VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE DOPRAVNÍCH ZNAČEK

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL JURČA
AUTHOR

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DETEKCE DOPRAVNÍCH ZNAČEK

ROAD SIGN DETECTION

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE MICHAL JURČA

AUTHOR

VEDOUCÍ PRÁCE Ing. VÍTĚZSLAV BERAN, Ph.D.

SUPERVISOR

BRNO 2015

Abstrakt

Tato bakalářská práce se zabývá detekcí dopravních značek v obraze se zaměřením na klasifikaci dopravních značek. Nejprve budou představeny stručně dostupné řešení, použité datové sady a metody zabývající se objektovými příznaky. V dalším kroku budou popsány postupy pro vytvoření modelu a následně jeho trénování pomocí klasifikátoru typu support vector machines. Nakonec budou popsány metody pro vyhodnocení modelu a zhodnoceny dosažené výsledky.

Abstract

This bachelor' thesis deals with the detection of traffic signs from image, focusing on the classification of traffic signs. First will be presented shortly available solution, used data sets and methods of dealing with object-signs. The next step will describe the procedures for creating a model and then his training classifier using Support Vector Machines. Finally they will be described a method for evaluation of the model and the evaluated results.

Klíčová slova

dopravní značky, klasifikace, support vector machines, kaskádový klasifikátor, trénování, LIBSVM

Keywords

traffic signs, classification, support vector machines, cascade classifier, training, LIBSVM

Citace

Michal Jurča: Detekce dopravních značek, bakalářská práce, Brno, FIT VUT v Brně, 2015

Detekce dopravních značek

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Vítězslava Berana, Ph.D.

. . . . . .
Michal Jurča
19. května 2015

Poděkování

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Vítězslavu Beranovi, Ph.D, za jeho vedení, trpělivost, konzultace a pomoc při tvorbě této práce. Dále bych chtěl poděkovat Janovi Duškovi za poskytnuté výstupy jeho bakalářské práce.

Obsah

Kapitola 1

Úvod

Už od 20.století s rozmachem automobilového průmyslu rostla potřeba detekce dopravních značek. Nyní s rozvojem počítačů jsou do automobilů implementovány vestavěné a počítačové systémy, které umožnují detekovat a klasifikovat dopravní značky. Systém může tak detokovat hrozící nebezpečí. S rozvojem autonomních vozidel bude potřeba detekce dopravních značek stále požadována.

Dnešní počítačové systémy v automobilu hlídají skoro všechno, jestli je řidič připoutaný, neusíná a dává pozornost. S rozvojem techniky dnešní systémy umožnují noční vidění, sledování chodců a dopravních značení. A posledním bodem se bude ve své práci zabývat. Detekcí dopravních značek.

V druhé kapitole této práce bude popsána historie dopravního značení od samotného začátku až po současnost. Následně budou představeny existující řešení a použité datové sady. V následující kapitole budou popsány algoritmy pro zpracování obrazu. Ve čtvrté kapitole se budu zabývat vytvořením modelu pro trénování a volbou vhodného klasifikátoru. Následně bude popsáno trénování klasifikátoru, volba vhodných pramatrů a implementace kaskáda klasifikátorů. Další kapitola bude věnovaná představením charakteristik pro vyhodnocení natrénovaného modelu. A následně pomocí charakteritik budou diskutovány dosažené výsledky. Závěř kapitily bude věnován experimentům. V poslední kapitole provedu zhodnocení řešení a představím návrhy na vylepšení.

Kapitola 2

Detekce dopravních značek

V této kapitole budou popsany dopravní značky, použité datové sady, existující řešení a na závěr popis procedury trénování a testování.

2.1 Dopravní značení

V dávných dobách funkci dopravního značení plnily pouze jen stopy, vyšlapané pěšiny a vyjeté koleje od vozů nebo koní. Již v Pompejích v době před 2 tisíci lety se objevily patníky, které oddělovaly peší zónu od ulice. Objevil se zde i přechod pro chodce v podobě vyvýšeného chodníku oproti silnici. Později Římané začali značit cesty pomocí svislých kamenů vyznačující vzdálenost od Říma. Ve středověku se pokračovalo ve značení směrů a vzdáleností. V roce 1868 byl použit první primitivní mechanický semafor. Potřeba dopravních značek vzrosla s rozmachem automobilního průmyslu ve 20.století. Od roku 1924 se koná každé čtyři roky silniční kongres, ale i přes veškeré snahy se dodnes nepovedlo všechny dopravní značky sjednotit.

V Československu v roce 1935 bylo zavedeno dopravní značení s pěti druhů výstražných značek. O pár let později přibyly další dopravní značky a byl zaveden pravostranný silniční provoz. Rychlost jízdy v obci nesměl být vyšší než rychlost koně v poklusu. Jelikož v té době, reflexní folie ještě nebyly, připevňovaly se ně odrazky pro dostatečnou viditelnost v noci. V součastnosti se prosazují elektronické a promněné dopravní značení, které se využívají při regulaci dopravy a zajištění plynulého a bezpečného provozu. V reálném čase poskytují řidiči informaci o výskytu dopravní nehody, zhoršeném počasí na konkrétních úsecích nebo provádění údržbových prací. Dále se prosazuje osvětlení značek s časovým ovládáním, které se využívá zejména u škol a používá se reflexní žlutozelené orámování.

