekvence, které jsou předem známy v celé délce (predikce není prováděna v reálném čase nad příchozím tokem dat) je možné využít apriorní znalosti dat pomocí tzv. obousměrné (bidirectional) RNN [28]. Obousměrná RNN využívá dva modely. První pracuje s původní sekvencí a druhý s touž sekvencí obrácenou v čase. Výsledný skrytý stav v čase t představuje konkatenaci skrytých stavů obou modelů v tomto čase.

V knihovně PyTorch je implementace LSTM včetně *bidirectional* varianty pod názvem *nn.LSTM* [23].

4 Faktory ovlivňující rychlost jízdy

Faktory ovlivňující rychlost automobilu lze rozdělit do následujících skupin [29]:

- vliv řidiče (věk, pohlaví, řidičské zkušenosti, vůle riskovat, nálada, únava, ...)
- vlivy vnější
  - charakter trasy (délka, účel, naléhavost)
  - vlastnosti automobilu (výkon, brzdná dráha)
  - vlastnosti vozovky (typ vozovky, povrch, rychlostní limity)
- vliv dopravní situace

Velký objem faktorů činí z dopravy pseudonáhodný jev [30].

Pro predikci syntetického rychlostního profilu podobajícího se možnému reálnému rychlostnímu profilu je třeba maximalizovat množství dostupných dat, která ho ovlivňují.

Tato data musí být dostupná ve fázi trénování ML modelu i ve fázi nasazení.

Některé z vlivů je možné snímat během reálné jízdy (zastavení v koloně, počet osob ve vozidle) a ve fázi nasazení modelu je lze modelu poskytnout jako uživatelský parametr. Data o některých vlivech lze získat z otevřených zdrojů (typ vozovky, povrch vozovky atp.) vzhledem k jejich stálosti v čase.

Jiné jsou z otevřených zdrojů obtížně dostupné (dopravní značení), těžko kvantifikovatelné (vůle riskovat) nebo vzhledem k proměnlivosti v čase a náhodnosti těžko simulovatelné ve fázi nasazení modelu (dopravní situace).

Některé z těchto nedostatků lze do jisté míry potlačit využitím většího množství dat. Toho využívá Google v nově implementovaných algoritmech pro odhad času dojezdu [31]. Googlem navržený postup pracuje s historickými daty a zároveň s aktuálními informacemi o dopravě. Zároveň je využito velkého množství různých modelů s ohledem na různou povahu dopravy na různých místech světa.

V této práci bohužel nemám k dispozici taková data ani takové výpočetní prostředky.

Pro úspěšné vytvoření modelu je nezbytné, aby byly minimalizovány náhodné vlivy a vlivy které nelze popsat na základě známých nebo zjistitelných informací. Je tedy vhodné, aby data jízd byla snímána v mírném nebo nulovém silničním provozu.

Dalším důležitým předpokladem je konzistence napříč řidiči, pokud jde o dodržování (a nedodržování) dopravních předpisů. Je zjevně nemožné sebrat data od všech řidičů, avšak lze pracovat s průměrným řidičem. Vycházím z toho, že rychlost vozidla napříč řidiči se řídí normálním rozdělením [30].

Doplňujícím požadavkem, který vyplývá z povahy algoritmů pro přimykání trasy k mapovým podkladům silniční sítě (map-matching), ale zároveň souvisí s dodržováním dopravních předpisů je eliminace náhlého otáčení a couvání vozidla na trase.

Pokud jde o predikci rychlosti pomocí ML modelu natrénovaného na datech zachycených při reálných jízdách, je cílem, aby bylo ve fázi trénování modelu v datech zachyceno maximální množství vlivů ovlivňujících rychlost. Dostupným zdrojům se věnuji v kapitole 2

Zároveň platí opak – aby při sběru dat z reálných jízd bylo dbáno na minimalizaci vlivů, které ve fázi produkce nelze z dostupných dat zjistit.

Metodiku záznamu dat reálných jízd lze shrnout do následujících bodů:

- Jízda v nulovém nebo mírném silničním provozu
- Dodržování dopravních předpisů při jízdě
- Jízda pouze po vozovce a pouze vpřed

Při dodržení této metodiky jsou získaná data konzistentní a vhodná pro využití při trénování ML modelu.

5 Datové sady

Za účelem provedení a vyhodnocení experimentů s ML modely jsem zpracoval dostupná data jízdních záznamů. Jízdní záznamy pocházejí ze tří různých zdrojů (Jízdní záznamy Entry 5.1, jízdy z aplikace GPS Logger a jízdy z knihy jízd TUL 5.2) a liší se konzistencí, kvalitou i formátem. Aby bylo možné s daty pracovat jednotným způsobem, bylo třeba je algoritmicky i manuálně zpracovat do standardizované formy.

5.1 Jízdní záznamy Entry

5.1.1 Specifikace vstupních dat

Společnost Entry Engineering dodala pro účely zhotovení tohoto projektu jízdní záznamy testovacích jízd. Snímání dat probíhá pomocí vlastního záznamového palubního počítače, který má přístup k datům na datové sběrnici CAN automobilu a zároveň data rozšiřuje o další, jako jsou vstupy od uživatele – řidiče. Výhodou je sběr velkého množství hodnot souvisejících s jízdou. Jedná se o 63, respektive 58 údajů v závislosti na verzi a vozidle.

Údaje jsou ukládány do souborů typu CSV a JSON. Každý jednotlivý soubor obsahuje jízdní údaje o délce do dvou minut jízdy. Tyto soubory jsou obsaženy v adresářové struktuře podle času měření *ROK / MĚSÍC / DEN / HODINA*.

Vzorkovací frekvence je přibližně 10 vzorků za sekundu, vzorkování neprobíhá rovnoměrně. Zaznamenávaná data nejsou rozdělena do jednotlivých jízd, ani není začátek a konec jízdy v datech nijak označen. V důsledku toho nelze data snadno rozdělit na jednotlivé jízdy.

5.1.2 Zpracování dat

Pro využití dat při učení ML modelu jsem navrhl algoritmus pro zpracování specifikovaných dat tak, aby výstupem byla vždy jednotlivá jízda. Následující část textu je věnována popisu jeho implementace.

Konkatenace dat

Všechny soubory typu CSV z kořenového adresáře vozu jsou načteny jako DataFrame a spojeny dohromady do jednoho DataFramu pro každý vůz, data jsou setříděna podle časové značky.

Filtrace dat

Klíčovými sloupci pro určení začátku a konce jízdy jsou rychlost podle jednotky ESP vozidla, zeměpisná šířka a délka a sloupec časových značek. Sloupce obsahují velké množství nevalidních hodnot. Nejprve jsou tedy zahozeny všechny řádky obsahující neznámé hodnoty nebo jiné než číselné hodnoty. U sloupců polohy zároveň řádky s hodnotami mimo možný rozsah a nulové hodnoty.

Není cílem modelu pro predikci rychlostního profilu na trase, aby odhadoval délku čekání na semaforu nebo železničním přejezdu. Vzhledem k tomu, že při snímání nebyl dodržen předpoklad o sběru dat v nulovém provozu, vyskytují se v datech i další úseky s nulovou rychlostí, které nepředstavují začátek ani konec jízdy. Pro další zpracování je vhodné tato zastavení odstranit s vědomím, že na základě rychlosti blízké nule je lze zpět doplnit, pokud to bude uživatel požadovat.

Všechny řádky, kde je rychlost podle ESP nulová jsou označeny binární maskou. Následně je tato maska obarvena pomocí algoritmu na barvení oblastí z knihovny OpenCV *connectedComponentsWithStats*, Pro každé zastavení je podle časových značek vypočteno jeho trvání. Pokud je trvání kratší, než je nastavená mez (výchozí nastavení 60 sekund), dojde k odstranění řádků a posunutí časové značky následujících vzorků zpět v čase o čas trvání daného zastavení.

Data jsou vzorkována desetkrát za sekundu, ale poloha je aktualizována pouze jednou za sekundu. Bylo by zřejmě možné polohu v mezidobí interpolovat, avšak vzhledem k přesnosti GPS a k paměťové a výpočetní režii, kterou takto vysoká vzorkovací frekvence obnáší jsem se rozhodl zachovat pouze ta data, kde dochází k vzájemné změně polohy.

Při filtrování lze zvolit ořezání začátku a konce jízdy podle dat o zařazeném zpětném chodu. Tato volba je určena pro situaci, kdy uživatel zpracovává pouze jednu jízdu, při zpracování všech jízd vozu je vhodné tato data zachovat, neboť jsou vhodným příznakem pro segmentaci jednotlivých jízd.

Nakonec filtrace jsou zahozeny řádky se zápornou rychlostí a rychlostí vyšší než 250 km/h podle ESP. Uživatel je informován o relativním množství řádků, které byly vyfiltrovány.

Segmentace jízd

Vyfiltrovaná data stále představují konkatenaci všech jízd včetně zastavení, která nebyla odstraněna. Dále je tedy provedena segmentace na jednotlivé jízdy. Filtrovaná data jsou segmentována podle několika kritérií:

- Doba mezi následujícími vzorky – pokud doba mezi dvěma následujícími vzorky přesáhne stanovenou mez (výchozí hodnota 5 sekund), je označen předěl mezi jízdami. Mez je stanovena tak, aby nedocházelo k falešně pozitivním označením v důsledku chybějících vzorků po filtraci.
- Vzdálenost mezi následujícími vzorky – pokud je vzdálenost, mezi následujícími vzorky vyšší než mez (výchozí hodnota 100 metrů), je označen předěl mezi jízdami.

- Zařazení zpětného chodu – Předpokládá se, že během jedné jízdy nenastává situace, kdy by vůz musel couvat. Zpětný chod tedy souvisí s manipulací s vozidlem při parkování na začátku nebo konci jízdy.

Všechny meze jsem zvolil po manuálním testování výsledků segmentace tak, aby byl dosažen optimální poměr mezi zahozenými daty z důvodu příliš krátké jízdy (k dalšímu zpracování jsem použil pouze jízdy delší, než 60 vzorků – typicky pojíždění po parkovišti) a falešně stanovenými konci jízd v důsledku ztráty signálu například při průjezdu podjezdem nebo tunelem.

5.2 Další datové zdroje

Oproti jízdním záznamům Entry z 5.1 jsou následující datové zdroje konzistentnější a zároveň jsou naměrené jízdy ukládány jednotlivě. Jejich zpracování je tedy snadnější. Zároveň jsou v práci použity jen okrajově a jsou tedy popsány stručněji.

5.2.1 Jízdy z aplikace GPS Logger

Za účelem zvýšení objemu dostupných dat pro trénování jsem sbíral data pomocí mobilní aplikace GPS Logger na šesti mobilních telefonech. Sběr probíhal souběžně na více zařízeních podle jejich aktuální dostupnosti. Data z aplikace jsou ukládána do souboru formátu CSV se vzorkovací frekvencí 1 Hz. Každý soubor odpovídá jedné jízdě a není tedy nutná segmentace. Při zpracování jsou pouze odstraněny vedoucí a koncové vzorky s nulovou rychlostí zachycené mezi začátkem záznamu a rozjezdem vozidla. Následně jsou namapovány odpovídající sloupce do standardního formátu. Řádky výstupu jsou ve tvaru [časová značka, zeměpisná šířka, zeměpisná délka]. Celkově jsem zaznamenal 116 jízd o celkové délce 3761 km.

5.2.2 Jízdy z knihy jízd TUL

Technická univerzita v Liberci využívá ke správě vozového parku aplikaci Webdispečink, které poskytuje webové API pro přístup a modifikaci dat. Za pomoci knihovny Zeep pro Python jsem vytvořil skript, který pro zvolené období stáhne všechny záznamy jízd všech dostupných vozů.

Jízdy jsou uloženy v souboru formátu CSV. Každý řádek jízdního záznamu je ve formátu [časová známka, zeměpisná šířka, zeměpisná délka]. Oproti předchozím dvěma sadám je vzorkovací frekvence nižší. To zřejmě souvisí s rozdílným účelem sběru dat. Data jsou navzorkována nerovnoměrně s mediánem 5 sekund. Celkový objem dostupných jízd je těžko vyčíslitelný. Data z tohoto zdroje nebyla dále v práci použita s ohledem na nízkou vzorkovací frekvenci a nutnost dotazování do databáze s limitovaným počtem dotazů za den. Jedná se o potenciální zdroj dat pro budoucí testování.

