Jihočeská univerzita v Českých Budějovicích
Přírodovědecká fakulta

Identifikace železničních vozů na základě obrazové informace

Diplomová práce

Bc. Pavel Šeps

Školitel: Ing. Skrbek Miroslav, Ph.D.
Konzultant: Ing. Miloslav Černý (Severočeské doly a.s.)

České Budějovice 2020

Bibliografické údaje

Bc. Šeps, Pavel, 2020: Identifikace železničních vozů na základě obrazové informace. [Identification of railcars based on image information. Mgr. Thesis, In Czech.] – 69p., Faculty of Science, University of South Bohemia, České Budějovice, Czech Republic.

Anotace

Tato magisterská práce se zaměřuje na návrh a vytvoření systému pro identifikaci železničních vozů na základě identifikační informace, která se nachází na boku každého vagónu. Popisuje nalezení optimálního předzpracování nalezeného snímku, porovnání OCR služeb, nalezení vhodných technologií a vyhodnocení kvality vytvořené aplikace. Součástí práce je také popis stávajících řešení, popis vývoje a funkčnosti aplikace.

Pro získání snímků vozů je navržena a sestavena fotopast, která je umístěna v blízkosti vlakového kolejiště. K nalezení identifikačních informací byla naučena vlastní neuronová síť. Pro vytvoření testovacích a trénovacích množin určených k učení neuronové sítě, byla vytvořena vlastní anotační aplikace. Tuto neuronovou síť implementuje aplikace pro identifikaci železničních vozů. Aplikace také využívá služby Azure pro OCR. Identifikační aplikace je určena k automatizaci manuálního procesu. Systém je připravený pro použití v reálném provozu.

Annotation

This master's thesis focuses on the design and implementation of railway wagon indetification system based on the identifying information on the side of each wagon. It describes the search for the optimal image preprocessing, comparison of OCR services, search for the suitable technologies and evaluation of the quality of the finished application. Part of this thesis also describes existing solutions and the application development and functionality.

To obtain the images of the wagons, a photo trap is designed and assembled, which is then located near the train track. A custom neural network was built to find the identifying information and a custom annotation application was developed to create the test and training sets. The final identification application then implements this neural network. It also uses Azure OCR services. It is designed to automate the manual process and is ready for use in real operation.

Prohlášení

Prohlašuji, že svoji diplomovou práci jsem vypracoval samostatně pouze s použitím pramenů a literatury uvedených v seznamu citované literatury.

Prohlašuji, že v souladu s § 47b zákona č. 111/1998 Sb. v platném znění souhlasím se zveřejněním své diplomové práce, a to v nezkrácené podobě elektronickou cestou ve veřejně přístupné části databáze STAG provozované Jihočeskou univerzitou v Českých Budějovicích na jejích internetových stránkách, a to se zachováním mého autorského práva k odevzdanému textu této kvalifikační práce. Souhlasím dále s tím, aby toutéž elektronickou cestou byly v souladu s uvedeným ustanovením zákona č. 111/1998 Sb. zveřejněny posudky školitele a oponentů práce i záznam o průběhu a výsledku obhajoby kvalifikační práce. Rovněž souhlasím s porovnáním textu mé kvalifikační práce s databází kvalifikačních prací Theses.cz provozovanou Národním registrem vysokoškolských kvalifikačních prací a systémem na odhalování plagiátů.

V Českých Budějovicích dne …….……. ……...

Pavel Šeps

Poděkování

Rád bych poděkoval panu Ing. Miroslavovi Skrbkovi Ph.D. za vedení mé diplomové práce, cenné rady a odborný dohled. Také bych rád poděkoval svým blízkým za trpělivost a podporu při mých studiích. Děkuji všem z oddělení aplikační podpory Severočeských dolů za možnost spolupráce a poskytnutí cenných rad.

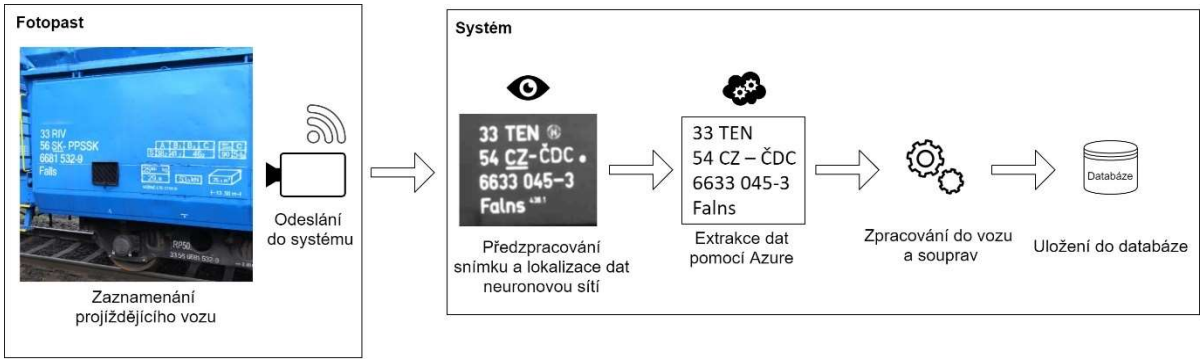
Obsah

1 Úvod 1
2 Cíle práce 2
3 Popis problematiky a stávající řešení 3
3.1 Značení vozů 3
3.2 Stávající řešení 5
3.3 Existující řešení 5
3.4 Logický rámec 7
4 Teoretická část 9
4.1 Předzpracování obrazu 9
4.2 Neuronové sítě 13
4.3 Tensorflow 14
4.4 SSD Inception v2 model 15
4.5 Azure 16
4.6 OCR 16
5 Návrh řešení 18
5.1 Příprava trénovacích množin pro neuronové sítě 19
5.1.1 Stávající řešení 19
5.1.2 Návrh aplikace pro anotaci dat 19
5.1.3 Scénáře použití 22
5.2 Návrh aplikace pro identifikaci vlakových souprav 23
5.2.1 Scénáře použití 25
5.2.2 Volba OCR technologie 28
5.2.3 Přehled vybraných technologií 29
5.2.4 Architektura systému 31
6 Implementace 32
6.1 Fotopast pro sběr experimentálních dat 32

6.1.1 Hardwarová část 33
6.1.2 Softwarová část 35
6.2 Aplikace pro anotaci dat 38
6.3 Naučení neuronové sítě pro detekci kódů 41
6.4 Aplikace pro identifikaci vlakových souprav 41
6.4.1 Serverová část 42
6.4.2 Klientská část 45
7 Experimenty 49
7.1 Učení sítě 49
7.2 Model sítě 50
7.3 Naučené sítě 52
7.4 Výsledky sítí 53
8 Testování 57
8.1 Skládání snímku 57
8.2 Testování fotopasti 58
8.3 Testování neuronové sítě 58
8.4 Testování aplikace pro identifikaci 58
8.5 Testovací provoz identifikační aplikace 58
9 Závěr 60
10 Literatura 62
Příloha A Hodnocení Severočeskými doly 64
Příloha B Vizuální část anotační aplikace 65
Příloha C Vizuální část identifikační aplikace 66

1 Úvod

Cílem diplomové práce je vytvořit systém pro identifikaci železničních vozů na základě snímků pořízených v železničním kolejišti. Práce vzniká ve spolupráci se Severočeskými doly. Motivací pro zadavatele je automatizace aktuálního manuálního procesu evidence vozů, který zajišťuje obsluha tratě. Systém je cílený na nakládku materiálu v úpravně uhlí Ledvice. Systém je navržen tak, aby bylo možné použití i na jiných lokacích. Identifikace vozů systémem je plně automatická, ale umožňuje také jejich manuální vkládání.



Obrázek 1 Schéma kompletního systému zpracování

Práce zpracovává popis stávajících řešení a aktuální manuální proces identifikace vozů. Popisuje postup získání snímku projíždějícího vozu a též k tomu potřebné navržení a vytvoření snímacího zařízení (fotopasti), které je určené k umístění do blízkosti kolejiště. Je popsán proces testování fotopasti před uvedením do provozu i po jejím umístění na určené místo. Práce se dále zabývá rozpoznáváním identifikačního kódu na boku každého vozu a popisem procesu učení vlastní neuronové sítě k tomu určené. Popisuje metody, testované pro nalezení optimálního předzpracování snímku vozu určeného pro neuronovou síť. Je popsáno, jakým způsobem cílový systém využívá vytvořenou fotopast a následně implementuje naučenou neuronovou síť k nalezení informace. Také je popsán výběr OCR technologie. Pro získání identifikačního kódu ze snímku je využito počítačové vidění externí služby Azure. Identifikační aplikace je vytvořena jako webová aplikace, která zpracovává snímky vozů a nalezené vozy zapisuje do databáze. Webové rozhraní slouží obsluze k administraci a případné manuální úpravě načtených vozů.

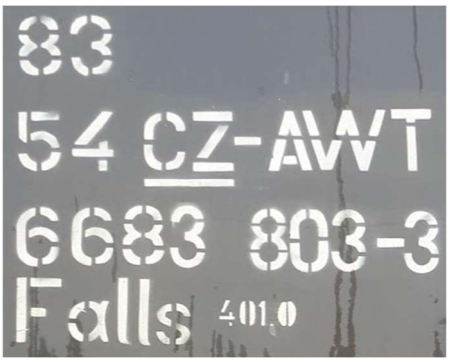
2 Cíle práce

- Vytvořit systém pro identifikaci vozu, který bude získávat a zpracovávat informace rychle a efektivně a nahradí tak současný zdlouhavý manuální proces.
- Navrhnout a sestavit fotopast pro sběr projíždějících vozů.
- Vyhledávat identifikační kód vlastní naučenou neuronovou sítí.
- Navrhnout a vytvořit aplikaci, pro správu identifikovaných vozu.
- Nasazení aplikace do reálného provozu a zjednodušení stávajícího procesu.

3 Popis problematiky a stávající řešení

3.1 Značení vozů

Každý vůz, který je provozovaný na dráze, musí být označený specifickým kódem. Kód je sjednocený ve všech členských státech evropské unie, musí být umístěný z obou stran vozu na viditelném místě a musí být čitelný. Na nákladních vozech se tento kód skládá ze čtyř řad čísel a písmen. První řada obsahuje dvě číslice nebo dvě číslice s písmeny – tato řada určuje interoperabilitu vozu. Druhá řada se skládá ze dvou číslic, mezinárodní značky, pomlčky a textové zkratky. Číslice a mezinárodní značka označuje zemi, ve které je vůz registrován, textová zkratka pak určuje provozovatele vozu. Třetí řada obsahuje čtyři číslice, označující technické vlastnosti vozidla, trojčíslí jako evidenční číslo vozu a poslední číslice za pomlčkou slouží jen jako kontrolní. Poslední čtvrtá řada obsahuje textovou zkratku technických vlastností vozů.



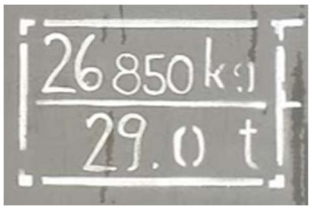
Obrázek 2 Identifikační kód vozu

Označení obsahuje celkem jedenáct číslic plus jednu ověřující číslici. Správnost kódu se provede vynásobením každé liché číslice dvěma a sudé číslice se ponechají. Následně se všechny číslice sečtou, pokud vyjde dvouciferné číslo, převede se na dvě jednociferné (např.: 16 na 1 a 6). Na výsledek součtu se provede modulo 10. Z 10 se odečte výsledek modulu a finálním výsledkem je kontrolní číslice [1].

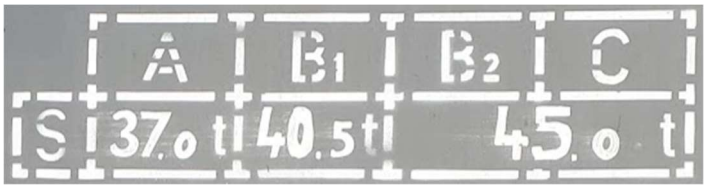
Tabulka 1 Ukázka výpočtu kontrolní číslice kódu vozu

Kód	8	3	5	4	6	6	8	3	8	0	3
Vynásobení každé liché	16	3	10	4	12	6	16	3	16	0	6
Součet	1 + 6 + 3 + 1 + 0 + 4 + 1 + 2 + 6 + 1 + 6 + 3 + 1 +6 + 0 + 6										
Modulo	10 – (47 mod 10)										
Výsledek kontrolní číslice	3										

Nákladní vozy na sobě mají natištěné další informace o vozu. Jednou z potřebných informací je váha prázdného vozu v kilogramech v tabulce s brzdící váhou vozu. Váha vozu se může lišit pro stejný typ – například z důvodu opravy. Další důležitou informací je tabulka ložných hmotností, která určuje maximální rychlost a váhu pro danou kategorii trati [2].



Obrázek 3 Tabulka váhy prázdného vozu



Obrázek 4 Tabulka kategorie tratí

Zpracováním identifikačního kódu na Obrázek 2 Identifikační kód vozu zjistíme z čísla 83, že vůz není v souladu s interoperabilitou. Z číslice 54 a kódu CZ, že se jedná o vůz registrovaný v České republice a zkratka za pomlčkou označuje provozovatele AWT. Číslice 6683 určuje maximální rychlost vozu do 160 km/h s hodnotou topné energie „3000 V~ 3000 V=“. Pomocí písmene F, vyskytujícím se v názvu Falls, zjistíme, že se jedná o otevřený vůz s vysokými postranicemi, s bočním vyklápěním a s plochou podlahou. Hodnoty z kódu jsou porovnávány s tabulkami pro překlad kódů, aby byl zjištěn jejich význam. Z Obrázek 3 Tabulka váhy prázdného vozu zjistíme, že váha prázdného vozu je 26 850 kg a brzdící váha

ruční brzdy je 29 t. Na Obrázek 4 Tabulka kategorie tratí písmeno S určuje maximální rychlost 100 km/h. Pro trať kategorie A je možnost váhy 37 t pro kategorii B1 40,5 t, pro kategorie B2 a C je maximální váha 45 t.

3.2 Stávající řešení

Uhlí z dolů se vyváží za pomoci vlakových souprav. Vozy vyčkávají seřazeny na osmnácti kolejích, dokud nebudou připraveny na naplnění, posléze naplněny a napojeny do správné soupravy a vypraveny. Příjezdy souprav jsou předem naplánovány, ale neví se přesný pohyb a čas příjezdu. Denně jsou nasmlouvány příjezdy a odjezdy více než třiceti souprav.

Aktuálně dispečeři hlásí obsluze příjezd nové vlakové soupravy a kolej, kde se nachází. Obsluha tratě musí následně fyzicky dojít k soupravě a všechna identifikační čísla z výše zmíněného kódu vozů ručně zapsat na papír. Poté je dispečeři na dispečinku opět ručně zapisují do systému, kde tímto úkonem teprve dojde k identifikaci soupravy. Tento postup se v současné době provádí 24 hodin denně, 7 dní v týdnu a za každého počasí.

Jedná se dlouhodobě o velmi pomalý, výrazně neefektivní a někdy též nebezpečný způsob identifikace. Severočeské doly se proto již v minulosti pokoušely o jeden způsob usnadnění identifikace vozů.

Metoda tohoto pokusu spočívala v použití QR kódů. Obsluha, která jindy provádí pouze zápis, během běžných úkonů lepila na každý daný vůz velkou samolepku s čitelným QR kódem. Výsledek tohoto postupu se bohužel ukázal jako neúspěšný. Návratnost stejných vozů s nalepeným kódem byla velmi nízká. Ukázalo se, že samolepky s kódem nemají dlouhou životnost a pokud už se nějaký takový vůz vrátil, často byly samolepky poškozené a nečitelné, tudíž nepoužitelné.

3.3 Existující řešení

Na celosvětovém trhu již v současné době existuje jedno řešení. Jeho název je Unirail a vyl vyvinut Českou společností Camea s.r.o [3]. V tomto řešení používají na usnadnění kontroly vozů multifunkční kontrolní brány, které nabízí mnoho technologických funkcí. Systém této brány je schopen detekovat identifikační čísla vozů, detekce plochých míst na kolech, vážení každého z vozů, zjištění přehřívání obručí, a navíc též obsahuje detektor horkoběžnosti ložisek a laserový 3D sken.

Kombinace všech těchto systémů je využita pro co nejpřesnější detekci vozu, zjištění rychlosti soupravy, směru pohybu a neméně důležité nalezení případné technické poruchy na některém z vozů. Výrobce udává 95% úspěšnost do rychlosti 160 km/h, u nákladních vozů pak 70 % a více.

Snímací systém je velmi propracovaný. Skládá se z čidel na kolejích, jež mají laserové skenery na obou stranách, a z centrály, která je instalována na panelovém podkladu kolejiště, nebo případně upevněna na blízkém sloupu. Stanice obsahuje také řádkovou vysokorychlostní kameru o vysokém rozlišení, která je připravena automaticky osvěcovat záznam v případě zhoršených světelných podmínek. Výstupem kamery je podélný záznam soupravy, který je použit k dalšímu zpracovávání a rozpoznávání.

3.4 Logický rámec

Tabulka 2 Logický rámec aplikace

Cíl projektu	**Objektivně ověřené ukazatele**	**Prostředky ověření**	
Vytvoření systému pro identifikaci vlakových souprav	*Používání systému v reálném provozu*	*Výstupy systému*	
Účel projektu	**Objektivně ověřené ukazatele**	**Prostředky ověření**	**Předpoklady**
Automatizace zadávání souprav do systému *Zjednodušení zadávání souprav obsluze*	*Úspěšnost systému nad 60%* *Zlepšení práce s daty*	*Výpis systému* *Reference obsluhy*	*Spuštění systému a používání*
Výstupy	**Objektivně ověřené ukazatele**	**Prostředky ověření**	**Předpoklady**
Popis stávajícího řešení *Aplikace pro administraci souprav* *Aplikace pro validaci souprav* *Kamera pro získání záznamů*	*Zdokumentování problematiky* *Použití aplikace obsluhou* *Propojení aplikací do systému*	*Analýza procesu* *Implementace do produkčního systému*	*Znalost problematiky* *Připravená infrastruktura pro provoz*
Činnosti	**Objektivně ověřené ukazatele**	**Prostředky ověření**	**Předpoklady**
Analýza problematiky *Analýza možností detekce souprav*	*Dodržení harmonogramu vývoje*	*7/2019 – seznámení s problematikou, analýzy, výběr vhodných technologií*	*Výběr vhodných technologii* *Finanční prostředky*

Využití strojového učení pro detekci *Vytvoření kamerového agenta* *Vytvoření aplikace pro ohodnocení dat* *Získání dat pro strojové učení* *Ohodnocení dat pro strojové učení* *Implementace aplikace* *Dokumentace aplikace* *Testování* *Použití v produkčním provozu* *Zajištění infrastruktury pro provoz*		*8/2019 – vytvoření kamerového agenta* *11/2019 – testování kamerového agenta a získání dat* *12/2019 – implementace aplikace na ohodnocení dat a ohodnocení dat* *1/2020 – Implementace systému* *3/2020 – Testování, dokumentace* *4/2020 – Nasazení do provozu*	*Podpora vyššího managementu*

4 Teoretická část

4.1 Předzpracování obrazu

Obrazové zpracovávání snímků je klíčovým procesem pro následné rozpoznávání dat z každého vozu. Upravuje snímek tak, aby mohl být použit pro rozpoznání neuronovou sítí. Je to proces, který upravuje data pro lepší následnou práci se snímkem. Obrazové zpracování je využíváno pro veškeré změny, jež potřebujeme na snímku udělat – změnu velikosti snímku, rotaci, úpravu barev, jasu, kontrastu, odebrání šumu nebo detekci hran. Bez tohoto předzpracovávání by některé snímky nebylo možné použít pro následné zpracování v neuronových sítích.

S následujícími metodami předzpracování bylo experimentováno na počátku práce s cílem získat snímek, který bude snáze zpracovatelný. Ne všechny techniky se osvědčily jako zlepšující pro potřeby práce a v cílové aplikaci byly tudíž následně použity pouze některé z nich.

Zvýšení jasu

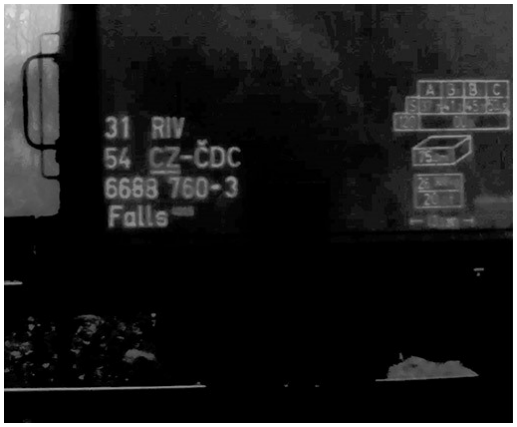
Cílem zvýšení jasu je plošné zesvětlení snímku, aby především tmavá místa byla lépe viditelná a čitelná. Zesvětlení se provádí přičtením konstanty ke každému bodu snímku. Maximální hodnota bodu je 255, která reprezentuje bílou barvu. [4]

Obrázek 5 Originální snímek

Obrázek 6 Zesvětlený snímek

Vyrovnání histogramu a gamma korekce

Jedná se o metodu pro zvýšení kontrastu snímku roztažením hodnot po celé šířce histogramu. Pokud je snímek tmavý, histogram je vyvýšený pouze v lokální oblasti. Vyrovnání histogramu způsobí roztažení této malé oblasti do celé jeho šíře a tím se rozloží světelnost snímku. [5]



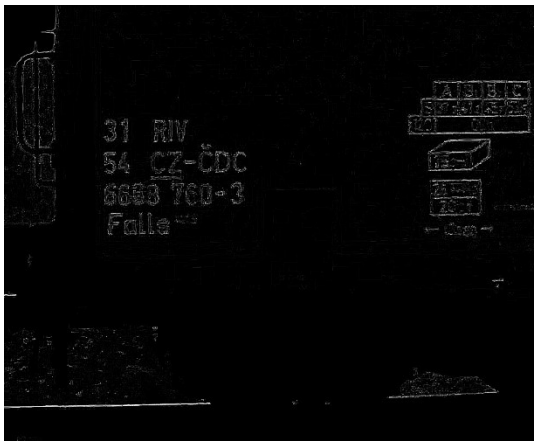
Obrázek 7 Vyrovnání histogramu a gamma korekce

Detekce hran pomocí masky

Tento princip je zajištěný pomocí konvoluce. Konvoluce je používaná pro úpravu jednotlivých bodů snímku. Jelikož snímek je reprezentován maticí bodů. Na jednotlivé body aplikujeme masku. Maska může být o rozměrech 3x3. Hodnoty v matici masky ukazují, jak okolní body ovlivní výsledný bod, na který je maska aplikována. Různým sestavením masky můžeme dokázat rozmazání, doostření anebo v této práci testovaný detektor hran. [6] Použití detektoru hran pro předzpracování se ukázalo jako nepřínosné oproti použitým metodám.

$$\begin{pmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{pmatrix}$$

Použitá matice masky pro detekci hran



Obrázek 8 Aplikace detekce hran pomocí masky

Cannyho hranový detektor

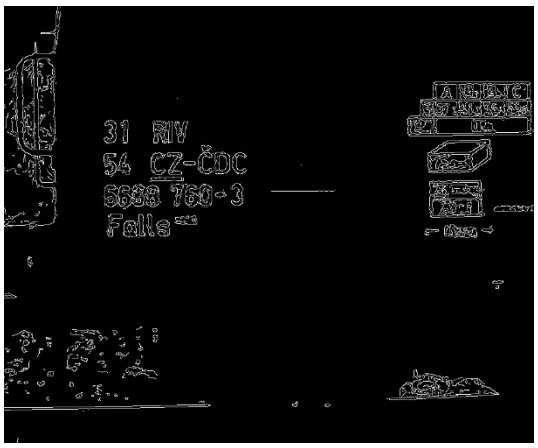
Jedná se o algoritmus vyvinutý Johnem F. Cannym v 80. letech 20. století, který se využívá pro detekci hran v obrazu. Algoritmus je spojený z více kroků.

První krok je odstranění šumu Gaussovým filtrem, aby nedocházelo k chybným počátkům hran. Na zjemněném snímku je následně provedeno vyhledávání gradientu pomocí Sobelova operátoru v horizontálním i vertikálním směru. Z první derivace těchto dvou směrů jsme schopni získat velikost gradientu a jeho směr pro každý pixel. Směr gradientu je vždy kolmý na hranu.

Dalším krokem je ztenčit získané gradienty pomocí vyhledávání lokálních maxim. Díky tomu získáme tenkou linii hrany.

Posledním krokem algoritmu je nalézt reálné hrany, jelikož algoritmus může detekovat nesprávné hrany způsobené například zbylým šumem. Tím si určíme maximální a minimální hodnotu prahu. Reálná hrana se vyznačuje tím, že minimálně jeden její bod je vyšší než maximální hodnota prahu a další napojené body hrany se pohybují v intervalu mezi minimální a maximální hodnotou prahu. Pokud ani jeden bod z hrany nepřesáhne maximální hodnotu, nejedná se o reálnou hranu [7]. Jedná se o dobře použitelný algoritmus na detekci hran na

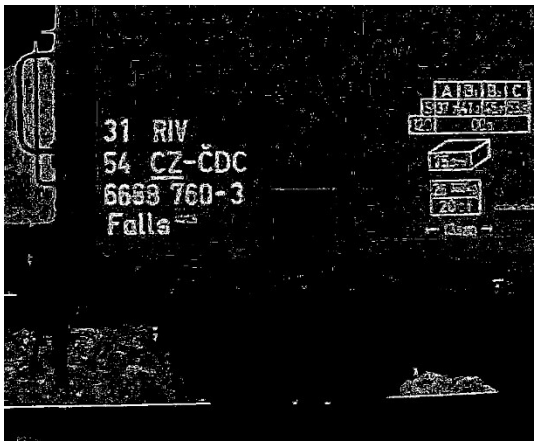
různých typech snímků, ale pro snímky vozů se ukázal jako nevhodný, z důvodu splynutí detailních informací.



Obrázek 9 Aplikace Cannyho hranového detektoru

Adaptivní prahování

Obecně řečeno, prahování je převedením snímku na binární snímek podle určitého prahu. Pokud je pod prahem, tak je bod tmavý, tudíž pokud jej překročí, je světlý. Globální prahování má narozdíl od adaptivního jen jeden práh pro celý snímek. Adaptivní prahování si vždy určí vlastní práh pro určitou sekci snímku. Práh sekce se určuje pomocí průměrováním hodnot v okolí bodů, nebo lépe určené gausovským filtrem [8]. Výhodou adaptivního prahování je možnost použití na snímcích s různým osvětlením. Pro předzpracování snímku v této práci se nejprve jevilo jako vhodné, ale produkuje stejné nedostatky jako předchozí metoda, a to splývání detailních informací.



Obrázek 10 Aplikace adaptivního prahování

4.2 Neuronové sítě

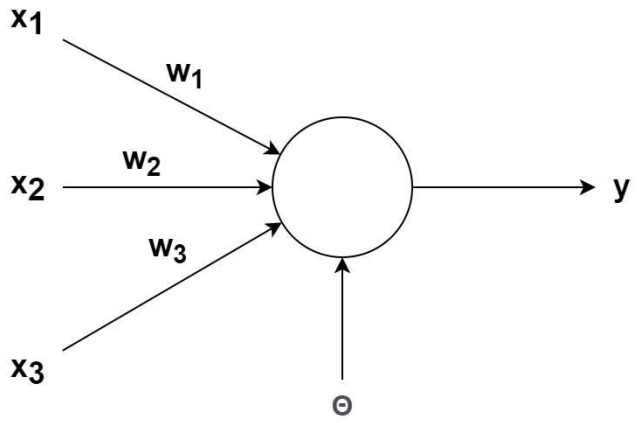
Neuronové sítě jsou modelem pro zpracování dat, který je inspirovaný biologickými strukturami. Model je složený z neuronů, které jsou vzájemně propojeny. Neurony jsou rozděleny do vrstev, kdy první vrstva je vstupní a poslední vrstva je výstupní. Vrstvy mezi vstupní a výstupní vrstvou se nazývají skryté vrstvy a jejich počet není omezen. Neuronové sítě se nejčastěji využívají pro zpracování obrazu, zvuků, či predikce číselných řad.

V této práci jsou neuronové sítě využity pro obrazové zpracování. Umožní detekovat pozici a identifikovat potřebné objekty na snímku.

Pro získání potřebného výsledku na základě vstupů jde potřeba neuronové sítě provést procesem učení. Jedním z postupů je učení s učitelem, kdy je vytvořena trénovací množina dat, která obsahuje vstupy a požadované výstupy neuronové sítě. Proces probíhá porovnáváním výstupu sítě s odpovídajícím požadovaným výstupem z množiny pro každý vzor z trénovací množiny. Ze získané chyby se vypočítá potřebná korekce hodnot neuronů a provede se další porovnání výstupů. Celý tento proces se opakuje, dokud není chyba výstupů co nejmenší. Správně naučené sítě jsou schopny reagovat i na data, na kterých nebyla síť naučena a dochází ke generalizaci. [9]

Neuron

Jedná se o základní jednotku neuronové sítě. Neuron může mít více vstupů na základě kterých určuje výstup – ten je vždy pouze jeden. Obsahuje vnitřní proměnné. První proměnnou jsou váhy, kterých je stejný počet jako vstupů. Váhy určují důležitost daného vstupu do neuronu. Další proměnnou, kterou obsahuje neuron typu perceptron, je práh. Neurony obsahují výstupní funkci, která je v hodnotách od 0 do 1 a může se jednat o skokovou funkci, sigmoidní funkci nebo modifikovanou Gaussovu křivku, dle které se určuje výstupní hodnota neuronu.



Obrázek 11 Ukázka umělého neuronu. Vstupy x, váhy w, práh Θ a výstup y

Hluboké sítě

Hluboká neuronová síť je typ použití strojového učení. Vyznačuje se vysokým počtem skrytých vrstev, které mohou být až v desítkách. Pro učení sítě se využívají gradientní metody učení. Neurony jsou uspořádány do vrstev, ze kterých se skládá neuronová síť. Neuronové sítě se tedy skládají ze střídajících se vrstev – konvoluční a MaxPooling, které se využívají při zpracování obrazu pro redukování velikosti vrstvy. **[10]**

Konvoluční vrstvy se využívají pro zpracování obrazu v hlubokých neuronových sítích. Úkolem je získat informace o oblastech z obrazu. Konvoluční vrstva získává data z malých oblastí například o velikosti 3x3 nebo 5x5 pixelů. Data z této oblasti jsou vstupy do jednoho neuronu. Oblasti se navzájem překrývají a pro každou oblast je jeden neuron. [11]

MaxPooling se využívá pro redukování rozměrů obrazu. Obraz je rozdělen do nepřekrývajících se oblastí a z každé z nich se vybere jedna hodnota – nejčastěji se jedná o nejvyšší hodnotu v každé oblasti. Nalezená hodnota z oblasti je novou hodnotou v obrazu, která následně nahradí celou oblast. [12]

4.3 Tensorflow

Tensorflow je platforma pro strojové učení vytvořena společností Google. Jedná se o high-level API, které využívá standardu Keras[1] API pro vytváření modelů. Je využitelná začátečníky, ale i pokročilými uživateli. Nabízí širokou skupinu použití v různých

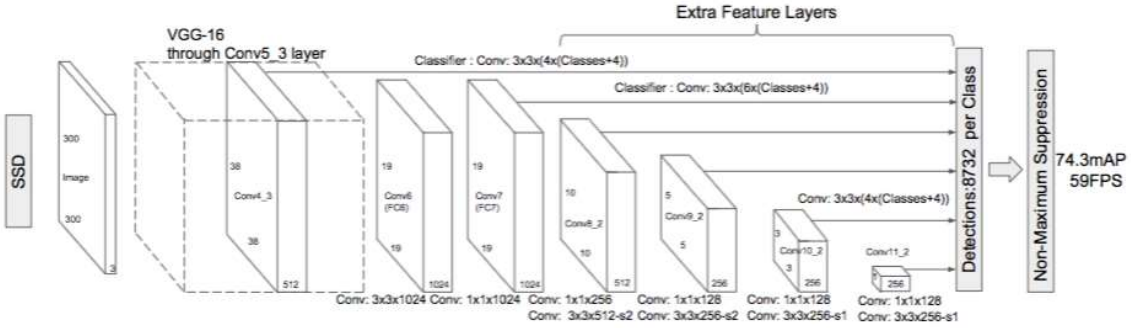
[1] Open-source knihovna pro vytváření neuronových sítí

programovacích jazycích. Platforma obsahuje kvalitní dokumentaci a mnoho příkladů, které zjednodušují použití. [13]

Jednou z možností použití Tensorflow je zpracování obrazu. Platforma nabízí přednaučené modely pro klasifikaci snímku, odhad postavy osoby, segmentaci snímku a detekci objektů na snímku. Připravený model pro detekci objektů detekuje 80 typů boxů. Tyto modely jsou nadále využívány pro přeučení se na vlastních datech pro specifické případy.

4.4 SSD Inception v2 model

Platforma Tensorflow poskytuje mnoho přednaučených modelů ve svém repozitáři, kterých se dá využít k přeučení na vlastní detekci objektů na snímku. Jedním z těchto modelů je SSD Inception v2, který se skládá ze dvou hlavních částí. Model je vytvořen na principu „Single Shot MultiBox Detektor“ (SSD). Tento model se začal používat kvůli potřebě zpracování videa v reálném čase. Konvoluční neuronové sítě neposkytovaly dostatečnou rychlost zpracování. Jedním průchodem je SSD model schopen určit lokalizaci s klasifikaci boxu [14] [15].



Obrázek 12 Schéma SSD modelu (převzato z [14])

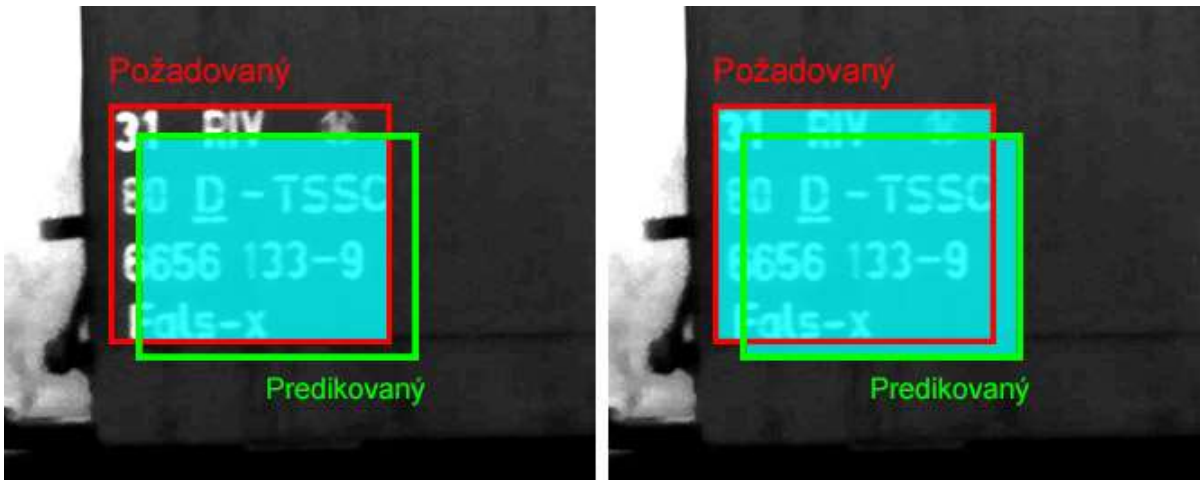
Prvotní zpracování dat provádí Inception v2. Tato část zpracovává data na více konvolučních filtrech o velikostech 1x1 a 3x3. Cílem je, aby model nerostl tolik do hloubky, ale spíše do šířky. Různé velikosti filtrů zajišťují vyhledávání větších objektů na snímku, ale i vyhledání detailnějších objektů. Konvoluční vrstva rozděluje snímek do oblastí, kdy pro každou oblast je určen jeden neuron [16].

Po zredukování počtu vstupů pomocí Inception v2, jsou data zpracovávána dalšími konvolučními vrstvami, které jsou určeny k detekci objektu. Data jdou od konvoluční vrstvy s vyšším rozlišením pro hledání detailnějších objektů, až po vrstvu s nejnižším rozlišením,

která je určena pro větší objekty. Pro hodnocení těchto modelů se využívá IoU (intersection over the union). Pro výpočet této hodnoty se využívá poměr přesahující oblasti ku celkové oblasti. Pokud hodnota IoU se blíží k nule, znamená to, že nalezené oblast se nepřekrývá s požadovanou. Pokud je hodnota jedna, oblasti se identicky překrývají. Cílem neuronových sítí je, aby tato hodnota byla co nejvyšší. [14]

$$IoU = \frac{Přesahující\ plocha}{Celková\ plocha}$$

Vzoreček pro výpočet IoU hodnoty



Obrázek 13 Vlevo přesahující oblast, vpravo celková oblast

4.5 Azure

Azure je globální platforma od společnosti Microsoft poskytující cloudové služby. Cílem platformy Azure je centralizovat služby firem do jednoho systému. Azure poskytuje velké množství produktů, například uložiště, databáze, webový hosting, virtuální počítače, internet věcí anebo strojové učení. Vše je obsluhováno pomocí webového rozhraní a výkonnější služby jsou zpoplatněny. Komunikace s aplikací je zprostředkována pomocí HTTP API požadavků na určené url služby [17].

4.6 OCR

Optické zpracování znaků je proces pro převedení tištěného nebo ručně psaného textu na strojově čitelný text. Pro čtení textu se využívají naskenované nebo vyfocené textové

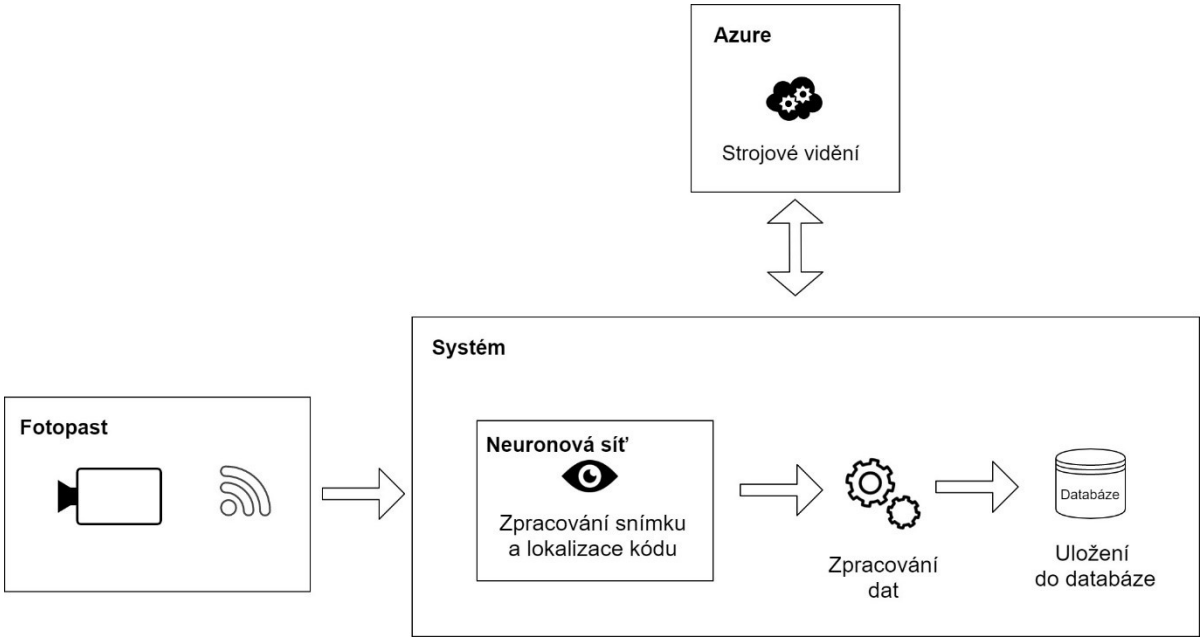
dokumenty. Znaky se mohou vyhledávat pomocí porovnávání vzorů ve snímku nebo pomocí strojového učení. [18]

Na trhu se pohybuje mnoho technologií pro optické zpracování znaků. V této práci byly porovnány dvě nejdostupnější. Knihovna Tesseract[2] a cloudová služba Azure. Pro implementaci byl vybrán Azure. Důvod rozhodnutí a postup porovnaní je popsán v kapitole 5.2.2.

[2] https://github.com/tesseract-ocr/tesseract

5 Návrh řešení

Pro zajištění automatizace je potřeba zkompletovat několik systémů dohromady. Je potřeba zajistit získání snímku vozu levným a univerzálním řešením, které bude snadno umístitelné ke kolejišti. Na získaných snímcích je následně potřeba nalézt a přečíst identifikační kód vozu, který je poté zpracován do dat srozumitelných pro obsluhu, jenž bude tuto aplikaci využívat.



Obrázek 14 Návrh řešení

Ke sběru dat je potřeba využit obrazový snímač. Jelikož v současné době nejsou koleje vybaveny potřebnými kamerami, je zapotřebí sestavit levné a univerzální řešení. Pro sběr dat je vytvořena fotopast, která je nezávislá na elektrické síti a též nepotřebuje připojení k internetu. Fotopast ukládá videa pouze po zaznamenání pohybu před kamerou.

Ze snímků potřebuje aplikace vyhledat a přečíst identifikační kód vozu. Pro detekování kódu je vytvořena vlastní neuronová síť, která na snímku vyhledá pozici a typ daného identifikačního kódu. Pro vytvoření vlastní neuronové sítě je zapotřebí získat dostatečný dataset vozů, který je následně zpracován na trénovací a testovací množinu dat. Detailnější popis a postup vytvoření neuronové sítě je v kapitole 7.

Po nalezení kódu je potřeba převést snímek na strojově čitelný text. Tento proces zajišťuje strojové vidění cloudové služby Azure. Přečtením snímku získáme kód, který je možné dál zpracovávat.

5.1 Příprava trénovacích množin pro neuronové sítě

5.1.1 Stávající řešení

Pro anotaci dat ozkoušena níže vypsaná stávající řešení. Žádná z těchto aplikací nenaplnila očekávání a nedosáhla potřebné plynulosti zpracovávání dat z důvodu nepříznivého uživatelského prostředí. Z tohoto důvodu byla pro anotaci dat vytvořena vlastní aplikace.

Labelbox

Jedná se o online platformu pro přípravu dat k učení neuronové sítě. Vyhledávané objekty na snímku je možné označovat pomocí boxů nebo pomocí polygonů. Data je možné vyexportovat ve formátu JSON. Jelikož se jedná o webovou aplikaci je vhodná též pro práci v týmu. Obsahuje i možnost kontroly kvality hodnocení. Její rozsáhlejší použití je zpoplatněno. [19]

LabelImg

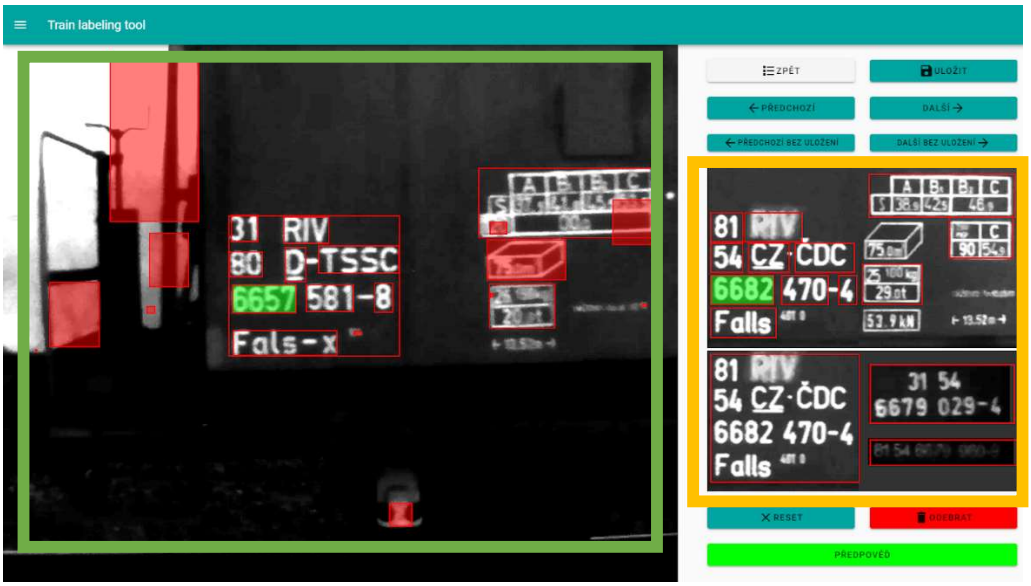
LabelImg je grafická aplikace vytvořena v jazyku Python s grafickým rozhraním v Qt. Aplikace je pod licencí MIT a zdrojové kódy jsou publikovány na githubu[3]. Snímky je možné hodnotit pomocí označování boxů a výstupem aplikace jsou .xml soubory se souřadnicemi. Aplikace umožňuje rychlé zpracování použitím klávesových zkratek. [20]

5.1.2 Návrh aplikace pro anotaci dat

Pro anotaci dat byla vytvořena vlastní aplikace z důvodu nedostatku funkcí aktuálních řešení. Jednou z výhod navrhované anotační aplikace oproti stávajícím řešením je samotný výběr typu boxu. Ten není volen podle názvu anotovaného boxu, ale přímo z ukázkového snímku, na kterém je vidět, který typ boxu je anotován.