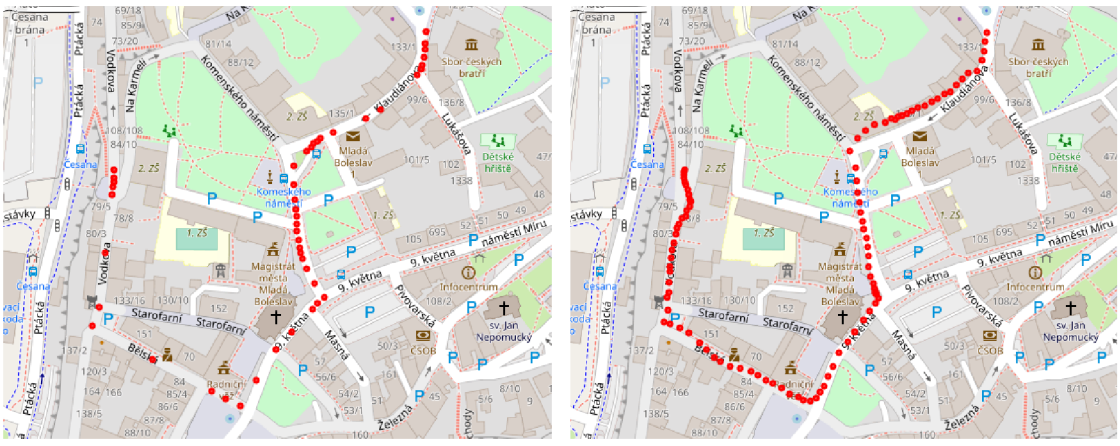
6 Proces predikce

6.1 Vstupní data

Mým cílem v průběhu této práce bylo navrhnout systém, který pro libovolnou vstupní trasu bude schopen predikovat takový rychlostní profil, který by nebylo možné odlišit od možného reálného rychlostního profilu, který byl naměřen pomocí palubního počítače.

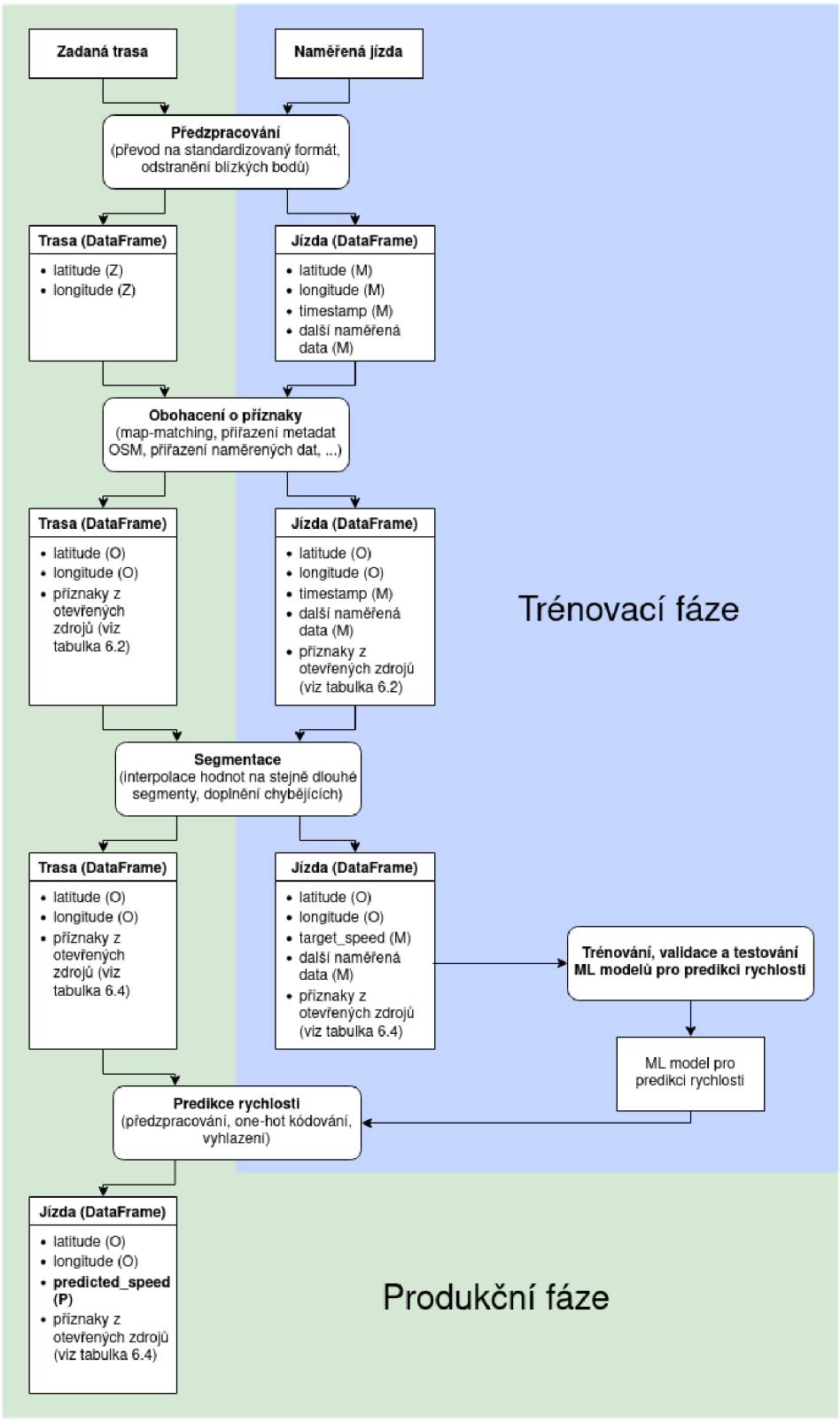
Diagram aktivit navrženého systému je popsán na obrázku 6.2. Následující kapitola je věnována podrobnému popisu postupu, jakým je dosaženo predikce rychlostního profilu.

Vstup systému je vždy trasa. Může se jednat buď o zadanou trasu v produkční fázi, nebo naměřenou jízdu ve fázi trénování. Doporučeným způsobem zadávání trasy v produkční fázi je export trasy z Google Maps do formátu KML. Na obrázku 6.1 Je vyznačena trasa se stejným startovním a cílovým bodem jako výsledek exportu z Google Maps (a) a jako výsledek měření jízdy vozidla palubním počítačem (b).



(a) (b)

Obrázek 6.1: Porovnání vstupních dat



Obrázek 6.2: Diagram predikce rychlostního profilu. Atributy jsou podle původu označeny jako M – naměřené, Z – zadané, O – z OSM a P – predikované.

6.2 Předzpracování vstupní trasy

6.2.1 Převod na standardizovaný formát

Vstupem celého systému pro predikci rychlosti může být soubor formátů KML, CSV, GPX, TCX a GeoJSON. Tento soubor představuje jednu trasu. V prvním kroku dochází k převodu jmenovaných formátů do struktury pandas.DataFrame. Toto sjednocení je nutné, vzhledem k rozdílným názvům sloupců napříč naměřenými daty ve fázi tréninku i vstupními daty v produkci. Výsledkem převodu je standardizovaný DataFrame, který zaručeně obsahuje sloupce *longitude* a *latitude* určující body trasy. Pokud je vstupní soubor formátu CSV, jsou zachovány všechny ostatní sloupce. Dalším povinným sloupcem je sloupec *timestamp*. Pokud se sloupec ve vstupních datech nachází, je zachován (jedná se o naměřenou jízdu). Pokud ne, je inicializován na hodnotu *NaN*.

6.2.2 Odstranění blízkých bodů

Při sběru dat dochází k zastavení vozidla v situacích, kdy si to vyžadují dopravní předpisy - značka "Stůj, dej přednost v jízdě!" (dále STOP), světelné signalizační zařízení (dále semafor) atd. Dále pak při jízdách, kde nebyl dodržen předpoklad volného dopravního toku a vozidlo je nuceno zastavovat v kolonách a nakonec z vůle řidiče (nejčastěji zastavení na začátku, nebo konci jízdy).

Všechny tyto situace v kombinaci s nepřesností měření zeměpisné polohy vytváří v datech shluky bodů. Jedná se o šum, který je pro další zpracování problematický a může způsobit selhání predikce. Proto dochází k filtraci bodů. Vstupní body jsou filtrovány na základě vzdálenosti. Algoritmus filtrace je popsán v kódu 6.1

```
pnts = list bodu
thr = 1  # vzdalenost v metrech, vychozi hodnota 1 metr

pro i od 0 do velikost(pnts) - 2
  dst = vzdalenost(pnts[i], pnts[i+1])
  dokud dst < thr
    odstran(pnts[i+1])
    dst = vzdalenost(pnts[i], pnts[i+1])
```

Kód 6.1: Algoritmus filtrace blízkých bodů v pseudokódu

Pokud dojde při filtraci ke ztrátě více než 10 procent dat, je uživatel varován před potenciálně poškozenými daty.

Odstranění blízkých bodů řeší problém naměřených dat. Vstupní data zadané trasy (ručně nebo z Google Maps by podobným hendikepem trpět neměla (vzhledem k prahu 1 metr) a tedy zůstanou algoritmem nezměněna.

6.3 Obohacení o příznaky

Zdrojem většiny příznaků jsou metadata bodů geometrie OSM. Vstupní body trasy a body odpovídající trasy v OSM jsou zpravidla odlišné. Poloha naměřených bodů je poloha vozu v době měření. Poloha bodů OSM je motivována zachycením optimálně jemné geometrie s ohledem na datovou velikost mapových podkladů a uživatelskou přívětivost.

Z toho pramení zásadní problém ve snaze využít metadat z OSM, neboť neexistuje jednoznačné přiřazení naměřených bodů a bodů OSM. Nabízí se dvojí řešení:

1. Zachovat naměřené body a přiřadit jim body z OSM
2. Zachovat body OSM a doplnit je o naměřené body

Pozn.: Naměřené body jsou v tomto kroku významově zaměnitelné s uživatelsky zadanými body ve fázi produkce.

Při tvorbě mapových podkladů OSM je využito fotogrammetrie, vícera mapových podkladů, uživatelských úprav vycházejících z místní znalosti a více měření pro jednotlivou trasu v místech, kde nelze ze satelitních snímků určit polohu vozovky (obr. 6.3) [32]. Zároveň narozdíl od naměřených dat nepředstavují problém části trasy, kde došlo ke ztrátě signálu a polohových dat (tunely, podjezdy). Rozdíl mezi naměřenou trasou (modře) a trasou podle OSM (červeně) je vidět na obrázku 6.4.

Po zvážení jsem dospěl k závěru, že vzhledem k způsobu mapování OSM lze prohlásit body geometrie OSM za bližší reálné trase než vstupní body. Zároveň lze pro namapování naměřených bodů na geometrii OSM použít existujících map-matchingových algoritmů (viz 2.3).

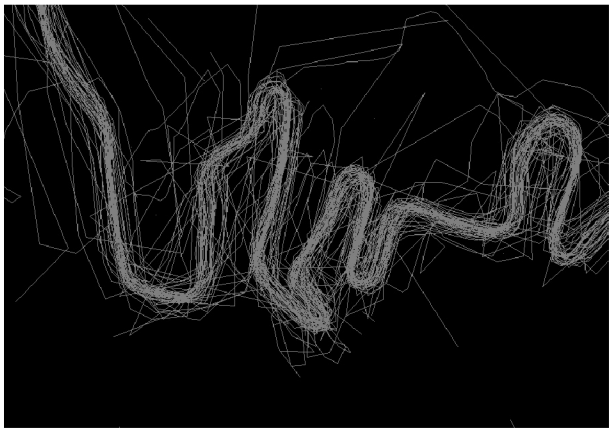
6.3.1 Map-matching

V prvním kroku jsou za pomoci map-matchingového algoritmu knihovny OSRM naměřené body přimknuty k silniční síti OSM. OSRM map-matching API je odeslán požadavek a návratovou hodnotou je objekt GeoJSON. Pokud jsou vstupní data nepřesná, může dojít k přimknutí pouze části bodů. V tomto případě je uživatel varován. Odpověď také obsahuje míru úspěšnosti přimknutí (confidence) v intervalu 0-1. Pokud je hodnota nižší než 0,8 (hodnotu prahu jsem stanovil ručním zkoumáním úspěšných a neúspěšných přimknutí), je uživatel varován.