Výhodou je možnost implementace vlastního kódu, který usnadní anotování snímku. Součástí aplikace je také samotné předzpracování snímku. Například na Obrázek 15 vidíme dvě barevně rozlišené části. Zeleně ohraničen je snímek určený k anotaci dat. Žlutě ohraničen je příklad snímku, který obsahuje všechny boxy určené k anotaci. Kliknutím na tyto jednotlivé boxy ve žluté části vybereme typ boxu, který obsluha chce anotovat na snímku v zelené části. Obsluha tahem myši označí anotovanou část snímku daným typem boxu. Tímto postupem získáme typ a pozici boxu na anotovaném snímku.

[3] https://github.com/tzutalin/labelImg

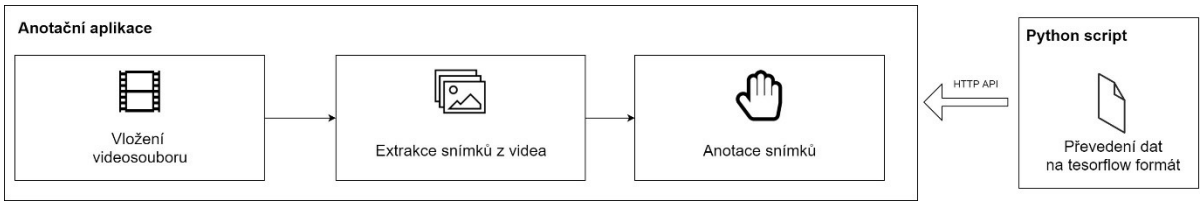


Obrázek 15 Ukázka anotační aplikace

Dalším důvodem k vytvoření vlastní aplikace k anotaci, je seznámení autora práce s procesem zpracování množin pro učení neuronových sítí a nezávislost na externím systému.

Získaná data je potřeba anotovat pro naučení vlastní neuronové sítě. Cílem je obrazový záznam anotovat a převést na data, která poté využívá síť ve frameworku Tensorflow[4] pro učení.

Anotace je informace, která je přidána ke snímku a tímto rozšíří jeho informační hodnotu. Anotace v tomto kontextu je zadání pozice a typu určité informace na snímku.



Obrázek 16 Proces anotace snímku

Jedním z cílů je tedy převést video na statický obraz, který je použitelný pro anotaci. První pokus byl inspirován řádkovou kamerou. Bylo potřeba získat tenký pruh ze snímků videa o určité frekvenci, poskládat pruhy za sebe a sestavit tak dlouhý snímek celé vlakové soupravy. Jelikož systém nebyl opatřen čidly pro zjištění aktuální rychlosti soupravy, bylo získání rychlosti z obrazového záznamu poměrně obtížné. Celý proces pokusu je popsán v kapitole 8.1.

[4] https://www.tensorflow.org/

Jako nejlepší postup se nakonec osvědčilo snížení frekvence snímání a převedení snímků na statický obraz ve formátu png. Frekvence se snížila na 10 snímků za vteřinu. Díky tomu se anotované objekty opakovaly na různých pozicích snímku. Vygenerovaný snímek je převeden na stupně šedé a z důvodu snížené kompenzace expozice kamery je zároveň zesvětlen. Na snímek byl také vyzkoušen sharpening pomocí matice nebo Cannyho detekce hran, ale všechny tyto pokusy měly horší výsledky než samotný zesvětlený obraz.

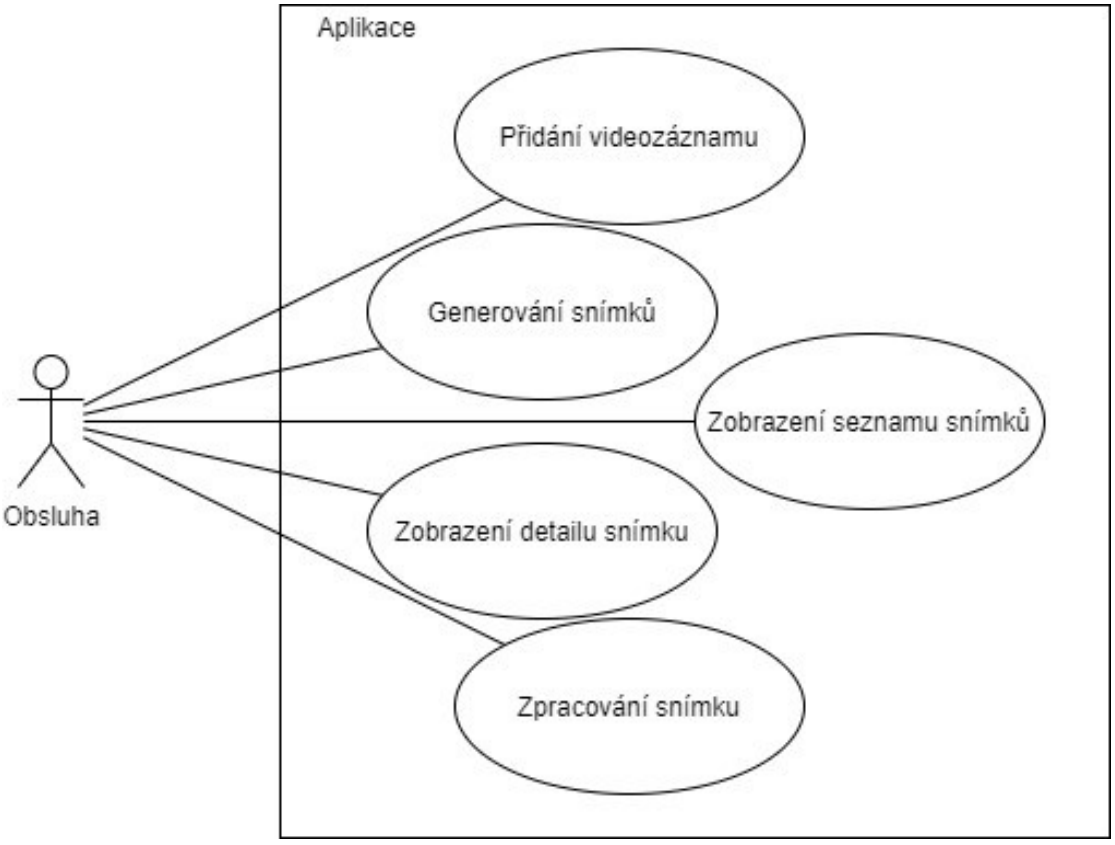
Na výsledných statických snímcích jsou anotovány objekty potřebné pro detekci, což jsou v tomto případě texty a tabulky. Součásti anotační aplikace je knihovna Tesseract, která vyhledává texty na snímku. Předem vyhledané texty usnadňují anotaci, kde se daný objekt případně pouze doupraví.

Použití aplikace probíhá tak, že obsluha na začátku vloží do aplikace cestu k videu, který má být zpracováno. Aplikace z vybraného videa extrahuje snímky pro anotaci. Obsluha prochází extrahované snímky, na kterých anotuje boxy pomocí postupu, který je popsaný výše. Funkcionalita programu je formálně popsána v následující kapitole 5.1.3.

Pomocí anotační aplikace obsluha anotuje na snímcích 15 boxů. Jedná se o boxy, které označují bloky informací, ale i také informace dílčích elementů. Jednotlivé boxy jsou dle struktury identifikačního kódu popsané v kapitole 3.1.

Snímky jsou ukládány na disky. V databázi aplikace je uložena cesta ke snímku se všemi vytvořenými anotacemi. Množina zpracovaných snímků je převedena pomocí Python scriptu na testovací a trénovací množiny, které jsou převedeny do formátu pro učení pomocí Tensorflow.

5.1.3 Scénáře použití



Obrázek 17 Diagram užití pro anotační aplikaci

Přidání videozáznamu

Registrace videozáznamu do systému

Základní scénář

1. Systém nabídne obsluze formulář.
2. Obsluha vyplní cestu k souboru s videozáznamem a odešle.
3. Systém uloží údaje do databáze.

Generování snímků

Generování a předzpracování snímků z videozáznamu.

Vstupní podmínky

Obsluha vybrala daný videozáznam.

Základní scénář

1. Obsluha požaduje po systému generování na daném videozáznamu.

2. Systém zpracuje video na snímky a provede předzpracování. Systém uloží vytvořené snímky a zavede do databáze.

Zobrazení seznamu snímků

Zobrazení vygenerovaných snímků z videozáznamu.

Vstupní podmínky

Obsluha vybrala daný video záznam.

Základní scénář

1. Systém zobrazí seznam vygenerovaných snímků daného videozáznamu.

Zobrazení detailu snímků

Zobrazení detailu pro daný snímek.

Vstupní podmínky

Obsluha vybrala daný snímek.

Základní scénář

1. Systém zobrazí detailní popis daného snímku.

Zpracování snímku

Ohodnocení daného snímku.

Vstupní podmínky

Obsluha má zobrazený detail daného snímku.

Základní scénář

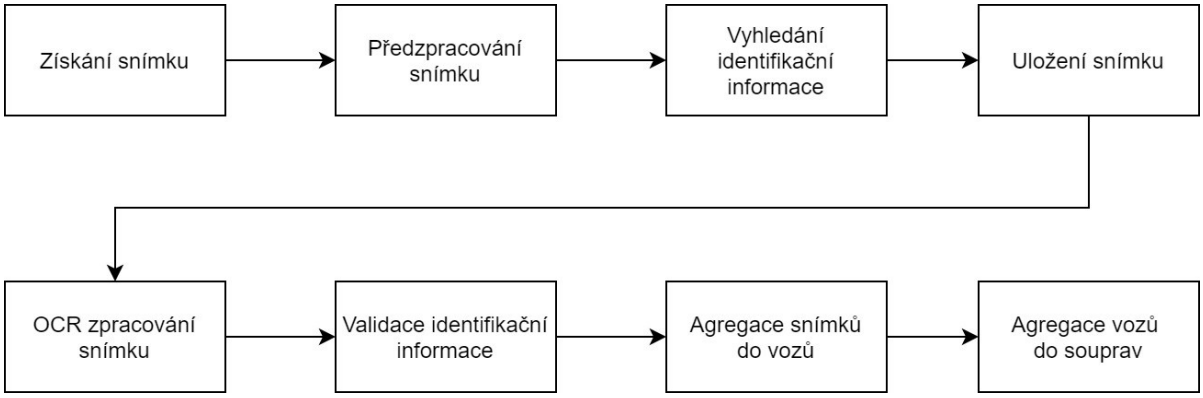
1. Provedení scénáře: Zobrazení detailu snímků
2. Obsluha oklasifikuje daný snímek a uloží.
3. Systém uloží data do databáze.

5.2 Návrh aplikace pro identifikaci vlakových souprav

Ve této kapitole je popsán návrh řešení aplikace pro identifikaci vlakových souprav. Aplikace zajišťuje automatické zpracování snímků, na kterých bude pomocí neuronové sítě nalezen

identifikační kód vozu. Pomocí strojového vidění Azure bude zajištěno převedení kódu ze snímku na text. Aplikace bude poté získané snímky s kódy seskupovat do dat přehlednějších pro obsluhu.

Aplikace bude sloužit zaměstnancům Severočeských dolů k získání seznamu souprav a vozů, které byly zpracovány. Obsluha bude moci načtené vozy libovolně upravovat a případně manuálně dopsat ty, jenž nebylo možné automaticky zpracovat, například z důvodu špatné kvality identifikačního kódu. Návrh procesu automatizovaného zpracování na snímku je blokově popsán na Obrázek 18.



Obrázek 18 Proces zpracování snímku identifikační aplikací

Proces automatického zpracování bude začínat přijmutím snímku pořízeného fotopastí do aplikace. Snímky budou seřazeny do fronty, aby bylo předcházeno přetížení aplikace a zajištěno plynulé zpracování pouze omezeného množství v daný moment.

Jednotlivé snímky budou otočeny do správné orientace, převedeny do šedotónu a zesvětleny. Snímek tímto bude předzpracován pro použití v neuronové sítí, která bude naučena pro vyhledávání identifikačního kódu vozu na snímku.

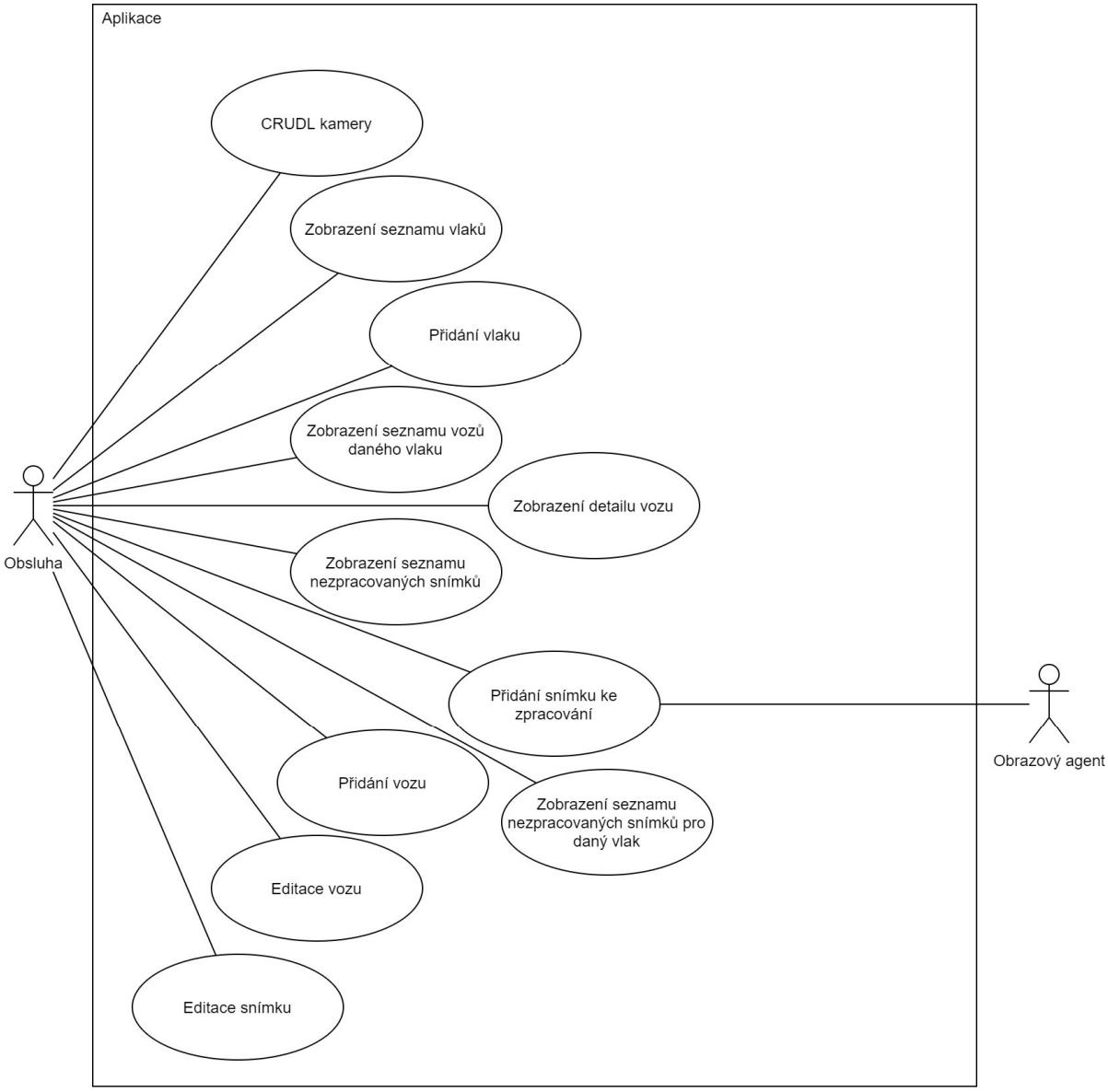
Snímky, které neobsahují identifikační kód budou zahozeny, uloženy na disk budou pouze snímky s identifikačním kódem. Dále bude snímek odeslán na cloudovou službu Azure, která pomocí OCR přečte identifikační kód. Pozice přečteného textu budou porovnány s pozicemi získanými z neuronové sítě pro ověření, že se skutečně jedná o identifikační kód vozu.

Přečtený identifikační kód bude validován pomocí algoritmu, který je detailně popsaný v kapitole 3.1. Do databáze bude uložena cesta ke snímku, pozice nalezených identifikačních kódů, přečtený kód a zdali je validní. Po vyprázdnění fronty budou snímky agregovány do

jednotlivých vozů, jelikož jeden vůz může být na více snímcích. Následně budou vozy agregovány do vlakových souprav. Vozy i vlakové soupravy budou uloženy do databáze.

Funkcionalita celé identifikační aplikace s automatickým zpracováním a možnostmi pro obsluhu je popsána formálně v následující kapitole 5.2.1.

5.2.1 Scénáře použití



Obrázek 19 Diagram užití identifikační aplikace

CRUDL[5] kamery

Vytvoření nového záznamu kamery, editace záznamu kamery, smazání záznamu kamery, zobrazení seznamu kamer, zobrazení detailu kamery.

Zobrazení seznamu vlaků

Zobrazení seznamu vlaků ze systému.

Základní scénář

1. Systém zobrazí seznam vlaků.

Přidání vlaku

Vytvoření nového záznamu vlaku.

Základní scénář

1. Obsluha zvolí nový vlak.
2. Systém vytvoří nový záznam vlaku.

Zobrazení seznamu vozů daného vlaku

Zobrazení seznamu vozů pro vybraný vlak.

Vstupní podmínky

Vybraný daný vlak.

Základní scénář

1. Obsluha vybere daný vlak.
2. Systém zobrazí seznam vozů po daný vlak.

Zobrazení detailu vozu

Detailní popis vozu.

Vstupní podmínky

Vybraný daný vůz.

Základní scénář

1. Obsluha vybere daný vůz.
2. Systém zobrazí detailní popis vozu.

[5] Create, read, update, delete, list (Vytvořit, číst, upravit, smazat, seznam)

Zobrazení seznamu nezpracovaných snímků

Zobrazení všech systémem nezpracovaných snímků.

Základní scénář

1. Systém zobrazí seznam všech vyfiltrovaných snímků, které nemají přiřazený vůz.

Zobrazení seznamu nezpracovaných snímků pro daný vlak

Zobrazení všech nezpracovaných snímků systémem pro daný vlak

Vstupní podmínky

Vybraný daný vlak.

Základní scénář

1. Obsluha vybere daný vlak
2. Systém zobrazí seznam všech vyfiltrovaných snímků, které nemají přiřazený vůz a zasahuje do časového intervalu vlaku.

Přidání vozu

Manuální přidání vozu pro vlak

Vstupní podmínky

Vybraný daný vlak.

Základní scénář

1. Obsluha vybere daný vlak.
2. Systém zobrazí formulář pro zadání dat.
3. Obsluha zadá data a odešle.
4. Systém vytvoří nový vůz o daných datech a přiřadí je danému vlaku.

Editace vozu

Upravení záznamu vozu.

Vstupní podmínky

Vybraný daný vůz.

Základní scénář

1. Obsluha vybere daný vůz.
2. Systém zobrazí formulář pro editaci dat.
3. Obsluha upraví data a odešle.

4. Systém aktualizuje data.

Editace snímku

Upravení dat u snímku.

Vstupní podmínky

Vybraný daný snímek.

Základní scénář

1. Obsluha vybere daný snímek
2. Systém zobrazí formulář pro zpracování snímku.
3. Obsluha upraví data a odešle.
4. Systém aktualizuje data u snímku a pokusí se přiřadit snímek k vozu.

Přidání snímku ke zpracování

Zavedení nového snímku do systému

Vstupní podmínky

Požadavek obsahuje snímek.

Základní scénář

1. Obsluha vloží snímek systému.
2. Systém snímek předzpracuje, vyhledá značky a zpracuje.

5.2.2 Volba OCR technologie

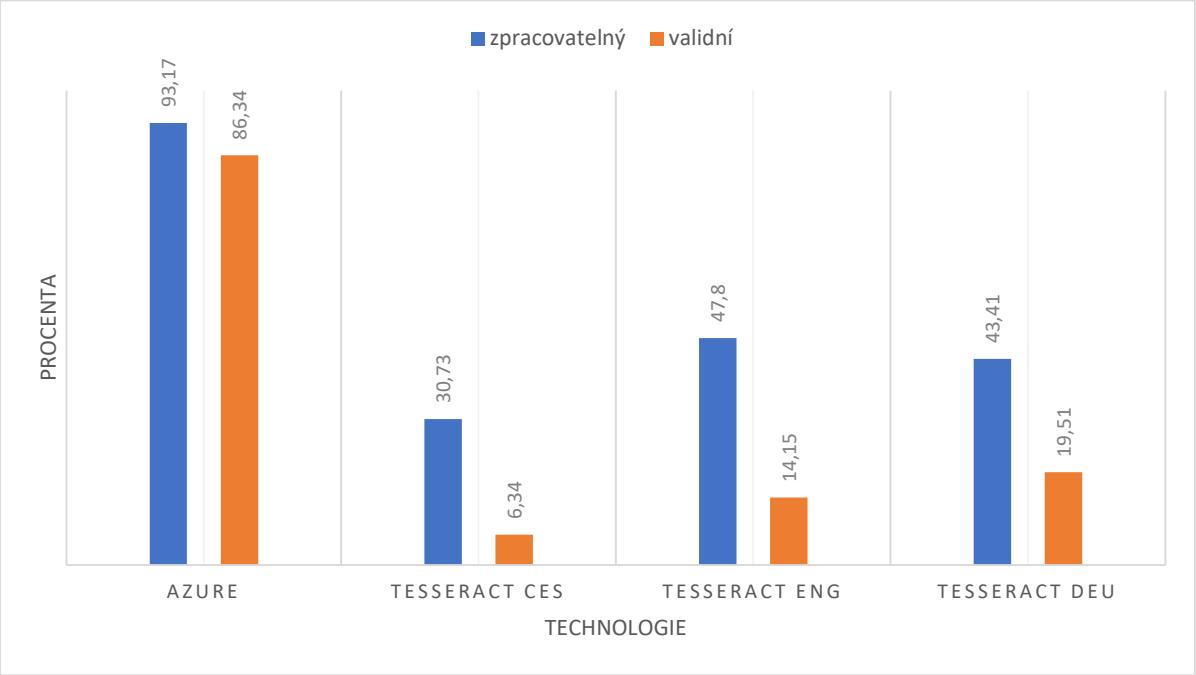
Před vlastní implementací aplikace byly pro obrazové zpracování textu otestovány dvě technologie. První technologií je počítačové zpracování obrazu od Microsoft Azure. Tato technologie obsahuje mnohé možnosti zpracování obrazu a jednou z nich je i zpracování textu. Jedná se o cloudovou službu, které je pro větší počet požadavků zpoplatněna. Výhodou je, že pro implementaci nejsou potřeba znalosti se strojovým učením a je ulehčeno aplikaci od složitějších výpočtů.

Druhou porovnávanou technologií je Tesseract. Jedná se o OCR[6] Engine, který je pod licencí Apache. Tesseract vyžaduje důkladnější předzpracování obrazu. Podporuje více jak 100 jazyků.

[6] Optical Character Recognition (Optické rozpoznávání znaků)

Porovnání bylo provedeno na 205 snímcích, na nichž byl k přečtení celý identifikační kód. Výsledek z technologií byl testován, aby se ukázalo, zda je možné zpracovat text pomocí regulárního výrazu na identifikační kód[7] a následně zdali je kód číselně validní[8]. Technologie Tesseract byla otestována s českým, anglickým a německým modelem, jelikož vozy pocházejí z různých zemí. Na Azure byly snímky odesílány v plném rozlišení a následně se vyhledával text, Tesseractu byly předkládány výřezy identifikačního kódu.

Z porovnání vyšlo lépe Azure, které bylo následně v aplikaci použito. Důvodem může být nedostatečné předzpracování obrazu pro technologii Tesseract, ale výsledek je důkazem toho, jak je použití Azure jednoduché a kvalitní i na zhoršené kvalitě dat.



Obrázek 20 Provnání Technologie Azure a Tesseract

5.2.3 Přehled vybraných technologií

Server

Na serverové části, která jen napsána v programovacím jazyku C# .NET byly využity knihovny pro ukládání dat, aplikování neuronových sítí a dotázání do externích systémů.

Pro práci s daty je využitý Entity framework. Zajišťuje napojení do databáze, mapování objektů s databází a migrace schématu databáze. Entity framework byl použitý

[7] O regulárním výrazu více v kapitole Serverová část
[8] O validitě kódu více v kapitole Značení vozů

stylem code first, kdy je z kódu generována struktura databáze. K implementaci Tensorflow byla využita knihovna TensorFlowSharp[9], která zajišťuje možnost použití Tensorflow v C#. Ke komunikaci s externími systémy byla využita knihovna RestSharp[10], jenž generuje REST API požadavky.

Azure

Na platformě Azure byla vytvořena služba počítačového zpracování textu[11]. Tato externí služba zajišťuje čtení textu ze snímku. Platforma Azure je detailněji popsána v kapitole 4.5.

Klientská část

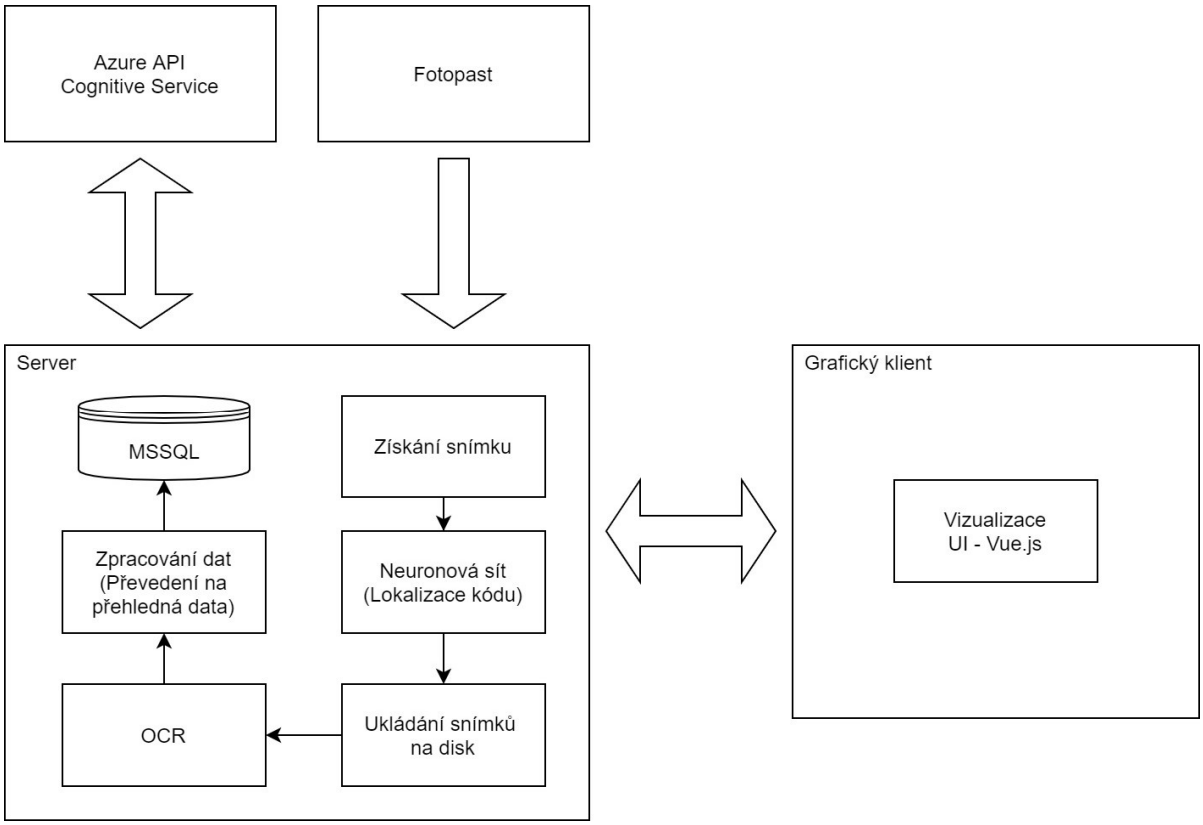
Uživatelská strana je vytvořena jako „single page application“ (SPA), která se načte pouze jednou, následně se při uživatelských interakcích obnovuje a získává data pouze pro změněné komponenty. Tento způsob použití zajišťuje vysokou rychlost práce s aplikací.

Pro lepší práci s Javascriptem je využitý Typescript[12], jenž zajišťuje typování Javascript a přidává další funkčnost pro zjednodušení vývoje a získání lepší přehlednosti kódu. Interaktivní SPA je vytvořena ve frameworku Vue.js, ve kterém se vytvářejí komponenty a všechna logika uživatelské části aplikace. Aby uživatel mohl přistupovat ke komponentám pomocí url, je použitý Vue-router[13], který routuje jednotlivé stránky dle url. Data fronendu jsou uložena pomocí Vuex[14] v globálním storu aplikace. Globální store se využívá z důvodu lepší práce s daty napříč mnoha komponentami. Ke komunikaci se serverem je použitá knihovna Axios[15], která generuje REST API požadavky na serverovou stranu. Uživatelská část je graficky ostylována pomocí frameworku Vuetify[16], jenž je sadou předem graficky nadefinovaných a před vytvořených komponent. Pro vlastní styly byl použit sass[17]. Na frontendu je použita neuronová sít, která je implementována pomocí knihovny Tensorflow.js[18]. Celý vývoj systému je verzovaný pomocí nástroje git[19], kde je vidět proces vývoje kódu.

[9] https://github.com/migueldeicaza/TensorFlowSharp
[10] http://restsharp.org/
[11] https://azure.microsoft.com/cs-cz/services/cognitive-services/computer-vision/
[12] https://www.typescriptlang.org/
[13] https://router.vuejs.org/
[14] https://vuex.vuejs.org/
[15] https://github.com/axios/axios
[16] https://vuetifyjs.com/en/
[17] https://sass-lang.com/
[18] https://github.com/tensorflow/tfjs
[19] Systém správy verzí

5.2.4 Architektura systému

Systém je rozdělen na dvě hlavní části, které mezi sebou komunikují. Z důvodu použití frameworku Vue.js se mezi serverem a uživatelskou částí nepředává celé HTML[20], ale pouze potřebná data a HTML kód je následně sestavený v klientské části. Data si předávají ve formátu JSON[21] pomocí HTTP API.



Obrázek 21 Architektura systému

Serverová část aplikace udržuje všechna data, na kterých provádí náročnější operace, které se netýkají zobrazovací logiky. Také zajišťuje komunikaci s externími API a použití neuronové sítě. Zároveň je cílovým bodem pro zasílání snímků z fotopastí, které jsou zde zpracovány. Klientská část zpracovává uživatelsky přívětivé zobrazení dat a díky javascriptovému frameworku je zde přesunuta část logiky aplikace, která umožní snížit zátěž na serverovou část a zrychlit celou aplikaci.

[20] Hypertext Markup Language
[21] JavaScript Object Notation. Datový formát pro přenos dat.

6 Implementace

6.1 Fotopast pro sběr experimentálních dat

Ke sběru dat byla vytvořena fotopast, která na pohyb před fotopastí ukládá videozáznam. Pro účel natrénování vlastní neuronové sítě je zapotřebí získat dostatek dat. Pro získávání dat bylo vybráno místo s nejvyšší četností průjezdů vozů. Dané místo se nacházelo mezi rozvětvením kolejiště, kde jsou dvě odjezdové (příjezdové) koleje a jedna manipulační kolej. Velkou nevýhodou odlehlého místa je nezavedení elektřiny, žádné připojení k síti, ale i vysoké riziko krádeže. Zaznamenávané vozy se pohybují ve vzdálenosti 9 metrů od umístěné kamery. K pořízení dostatečně kvalitního snímku je zapotřebí snímač s dostatečným rozlišením, nebo čočka s nižším úhlem záběru, která obraz zvětší.

Obrázek 22 Umístění fotopasti na sloupu vedle kolejiště

První sběry dat probíhaly pomocí webové kamery Microsoft LifeCam Cinema[22], kdy autor byl fyzicky přítomen a manuálně zaznamenával. Tento postup byl časově poměrně náročný, neproduktivní a kamera se ukázala jako nedostatečná.

[22] https://www.microsoft.com/accessories/cs-cz/products/webcams/lifecam-cinema/h5d-00004

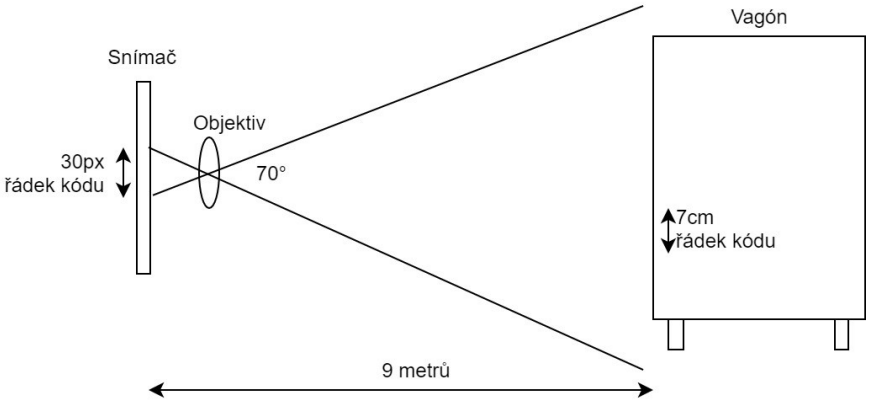
Dalším krokem byla snaha tento postup co nejvíce osamostatnit a zautomatizovat. S využitím Raspberry Pi Model B+[23], infračervené kamery s přisvícením a externí baterie autor sestrojil fotopast na zaznamenávání projíždějících vozů.

Díky programu napsaném v pythonu fotopast zaznamenává pouze v případě pohybu v zorném poli kamery. Pro jednodušší administraci se připojí na wifi, kde díky HTTP [24] serveru poskytuje potřebná data ke kontrole. Box, předem vytištěný na 3D tiskárně z modelu, byl přivázán na stožár vedle kolejí.

Vývoj fotopasti se neobešel bez komplikací. Nastal problém ve spojení Microsoft LiveCam a Raspberry Pi. Po otestování na novější verzi Raspberry Pi[25] bylo zjištěny limity použitého USB 2.0 pro propojení kamery a Raspberry Pi. Datová propustnost USB 2.0 není dostatečná[26] pro HD video o 30 snímcích za vteřinu. Stávající kamera vyřešila tento problém, poněvadž je napřímo připojena ke grafické kartě přes proprietární konektor a nevyžaduje řadič USB. Fotopast byla vyvíjena a testována na projíždějících automobilech, které byly pro vývoj lepe dostupné než vlakové kolejiště.

6.1.1 Hardwarová část

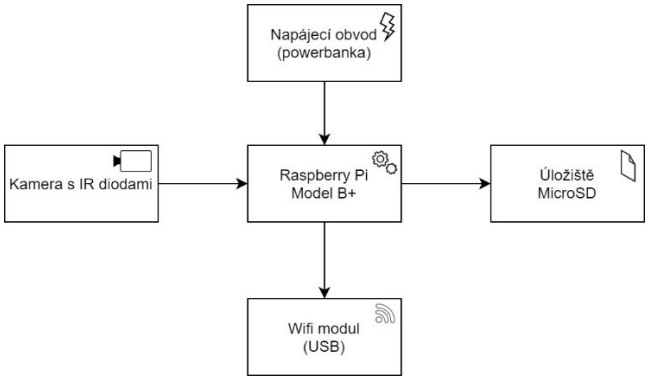
Pro spojení všech komponent fotopasti je jako jádro použito Raspberry Pi Model B+, které je napojené na kameru s rozlišením 1920px na 1088px a snímací frekvencí 30 snímků za vteřinu.



Obrázek 23 Ilustrační obrázek snímání fotopasti

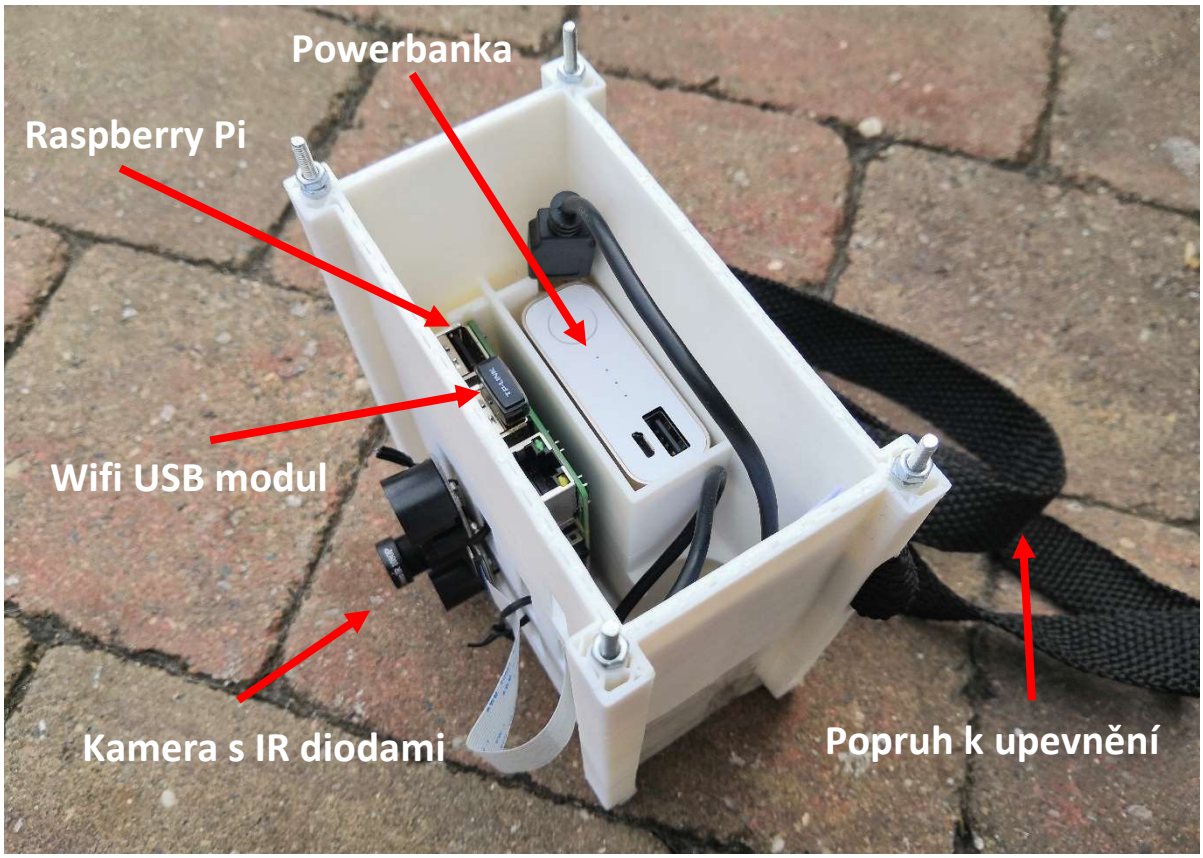
[23] https://www.raspberrypi.org/products/raspberry-pi-1-model-b-plus/
[24] Hypertext Transfer Protocol
[25] Raspberry Pi 3 Model B+
[26] 60 MB/s

Úhel záběru kamery je 70°. Při tomto uspořádání se výška řádku identifikačního kódu pohybuje kolem 30px, je tudíž dostačující pro následné zpracování. Kamera snímá v infračerveném spektru, osvětluje záběr dvěma infračervenými diodami a je otočena vertikálně, aby snímala vůz po celé výšce. Ilustrace snímání je na Obrázek 23.



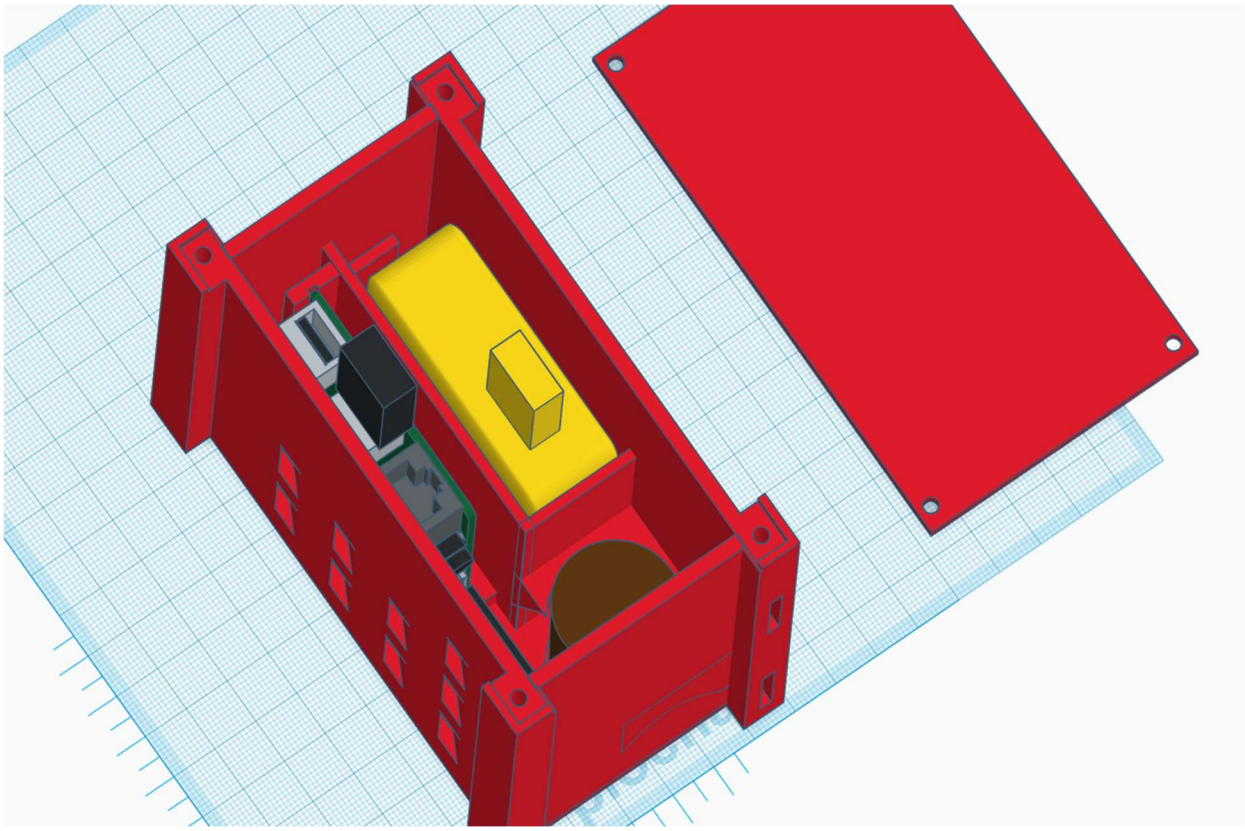
Obrázek 24 Blokové schéma fotopasti

Pro možnost jednodušší správy fotopasti je Raspberry Pi rozšířeno o USB wi-fi modul. Raspberri Pi je napájeno z 10050 mAh baterie, která zajišťuje chod přes více než 10 h. Vše je umístěno ve vlastně vymodelovaném boxu, vytištěném na 3D tiskárně.



Obrázek 25 Sestavená fotopast

Pro vymodelování boxu byl použitý software Tinkercad[27] od společnosti Autodesk. Tinkercad je bezplatná webová aplikace na vytváření 3D návrhů. Jedná se jednoduchou a intuitivní aplikaci. Všechny komponenty, které bylo potřeba umístit do boxu, byly změřeny a následně převedeny do 3D světa. Box byl vytvořen pro využití Microsoft LiveCam kamery, která následně použita nebyla. Vytvořený otvor pro tuto kameru byl zacelen a pro připojení stávající kamery bylo použito předem vymodelované větrání.