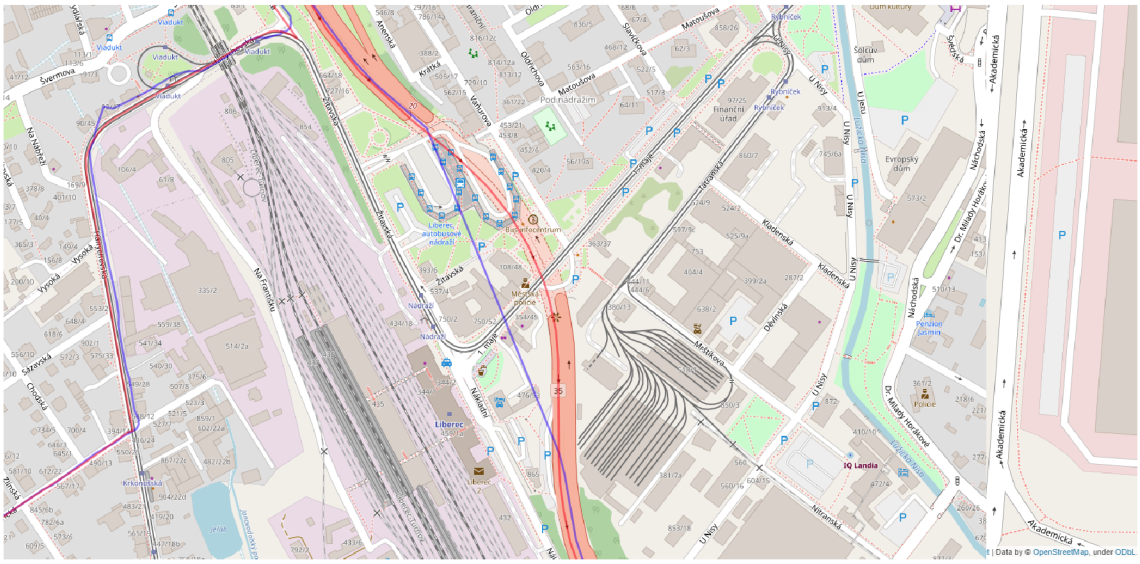
Objekt GeoJSON není vhodný pro přímé další využití, jelikož obsahuje řadu nepotřebných nebo duplicitních dat [33]. Je proto převeden do formátu DataFrame. Ten obsahuje všechny unikátní body OSM a body, k nimž byly přimknuty naměřené body. Tyto dvě množiny jsou zpravidla disjunktní.

6.3.2 Přiřazení identifikátorů uzlů OSM

Z OSRM odpovědi lze získat identifikátory bodů z OSM, které se na trase nachází. Identifikátory jsou nezbytné pro získávání metadat v následujících krocích. Odpověď



Obrázek 6.3: Více průjezdů stejnou trasou [32]



Obrázek 6.4: Ztráta signálu v tunelu

však neobsahuje vazbu mezi identifikátorem a bodem trasy. Platí, že jedna jízda může obsahovat stejný bod OSM vícekrát (opakovaný průjezd). Zároveň platí, že odpověď OSRM může obsahovat i identifikátor, který trasa neobsahuje. Z těchto důvodů je přiřazení bodů trasy a identifikátorů OSM provedeno následovně:

1. Pro každý identifikátor bodu z množiny všech vrácených z OSRM je proveden dotaz na metadata do Overpass API. Z metadat jsou získány ty značky, která obsahují relevantní data pro další použití (dopravní značení, železniční křížení atp.) Výsledkem je DataFrame obsahující polohu bodů OSM, identifikátory OSM a jejich metadata.

2. K DataFramu A vytvořeném transformací objektu GeoJSON OSRM při map-matchingu je připojen dataframe B z bodu 1) a to tak, že každému bodu bodu z A je vytvořen rádius přibližně 0.5 m a pokud do tohoto rádia náleží bod z

B, jsou mu přiřazeny jeho metadata. Hodnota rádia je zvolena experimentálně a její nutnost vychází z chyby vzniklé zaokrouhlením daným formátem GeoJSON. Poloha obsažená v OSM takto zaokrouhlena není. Ke spojení je využita funkce *smerge* knihovny GeoPandas.

6.3.3 Přiřazení naměřených bodů a příslušných hodnot

Vazební dvojice přimknutých bodů a naměřených bodů jsou v oddělené datové struktuře a bohužel neobsahují jednoznačnou referenci na body trasy. Zároveň naměřené body nejsou při map-matchingu použity všechny. To činí hledání spojení mezi nimi složité, především vzhledem k tomu, že pokud vozidlo projede v rámci jedné jízdy stejné místo vícekrát, zvyšuje se šance, že dva naměřené body z různých průjezdů dostanou přiřazeny velmi blízké přimknuté body. Tento problém se týká především reálně naměřených (ne uživatelsky vložených) bodů, které jsou zpravidla hustší. Dvojice bodů jsou vytvořeny takto: Pokud je manhattanská vzdálenost přimknutého bodu a bodu z vazební tabulky menší než $1 \cdot 10^{-9}$ a zároveň je rozdíl kumulativní vzdálenosti na naměřené trase a přimknuté trase maximálně 5%, jedná se o stejné body.

Metoda varuje uživatele, pokud nedošlo ke vzájemně jednoznačnému přiřazení a pokud nebylo použito více než 10 naměřených bodů, což naznačuje selhání.

Na základě tohoto přiřazení jsou do DataFramu přidány sloupce s naměřenými veličinami (*engine_rpm*, *engine_gear*, *timestamp* atp.).

Pokud je mezi některými po sobě následujícími body s časovou známkou více než 50 sekund, ukazuje to na selhání map-matchingu, ztrátu signálu GPS nebo jinak poškozená data a zpracování končí výjimkou.

Podobně pokud přes výše zmíněná protiopatření dojde k přimknutí bodu dřívějšího průjezdu, způsobí to v datech negativní rozdíl času – jízdu zpět v čase a zpracování končí výjimkou.

6.3.4 Přiřazení metadat vozovky

Dále jsou přidány metadata o vozovce (entita Way v OSM). Pro všechny unikátní identifikátory vozovky dojde k dotazování do Overpass API. Výsledky jsou převedeny do vhodného formátu DataFrame a pomocí funkce merge jsou metadata namapována podle klíče *way_id*.

6.3.5 Označení křižovatek

V DataFramu jsou následně označeny body, které v OSM představují křižovatku. OSM neobsahuje metadata s podobnou informací přímo, ačkoliv je to jedna z navrhovaných vlastností komunitou [34]. Výjimku z tohoto pravidla představují body kruhového objezdu, které jsou označeny klíčem ‘intersection‘ s hodnotou ‘roundabout‘. Vedle kruhových objezdů jsou jako křižovatka označeny ty body, které představují uzly OSM (entita Node), jež náleží více než jedné cestě (entita Way). Dojde k rekur-

zivnímu dotazu do Overpass API, který získá metadata o všech cestách, jimž uzel náleží.

Cesta ve smyslu základní entity Way v OSM označuje významově menší celek, než je vhodné pro účel označení křižovatky. Například pokud trasa překračuje hranici katastru obce nebo jiného správního celku, mění se identifikátor cesty Way. V takovém případě je podle výše zmíněného pravidla bod označen jako křižovatka, ačkoliv uzel, jenž obě cesty sdílí nenáleží křižovatce. Tomuto problému je zabráněno sjednocením cest, kterým uzel náleží podle hodnot klíče ref a name v metadatech cest. Pokud je i po tomto sjednocení počet cest vyšší než 1, je uzel označen klíčem *intersection* s hodnotou *indistinct*. V dalším kroku jsou křižovatky dále odlišeny s ohledem na způsob, jakým jimi trasa prochází. Odlišení vychází z předpokladu, že vzájemný typ vozovek (v OSM klíč *highway*) v křižovatce naznačuje rozložení předností v jízdě. Například křížení servisní cesty a dálnice nebude mít vliv na změnu rychlosti vozidla jedoucího po dálnici. Naopak pokud vozidlo přejíždí z cesty v obytné čtvrti na hlavní tah, pravděpodobně zpomalí. Toto odlišení jsem stanovil hierarchicky podle priorit v tabulce 6.1. Čím nižší je priorita, tím pravděpodobnější je právo přednosti v jízdě. Podle tohoto předpokladu nabývá klíč intersection následujících hodnot:

- indistinct, pokud jsou všechny vozovky v křižovatce stejné priority.
- side_to_side, pokud jsou příjezdová a odjezdová vozovka trasy stejné priority a alespoň jedna vozovka v křižovatce je nižší priority.
- main_to_main, pokud jsou příjezdová a odjezdová vozovka trasy stejné priority a žádná vozovka není nižší nebo stejné priority.
- main_to_side, pokud je příjezdová vozovka nižší priority, než odjezdová vozovka.
- side_to_main, pokud je příjezdová vozovka vyšší priority, než odjezdová vozovka.

Tabulka 6.1: Priority křižovatek podle *way_type*

way_type	**Priorita**
motorway	1
motorway_link	2
trunk	3
trunk_link	4
primary	5
primary_link	6
secondary	7
secondary_link	8
tertiary	9
tertiary_link	10
residential	11
living_street	12
service	13
unclassified	14

Doposud jsou jako kruhový objezd označeny pouze body, které odpovídají bodům z OSM. Body přimknuté, které se nachází na kruhovém objezdu takto označeny nejsou a tedy vznikají mezery. Tento problém jsem vyřešil tak, že po binární masce označující body trasy náležící kruhovému je postupně posouváno jádro [1, 0, 1] a [1, 0, 0, 1]. Pokud dojde k překryvu tak, že součet součinu masky a jádra je 2, je nahrazen daný segment binární masky jedničkami. Tímto způsobem jsou tedy detekovány mezery délky jedna a dva. Délky jader jsem zvolil na základě inspekce délky mezer v naměřených datech.

6.3.6 Nadmořská výška

Pro všechny body DataFramu trasy je proveden dotaz do Open-Elevation API (v dávkách à 2000 bodů) a je jim přiřazena nadmořská výška.

6.3.7 Postproces příznaků

Metadata získaná z OSM mohou obsahovat exotické hodnoty, které se vyskytují velmi zřídka. To vede k nárůstu dimenzionality příznaků, kdy binární příznaky exotických hodnot představují řídké sloupce, ve kterých se daná vlastnost objevuje například jednou za 1000 kilometrů. Dále obsahují hodnoty, které nemají pro účel predikce rychlostního profilu smysl. Postproces má za cíl sloučit hodnoty, které jsou významově stejné a odebrat hodnoty, které nenabízí potenciální zlepšení výsledku predikce. To vše tak, aby byla zachována rovnováha mezi dimenzionalitou příznaků a ztrátou informací.

Například klíč *crossing* může obsahovat hodnoty *zebra*, *marked*, *unmarked* [35], které se významově překrývají s hodnotou *uncontrolled*, tedy přechod pro chodce bez semaforu. Příkladem hodnoty, která nemá vliv na rychlost může být situace, kdy klíč *direction* nabývá hodnoty *backwards* u bodu reprezentujícího semafor a ovlivňuje tedy jízdu v opačném směru. Výsledná struktura dat je popsána v tabulce 6.2. Řádky označené hvězdičkou představují veličiny, které jsou měřeny při sběru dat. Hodnoty těchto veličin jsou dostupné pouze ve fázi trénování. Ve fázi nasazení systému představují predikované veličiny.

Hodnoty výstupního DataFramu jsou zvoleny tak, aby zachovávaly maximální množství informací získaných ze všech zdrojů. Předpokládá se, že výzkumník může pracovat nad těmito daty, aniž by potřeboval přístup k endpointům použitých API a jsou mu všechna data dostupná za cenu větších nároků na úložiště. Postproces příznaků způsobuje ztrátu některých dat, ale je volitelnou součástí zpracování.