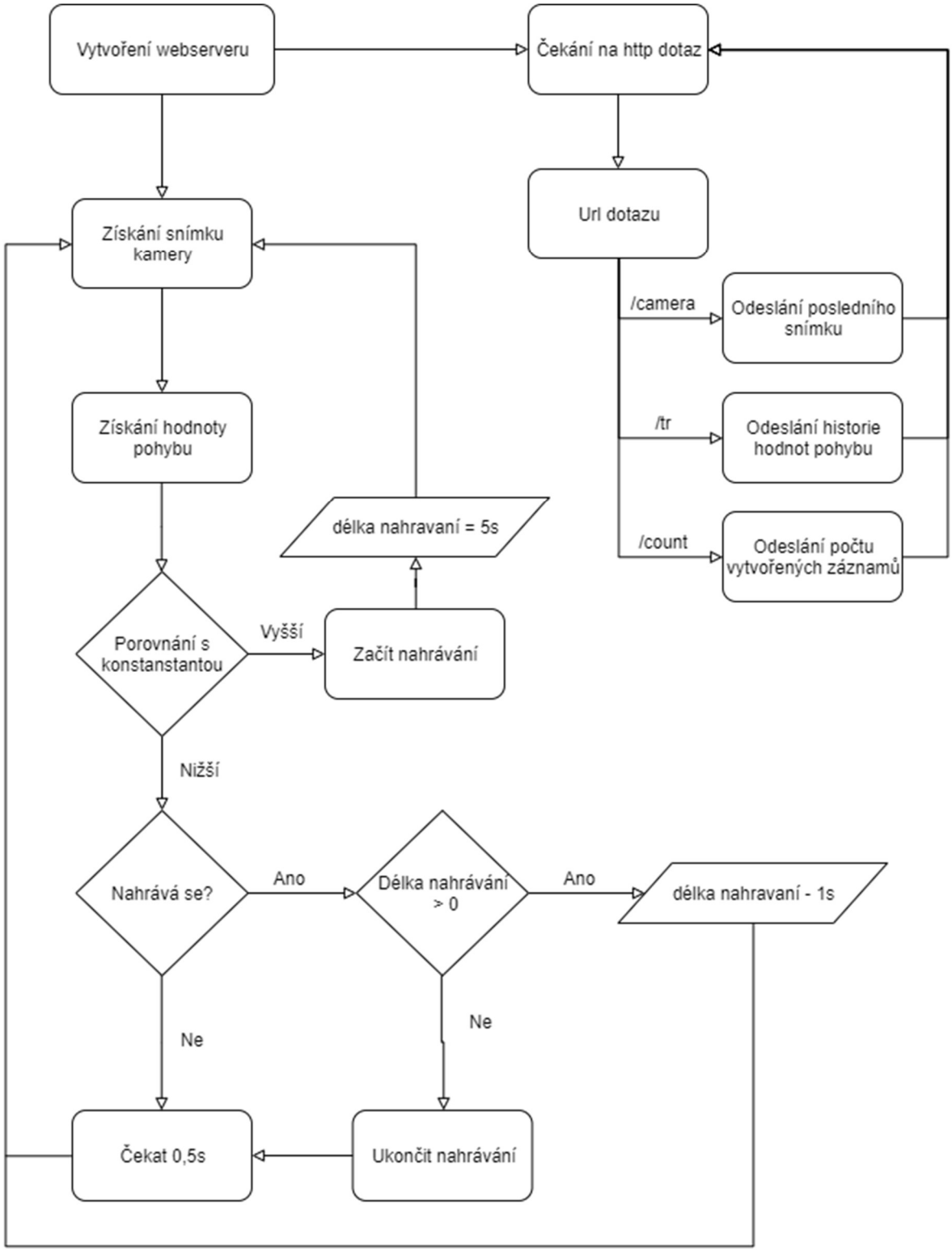
Obrázek 26 Ukázka 3D modelu fotopasti ze software Tinkercad

6.1.2 Softwarová část

Během startu Raspberry Pi se automaticky spouští aplikace napsaná v jazyce Python [21]. Tato aplikace zajišťuje získání videozáznamu pouze na pohyb před kamerou. Aplikace je rozdělena na tři části. První z částí je určena pro sledování pohybu, druhá rozhoduje o spuštění a ukončení nahrávání a poslední částí je webserver určený k administraci fotopasti. Celý proces je popsán na Obrázek 27.

[27] https://www.tinkercad.com/

Wifi modul je nastaven, aby se automaticky připojoval do určených sítí. Například do domácí sítě pro možnost vývoje anebo připojení na hotspot telefonu pro administraci v terénu.



Obrázek 27 Diagram programu ve fotopasti

Sledování pohybu

Fotopast využívá knihovnu OpenCV[28] pro zjištění změny stávajícího snímku oproti předchozímu. Porovnání snímku probíhá pomocí metody absdiff(InputArray src1, InputArray src2, OutputArray dst)[29], která získá absolutní hodnotu rozdílu jednotlivých bodů snímku. Metodou treshold(InputArray src, OutputArray dst, double thresh, double maxval, int type)[30] nastavenou na typ THRESH_BINARY a s nastavením thresholdu na 25 je získaný binární snímek veškerých rozdílných bodů. [22] Jakmile počet rozdílných pixelů překročí určitou hodnotu, fotopast spustí nahrávání. Pokud se během 5 vteřin opět nepřekročí daná hodnota, nahrávání se vypne. Díky tomu se docílí určité rezervy pro případ, že by došlo ke krátkému přerušení pohybu – výstupem by tak i v tomto případě byl pouze jeden videozáznam. Pokud je fotopast ve stavu, kdy nenahrává, tak testuje pohyb pouze jednou za 0.5 vteřiny, aby se snížila náročnost na procesor a zvýšila se výdrž baterie.

Nahrávání

Pro nahrávání je použita knihovna PiCamera[31], která je určena pro komunikaci s rozhraním, do něhož je zapojena kamera. Knihovna zajišťuje získávání cyklu aktuálních snímku kamery a také zajišťuje nahrávání.

Kamera je nastavena na maximální možné rozlišení (1088 x 1920px) a též na maximální snímkovací frekvenci (30 snímků za vteřinu). Kvůli rychlejšímu pohybu projíždějícího vlaku je expozice kamery nastavena do režimu sport a je snížena kompenzace expozice na hodnotu -15. Výsledkem je mnohem kratší čas expozice, ale také zhoršené snímání během nižších světelných podmínek.

Vytvořený videozáznam je ukládán na SD kartu, na které je nahrán filesystém Raspberry Pi s názvem, který obsahuje počáteční čas nahrávání. Výsledný soubor je ve formátu .h264.

[28] https://opencv.org/
[29] https://docs.opencv.org/2.4/modules/core/doc/operations_on_arrays.html#absdiff
[30] https://docs.opencv.org/master/d7/d1b/group__imgproc__misc.html#gae8a4a146d1ca78c626a53577199e9c57
[31] https://picamera.readthedocs.io/

Obrázek 28 Porovnání základního nastavení fotopasti (vlevo) a režimu sport (vpravo)

Webserver

Webserver je implementovaný pomocí knihovny socket[32], která vytvoří lokální webserver na portu 8000, jenž je spuštěný v samostatném vlákně, aby byl oddělen od nahrávání.

Vlákno s webserverem vyčkává na zaslané HTTP požadavky na definovaný port. Zpracovává požadavky na „/count“, kde vypisuje počet nahraných videozáznamů. Na url „/tr“ vypisuje hodnoty změn na snímcích za určitý časový interval. Slouží k ověření, že daný pohyb překročí určenou hodnotu pro zahájení nahrávání. Poslední url „/camera“ je určena pro aktuální pohled kamery.

6.2 Aplikace pro anotaci dat

Tato aplikace se dělí na 3 hlavní části. První část je převedení videa na statický záznam a předzpracování, druhá je určena na samotnou lokalizaci a klasifikaci, poslední část se stará o převedení dat pro Tensorflow.

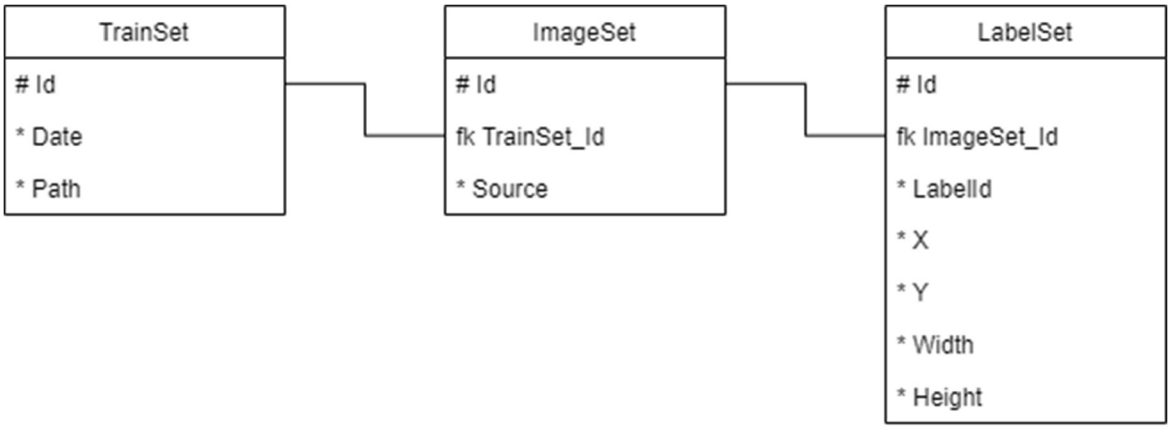
[32] https://docs.python.org/3/library/socket.html

Zpracování videa a preprocessing

Vstupem do této komponenty je video pořízené z fotopasti. Video je následně převedeno do .png souborů o frekvenci 10 snímků za vteřinu. Snímek je převeden do šedo tónu a zesvětlen.

Anotace

Tato část zajišťuje vložení snímků do systému, ukládání stavu a označení hledaných objektů. Seznam snímků se získá z předešlé komponenty a vloží se do databáze pro daný vlak. Data o vlaku, snímcích a labelech jsou držena v databázi.

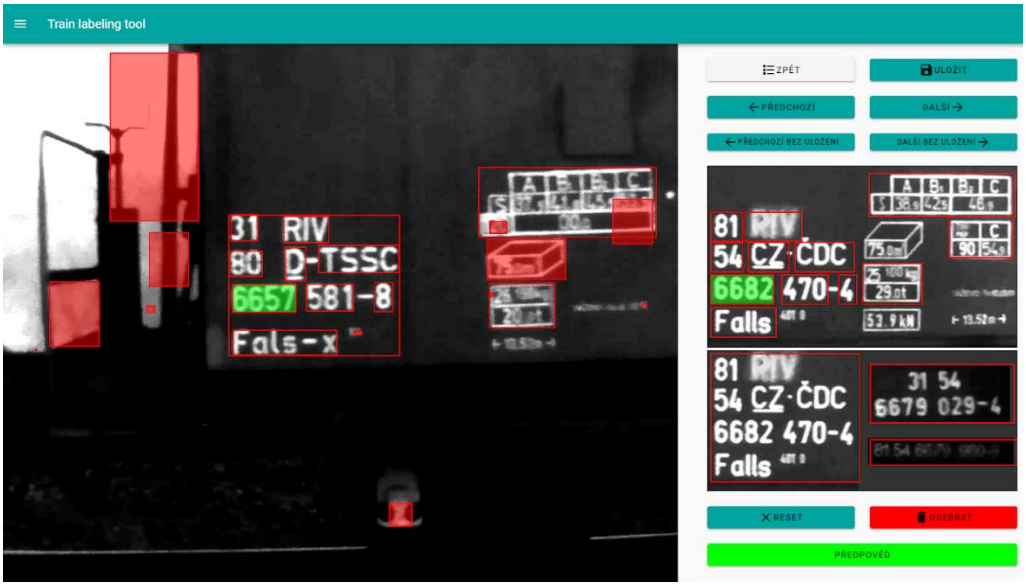


Obrázek 29 Struktura databáze anotační aplikace

Je to webová aplikace napsána v jazyku C# .NET[33] s využitím Entity frameworku[34] a code first stylu programování. Jedná se o napsání datových modelů a následně je z těchto modelů vygenerována struktura databáze. Backendová část obstarává datové úložiště a generování snímků za pomoci předchozí komponenty. Samotná práce nad snímky je prováděna v Javascriptu. Frontendová část aplikace využívá framework Vue.js[35], který zajišťuje chod aplikace. Pro práci nad snímky je použitý framework Fabric.js[36], který umožňuje zakreslovat obdélníky nad snímek a získat jejich polohu. Pro usnadnění označení hledaného textu je použitý Tesseract.js[37]. Tesseract.js je javascriptová implementace neuronové sítě Tesseract pro vyhledávání textu.

[33] https://www.microsoft.com/net/
[34] https://msdn.microsoft.com/en-us/library/aa937723(v=vs.113).aspx
[35] https://vuejs.org/
[36] http://fabricjs.com/
[37] https://tesseract.projectnaptha.com/

Výhodou vlastního vývoje je možnost co nejvíce zjednodušit a zrychlit zpracování snímku. Na Obrázek 30 je ukázka obrazovky, na které obsluha anotuje snímek vozu. Jelikož se anotuje 15 různých objektů, tak by mohlo být zmatečné nebo složité anotovat podle názvů. Pro zjednodušení je na pravé straně obrazovky ukázaný vzor, na kterém se vybírá, jaký objekt je zrovna anotován. Další zajímavostí je zelené tlačítko "předpověď" v pravém dolním rohu, které spustí Tesseract a označí na snímku všechny texty nalezené touto knihovnou. Všechny vybrané objekty jsou odeslány na server a uloženy do databáze. Další obrazovky aplikace pro anotaci jsou přiloženy v Příloha B.



Obrázek 30 Snímek z aplikace pro labeling. Na pravé straně je vidět vzor pro výběr typu a na levé straně právě zpracovávaný snímek. Červené objekty jsou nalezeny pomoci Tesseractu, ale nejsou anotovány, tudíž se nezapočítají.

Převedení zpracovaných dat na formát pro učení sítí Tensorflow

Práce s frameworkem Tensorflow je mnohem jednoduší v jazyku Python a z toho důvodu byl využit pro exportování dat. Byl vytvořen program v Jupyteru[38], který dotazuje na HTTP API[39] anotační aplikace pro určité snímky vozů a jejich značky. Tyto snímky se značkami jsou získány ve formátu JSON a pomocí skriptu jsou převedeny na typ „tf.train.Example" využívaný Tensorflow. Snímky jsou náhodně rozděleny poměrem 80 % na trénovací množinu a 20 % na testovací množinu.

[38] https://jupyter.org/
[39] Application Programming Interface

```
{
    'Id': integer,
    'Labels': [
        'Id': integer,
        'LabelId': integer,
        'X': integer,
        'Y': integer,
        'Width': integer,
        'Height': integer,
    ],
    'Source': string,
}
```

Obrázek 31 JSON z anotační aplikace

```
{
    'image/height': integer,
    'image/width': integer,
    'image/filename': string,
    'image/source_id': string,
    'image/encoded': bytes,
    'image/format': string,
    'image/object/bbox/xmin': float[],
    'image/object/bbox/xmax': float[],
    'image/object/bbox/ymin': float[],
    'image/object/bbox/ymax': float[],
    'image/object/class/text': string[],
    'image/object/class/label': int[],
}
```

Obrázek 32 Struktura pro Tensorflow tf.train.Example

Z výsledných množin jsou za pomoci frameworku vytvořeny dva .record soubory, které obsahují data potřebná pro naučení vlastní neuronové sítě.

6.3 Naučení neuronové sítě pro detekci kódů

Pro potřeby dosažení cílů aplikace bylo potřeba vytvořit neuronovou síť, vyhledávající na vozech identifikační kódy a další vedlejší informace z obrazových podkladů. Pro zpracování sítě byla využita platforma Tensorflow. Proces učení, popis modelu a výsledky naučené sítě jsou popsány v kapitole 7.

6.4 Aplikace pro identifikaci vlakových souprav

Tato aplikace je klíčová ke splněné cíle práce. Jedná se o spojení fotopasti, která dodává do této aplikace snímky z kolejiště a implementace neuronové sítě pro detekci identifikačních

kódů. Identifikační aplikace zajišťuje koncový bod pro fotopast, lokalizaci a přečtení kódu ze snímku a následné uložení do databáze. Aplikace je mířena pro používání drážní obsluhou, která aktuálně zpracovává identifikační kódy manuálně.

6.4.1 Serverová část

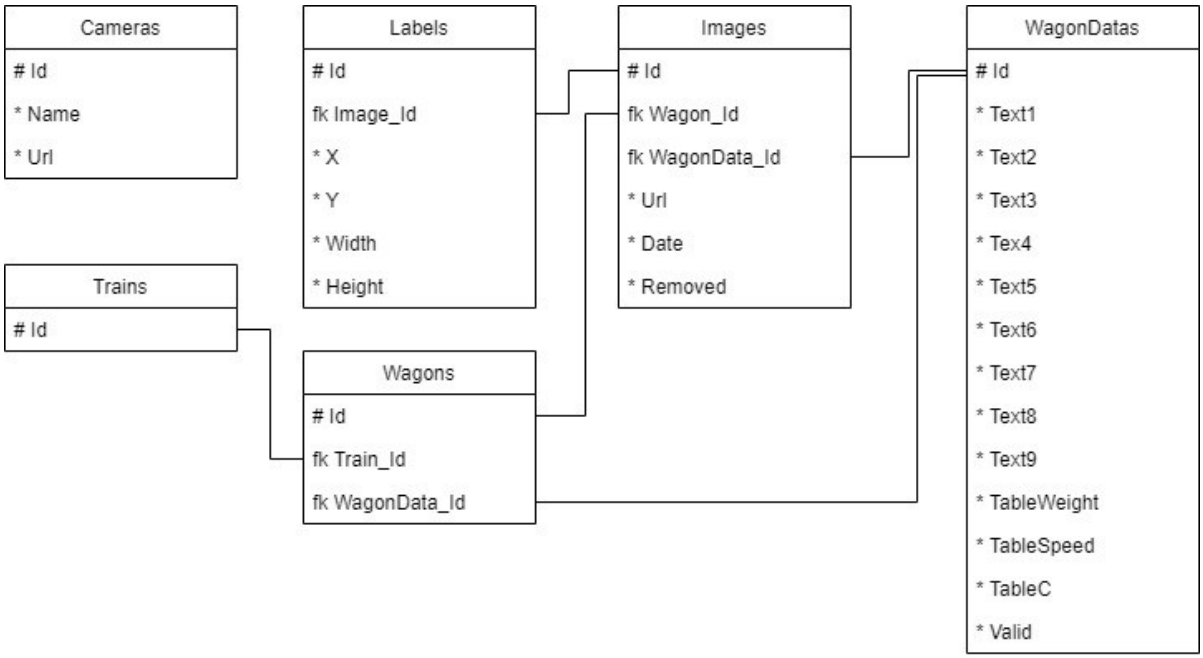
Serverová část aplikace zajišťuje veškerou logiku pro předávání dat do uživatelské části, ukládání dat a zpracování snímku vozu, včetně vygenerování vozů a vlakových souprav, na kterou je celá aplikace zaměřena. Server pro komunikaci s okolím a klientskou částí využívá HTTP API požadavky, díky kterým je zajištěna nezávislost.

Logika aplikace je rozdělena na samostatně fungující části nazývané service. API je definováno pomocí kontrolerů, které používají přidělené service – ty následně zajišťují celkovou logiku aplikace a práci s daty.

Na serverové části je využita samostatně naučená bloková neuronová síť pro vyhledávání boxů (labelů) na snímcích, které by mohli obsahovat potřebné informace k identifikaci vozu. Pro získání dat ze snímku je dotazováno do Azure pro OCR čtení a získání potřebného textu.

Aplikace je napojena na databázi MSSQL[40] pomocí Entity frameworku. V databázi jsou uloženy všechny snímky s labely nalezenými neuronovou sítí a s přečtenými daty z Azure. Následně jsou zde uloženy všechny nalezené vozy a vygenerované vlakové soupravy. Dalšími uloženými daty jsou fotopasti.

[40] Microsoft SQL Server

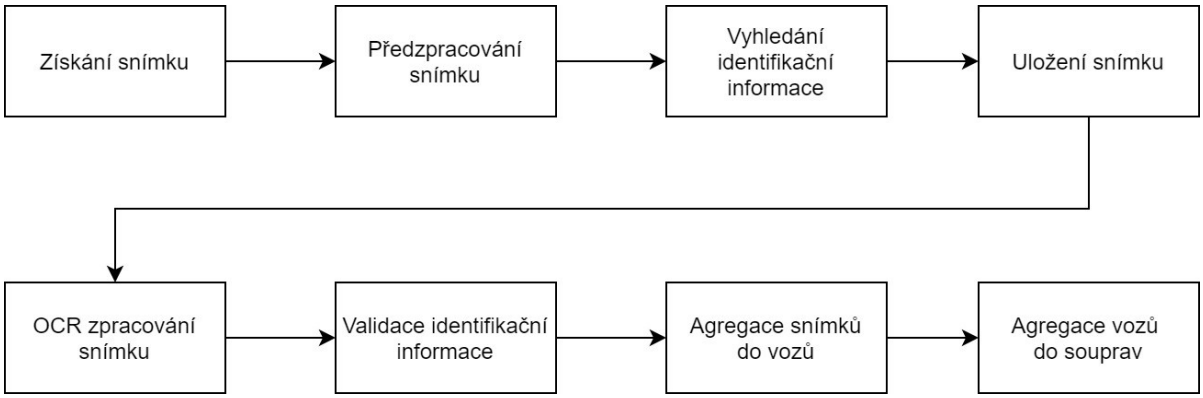


Obrázek 33 Struktura databáze identifikační aplikace

Aplikace zpracovává velké množství snímků z fotopastí. Na vstupu pro odeslání snímku je vytvořena fronta. Velikost je upravena dle výkonu serveru, aby nedocházelo k naprostému zahlcení prostředků. Snímky jsou zpracovávány asynchronně z důvodu čekání odpovědi z Azure, aby bylo zajištěno plynulé zpracování.

V předchozí kapitole je popsáno, že logika aplikace je rozdělena do samostatných service (komponent). Server obsahuje méně zajímavé komponenty, které zajišťují CRUDL dat anebo ukládání snímků na disk.

Největšími a nejzajímavějšími komponenty jsou ty, které jsou využívány ke zpravování snímků až po vytvoření vlakové soupravy. Celý tento proces je popsán na Obrázek 34 Proces zpracování snímku na serveru.



Obrázek 34 Proces zpracování snímku na serveru

Získání snímku

Získání snímku je vstupní bod celého procesu. Zde je předaný snímek z fronty, převedený z base64 kódování na datový typ Image.

Předpracování snímku

Na snímku je provedeno předzpracování, kdy je snímek otočen do správné orientace, převeden do šedo tónu a zesvětlen pro další zpracování.

Vyhledání identifikační informace

Snímek je zpracovaný vlastní naučenou neuronovou sítí na vyhledávání blokových dat. Pokud na snímku nenalezne identifikační kód vozu, je snímek zahozen a dál nezpracováván.

Uložení snímku

Snímek zpracovaný neuronovou sítí je uložen na disk. Následně je vloženo do databáze jeho místo uložení, nalezené boxy a čas záznamu.

OCR zpracování snímku

V této komponentě je zajištěno propojení s Azure API. Snímek je odeslaný jedním požadavkem na zpracování, kdy v odpovědi požadavku se nachází ID procesu. Dalším požadavkem se získaným ID, je dotazování na stav procesu. Dokud není proces zpracován, je dotazování opakováno. Jakmile je proces dokončen, součástí odpovědi jsou nalezené texty s pozicemi.

Tento proces je proveden na celém snímku z důvodu zmenšení počtu zpoplatněných požadavků na Azure OCR. Z množiny nalezených textů jsou vybrány pouze ty, které spadají do lokace nalezeného identifikačního kódu boxu.

Validace identifikační informace

Všechny texty, které jsou lokalizovány v oblasti identifikačního kódu, jsou seřazeny do jednoho. Tento text je rozsekaný na podtexty pomocí vytvořeného regulárního výrazu.

```
(?<text1>\d{2}).(?<text2>\b[\w|]*\b).*(?<text3>\d{2}).*(?<text4>\b\w+\b).*(?<text5>\b\w+\)
.*(?<text6>\d{4}).*(?<text7>\d{3}).*(?<text8>\d{1}).(?<text9>\b[\w|-]+\b)
```

regulární výraz na rozsekání identifikačního kódu vozu

Pokud se zpracování regulárním výrazem zdaří, je zpracovaný identifikační kód uložen do databáze ke snímku. Identifikační kód je validován pomocí validačního algoritmu z kapitoly Značení vozů. Validita kódu je uložena do databáze.

Agregace snímků do vozů

Po každém vyprázdnění fronty se snímky je zavolána metoda na vytvoření vozů ze snímku. Frekvence zaznamenávání je vyšší a identifikační kód jednoho vozu se vyskytuje 2–4x, záleží na rychlosti projíždějícího vozu.

Od snímku, který byl naposledy takto zpracován, je zahájeno spojování snímků se stejným identifikačním kódem. Z nichž je následně do databáze vytvořen záznam se seznamem všech nalezených snímků vozu.

Agregace vozů do souprav

Po zpracování agregace snímků do vozů, je potřeba vozy spojit do projíždějící soupravy. Od poslední nalezené soupravy jsou všechny vozy spojeny. Nová vlaková souprava je vytvořena, pokud je mezi vozy rozestup alespoň jedné minuty. Nalezené vlakové soupravy jsou následně uloženy do databáze.

Ruční přidání a editace vozu

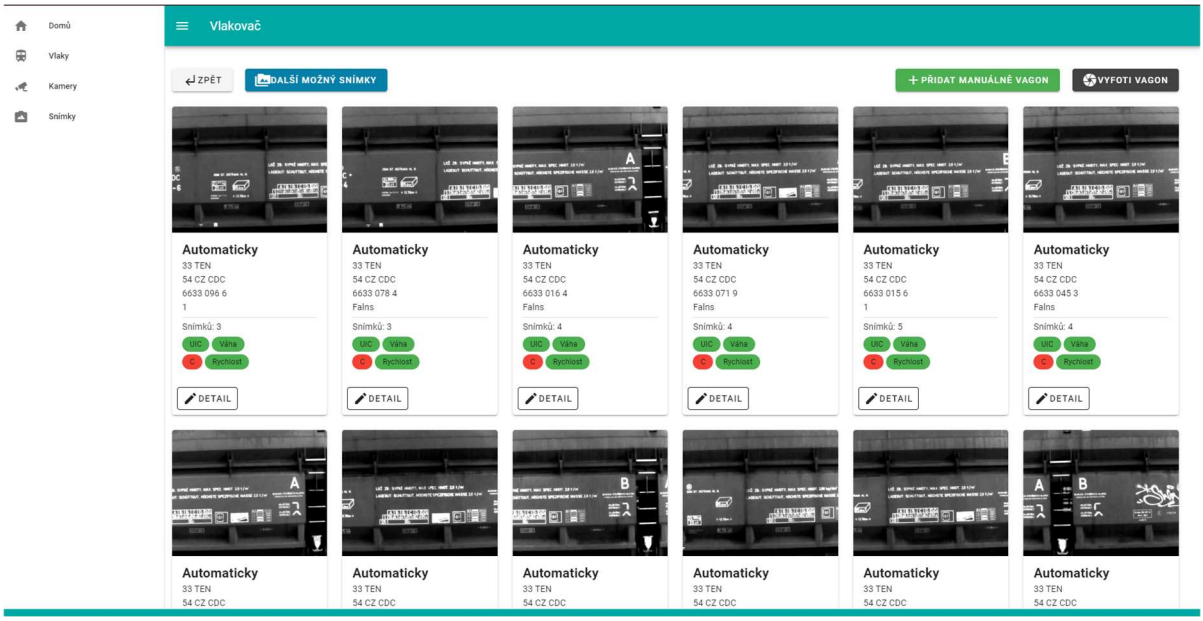
Všechny dříve vypsané komponenty jsou plně automaticky zpracovány. Některé snímky neprošly validací nebo se nezdařilo přečtení pomocí OCR. Obsluha systému má možnost nechat si vypsat seznam těchto nezpracovaných snímků a následně dopsat nebo upravit identifikační kód ručně.

Takto dopsaný identifikační kód je též zapsán do databáze. Prvním krokem je zkontrolovat, zda v blízkém časovém horizontu není vůz, který je už zpracovaný, se stejným kódem. V takovém případě by se snímek rovnou přiřadil. Pokud daný vůz neexistuje, je vytvořen nový a k němu jsou přiřazeny všechny snímky v rozmezí dvou vteřin. Následně je tento vůz porovnán s existujícími soupravami, přičemž se hledá shoda v časovém rozmezí. V případě nalezení soupravy je vůz přiřazen.

6.4.2 Klientská část

Klientská část je určena ke grafickému zobrazení dat a interakci s uživatelem. Díky použití javascriptového frameworku je část logiky aplikace přesunuta do klientské části. Jednou z věcí, která je přesunuta na frontend, je zpracování url stránky. Zobrazení stránky se

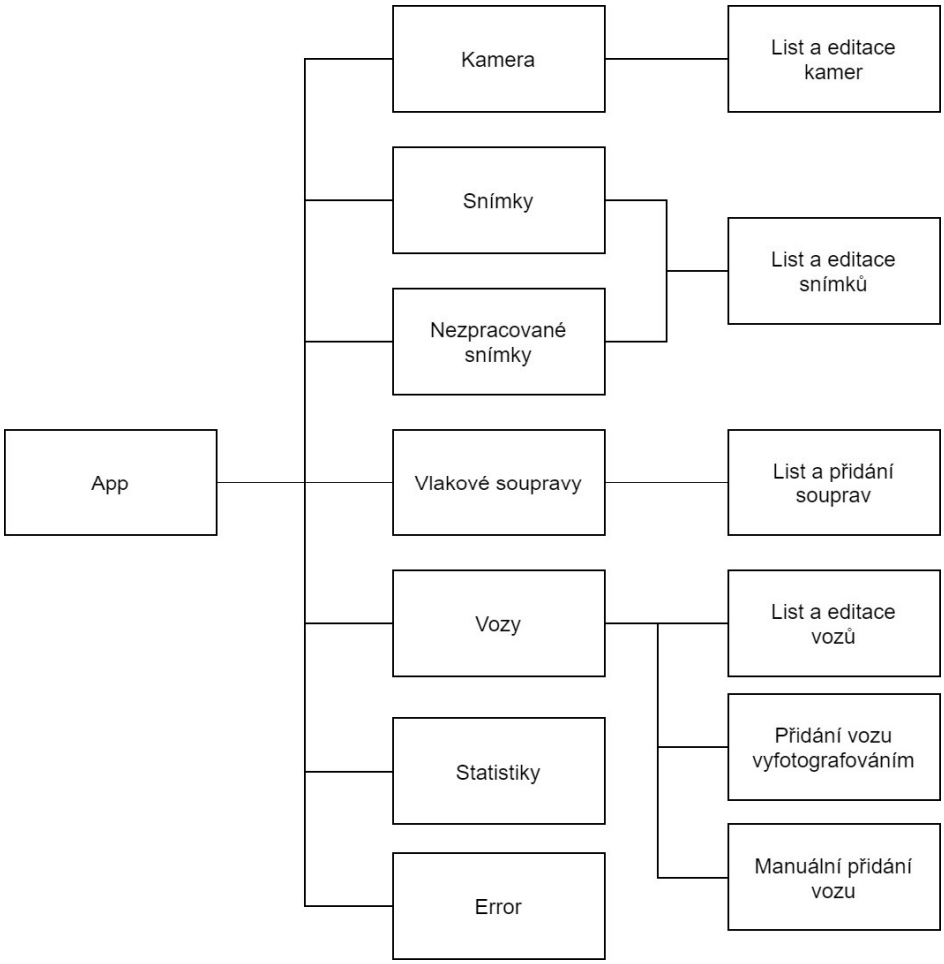
rozhoduje na straně javascriptu a tím, že se nemusí dotazovat na server, se urychlí změna stránky. Zobrazovaná data jsou oddělena do globálního store, který zajišťuje získávání dat z API serveru, filtraci dat a uchování stavu aplikace. Na Obrázek 35 je snímek aplikace, který ukazuje seznam všech zpracovaných vozů aplikací. Další snímky z aplikace pro identifikaci vlakových souprav jsou přiloženy v Příloha C.



Obrázek 35 Ukázka identifikační aplikace se seznamem vozů

Na straně uživatele je použit naučený model blokové neuronové sítě. Model byl převeden do formátu, který je čitelný knihovnou Tensorflow.js, a je využitý pro možnost vyfotografování vozu a jeho zapsání do systému.

Výsledná aplikace je vytvořena formou progresivní webové aplikace (PWA), která se chová jako běžná webová stránka, ale je možné jí uložit do telefonu, kde se chová jako nativní aplikace.



Obrázek 36 Komponenty klientské části

Kamera

Komponenta s kamerami je udělaná ve formě CRUDL pro možnost uložení kamer a zobrazení aktuálního pohledu kamery.

Nezpracované snímky

Na tomto pohledu je zobrazený seznam obsahující veškeré snímky všech vlakových souprav, které se nepovedlo automaticky zpracovat. Stránka je určena pro obsluhu k manuální identifikaci snímku.

Vlakové soupravy

Tato komponenta zobrazuje všechny vygenerované soupravy s informací o počtu vozů a o čase, ve kterém projížděla. Je zde i možnost přidat ručně další vlakovou soupravu. Též jsou zde vyfiltrovány všechny nezpracované snímky pro danou soupravu, jež je následně potřeba zpracovat manuálně.

Vozy

Zobrazení všech vozů dané vlakové soupravy. Jsou zde vidět snímky, jejich počet, čas a ikony nalezených dat na snímku. Na detailu vozu pak nalezneme všechny snímky týkající se konkrétního vozu s možností upravit přečtená identifikační data. Vůz je možno doplnit ručně nebo využít přidání vyfotografováním.

Přidání vozu vyfotografováním

Vyfotografování vozu je mířeno na mobilní zařízení. Obsluha touto komponentou získává možnost vůz vyfotit mobilem a v javascriptu je vzápětí vyhledán identifikační kód pomocí blokové neuronové sítě. Snímek je následně odeslán na server a zpracován pro předem daný vůz.

Statistiky

Statistiky jsou určeny k rychlejšímu zorientování obsluhy ve výpise časů projíždějících vozů a počtu snímků potřebných k manuální opravě.

7 Experimenty

Pro lokalizaci identifikačních kódů na boku vozu bylo potřeba vytvořit neuronovou síť, která bude vyhledávat identifikační kódy na vozech a další vedlejší informace z obrazových podkladů. Pro zpracování sítě byla využita platforma Tensorflow. Pro zrychlení učení už byl použitý přednaučený model SSD Inception v2, který je popsaný v kapitole 4.4.

Snímek je před vstupem do sítě předzpracovaný. Jako dostatečné předzpracování se ukázalo převedení do šedotónu a zesvětlení snímku. Byly ozkoušeny také detekce hran nebo využití prahování, ale nebylo to potřeba.

Z důvodu převedení problému na menší řešitelné části byly naučeny dva modely sítě. První model, pojmenovaný blokový, byl vytvořen pro vyhledávání bloků, jenž obsahují více informací. Druhý model, pojmenovaný elementární, byl vytvořen pro vyhledávání informací z identifikačního kódu vozu, skládajícího se z dílčích informací podstatných pro tento model.

7.1 Učení sítě

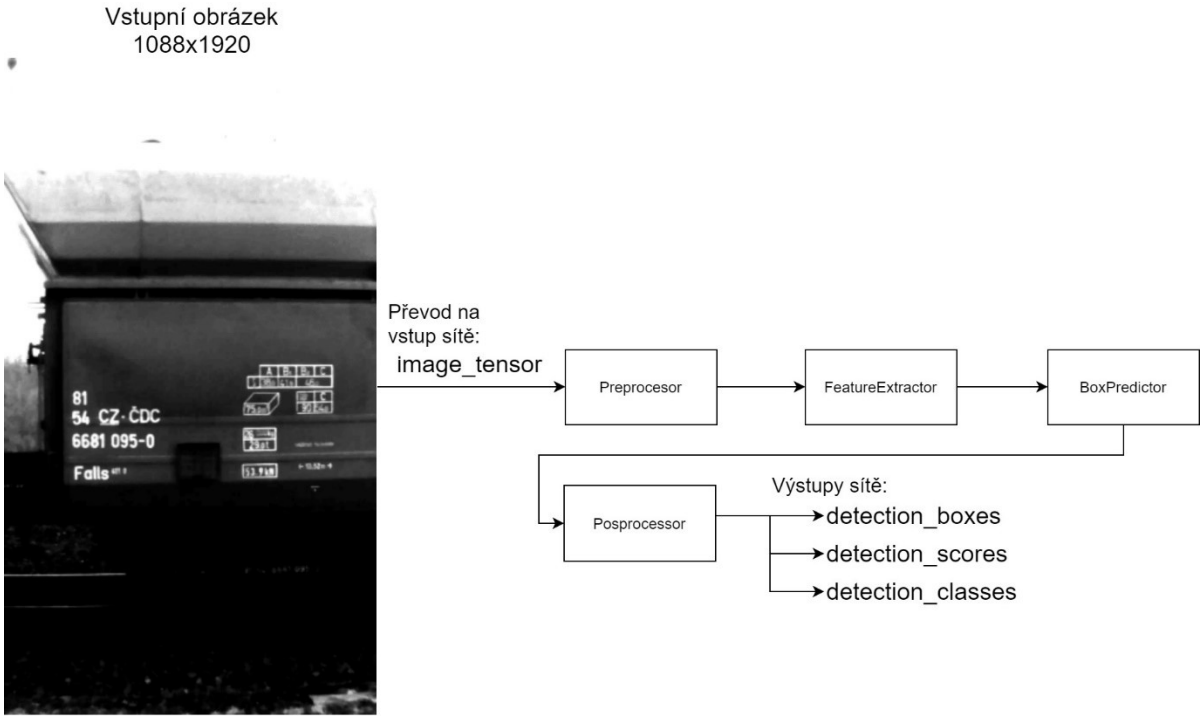
Vytvořená množina předzpracovaných a ohodnocených snímků byla rozdělena v poměru 80 % na testovací množinu a 20 % na trénovací množinu. Každá z těchto množin byla převedena na soubor typu „.record“, který obsahuje převedenou množinu dat na typ „tf.train.Example“, který je Tensorflow schopno načíst pro proces učení. Učení bylo prováděno lokálně na grafické kartě Nvidia GeForce GTX 860M s procesorem Intel Core i7-4720HQ a 8 GB RAM paměti.

Virtuální prostředí pro Python balíčky bylo vytvořeno pomocí programu Miniconda[41]. Seznam balíčků je přiložen u zdrojových kódů. V konfiguračním souboru Tensorflow byl nastaven počet typů boxů, cesta k jejich názvům a cesta k souborům s trénovací a testovací množinou. Po spuštění se z konfiguračního souboru vygeneroval soubor s modelem sítě pro učení. Během učení se vytvářejí soubory s aktuálním mezistavem naučené sítě. Tento mezistav může být využit pro obnovení učení při případném výpadku. Z těchto souborů je vygenerovaný model potřebné sítě, který je určen k použití v implementačních aplikacích.

[41] https://docs.conda.io/en/latest/miniconda.html

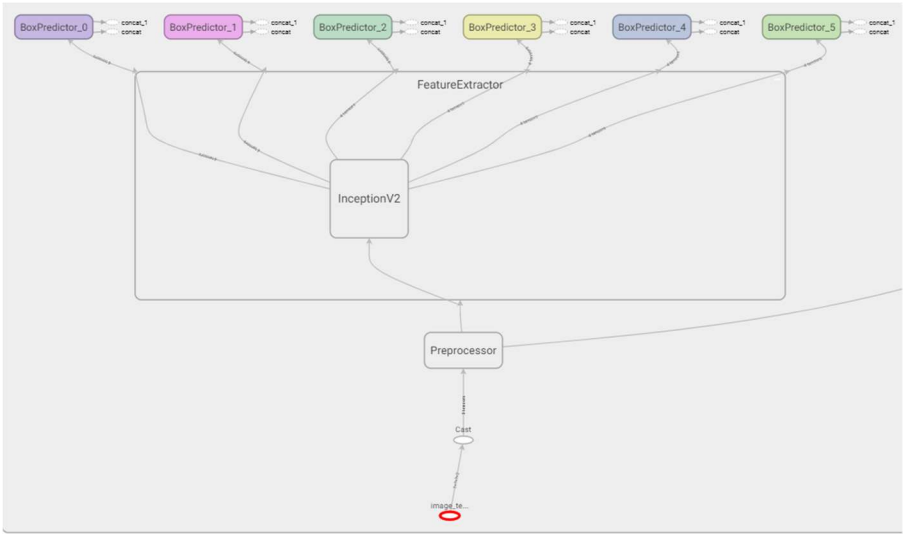
7.2 Model sítě

V této kapitole je popsaný model vytvořený ve frameworku Tensorflow, který je využitý pro vyhledávání boxů na snímku. Na Obrázek 37 je digram sítě s vyobrazenými nejhlavnějšími komponentami a dále v textu je popsán tok dat přes jednotlivé komponenty.



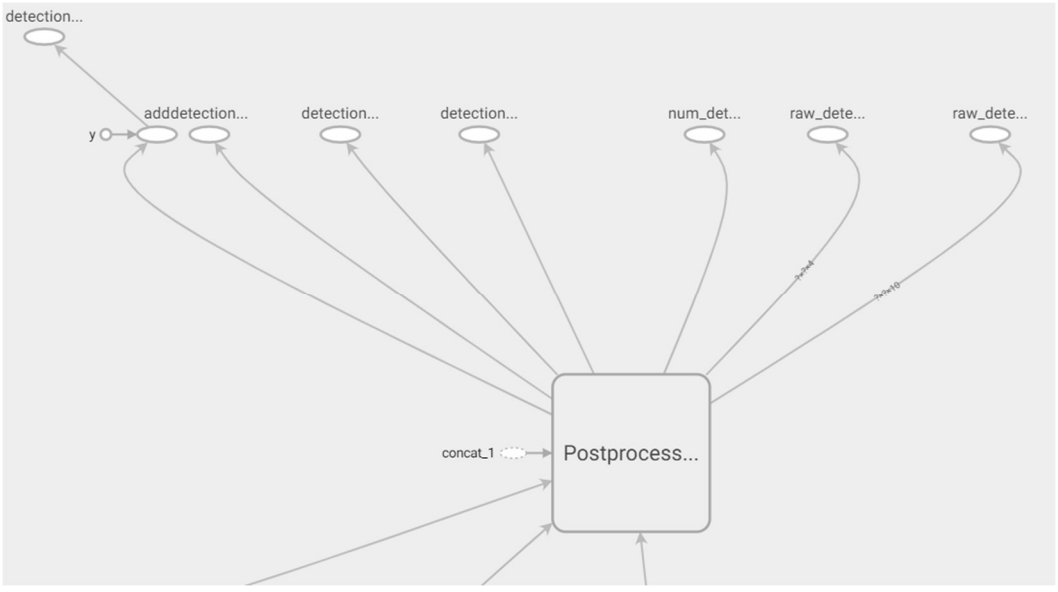
Obrázek 37 Diagram modelu

Vstupem do modelu je image_tensor. Jedná se o řadu bytů vytvořenou ze snímku o rozměrech 1088 x 1920px. Řada dat vstupuje do preprocesoru, zmenší snímek na potřebné rozměry, které jsou 300 x 300px o hloubce 3 byty. Po zredukování velikosti vrstvy následuje FeatureExtractor, který implementuje Inception v2 popsaný v kapitole 4.4. Také vyhledává potřebné informace ze snímku. FeatureExtractor používá jako aktivační funkci relu. Skupina BoxPredictorů získává výstup z FeatureExtraktoru a z těchto vstupů predikuje pozice boxů a typ pro každý box. BoxPredictor též používá jako aktivační funkci relu. Tato popsaná část modelu je zobrazena na Obrázek 38.



Obrázek 38 Ve spodní části je vstup do sítě a dále pokračuje přes Preprocessor, FeatureExtractor do BoxPrediction

Vstupem do postprocesoru jsou předpovědi BoxPredictoru převedeny na výstupy modelu. Výstupy modelu jsou typu DT_FLOAT. Výstupní funkce modelu je sigmoid. Jedná se o hodnotu v rozsahu 0 až 1. Detection_boxes je pole se souřadnicemi nalezeného boxu. Jelikož se jedná o hodnotu 0 až 1, tak pozice je popsána procentuálně vůči snímku. Detection_scores je pole pravděpodobnosti nalezeného boxu. Výstup detection_classes určuje typ daného boxu, převedený na jeho ID. Ukázka této části modelu je na Obrázek 39.



Obrázek 39 Data z BoxPrediction vstupují do Postprocessoru, který určuje výstupy sítě

7.3 Naučené sítě

Neuronová síť pro vyhledávání bloků

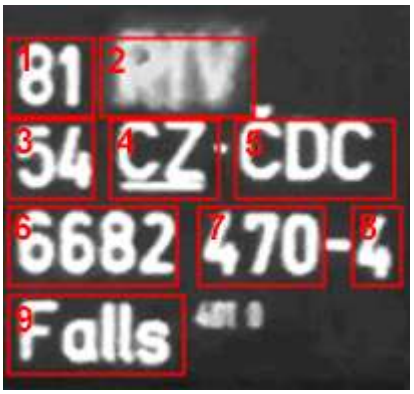
Tato síť vyhledává větší bloky informací. Nejdůležitějším blokem, který lokalizuje, je celá identifikační informace. Nejvíce viditelná je na levé části vozu a skládá se ze čtyř řádků. Další informace, jež vyhledává, jsou tabulky ložných hmotností a tabulka s hmotností vozu. Vstupem do sítě je předzpracovaný snímek vozu a výstupem sítě jsou pozice nalezených boxů, jejich typ a procentuální kvalita.

Obrázek 40 Ukázka boxů vyhledávaných blokovou sítí

Velikost trénovací množiny byla 1070 snímků, množiny testovací pak 261 snímků. Učení bylo ukončeno po 42 100 iteracích. Proces učení trval necelých 20 h. Vždy po dosažení určitého počtu kroků bylo provedeno ověření procesu učení, aby bylo jasné, v jakém stavu se síť nachází. Během procesu byla několikrát snížena hodnota rychlosti učení, aby se chyba sítě ustálila. Rychlost učení na počátku byla 0,004, po 18 000 krocích byla snížena na 0,0004 a po 27 000 krocích na 0,00004.

Neuronová síť pro vyhledávání dílčích elementů

Elementární síť byla vytvořena za účelem získání samostatné informace z identifikačního kódu vozu. Tento kód obsahuje 9 informačních bloků popsané v kapitole 3.1. Vstupem do sítě je předzpracovaný snímek vozu a výstupem jsou pozice, typ a kvalita boxu jako u předchozí neuronové sítě.