6.4 Segmentace trasy

Trasa obohacená o příznaky z předchozího procesu obsahuje body vzájemně nerovnoměrně vzdálené. V místech s vyšší hustotou bodů trasy (např. ve městě) jsou následující body vzdáleny jednotky metrů a v místech s nižší hustotou (např. dálnice) desítky i stovky metrů. Smyslem segmentace je převzorkování těchto bodů takovým

Tabulka 6.2: Data po obohacení o příznaky

Název sloupce	**Popis**	**Zdroj**	**Hodnota**
latitude	z. šířka bodu	OSRM map-matching	float
longitude	z. délka bodu	OSRM map-matching	float
speed_osrm	rychlost podle OSRM	OSRM map-matching	float
way_id	identifikátor cesty OSM	OSRM map-matching	int
original_latitude	z. šířka naměřeného (zadaného) bodu před přimknutím	uživatel	float
original_longitude	z. délka naměřeného (zadaného) bodu před přimknutím	uživatel	float
timestamp *	časová značka bodu	uživatel	int
elevation	nadmořská výška bodu	Open-Elevation	int
node_id	identifikátor uzlu OSM	Overpass / OSM	int
node:highway	hodnota klíče highway	Overpass / OSM	enum
node:crossing	hodnota klíče crossing	Overpass / OSM	enum
engine_rpm *	otáčky motoru	uživatel	float
engine_gear *	zařazený rychlostní stupeň	uživatel	int
brake_pressure *	tlak brzdového pedálu	uživatel	float
throttle_pedal *	tlak plynového pedálu	uživatel	float
BM_links *	sepnutí levé směrovky	uživatel	int
BM_rechts *	sepnutí pravé směrovky	uživatel	int
way_type	typ vozovky	Overpass / OSM	enum
way_maxspeed	maximální povolená rychlost	Overpass / OSM	int
way_surface	povrch vozovky	Overpass / OSM	enum
node_tags	tagy uzlu OSM	Overpass / OSM	dict
way_tags	tagy cesty OSM	Overpass / OSM	dict
intersection	typ křižovatky	odvozeno z dat OSM	enum
node:railway	hodnota klíče railway	Overpass / OSM	enum
node:stop	uživatelsky definované zastavení vozidla	uživatel	int

způsobem, že výstupní data budou reprezentovat stejně dlouhé úseky (segmenty) trasy. Pro vhodnost vzhledem k dalšímu zpracování jsem zvolil délku segmentu 1 metr. Taková reprezentace je vhodná pro využití při predikci rychlosti, neboť obsahuje implicitní informaci o vzdálenosti.

Proces segmentace je následující: Každému bodu trasy s časovou značkou je přidělena průměrná rychlost podle kumulativní vzdálenosti od předchozího bodu s časovou značkou. Tato rychlost je uložena jako *target_speed* a představuje hodnotu pravdy (ground truth) z pohledu následné predikce.

Jelikož způsob segmentace vede ke ztrátě některých informací, je dále doplněno maximální množství neznámých hodnot. Všem bodům je vypočten dopředný azimut, bodům s neznámou maximální úsekovou rychlostí je přiřazena hodnota modu v celém souboru dat 50 km/h a bodům s neznámým povrchem obdobným způsobem asfalt [36].

Jsou zahozeny sloupce, které nejsou pro další zpracování potřebné.

Příznaky lze sémanticky rozdělit na úsekové – popisující okolnost trasy, která je platná po nějakou vzdálenost (povrch vozovky, typ vozovky) a bodové – hodnoty, které reprezentují okamžitou vlastnost (dopravní značení, úrovňové křížení, přechod pro chodce). Data jsou rozdělena do dvou DataFramů podle tohoto dělení.

DataFramu obsahujícímu bodové příznaky je vytvořen index, jehož hodnoty představují zaokrouhlenou kumulativní vzdálenost bodů na trase. Pokud dojde k zaokrouhlení více bodů na stejnou kumulativní vzdálenost, je ponechán první z nich. Tento postup je ztrátový, ale situace nastává zřídka.

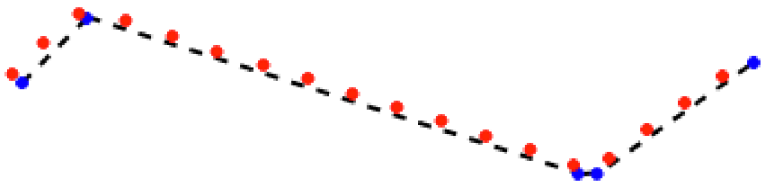
DataFrame s úsekovými příznaky je převzorkován pomocí interpolace. Pokud je dostupná vhodná metoda interpolace ve knihovně scipy, jsou hodnoty interpo-

lovány pomocí scipy. Pro číselné hodnoty je použita knihovní funkce interp1d ze scipy.interpolate, pro kategorické proměnné obdobně s využitím knihovní funkce LabelEncoder z sklearn.preprocessing. Metody interpolace pro jednotlivé sloupce jsou v tabulce 6.3.

Zvláštním případem je interpolace geometrie. Pro měření vzdálenosti je použit referenční elipsoid WGS84. Vzdálenost není měřena mezi novými body přímo, ale po trase původních bodů. Jako důsledek nejsou body po dvojicích vzdáleny vždy 1 metr. Interpolace geometrie je znázorněna na obrázku 6.5 (modré body – původní geometrie, červené body – interpolace).

Výsledné dva DataFramy jsou spojeny po řádcích pomocí funkce merge. Jako klíč je použita kumulativní vzdálenost. Sloupec vzdálenosti je následně zahozen, neboť vzdálenost je reprezentována indexem.

Na závěr jsou v datech označeny binárním příznakem začátek a konec jízdy (první a poslední řádek) a přidány sloupce první diference azimutu a nadmořské výšky.



Obrázek 6.5: Interpolace polohy při segmentaci

Tabulka 6.3: Typy interpolace při segmentaci

Sloupec	**Typ interpolace**
target_speed	lineární
speed_osrm	lineární
way_maxspeed	nejbližší
elevation	lineární
fwd_azimuth	předchozí
engine_rpm	lineární
engine_gear	předchozí
brake_pressure	lineární
throttle_pedal	lineární
BM_links	předchozí
BM_rechts	předchozí
way_type	nejbližší
way_surface	nejbližší

6.5 Predikce rychlosti pomocí ML modelu

Modul speed_predictor obsahuje všechny nástroje související přímo s predikcí rychlosti. Trénovací a testovací skripty a produkční modely včetně vah. Vstupními daty jsou segmentovaná data a výstupem je predikovaná rychlost. V produkční fázi je API co nejjednodušší tak, aby byl vývojář zbytku systému co nejvíce odstíněn od logiky predikce. Příklad predikce je popsán v kódu 6.2

Tabulka 6.4: Data po segmentaci

Název sloupce	Popis	Zdroj	Hodnota
latitude	z. šířka bodu	OSRM map-matching / OSM	float
longitude	z. délka bodu	OSRM map-matching / OSM	float
speed_osrm	rychlost podle OSRM	OSRM map-matching / OSM	float
target_speed	vypočtená rychlost podle časových značek	odvozeno z uživatelských dat	int
elevation	nadmořská výška bodu	Open-Elevation	int
fwd_azimuth	dopředný azimut z bodu	odvozeno	int
node:highway	hodnota klíče highway	Overpass / OSM	enum
node:crossing	hodnota klíče crossing	Overpass / OSM	enum
target_engine_rpm	otáčky motoru	uživatel	float
target_engine_gear	zařazený rychlostní stupeň	uživatel	int
target_brake_pressure	tlak brzdového pedálu	uživatel	float
target_throttle_pedal	tlak plynového pedálu	uživatel	float
target_BM_links	sepnutí levé směrovky	uživatel	int
target_BM_rechts	sepnutí pravé směrovky	uživatel	int
way_type	typ vozovky	Overpass / OSM	enum
way_maxspeed	maximální povolená rychlost	Overpass / OSM	int
way_surface	povrch vozovky	Overpass / OSM	enum
node_tags	tagy uzlu OSM	Overpass / OSM	dict
way_tags	tagy cesty OSM	Overpass / OSM	dict
node:intersection	typ křižovatky	odvozeno z dat OSM	enum
node:railway	hodnota klíče railway	Overpass / OSM	enum
node:stop	uživatelsky definované zastavení vozidla	uživatel	int
start_stop	binární označení začátku a konce jízdy	odvozeno	int
azimuth_diff	první diference dopředného azimutu	odvozeno	float
elevation_diff	první diference nadmořské výšky	odvozeno	float

```
# Importuj modul
import speed_predictor.models as models

# Vytvor objekt prediktoru
model = models.SpeedPredictorSingle()

# Predikuj, vstupem DataFrame ze segmentace
predicted_speed = model.predict(df)

# Uloz predikci do sloupce "predicted_speed"
df["predicted_speed"] = predicted_speed
```

Kód 6.2: Ukázka predikce rychlosti v jazyce Python

Každý prediktor má přiřazen způsob předzpracování, scaler, ML model s předtrénovanými vahami a způsob postprocessingu.

6.5.1 Roura predikce

Při inicializaci modelu je inicializován jemu náležící ML model a načteny váhy. Model je přesunut na GPU, pokud je dostupná.

Metadata obsažená v OSM nejsou vzhledem k otevřené koncepci OSM omezena z pohledu možných hodnot, jichž mohou jednotlivé klíče nabývat. Při předzpracování vstupních dat je podle specifikace v souboru *data_structure.json* vymezeno, jaké sloupce budou zpracovávány a jakých hodnot mohou nabývat kategorické proměnné.

Tento krok je nutný, jelikož model pro predikci očekává pevně daný počet příznaků a kategorické příznaky jsou kódovány pomocí one-hot kódování.

Po zakódování kategorických příznaků je ze stejného důvodu provedena kontrola přítomnosti všech sloupců a v případě chybějícího sloupce je tento přidán.

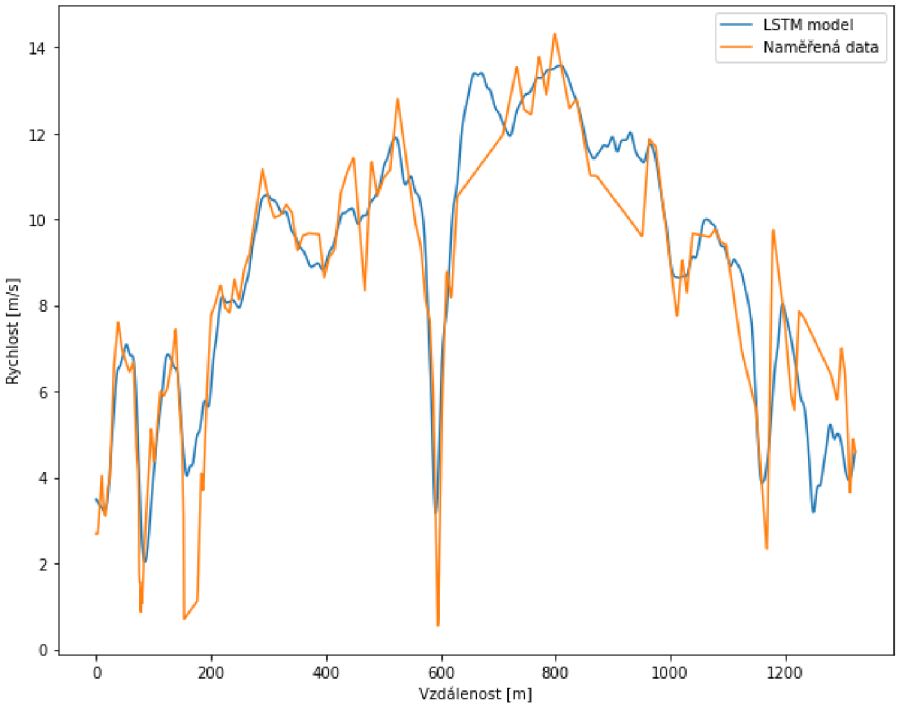
Sloupce jsou poté setříděny podle abecedy s výjimkou sloupce *target_speed*, který je zařazen jako první.

Předzpracování dále umožňuje pomocí přepínače tyto volby: vypuštění sloupců vzniklých zakódováním one-hot představujících chybějící hodnotu, rozmazání bodových příznaků pomocí trojúhelníkového konvolučního filtru a přidání zastavení jako příznaku. Tyto volby jsou blíže popsány v sekci věnované hledání vhodného modelu (viz 7.4.2) a jsou inherentní použitému modelu.

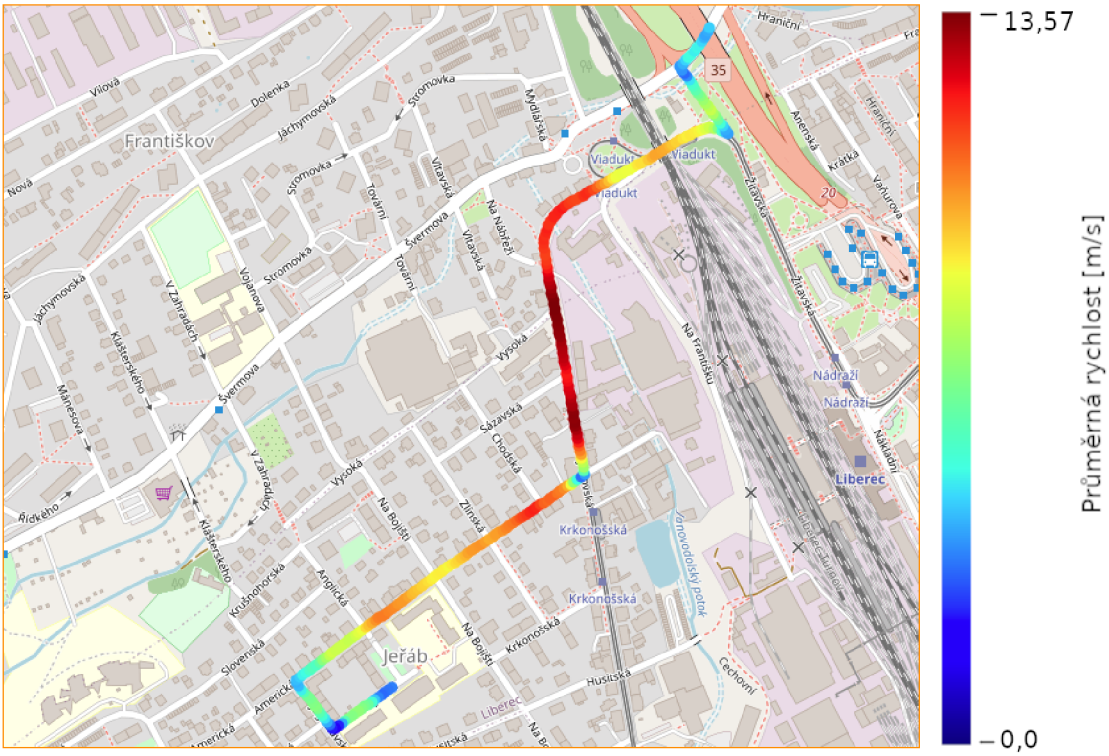
Data jsou přeškálována scalerem naučeným na trénovacích datech při trénování použitého ML modelu. Dojde k transformaci dat z DataFramu na formát očekávaný ML modelem. Výsledkem predikce je rychlostní profil, který je zpětně přeškálován do původního rozsahu.

V posledním kroku je provedeno vyhlazení rychlostního profilu pomocí zvoleného filtru. Prediktoru je ručně přiřazen filtr s ohledem na zašuměnost výsledku ML predikce. Bližší informace o filtraci obsahuje sekce 7.5.

Na grafu 6.6 je vidět porovnání naměřeného a predikovaného rychlostního profilu na doposud neznámých datech z 5.2, která jsem naměřil. Je vidět, že zvláště pro vyšší rychlosti je predikce podobná měřenému rychlostnímu profilu. Větší rozdíly nastávají ve zpomaleních souvisejících s průjezdem křižovatek, což je možné blíže vidět na zobrazení rychlostního profilu v mapě (viz 6.7).



Obrázek 6.6: Porovnání naměřeného rychlostního profilu a rychlostního profilu predikovaného LSTM modelem



Obrázek 6.7: Vizualizace predikovaného rychlostního profilu z 6.6 v mapě

7 Vývoj modelu pro predikci rychlostního profilu

7.1 Trénovací, validační a testovací data

Jako trénovací data jsem použil data jízd šesti vozy společnosti Entry od února do července 2021. Testovací sada obsahuje data z týchž vozů z období od července do září. Tyto sady jsou vzájemně disjunktní. Pokud není uvedeno jinak, jsou použity v průběhu trénování celé. Před použitím byla data zpracována způsobem popsaným v sekci 5.1 a kapitole 6 (segmentace jednotlivých jízd →předzpracování pro map-matching →map-matching →obohacení o příznaky →segmentace trasy na stejně dlouhé úseky).

Po tomto zpracování sestává trénovací a validační sada ze 703 jízd o celkové délce 6120 kilometrů (6120000 vzorků) a testovací sada ze 307 jízd o celkové délce 2574 kilometrů (2574000 vzorků). Mezi trénovací a validační sadu jsou jízdy rozděleny náhodně poměrem 8:2.

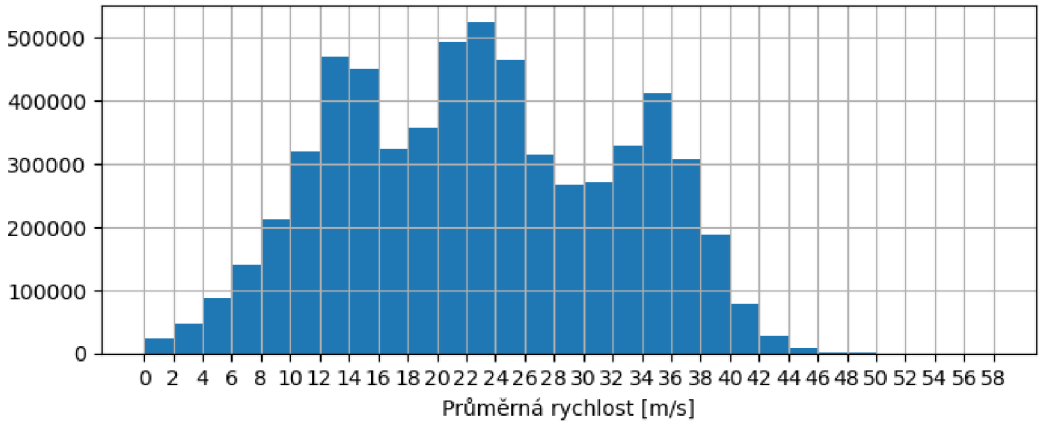
Každá jízda představuje matici tvaru $[\textit{délka jízdy v metrech} \times \textit{počet příznaků}]$ jednotlivé příznaky odpovídají veličinám v tabulce veličin po segmentaci 6.4. Na obrázku 7.3 je vidět vizualizace trénovací a validační sady ve 2D metodou t-SNE. Vstupními daty jsou průměrné příznakové vektory pro každých 50 vzorků z důvodu snížení výpočetní náročnosti. Data nevytváří výrazně oddělené shluky, avšak je zde vidět, že vzhledem k naměřené rychlosti jsou oddělené shluky segmentů s vysokou rychlostí, což zřejmě souvisí s podobností segmentů představujících rychlostní komunikace.

7.2 Příznaky

Předzpracování je popsáno v kapitole 6, výsledná data po zakódování jsou popsána v tabulce 7.1 a představují vstupní příznaky pro predikci rychlosti ML modelem.

Příznaky lze rozdělit na tyto tři skupiny:

- **spojitý** – příznak nabývající spojitých hodnot
- **binární** – příznak úsekový, označuje úseky vozovky s určitou vlastností (povrch, třída)



Obrázek 7.1: Histogram průměrných naměřených rychlostí

- **binární řídký** – příznak bodový, označuje jev, který nastane pouze na jednom segmentu, respektive na několika málo po sobě jdoucích (kruhový objezd, dopravní značení, křižovatka, ...)

7.3 Měření úspěšnosti predikce

Řešená úloha je specifická tím, že nelze určit jednu správnou hodnotu predikce. Objem proměnných, které nejsou a nemohou být modelu předány (nálada řidiče, dopravní situace, . . .) způsobují, že dvě jízdy po stejné trase se mohou výrazně lišit. Hodnocení výsledků modelu proto není vhodné provádět pouze jednou číselnou metrikou. U každého z modelů posuzuji střední absolutní chybu (uváděna v m/s a značena MAE) a mediánovou absolutní chybu (uváděna v m/s a značena MedianAE) pro její vyšší odolnost vůči extrémním hodnotám. Zároveň jsem při testování využíval soubor grafických znázornění, které mi pomohly lépe se zorientovat ve výsledcích a lépe odhalit problémy jako je například přeučení na určité rychlostní hladiny nebo vozy [37].

7.4 Hledání modelu

Při hledání vhodného modelu jsem postupoval iterativně od jednodušších po složitější NN architektury. Průběžně jsem zároveň testoval úpravy návrhů příznaků s potenciálem ke zlepšení úspěšnosti modelu. Hlavní myšlenkou bylo postupné přidávání kontextu okolních segmentů různými způsoby. Ve studii 1 a 2 zakódováním kontextu přímo do vstupních dat, ve studii 3 a 4 předáním okolních segmentů jako vstupu dopřednému modelu a ve studii 5 s využitím RNN. Pro práci s ML modely jsem využil knihovnu PyTorch. Pro optimalizaci hyperparametrů knihovnu Optuna.

Tabulka 7.1: Příznaky po zakódování

Název sloupce	**Typ příznaku**	**Význam**
target_speed	spojitý	naměřená rychlost m/s
azimuth_diff	spojitý	první diference azimutu [°]
elevation	spojitý	nadmořská výška [m]
elevation_diff	spojitý	první diference nadmořské výšky [m]
fwd_azimuth	spojitý	dopředný azimut [°]
node:crossing_traffic_signals	binární řídký	světelně řízený přechod pro chodce
node:crossing_uncontrolled	binární řídký	neřízený přechod pro chodce
node:highway_bus_stop	binární řídký	autobusová zastávka
node:highway_crossing	binární řídký	přechod pro chodce
node:highway_give_way	binární řídký	dopravní značení dej přednost v jízde
node:highway_motorway_junction	binární řídký	sjezd z rychlostní komunikace
node:highway_speed_camera	binární řídký	měření rychlosti
node:highway_stop	binární řídký	dopravní značení STOP
node:highway_traffic_signals	binární řídký	světelné signalizační zařízení
node:intersection_indistinct	binární řídký	průjezd křižovatkou bez rozlišení přednosti
node:intersection_main_to_main	binární řídký	průjezd křižovatkou po vozovce vyšší priority
node:intersection_main_to_side	binární řídký	průjezd křižovatkou na vozovku nižší priority
node:intersection_roundabout	binární řídký	kruhový objezd
node:intersection_side_to_main	binární řídký	průjezd křižovatkou na vozovku vyšší priority
node:intersection_side_to_side	binární řídký	průjezd křižovatkou po vozovce nižší priority
node:railway_level_crossing	binární řídký	úrovňový železniční přejezd
node:stop_stop	binární řídký	uživatelsky zadané dopravní značení STOP
speed_osrm	spojitý	rychlost podle modelu OSRM [m/s]
start_stop_1	binární řídký	označení prvního a posledního segmentu trasy
way_maxspeed	spojitý	maximální povolená rychlost [m/s]
way_surface_asphalt	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_surface_cobblestone	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_surface_concrete	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_surface_sett	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_surface_unpaved	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_living_street	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_motorway	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_motorway_link	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_primary	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_primary_link	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_residential	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_secondary	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_secondary_link	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_service	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_tertiary	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_tertiary_link	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_trunk	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway
way_type_trunk_link	binární	typ vozovky https://wiki.openstreetmap.org/wiki/Key:highway

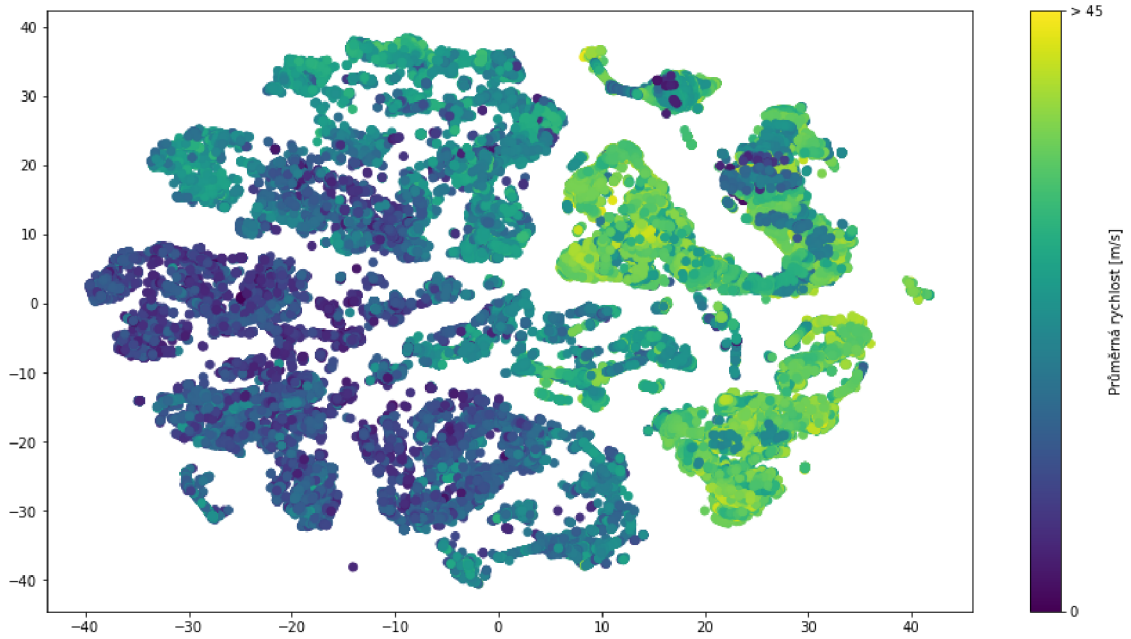
Obrázek 7.2: Hodnoty Pearsonova r pro spojité příznaky