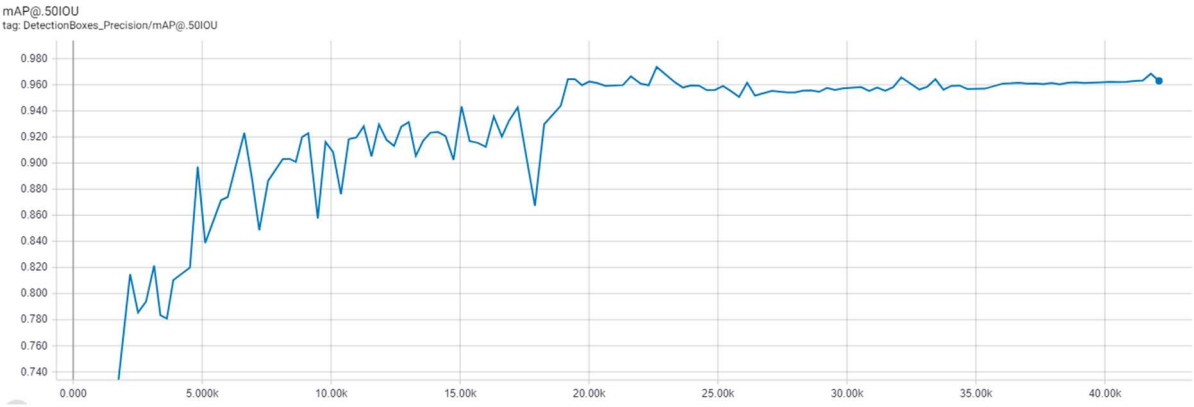


Obrázek 41 Ukázka boxů vyhledávaných elementární sítí

Trénovací množina byla o velikosti 649 snímků a testovací množina o velikosti 154 snímků. Učení bylo ukončeno po 66 540 iteracích. Proces učení trval kolem 30 h. Jako u předchozí sítě byl proces průběžně testován za cílem zjistit kvalitu naučení sítě. Rychlost učení byla zpočátku nastavena na hodnotu 0,004, po 46 000 krocích byla snížena na 0,0004 a po 64 000 krocích na 0,00004.

7.4 Výsledky sítí

Neuronová síť pro vyhledávání bloků

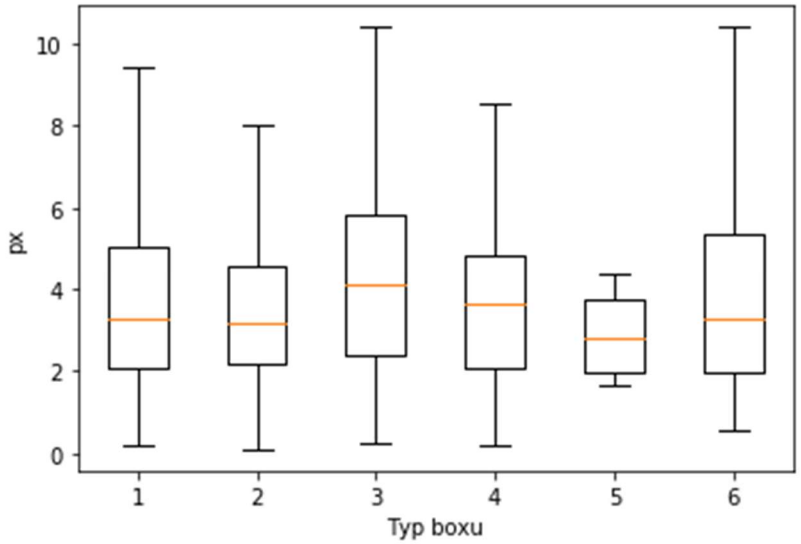


Obrázek 42 Vývoj hodnoty mAP@.50IOU během učení sítě pro vyhledávání bloků

Učení bylo ukončeno na základě zastavení stoupání hodnoty mAP@.50IOU, která se tímto ustálila na 0,9629. Výpočet této hodnoty je popsán v kapitole teoretické části 4.4. Kvalita sítě byla ověřena vlastním porovnáním výsledků sítě oproti očekávaným výstupům. Byla změřena průměrná odchylka vzdálenosti středu boxů, která je v průměru 5,26px. Odchylky jednotlivý středů boxů jsou zobrazeny na Obrázek 43. Následně byly porovnávány boxy, jenž byly nalezeny sítí navíc a boxy, jenž nalezeny nebyly. 1,37 % boxů bylo nalezeno navíc a 0,16

% boxů nebylo nalezeno. Pro test bylo použito 400 snímků. Zvýšený rozptyl sítě je z důvodu, že se ve výsledku vyskytovali jednotky boxů, které měli odchylku 200px a více. V těchto případech se jednalo o naprosto chybný nález. Kvalita sítě je popsána v Tabulka 3.

Síť dosahuje dostačujících výsledků a je použita v identifikační aplikaci.

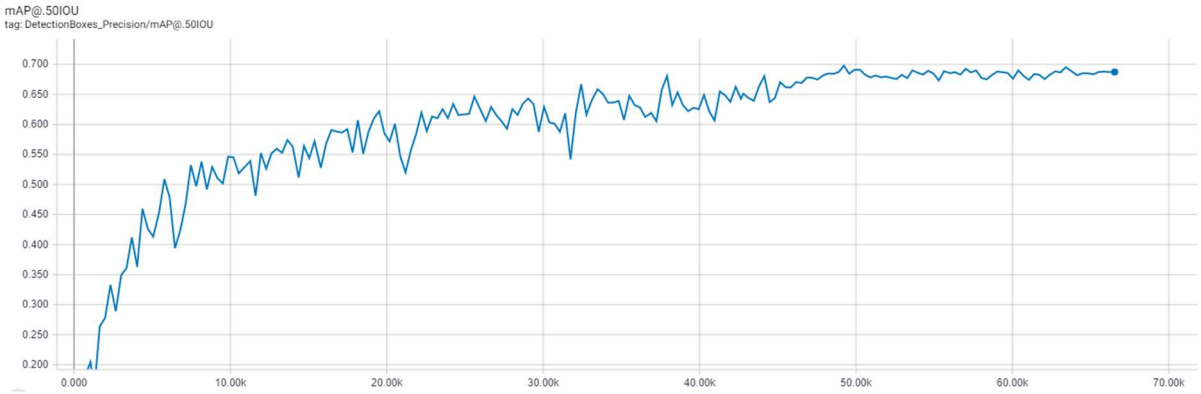


Obrázek 43 Odchylka středů jednotlivých boxů nalezených sítí pro vyhledávání bloků

Tabulka 3 Kvalita sítě pro vyhledávání bloků

Průměrná odchylka středů	5,26px (rozptyl 959,54) ze 400 vzorků
Nalezené boxy navíc	1,37 % ze 400 vzorků
Nenalezené boxy	0,16 % ze 400 vzorků
mAP@.50IOU	96,29 %
Přesnost klasifikace neuronovou sítí	**96 %**

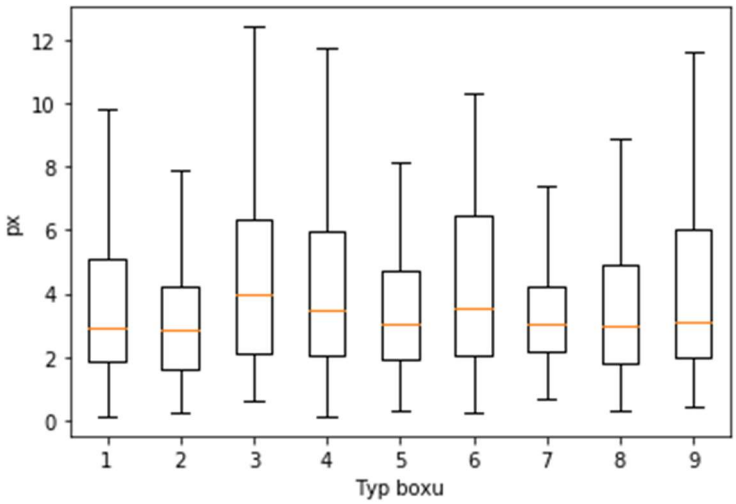
Neuronová síť pro vyhledávání dílčích elementů



Obrázek 44 Vývoj hodnoty mAP@.50IOU během učení sítě pro vyhledávání dílčích elementů

Učení sítě bylo ukončeno po ustálení hodnoty mAP@.50IOU na hodnotě 0,6873. Výpočet této hodnoty je popsán v kapitole teoretické části 4.4. Kvalita byla následně znovu ověřena vlastními postupy na základě porovnávání výstupů a očekávaných výstupů. Průměrná odchylka středů boxu se pohybuje kolem 5,11px. Odchylky jednotlivý středů boxů jsou zobrazeny na Obrázek 45 Odchylka středů jednotlivých boxů . Nalezených boxů navíc bylo 1,72 % a nenalezených boxů 9,88 %. Pro test bylo použito 400 snímků. Kvalita sítě je popsána v Tabulka 4.

Tato síť byla vytvořena za účelem experimentu, zdali je možné tohoto výstupu docílit. Výsledek sítě nebyl dostatečně kvalitní a nebylo tudíž potřeba použití této sítě v identifikační aplikaci.



Obrázek 45 Odchylka středů jednotlivých boxů vyhledávaných sítí pro vyhledávání dílčích elementů

Tabulka 4 Kvalita sítě pro vyhledávání dílčích elementů

Průměrná odchylka středů	5,11px (rozptyl 54,78) ze 400 vzorků
Nalezené boxy navíc	1,72 % ze 400 vzorků
Nenalezené boxy	9,88 % ze 400 vzorků
mAP@.50IOU	68,73 %
Přesnost klasifikace neuronovou sítí	**69 %**

8 Testování

Během vývoje aplikace byla provedena řada testů, které se ukázaly jako slepá cesta, ale byly nalezeny též postupy, které byly následně použity ve vytvořených aplikacích. V následujících kapitolách je popsán proces testování aplikací během vývoje.

8.1 Skládání snímku

Nejprve byly provedeny experimenty pro zjištění rychlosti vozu ze snímku. Jedním z testů bylo vyhledávat podobnosti mezi dvěma snímky a následně zjistit posun podobností na snímku, ale úspěšnost nebyla moc vysoká a jednalo se o výpočetně náročný proces.

Dalším testem pro zjištění rychlosti bylo snímání podvozku na začátku a konci snímku. Jelikož podvozek vozu je tmavý oproti světlému pozadí lze snáze rozpoznat tuto změnu na snímku. Byl měřen čas mezi změnou světelnosti v boxu na počátku a na konci snímku, ale ani tento způsob nebyl příliš spolehlivý. Docházelo k neúplnému vykreslení identifikačního kódu nebo jeho duplikování (viz. Obrázek 46 Ukázka neúspěšného skládání snímku). Tento postup by mohl být úspěšný v případě, že by byla k dispozici kamera s vyšší frekvencí snímání a případně také senzory na kolejišti pro zjištění přesné rychlosti vlakové soupravy. Skládání snímků do dlouhého pruhu se v tomto experimentu ukázalo jako zbytečné.

Obrázek 46 Ukázka neúspěšného skládání snímku

8.2 Testování fotopasti

První vyvíjenou částí byla fotopast. Z počátku bylo otestováno, zdali bude rozlišení dostačující a po úspěšném ověření kvality byl software fotopasti testován na projíždějících automobilech, které byly lépe přístupné než vlakové kolejiště. Poslední fází testování fotopasti bylo ověření funkčnosti v reálném provozu. Po prvním dni provozu byly opraveny drobné nedostatky a fotopast započala sběr dat.

8.3 Testování neuronové sítě

Učení neuronové sítě bylo průběžně testováno a kontrolováno pomocí TensorBoard[42], který je určený pro správu učení modelů pomocí Tensorflow. Během učení bylo pravidelně prováděno opakované vyhodnocení sítě, jež bylo pokaždé porovnáváno s předchozím výsledkem, pro zjištění kvality sítě. Naučená síť byla dodatečně otestována vlastním programem, kde byla měřena vzdálenost středů nalezených boxů od boxů predikovaných, nebo počet správného nalezení boxu. Výsledky testování jsou k nalezení v kapitole 7.4.

8.4 Testování aplikace pro identifikaci

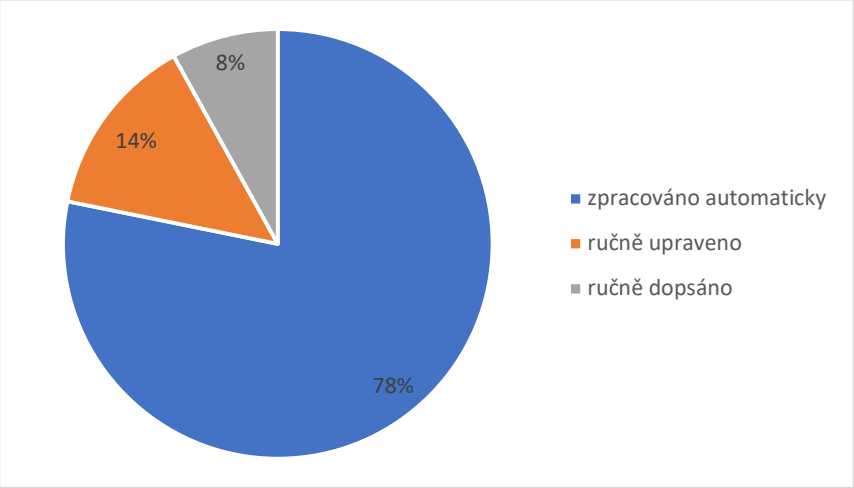
Aplikace pro identifikaci byla testována během průběhu celého vývoje. Vždy byla otestována daná komponenta, jíž se týkal aktuální vývoj. Testová no bylo vždy odesíláním testovacích požadavků na API kontroler pomocí aplikace Postman[43], kde byly následně kontrolovány výstupy z identifikační aplikace.

8.5 Testovací provoz identifikační aplikace

Po spojení aplikace s vizuální částí probíhalo testování několika manuálními průchody, aby byla každá komponenta otestována co možná nejefektivněji a nejdůkladněji. Na aplikaci byl následně poslán simulovaný provoz skriptem v Pythonu, který simuloval chování fotopasti, před kterou projíždí vlak. Bylo odesláno 1000 požadavků na aplikaci se snímkem, z nichž na 381 byla nalezena relativní informace pro zpracování. Pro detekci textu bylo odesláno na

[42] https://www.tensorflow.org/tensorboard
[43] Aplikace pro pohodlné otestování API požadavků na server - https://www.getpostman.com/

Azure 796 požadavků. Plně automatická úspěšnost zpracování byla na 78 % a 14 % potřebovalo pouze lehkou úpravu v rozpoznaném textu. 8 % snímků rozpoznáno nebylo.



Obrázek 47 Úspěšnost testovacího provozu

9 Závěr

Cíl práce byl splněn. Byl vyvinut funkční systém na identifikaci železničních vozům, který je připraven na použití v reálném provozu a zautomatizování aktuálního manuálního procesu. Práce se zabývá problematikou sběru dat, pro kterou byla navržena a vytvořena fotopast jak po stránce softwarové tak hardwarové, jíž je možné umístit ke kolejišti. Získaná data, určená pro naučení neuronové sítě, byla anotována pomocí vlastní navržené a implementované anotační aplikace specializované pro anotaci snímků identifikačních kódů na boku vozů. Na vytvořených testovacích a trénovacích množinách byly naučeny dvě neuronové sítě. První síť je určena pro vyhledávání bloků a druhá pro vyhledávání dílčích elementů. Pro získání zpracovatelného textu ze snímku je použita externí cloudová služba Azure. Je navržena a implementována aplikace pro identifikaci železničních vozů. Tato aplikace automatizuje daný proces s využitím vlastní naučené neuronové sítě pro vyhledávání bloků a využívá API Azure OCR pro získání textu.

V průběhu vývoje práce byly provedeny testy pro zjištění optimálního předzpracování snímku pro neuronovou síť. Jako plně dostačující předzpracování se ukázalo pouhé zesvětlení snímku. Také byly otestovány OCR technologie pro nalezení nejvhodnější služby. S nejlepšími výsledky se osvědčila služba Azure. Dalším testování se týkalo zjištění rychlosti vozu ze snímku, ale tento test skončil neúspěchem, což nijak neovlivnilo kvalitu aplikace, jelikož tento proces se později ukázal jako nepotřebný. Vývoj aplikací byl též doprovázen průběžnými testy. Neuronové sítě byly testovány vlastními metrikami. Síť pro vyhledávání bloků dosahuje úspěšnosti 96 %, která je dostačující pro použití v aplikaci. Druhá neuronová síť sloužící pro vyhledání dílčích elementů dosahuje úspěšnosti pouze 69 %, není tudíž dostatečně kvalitní a nebyla v aplikaci použita. Tato síť se později též ukázala jako nepotřebná v identifikační aplikaci. Identifikační aplikace byla otestována zkušebním provozem. Plně automaticky bylo zpracováno 78 % vozů, 14 % vozů muselo být lehce ručně upraveno a pouze 8 % vozů nebylo rozpoznáno a bylo potřeba je doplnit manuálně.

Fotopastí bylo získáno 66 videozáznamů vozů se stopáží 1 hodiny a 13 minut. Program pro fotopast se pohybuje okolo 140 řádků kódu. Anotační aplikace se serverovou a klientskou částí má okolo 1 500 řádků. Aplikace pro identifikaci železničních vozů se serverovou a klientskou částí má přibližně 4 500 řádků kódu. Autor práce uplatnil celou škálu technologií. Programovací jazyk Python byl použit pro fotopast a převodové scripty množin pro neuronové sítě. Serverové části aplikací byly psány v programovacím jazyku C# .NET

s Entity frameworkem. Programovací jazyk Javascript byl využitý na klientských částech s použitím frameworku Vue.js. Práce se snímky byla ulehčena pomocí knihovny OpenCV. Pro implementaci neuronových sítí bylo využito Tensorflow. Bylo také využito externí služby Azure.

Systém byl vyvinut s nízkými náklady za hardware a je tudíž spíše ověřením konceptu této problematiky, na kterém se bude dále rozvíjet a zvyšovat úspěšnost. Nejslabším bodem systému je fotopast, kterou by bylo potřeba osadit kvalitnějším snímačem. Další možností zlepšení by bylo přesunout implementaci vlastní neuronové sítě na vyhledávání identifikační informace přímo do fotopasti, tím snížit počet požadavků a odlehčit zátěž na serveru. I přes tyto nedostatky se stále v tuto chvíli jedná o funkční a přínosný systém.

Zdrojové kódy jsou nahrány na přiloženém DVD s popisujícími README.md soubory daných složek. Systém byl předán Severočeským dolům a vzájemná spolupráce byla hodnocena kladně. Celé vyjádření je k nahlédnutí v Příloha A Hodnocení Severočeskými doly.

10 Literatura

[1] **Označení železničních vozů.** ***Vagóny.*** **[Online] 2020. https://www.vagony.cz/vagony/oznaceni.html.**

[2] Železniční vůz. ***Vagóny.*** **[Online] 2020. https://www.vagony.cz/vagony/vagony.html.**

[3] Camea unirail overview. ***Camea.*** **[Online] 2019. https://camea.cz/underwood/download/files/unirail_overview.pdf.**

[4] Arar, Steve. Digital Image Processing: Point Operations to Adjust Brightness and Contrast. ***All about circuits.*** **[Online] https://www.allaboutcircuits.com/technical-articles/digital-image-processing-point-operations/.**

[5] Histogram Equalization. ***OpenCv.*** **[Online] 2020. https://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_equalization/histogram_equalization.html.**

[6] Obrazové filtry. ***Mendelu.*** **[Online] 2020. https://is.mendelu.cz/eknihovna/opory/zobraz_cast.pl?cast=18431.**

[7] Canny Edge Detection. ***OpenCV-Python Tutorials.*** **[Online] 2020. https://opencv-python-tutroals.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_canny/py_canny.html.**

[8] Image Thresholding. ***OpenCV documentation.*** **[Online] 2020. https://docs.opencv.org/3.4/d7/d4d/tutorial_py_thresholding.html.**

[9] Yiu, Tony. Understanding Neural Networks. ***towards data science.*** **[Online] 2020. https://towardsdatascience.com/understanding-neural-networks-19020b758230.**

[10] Using neural nets to recognize handwritten digits. ***neuralnetworksanddeeplearning.*** **[Online] 2020. http://neuralnetworksanddeeplearning.com/chap1.html.**

[11] Saha, Sumit. A Comprehensive Guide to Convolutional Neural Networks — the ELI5 way. ***towards data science.*** **[Online] 2020. https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.**

[12] Max-pooling / Pooling. ***computersciencewiki.*** **[Online] 2020. https://computersciencewiki.org/index.php/Max-pooling_/_Pooling.**

[13] Why TensorFlow. ***Tensorflow.*** **[Online] 2020. https://www.tensorflow.org/about.**

[14] Forson, Eddie. Understanding SSD MultiBox — Real-Time Object Detection In Deep Learning. *Towards data science.* [Online] 2020. https://towardsdatascience.com/understanding-ssd-multibox-real-time-object-detection-in-deep-learning-495ef744fab.

[15] Asharul Islam Khana, Salim Al-Habsi. Machine Learning in Computer Vision. *ScienceDirect.* [Online] 2020. https://www.sciencedirect.com/science/article/pii/S1877050920308218.

[16] Raj, Bharath. A Simple Guide to the Versions of the Inception Network. *Toward data science.* [Online] 2020. https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202.

[17] Seznámení se s Azure. *Microsoft Azure.* [Online] 2020. https://azure.microsoft.com/cs-cz/overview/.

[18] Woodford, Chris. Optical character recognition (OCR). *Explainthatstuff.* [Online] 2020. https://www.explainthatstuff.com/how-ocr-works.html.

[19] Detect objects with superhuman ability. *Labelbox.* [Online] 2020. https://labelbox.com/product/object-detection.

[20] LabelImg. *Github.* [Online] 2020. https://github.com/tzutalin/labelImg.

[21] Lutz, Mark. *Learning Python.* místo neznámé : O'Reilly Media, Inc, USA, 2013. 1449355730.

[22] Bradski, Gary. *Learning OpenCv.* místo neznámé : O'Reilly Media, Inc, USA, 2008. 0596516134.

Příloha A Hodnocení Severočeskými doly

Projekt Identifikace železničních vozů na základě obrazové informace vychází z praktické potřeby nahrazení neproduktivní činnosti obsluhy při přejímce železničních nákladních vozů. Obsluha v současné době musí obejít jednotlivé vozy vlaku, zapsat jejich údaje do tzv. soupisky a následně je přepsat do PC. To je náročné jak z pohledu časového, tak z pohledu možného vzniku chyb při přepisování údajů. Vzhledem ke skutečnosti že IT Severočeských dolů často zkouší nové technologie jsme uvítali možnost zpracování diplomové práce, která by tento problém řešila s využitím AI. Spolu se zpracovatelem P. Šepsem jsme zvolili metodu Proof of concept, tedy vývoje praktického prototypu, který by ukázal vhodnost použití takovéto metody v praxi. Zpracovatelem jsme byli při konzultacích průběžně seznamováni s postupem prací od vývoje fotopasti a zachycení snímků jednotlivých vagónů, přes jejich zpracování až k závěrečnému vyhodnocení. Práce byla zakončena prezentací příslušným pracovníkům a ukázkou zpracování dat.