7.4.1 Studie 0 – Regresní MLP s jedním segmentem

Studie 0 využívá pro predikci rychlosti na segmentu n pouze příznaky ze stejného segmentu. Model tedy nemá k dispozici žádná data z předcházejících nebo následujících segmentů. Model je MLP se dvěma skrytými vrstvami (viz kód 7.1). Vzhledem k relativně rychlému cyklu trénování-validace-testování jsem při optimalizaci hyperparametrů do hledání zahrnul více parametrů. Pomocí metody RandomSearch jsem hledal optimální learning rate, optimizer a aktivační funkci modelu. Nejlepších výsledků bylo dosaženo s optimizerem Adam a aktivační funkcí ReLU, což je v souladu s doporučeními k trénování [22, 38]. Model nebyl schopen přeučení na trénovací data při použití jedné dávky o 1024 vzorcích. To lze přisoudit velikosti modelu.

```
Sequential(
  (fc1): Linear(in_features=42, out_features=42, bias=True)
  (relu1): ReLU()
  (fc2): Linear(in_features=42, out_features=21, bias=True)
  (relu2): ReLU()
  (fc3): Linear(in_features=21, out_features=10, bias=True)
  (relu3): ReLU()
  (fc4): Linear(in_features=10, out_features=1, bias=True)
  (sigmoid): Sigmoid()
)
```

Kód 7.1: Model studie 0



Obrázek 7.3: Vizualizace datasetu metodou t-SNE

Po následném vytrénování na všech datech dosáhl model testovacího výsledku MAE 3,8 a MedianAE 2,78.

7.4.2 Studie 1 – Kontext rozmazáním příznaků

Jak již bylo zmíněno, navržený model přijímá pouze příznaky popisující aktuální segment. V důsledku to znamená, že pokud po aktuálním segmentu následuje například segment křižovatky, která pravděpodobně významně ovlivní rychlost, tato informace není modelu dostupná. Při zachování malého modelu jsem se pokusil tuto informaci zpřístupnit modelu následovně: Vstupní data jsou sekvenční, což dosud není využito. Příznaky označené jako binární řídké jsem upravil tak, že namísto binární reprezentace jsem pomocí trojúhelníkového filtru w délky l nahradil hodnotu příznaku x na segmentu n tak, že:

$$x_n = max(w * [x_{n-\lceil l/2 \rceil}; x_{n+\lceil l/2 \rceil}]) \tag{7.1}$$

Namísto binární hodnoty reprezentující přítomnost či nepřítomnost dané vlastnosti je nová hodnota desetinné číslo v intervalu [0;1] reprezentující vzdálenost od vlastnosti. V případě použitého filtru délky 201 (výchozí) má příznak nenulovou hodnotu právě tehdy, když se v okolí 100 segmentů na obě strany daná vlastnost vyskytuje a hodnota příznaku stoupá s blízkostí vlastnosti.

Při zachování všech hyperparametrů ze studie 0 dosáhl model hodnot MAE 3,33 (zlepšení 12,4 %) a MedianAE 2,45 (zlepšení 11,9 %).

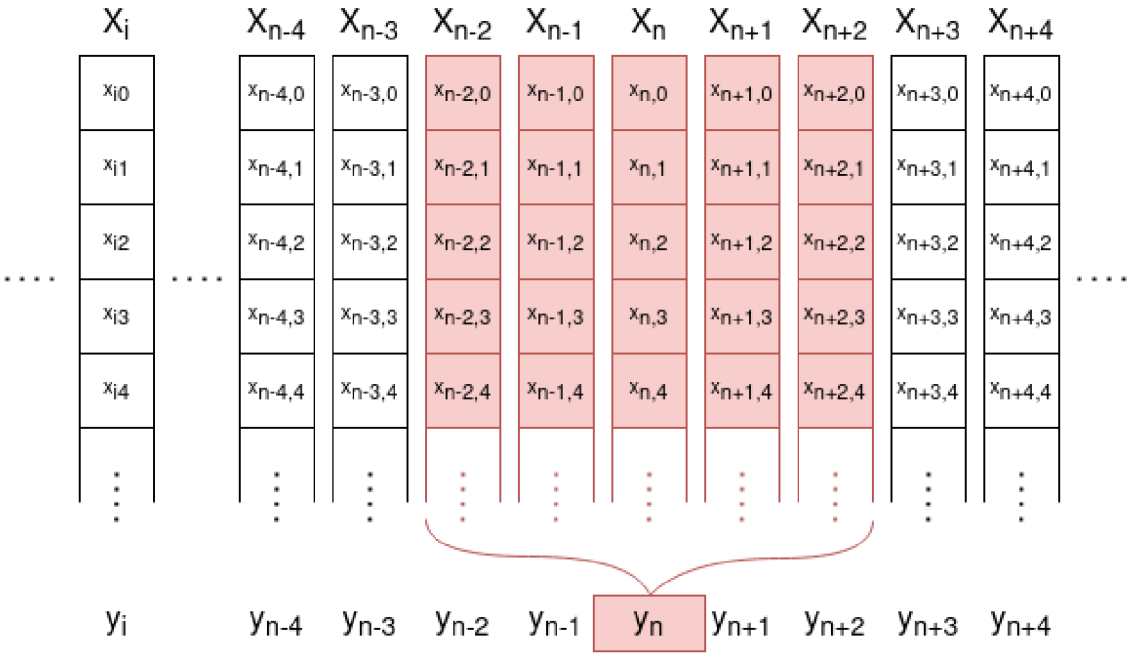
7.4.3 Studie 2 – Zastavení jako příznak

V produkční fázi má uživatel možnost, zadat body na trase, ve kterých vozidlo zastavilo. Zastavení na trase vycházející z dodržování dopravních předpisů nenastávají z důvodu dopravní situace na stejných místech. Příkladem může být značka STOP, která řidiči ukládá zastavit na takovém místě, odkud má náležitý výhled [39]. Což nemusí být místo umístění značky. Podobná situace nastává u světelné křižovatky, kde řidič nezastaví, pokud mu to není signalizováno. Tyto situace nejsou v dostupných datech označeny, ačkoliv by tomu tak při sběru dat mohlo být. Společně s jízdním stylem řidiče jde o důležité faktory odlišující dvě různé jízdy na stejné trase.

Potenciálním řešením je dodatečné označení zastavení na základě naměřené rychlosti vozidla. Vzhledem k možnosti uživatelsky označit zastavení se nejedná o „target leaking“. Pokud je jako práh k označení zastavení vozidla použita hodnota 1 m/s, je označeno v trénovacím/validačním datasetu 0,12 % segmentů.

Takto upravená data spolu s rozmazáním popsaným ve studii 1 jsem použil pro přetrénování modelu. Vytrénovaný model dosáhl horších výsledků oproti modelu ve studii 1: MAE 3,5 (zhoršení o 5,1 %) a MedianAE 2,53 (zhoršení o 3,3 %).

7.4.4 Studie 3 – Sekvence segmentů jako vstup



Obrázek 7.4: Využití okolních segmentů jako vstupních dat modelu

Dosavadní model pracoval pouze s jedním segmentem. Ve studii 3 jsem modelu předal širší kontext tak, že namísto jednoho segmentu je vstupem okno o délce 21 segmentů (viz kód 7.2 a schéma 7.4). Kromě aktuálního i 10 předcházejících a 10 následujících. V první vrstvě modelu dojde k jejich konkatenaci, výsledkem je vektor o délce [*počet příznaků* × *délka okna*]. Ostatní předzpracování vstupních dat

zůstává stejné jako u předchozích studií. Tato strategie naráží na absenci okolních segmentů pro krajní segmenty a pro tuto studii jsem ji vyřešil doplněním segmentů obsahujících nuly na začátek a konec v počtu požadovaných předcházejících respektive následujících vzorků.

Narozdíl od předchozího modelu dochází na 10 dávkách o 1024 vzorcích k přeučení. Při přetrénování na celém datasetu však model dosahuje horších výsledků než model předcházející (MAE 3,59; MedianAE 2,47).

Z průběhu trénovacího a validačního kritéria (lossu) je zřejmé, že zatímco trénovací kritérium (loss) velmi rychle konverguje k nule, validační kritérium (loss) se ustaluje na násobně vyšších hodnotách. To ukazuje na špatnou schopnost generalizace modelu nebo na velkou odlišnost validačních a trénovacích dat. Vzhledem ke způsou sběru dat a jejich dělení do trénovací a validační sady je pravděpodobnější první důvod. Zároveň jsem neměl možnost doplnit nová data. Problém by se dal vyřešit zmenšením modelu nebo pomocí regularizace (Dropout, L1, L2). Problém jsem se pokusil vyřešit pomocí L2 regularizace [21]. Vhodnou hodnotu regularizačního parametru jsem stanovil hledáním pomocí RandomSearch na menší datové sadě z důvodu výpočetní náročnosti. Po přetrénování s L2 regularizací dosáhl model lepšího validačního kritéria (lossu) a překonal v testovacím skóre předchozí model. Dosáhl hodnot MAE 3,18 a MedianAE 2,32.

```
Sequential(
 (flatten): Flatten()
 (fc1): Linear(in_features=882, out_features=880, bias=True)
 (relu1): ReLU()
 (fc2): Linear(in_features=880, out_features=512, bias=True)
 (relu2): ReLU()
 (fc3): Linear(in_features=512, out_features=256, bias=True)
 (relu3): ReLU()
 (fc4): Linear(in_features=256, out_features=128, bias=True)
 (relu4): ReLU()
 (fc5): Linear(in_features=128, out_features=64, bias=True)
 (relu5): ReLU()
 (fc6): Linear(in_features=64, out_features=32, bias=True)
 (relu6): ReLU()
 (fc7): Linear(in_features=32, out_features=1, bias=True)
 (sigmoid): Sigmoid()
)
```

Kód 7.2: Model studie 3

7.4.5 Studie 4 – 1D konvoluce

Lze si představit, že v závislosti na viditelnosti a místní znalosti řidič přizpůsobuje rychlost delšímu úseku, než je následujících 10 metrů. Potenciálním zlepšením by tedy mohlo být rozšíření okna – „zorného pole“ modelu. S ohledem na paměťové nároky plně propojených vrstev jsem se rozhodl rozšířit jej přidáním 1D konvoluční

vrstvy jako první vrstvy modelu (viz kód 7.3). Upravený model nově přijímá 10 předcházejících a 52 následujících segmentů. Při velikosti jádra 3 a kroku 3 zůstaly ostatní vrstvy modelu stejné.

```
Sequential(
 (conv1): Conv1d(
   in_channels=42,
   out_channels=42,
   kernel_size=(3,),
   stride=(3,)
 )
 (ac1): ReLU()
 (flatten): Flatten()
 (fc1): Linear(in_features=882, out_features=880, bias=True)
 (relu1): ReLU()
 (fc2): Linear(in_features=880, out_features=512, bias=True)
 (relu2): ReLU()
 (fc3): Linear(in_features=512, out_features=256, bias=True)
 (relu3): ReLU()
 (fc4): Linear(in_features=256, out_features=128, bias=True)
 (relu4): ReLU()
 (fc5): Linear(in_features=128, out_features=64, bias=True)
 (relu5): ReLU()
 (fc6): Linear(in_features=64, out_features=32, bias=True)
 (relu6): ReLU()
 (fc7): Linear(in_features=32, out_features=1, bias=True)
 (sigmoid): Sigmoid()
)
```

Kód 7.3: Model studie 4

Při zachování hyperparametrů z předchozí studie model po vytrénování dosáhl testovacích výsledků MAE 2,7 (zlepšení 15,1 %) a MedianAE 2,04 (zlepšení 12,2 %).

Na 10 dávkách o 1024 vzorcích jsem následně porovnal různé kombinace počtů předcházejících (11, 23, 41) a následujících (12, 54, 66, 87, 108, 72) segmentů. Počty byly zvoleny tak, aby velikost výsledného okna byla dělitelná třemi s ohledem na velikost konvolučního jádra a kroku). Z průběhu trénovacího a validačního kritéria (lossu) testovaných oken jsem stanovil odhadem optimální velikost okna na 23 předcházejících a 72 následujících segmentů. Takto upravený model dosáhl horšího testovacího skóre.

Výsledné predikované rychlostní profily testovacích jízd výrazně oscilují na začátku a konci jízdy. To by mohlo souviset s doplňováním neúplných vstupních dat na začátku a konci nulami. Na menších datech pouze z jednoho vozu jsem otestoval alternativní doplnění neúplných dat. Testovaná doplnění byla trojí:

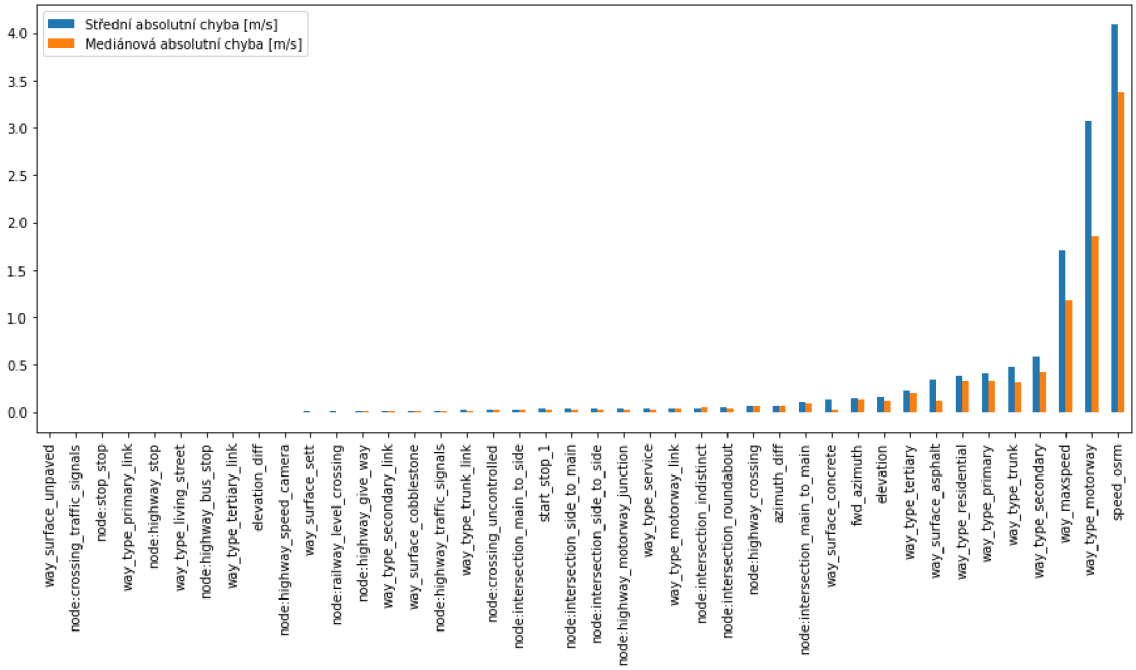
- Doplnění nulami – výchozí
- Doplnění opakováním prvního / posledního segmentu

- Doplnění zrcadlením (např. pro sekvenci začínající [1, 2, 3, 4] a 2 doplněné hodnoty je výsledkem sekvence [2, 1, 1, 2, 3, 4])

Z testovaných možností dosáhlo nejlepšího výsledku zrcadlení.

Stejným způsobem jsem otestoval změnu sigmoidy na výstupu modelu proti alternativám ReLU a úplnému vynechání nelinearity. Nelinearita nepředstavuje nezbytnou součást modelu vzhledem k následným úpravám v postprocesu. Z testovaných možností dosáhlo nejlepšího výsledku vypuštění nelinearity.

Na základě testování jsem model přetrénoval na všech datech se zrcadlením a bez nelinearity. Přetrénovaný model dosáhl mírně horšího skóre MAE 2,82 a MedianAE 2,08.



Obrázek 7.5: Zhoršení výsledku predikce při permutaci příznaků

Pro lepší pochopení důležitosti příznaků jsem otestoval změnu výsledku predikce na testovacích jízdách, pokud dojde k permutaci všech hodnot jednoho příznaku. Výsledky testu jsou zobrazeny v grafu 7.5. Lze si všimnout, že příznaky s nejmenším vlivem na výsledek jsou binární příznaky, které nastávají velmi zřídka. Zároveň je vidět, že permutace příznaků *way_maxspeed* a *speed_osrm*, které dosahují ze spojitých příznaků nejvyšší míry korelace s *target_speed* (viz 7.2), způsobuje výrazné zhoršení výsledku predikce.

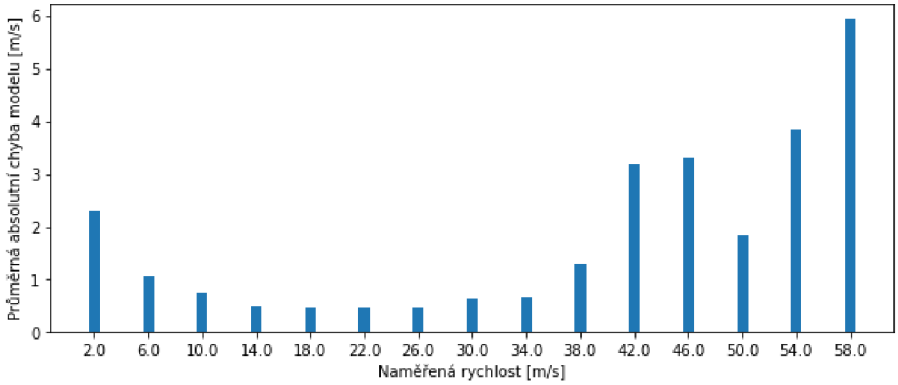
7.4.6 Studie 5 – Obousměrné LSTM

Doposud provedené studie pracovaly s dopřednými modely. Pro sekvenční data je potenciálně vhodnějším řešením využití zpětnovazební neuronové sítě (RNN). Vzhledem k offline povaze problému a tedy předchozí znalosti příznaků celé trasy se nabízí

využití obousměrné implementace LSTM, která prochází vstupní sekvenci dvěma modely dopředně a zpětně a výstup závisí na stavech obou modelů zároveň (viz kód 7.4).

Na menším počtu epoch (100) jsem pomocí RandomSearch hledal vhodnou hodnotu pro parametry *hidden_dim*, *n_layers* a *seq_len*. Nejlepšího výsledku dosáhl model s parametry *hidden_dim* a *n_layers* na horní hranici prohledávaného rozsahu. Pro delší trénink jsem zvolil hodnoty *hidden_dim* 512, *n_layers* 5 a *seq_len* 128. Po 500 epochách dosáhl vytrénovaný model na testovacích datech zatím nejlepšího výsledku 0,8 MAE (zlepšení 70,6 %) a 0,38 (zlepšení 81,4 %) MedianAE.

Obrázek 7.6: MAE v závislosti na rychlosti




```
Sequential(
  (lstm): LSTM(
    input_size=42,
    hidden_dim=512,
    num_layers=5,
    batch_first=True,
    bidirectional=True
  )
  (fc): Linear(in_features=1024, out_features=1, bias=True)
)
```

Kód 7.4: Model studie 5

7.4.7 Vyhodnocení výsledků

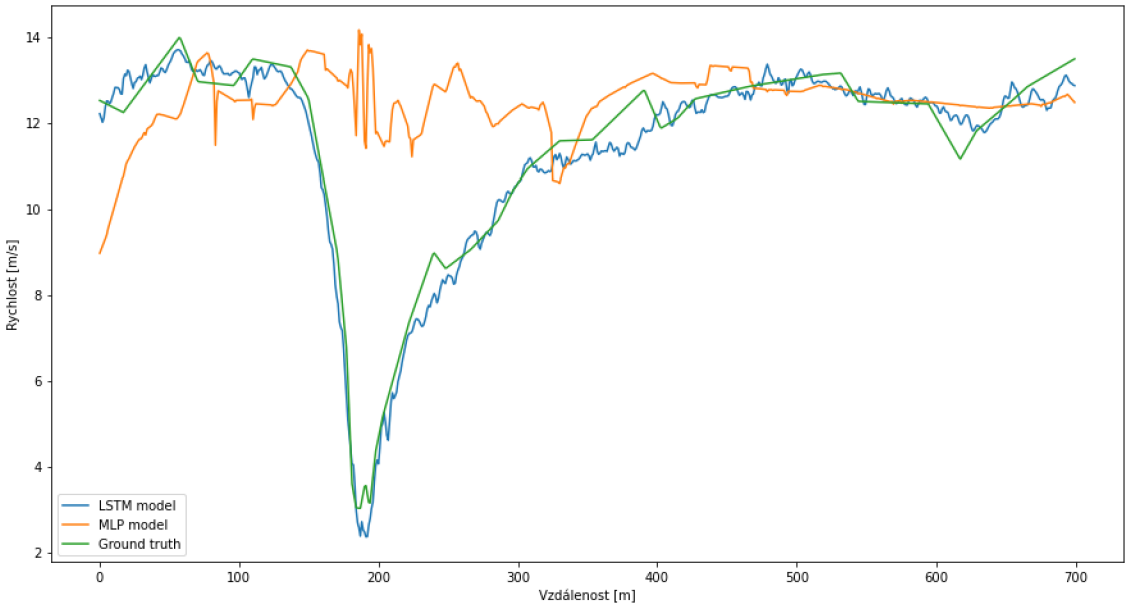
V tabulce 7.2 je shrnutí výsledků jednotlivých studií. Sloupec *doba trvání testu* představuje dobu běhu textu na všech testovacích jízdách (viz 7.1). Časy byly naměřeny při testování na grafické kartě GeForce RTX 3060.

Tabulka 7.2: Testovací výsledky modelů

Studie	MAE [m/s]	MedianAE [m/s]	Doba trvání testu [s]
0 – Regresní MLP s jedním segmentem	3,804	2,783	**52**
1 – Kontext rozmazáním příznaků	3,332	2,451	58
2 – Zastavení jako příznak	3,501	2,533	71
3 – Sekvence segmentů jako vstup	3,184	2,319	299
4 – 1D konvoluce	2,703	2,037	719
5 – Obousměrné LSTM	**0,796**	**0,379**	629

7.5 Vyhlazení rychlostního profilu

Jak je vidět na obrázku 7.7, výsledky predikce rychlostního profilu z různých modelů vykazují různou míru kmitání, které se v naměřených datech nenachází. Rozdíly mezi sousedními vzorky pak místy představují zrychlení nebo zpomalení, jakého by vozidlo nebylo reálně schopno. Aby byla maximalizována podobnost s naměřeným rychlostním profilem, je vhodné výsledky vyhladit.



Obrázek 7.7: Rychlostní profil před vyhlazením

Pro vyhlazení jsem testoval různé metody (Gaussovské rozmazání, klouzavý průměr, polynomická regrese, filtr Savitzky-Golay a klouzavý vážený průměr s trojuhelníkovým filtrem), pro jednotlivé metody vyhlazení jsem testoval různé délky oken a ostatních příslušných parametrů. Zhodnocení výsledku vyhlazení jsem provedl jednak subjektivně a zároveň na základě změny MAE a MedianAE. Dále jsem hodnotil dobu trvání vyhlazení.

Příklad objektivních výsledků testování je vidět v tabulce 7.3. Při vhodně zvolených parametrech průměrovací funkce nedochází ke zhoršení ale naopak mírnému zlepšení výsledků predikce. S výjimkou polynomické regrese je doba trvání vyhlazení velmi malým zlomkem celkové doby predikce a nepředstavuje důležitý faktor při

volbě metody (u váženého klouzavého průměru souvisí delší čas běhu s implementací pomocí Python for cyklu).