Výsledky z testovacího provozu ukazují, že tento způsob zpracování dat má velkou šanci být zaveden v praxi. Projekt považujeme za velmi zdařilý.

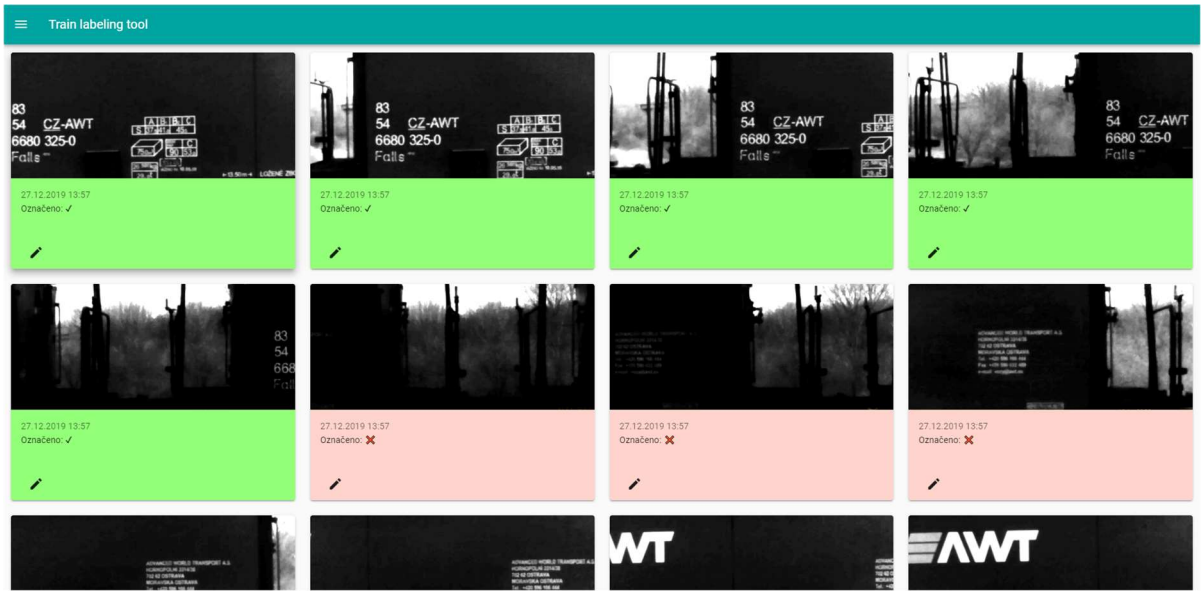
Jako další krok plánujeme návrh odpovídajícího (profesionálního) technického vybavení, zpracování investičního záměru a předložení projektu ke schválení investiční komisi Severočeských dolů.

Zpracovatel při vývoji prokázal vysokou míru samostatnosti a rychlou orientaci v problematice. Spolupráci s ním hodnotíme jako vynikající.

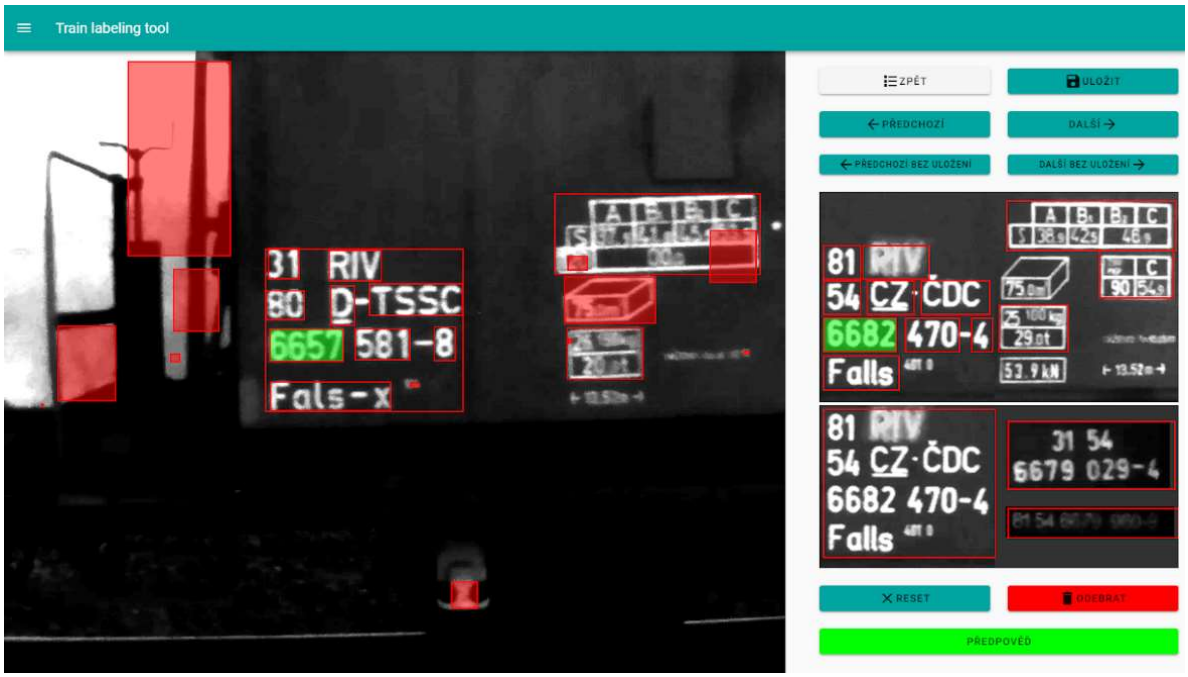
Ing. Miloslav Černý
vedoucí oddělení aplikační podpory

Severočeské doly a.s.
ul. 5. května 213
418 29 Bílina

Příloha B Vizuální část anotační aplikace

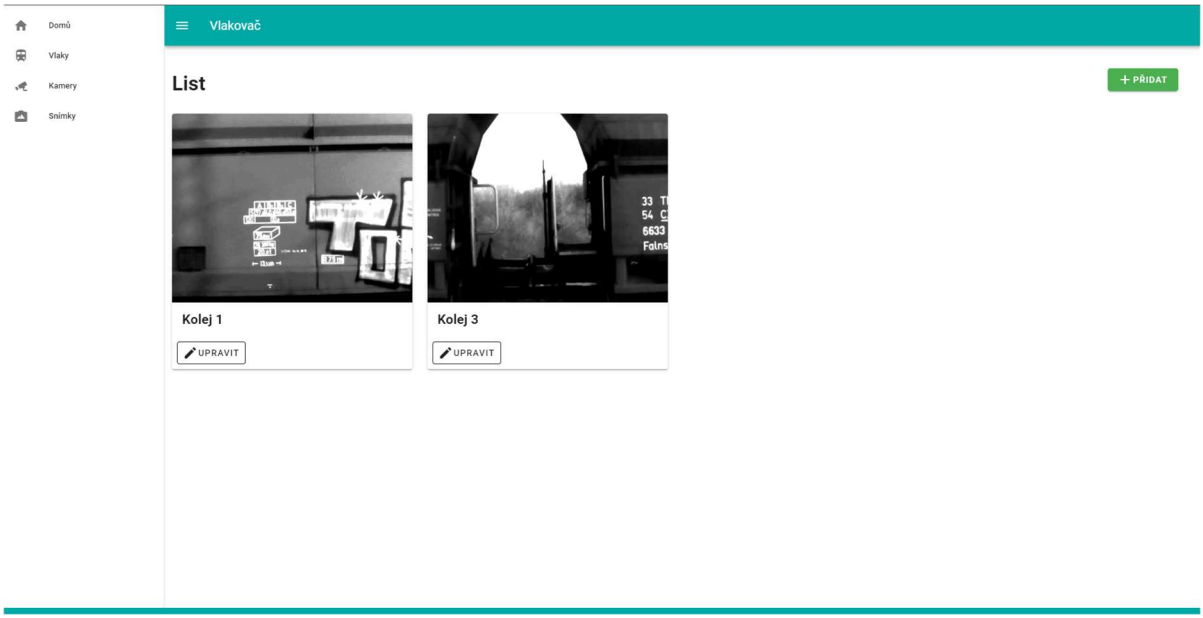


Obrázek 48 Seznam snímků v anotační aplikaci

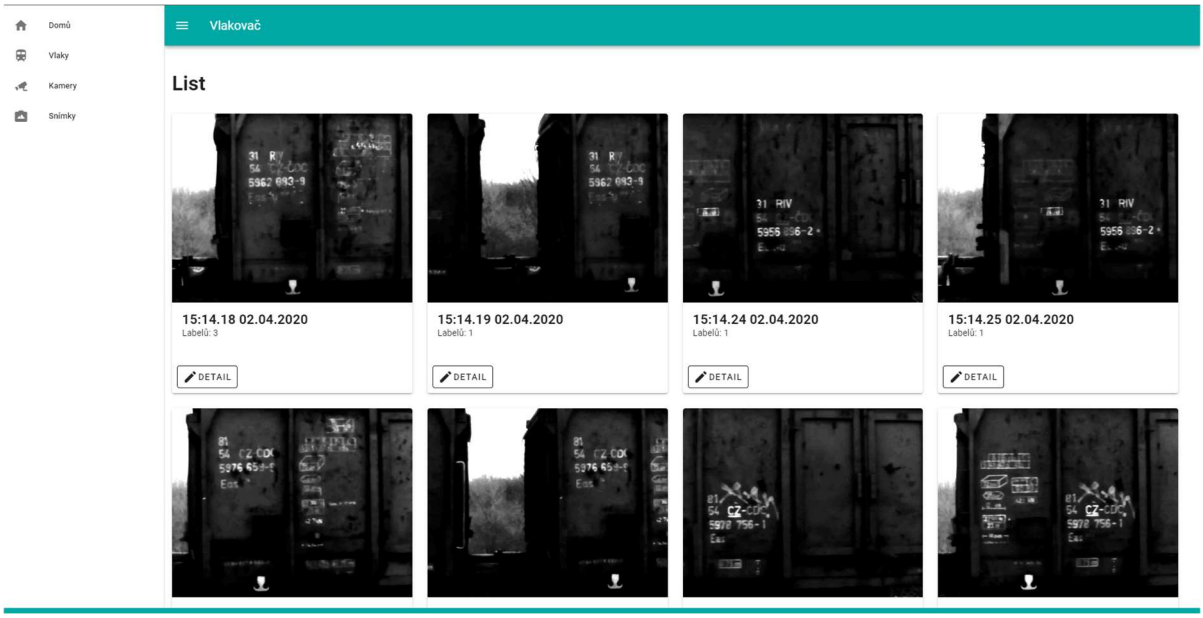


Obrázek 49 Detail snímku v anotační aplikaci (vlevo klasifikovaná snímek, vpravo legenda)

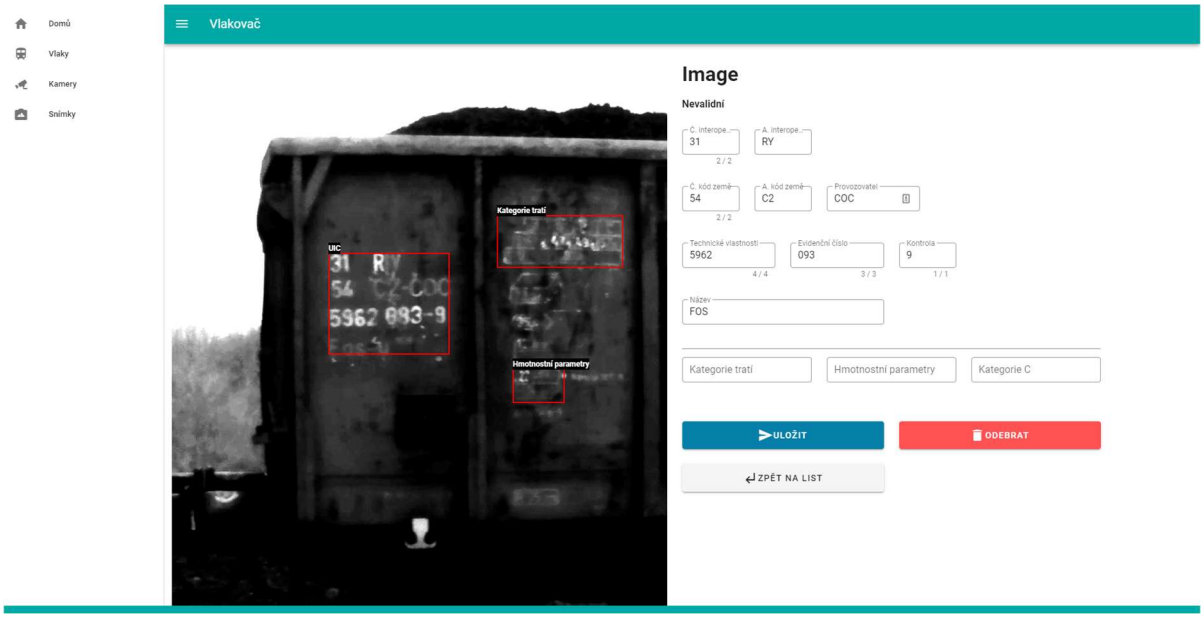
Příloha C Vizuální část identifikační aplikace



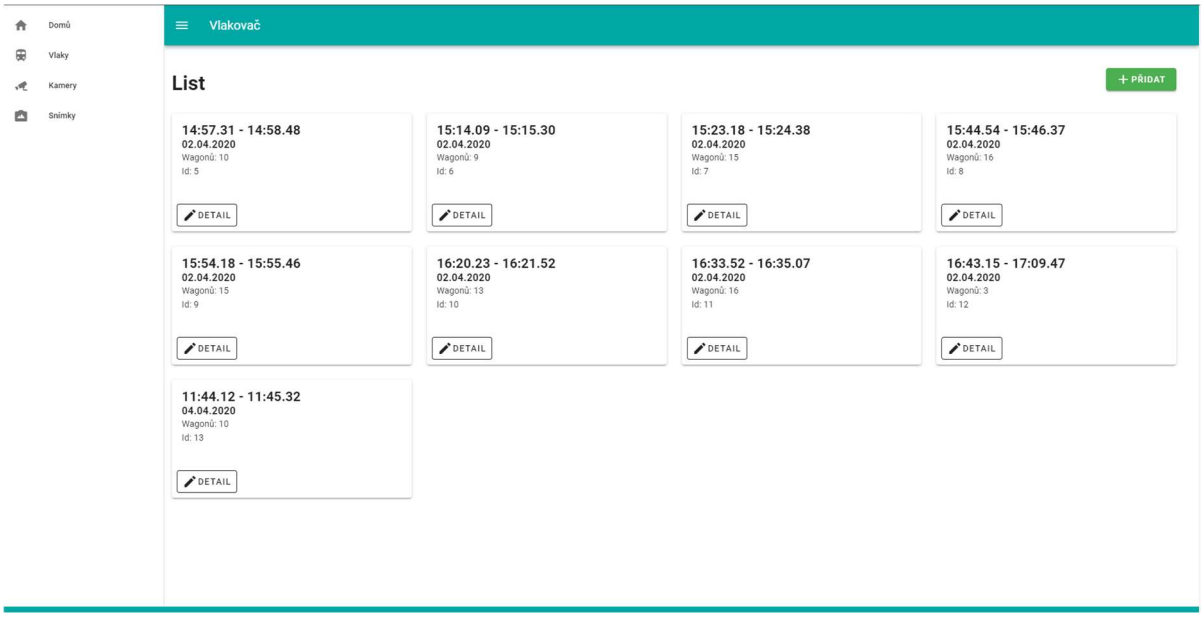
Obrázek 50 Seznam kamer identifikační aplikace



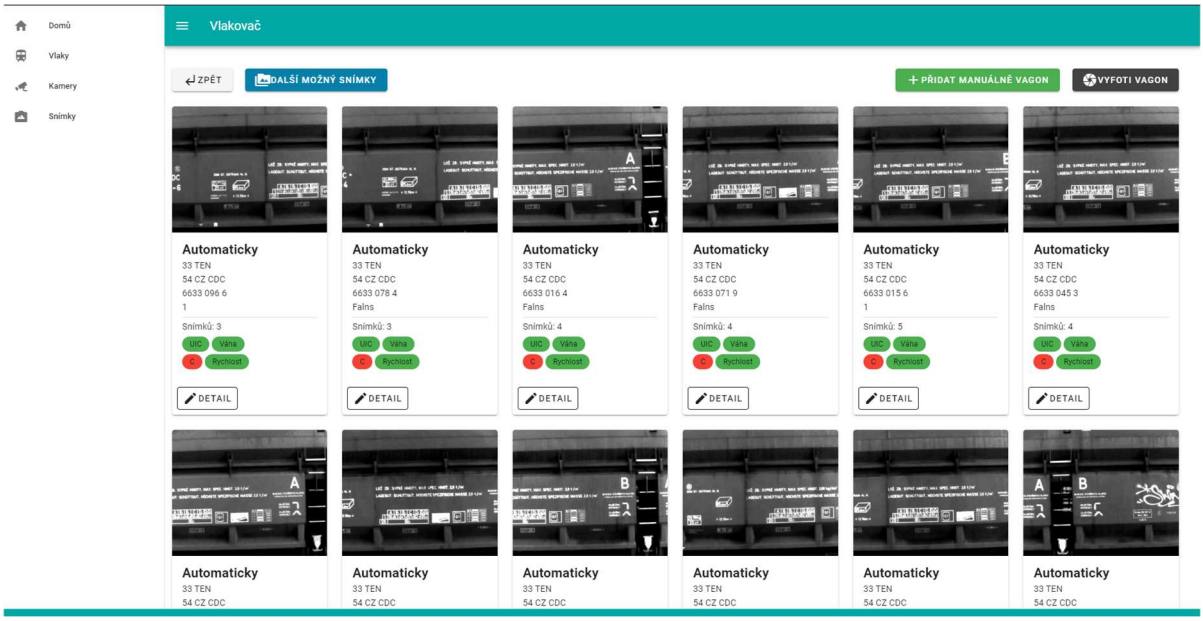
Obrázek 51 Seznam snímků potřebných k manuální identifikaci



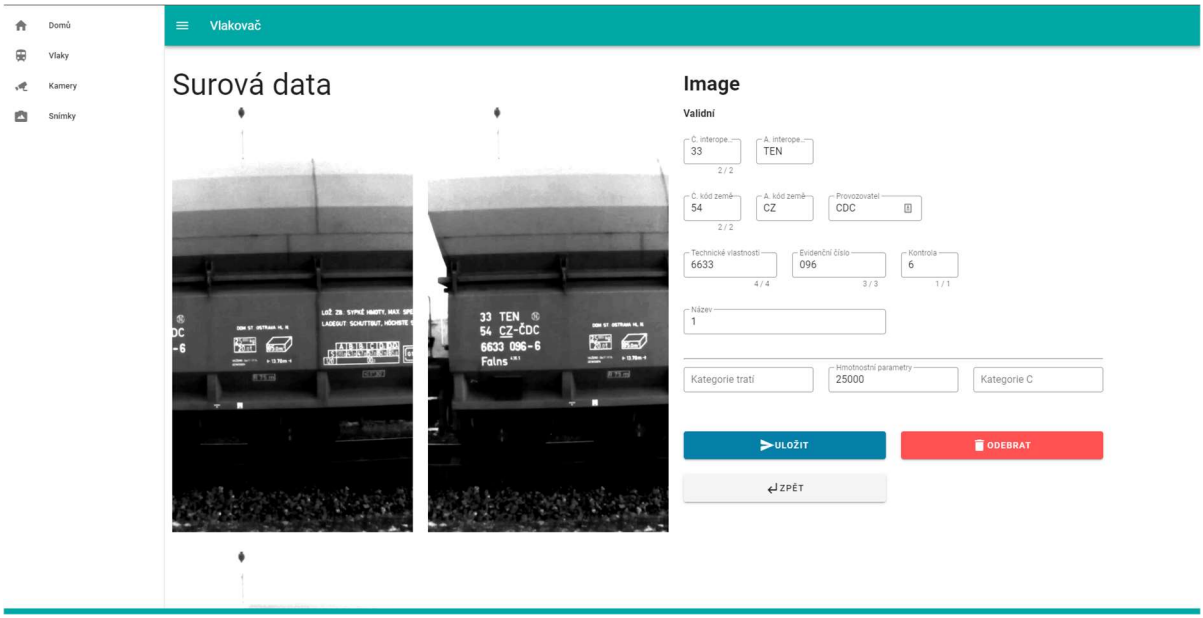
Obrázek 52 Detail snímku k manuální identifikaci



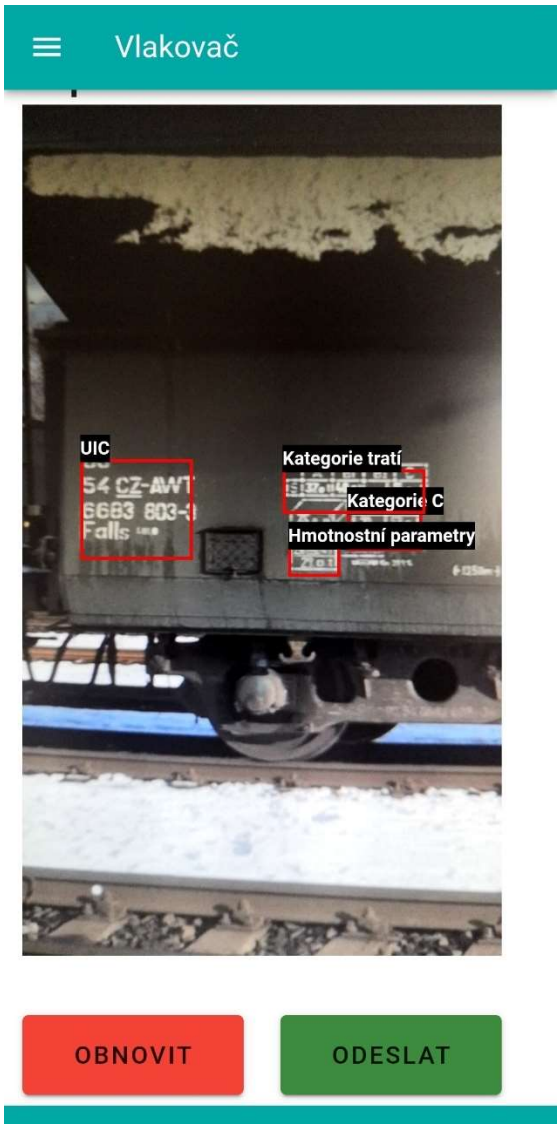
Obrázek 53 Seznam vlakových souprav



Obrázek 54 Seznam vozů vlakové soupravy



Obrázek 55 Detail vozu se snímky a identifikačníma informacemi



Obrázek 56 Ukázka identifikace vyfocením