Tabulka 7.3: Výsledky vyhlazení

Metoda	**Změna MAE [m/s]**	**Změna MedianAE [m/s]**	**Doba trvání [s]**
Gaussovské rozmazání	0,0081	**0,0047**	0,0799
klouzavý průměr	**0,0089**	0,006	**0,064**
polynomická regrese	0,002	0,0009	224,5711
Savitzky-Golay	0,0023	0,0013	0,153
vážený klouzavý průměr	0,0064	0,0036	42,8594

8 Závěr

V práci jsem se věnoval tvorbě ML modelu pro predikci rychlostního profilu na libovolné trase. Prozkoumal jsem zdroje geografických dat a porovnal je z pohledu vhodnosti pro tvorbu příznaků pro predikci rychlostního profilu. Rozebral jsem faktory ovlivňující rychlost jízdy z hlediska dostupnosti.

Navrhl a implementoval jsem datovou rouru pro zpracování naměřených jízdních logů společnosti Entry a zároveň jsem pomocí mobilní aplikace GPS Logger rozšířil dostupný objem naměřených jízd dostupných pro další zkoumání. Dále jsem implementoval skript, který umožňuje dávkové získávání jízd z API knihy jízd TUL. Zdroje dat jsem popsal s ohledem na vhodnost pro další využití k účelu predikce rychlostního profilu.

Hlavním výsledkem práce je datová roura pro predikci rychlostního profilu na libovolné trase. S využitím map-matchingového algoritmu jsem implementoval algoritmus, který vstupní trasu (naměřené nebo zadané) přimkne spolu s dostupnými daty k odpovídající trase v mapových datech OSM. Za využití dalších API třetích stran obohatí trasu o metadata OSM a data o nadmořské výšce. Zároveň jsem implementoval doplnění o příznaky, které lze dopočítat z dat OSM a jsou výhodné pro predikci rychlostního profilu (označení křižovatek, diference). Navrhl jsem způsob segmentace dat o trase takovým způsobem, aby každý vzorek představoval stejně dlouhý segment vozovky. Takto zpracovaná data slouží jako vstup ML modelu pro predikci rychlostního profilu, který jsem vyvinul v další části.

Provedl a vyhodnotil jsem několik experimentů, které zahrnovaly návrh ML modelu, jeho vytrénování, validaci a testování. Pokusil jsem se dosáhnout co nejlepšího výsledku optimalizací hyperparametrů metodou RandomSearch. Prováděl jsem dílčí změny v architektuře ML modelu i zpracování příznaků a testoval jejich vliv na výsledek. Naivní dopředný model pracující pouze s jedním, aktuálním segmentem trasy dosáhl po popsané optimalizaci skóre střední absolutní chyby 3,33 m/s. Nejlepšího výsledku dosáhl model využívající obousměrný LSTM. Po optimalizaci hyperparametrů dosáhl střední absolutní chyby 0,8 m/s.

S rozvojem práce je zřejmé, že pro další rozšiřování by byla vhodná reimplementace map-matchingového nástroje, který je v tuto chvíli do jisté míry černou skříňkou. Výsledkem je ztráta části naměřených dat, které by se takto dalo zabránit.

Analýzou naměřených dat jsem zjistil, že z pohledu faktorů ovlivňujících rychlost by bylo vhodné v budoucnu zahrnout důkladné zaznamenávání těch náhodných faktorů, které ve fázi trénování nelze jinak získat (zastavení na přechodu pro chodce atp.), ale zároveň je možné je v produkční fázi předat modelu jako uživatelský parametr.

S ohledem na časovou náročnost jednoho ML cyklu trénování–testování jsem zdaleka nevyčerpal možnosti co do úpravy a tvorby příznaků, jakož i do návrhu modelů. Zajímavým směrem by mohla být reformulace ML úlohy jako zpětnovazebního učení.

Bibliografie

1. JURKINA, Maria Ivanovna; PICK, Miloš. *Numerické výpočty ve světovém geodetickém referenčním systému 1984: Vojenský geografický obzor: sborník Geografické služby AČR*. 1. vyd. Praha: Ministerstvo obrany ČR, Hlavní úřad vojenské geografie, 2006. ISSN 1214-3707.
2. GRIFFITHS, Jamie. *Google Maps API pricing changes: what do they mean?* [Online]. 2018 [cit. 2021-12-20]. Dostupné z: https://manifesto.co.uk/google-maps-api-pricing-changes/.
3. ROSENFELD, Keren. *Mapping Technologies Market Share and Web Usage Statistics* [online] [cit. 2021-12-20]. Dostupné z: https://www.similartech.com/categories/mapping.
4. PŘISPĚVATELÉ OPENSTREETMAP. *Who uses OpenStreetMap? — OpenStreetMap* [online] [cit. 2020-06-21]. Dostupné z: https://welcome.openstreetmap.org/about-osm-community/consumers/.
5. PŘISPĚVATELÉ OPENSTREETMAP. *OpenStreetMap Wiki* [online] [cit. 2020-05-17]. Dostupné z: https://wiki.openstreetmap.org/wiki/Main_Page.
6. *Cs:Prvky – OpenStreetMap Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Cs:Prvky.
7. *Cs:Značka – OpenStreetMap Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Cs:Zna%C4%8Dka.
8. *Cs:Map Features – OpenStreetMap Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Cs:Map_Features.
9. *Cs:Česko/Editační standardy a konvence – OpenStreetMap Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Cs:%C4%8Cesko/Edita%C4%8Dn%C3%AD_standardy_a_konvence.
10. PŘISPĚVATELÉ OPENSTREETMAP. *List of OSM-based services* [https://wiki.openstreetmap.org/wiki/List_of_OSM-based_services#Routing]. 2017.

11. *GIScience/openrouteservice-docs* [online]. GIScience Research Group, 2020 [cit. 2020-06-21]. Dostupné z: https://github.com/GIScience/openrouteservice-docs. original-date: 2017-03-28T13:58:22Z.

12. *graphhopper/graphhopper* [GitHub] [online] [cit. 2020-06-21]. Dostupné z: https://github.com/graphhopper/graphhopper.

13. *OSRM API Documentation* [online] [cit. 2020-06-21]. Dostupné z: http://project-osrm.org/docs/v5.22.0/api/#tile-service.

14. *How does OSRM estimate "duration" when in route/match API? · Issue #4903 · Project-OSRM/osrm-backend* [GitHub] [online] [cit. 2020-06-21]. Dostupné z: https://github.com/Project-OSRM/osrm-backend/issues/4903.

15. RAIFER, Martin. *Overpass API User's Manual* [online] [cit. 2020-06-21]. Dostupné z: https://dev.overpass-api.de/overpass-doc/en/.

16. *Overpass API*. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP. Dostupné také z: https://wiki.openstreetmap.org/wiki/Overpass_API.

17. OLBRICHT, Roland. *drolbr/Overpass-API* [online]. 2020 [cit. 2020-06-21]. Dostupné z: https://github.com/drolbr/Overpass-API.

18. NIESIOBEDZKI, Wiktor. *Overpass-API* [online]. GitHub, 2021 [cit. 2020-06-21]. Dostupné z: https://github.com/wiktorn/Overpass-API.

19. EARTH RESOURCES OBSERVATION AND SCIENCE (EROS) CENTER. *Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global*. U.S. Geological Survey, 2017. Dostupné také z: https://doi.org/10.5066/F7PR7TFT.

20. FERNÁNDEZ-DELGADO, M.; SIRSAT, M.S.; CERNADAS, E.; ALAWADI, S.; BARRO, S.; FEBRERO-BANDE, M. An extensive experimental survey of regression methods. *Neural Networks*. 2019, roč. 111, s. 11–34. ISSN 0893-6080. Dostupné z DOI: https://doi.org/10.1016/j.neunet.2018.12.010.

21. GOODFELLOW, Ian J.; BENGIO, Yoshua; COURVILLE, Aaron. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. http://www.deeplearningbook.org.

22. JOHNSON, Justin. *Lecture 5: Neural networks - Deep Learning for Computer Vision* [online]. 2019 [cit. 2020-09-21]. Dostupné z: https://youtu.be/g6InpdhUblE.

23. PASZKE, Adam; GROSS, Sam; MASSA, Francisco; LERER, Adam; BRADBURY, James; CHANAN, Gregory; KILLEEN, Trevor; LIN, Zeming; GIMELSHEIN, Natalia; ANTIGA, Luca; DESMAISON, Alban; KOPF, Andreas; YANG, Edward; DEVITO, Zachary; RAISON, Martin; TEJANI, Alykhan; CHILAMKURTHY, Sasank; STEINER, Benoit; FANG, Lu; BAI, Junjie; CHINTALA, Soumith. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In: WALLACH, H.; LAROCHELLE, H.; BEYGELZIMER, A.; D'ALCHÉ-BUC, F.; FOX, E.; GARNETT, R. (ed.). *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, s. 8024–8035. Dostupné

také z: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

24. JOHNSON, Justin. *Lecture 7: Convolutional networks - Deep Learning for Computer Vision* [online]. 2019 [cit. 2020-09-21]. Dostupné z: https://youtu.be/ANyxBVxmdZ0.

25. *PyTorch Model Zoo* [online]. Ve spol. s PYTORCH SERVE CONTRIBUTORS. 2020 [cit. 2021-12-21]. Dostupné z: https://pytorch.org/serve/model_zoo.html.

26. OORD, Aaron van den; DIELEMAN, Sander; ZEN, Heiga; SIMONYAN, Karen; VINYALS, Oriol; GRAVES, Alex; KALCHBRENNER, Nal; SENIOR, Andrew; KAVUKCUOGLU, Koray. *WaveNet: A Generative Model for Raw Audio*. 2016. Dostupné také z: http://arxiv.org/abs/1609.03499. arxiv:1609.03499.

27. JOHNSON, Justin. *Lecture 12: Recurrent networks - Deep Learning for Computer Vision* [online]. 2019 [cit. 2020-09-21]. Dostupné z: https://www.youtube.com/watch?v=dUzLD91Sj-o.

28. SCHUSTER, Mike; PALIWAL, Kuldip. Bidirectional recurrent neural networks. *Signal Processing, IEEE Transactions on*. 1997, roč. 45, s. 2673–2681. Dostupné z DOI: 10.1109/78.650093.

29. C, Palmer; A, Quimby; G, Maycock; C, Palmer. The factors that influence a driver's choice of speed — a questionnaire study. *TRANSPORT RESEARCH LABORATORY*. 1999, č. 325. ISSN 0968-4107.

30. FALTUS, Vladimír. *Statistické charakteristiky v dopravě*. Praha: Ústav dopravní telematiky ČVUT v Praze Fakulta dopravní, 2018.

31. DERROW-PINION, Austin; SHE, Jennifer; WONG, David; LANGE, Oliver; HESTER, Todd; PEREZ, Luis; NUNKESSER, Marc; LEE, Seongjae; GUO, Xueying; WILTSHIRE, Brett; BATTAGLIA, Peter W.; GUPTA, Vishal; LI, Ang; XU, Zhongwen; SANCHEZ-GONZALEZ, Alvaro; LI, Yujia; VELICKOVIC, Petar. ETA Prediction with Graph Neural Networks in Google Maps. In: New York, NY, USA: Association for Computing Machinery, 2021, s. 3767–3776. ISBN 9781450384469. Dostupné také z: https://doi.org/10.1145/3459637.3481916.

32. *Accuracy Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Accuracy.

33. PŘISPĚVATELÉ OSRM. *OSRM API Documentation* [online] [cit. 2021-05-17]. Dostupné z: http://project-osrm.org/docs/v5.5.1/api/.

34. *Proposed features Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Proposed_features/highway%3Djunction.

35. *Tag:crossing=unmarked Wiki* [online]. Ve spol. s PŘISPĚVATELÉ OPENSTREETMAP [cit. 2020-06-21]. Dostupné z: https://wiki.openstreetmap.org/wiki/Tag:crossing%3Dunmarked.

36. DUBOUE, Pablo. *The Art of Feature Engineering: Essentials for Machine Learning*. Cambridge University Press, 2020. Dostupné z DOI: hb3f.

37. DUBOUE, Pablo. *The Art of Feature Engineering: Essentials for Machine Learning*. Cambridge University Press, 2020. Dostupné z DOI: hb3f.

38. KARPATHY, Andrej. *A Recipe for Training Neural Networks* [online]. 2019 [cit. 2021-12-21]. Dostupné z: http://karpathy.github.io/2019/04/25/recipe/.

39. [N.d.]. ČESKO. § 22 odst. 1 zákona č. 361/2000 Sb., o provozu na pozemních komunikacích a o změnách některých zákonů. In: Zákony pro lidi.cz [online]. © AION CS 2010-2021 [cit. 20. 12. 2021]. Dostupné z: https://www.zakonyprolidi.cz/cs/2000-361p22-1.