Dopravní značky jsou jednoduché piktogramy určené pro řízení a regulaci silničního provozu na pozemních komunikacích. Jedná se o zařízení, které upozorňuje řidiče na nebezpečná místa, ukládá jim příkazy, zákazy nebo omezení, poskytuje jim zpřesnění nebo inforace, doplňuje nebo omezuje význam jiní doporavní značky. Rozlišují se dopravní značky svislé nebo vodorovné.

Podle zákona o provozu na pozemích komunikacích §63 odst. 1 svislé dopravní značky jsou:

- výstražné značky, které upozorňují na místa, kde účastníku provozu na pozemních komunikacích hrozí nebezpečí a kde musí dbát zvýšené opatrnosti
- značky upravující přednost, které stanoví přednost v jízdě v provozu na pozemních komunikacích

- zákazové značky, které ukládají účastníku provozu na pozemních komunikacích zákazy nebo omezení
- příkazové značky, které ukládají účastníku provozu na pozemních komunikacích příkazy
- informativní značky, které poskytují účastníku provozu nutné informace, slouží k jeho orientaci nebo mu ukládají povinnosti stanovené tímto zákonem nebo zvláštním právním předpisem
- dodatkové tabulky, které zpřesňují, doplňují nebo omezují význam dopravní značky, pod kterou jsou umístěny.

V úloze detekce dopravních značek, se používá kromě dělení podle typu taky dělení podle tvaru, který může být určen čtvercem, kruhem, trojúhelnékem, osmiúhelníkem a obdelníkem. Tvar a velikost jsou pevně definovány a nesmí se měnit.

Kompletní přehled všech dopravních značek a více inforací o dopravním značení lze nalést na stránkách Ministerstva dopravy.

2.2 Detekce dopravních značek

Detekcí a klasifikací dopravních značek se zabývají vědecké skupiny a stále vytvářejí nové postupy, pro přesnější a rychlejší detekování a klasifikování značek. Dostupné řešení může být použito v autonomním vozidlech. Detekce značek by se dala rozdělit do 4 skupin - barevná segmentace, detekce tvarů, extrakce příznaků a klasifikace.

Barevná segmentace

Při barevné segmentaci se využívá, že mají značky zákonem definovanou barvu. A dochází tedy k výběru oblastí s barvou určené dopravními značkami. Nejčastěji červená, žlutá a modrá.

Toho využívají autoři práce [8] ke segmentaci používají barevný model HSV, který je popsán v kapitole 3.1. Vytváří se dvě LUT tabulky na základě získaných barených složek z modelu a to barevný tón a sytost. Po vytvoření tabulek jsou snímky normalizovány na maximální hodnotu 255. V následujícím kroku jsou využívány genetické algoritmy pro vyhledání a určení pozice dopravní značky v obraze. Autoři použili klasifikátor neuronové sítě se dvěmi vrstvy.

Řešení bylo testováno na stroji s AMD Duron o 1 GHz. Úspěšnost detekování dopravní značky činala 70%-90%.

Dalším řešením, které využívá segmentaci je práce od Hassan Shojania [16], která se skládá ze čtyř hlavních fází - barevná segmentace, detekce hran, detekce tvaru a klasifikace. Segmentace je založena na převedením vstupního obrázku do binárního obrázku. Autor sestrojil řadu filtračních masek k nalezení rohů v obraze. Následně je snímek dál zpracováván a testován jestli splňuje tvar dopravní značky. Klasifikátorem jsou použity neuronové sítě 4.3. Metoda je úspěšná pro detekci kruhových červených značek, čtverců, trojúhelníků a osmiúhelníků. Ale má jistá omezení, snímky se nesmí otáčet a je povolen jen menší náklon zhledem k pozici kamery. V podstatě stejný náklon jako to, co řidič vidí normálně.

Detekce tvarů

Metody založené na detekci tvarů jsou více stabilní a neovlivnují podnební podmínky.

oho využívají autoři práce [14]. Práce detekuje a rozpoznává evropské a americké výstražní značky. Model je založený na rozpoznávání kruhu nebo trojúhelníku. Pro rozpoznávání potencionální oblasti umístění značky používá šedotonový obraz a Houghovy transformace. Klasifikátorem jsou jsou použity neuronove sítě a na rozpoznávání číslic se používá ODR. Datová sada byla manuálně vytvořena z dopravních značek Francie a Ameriky.

Úspěšnost při detekování dosahovala přibližně 90%. Program zpracovával přibližně 20 snímků/s.

Klasifikace

Klasifikace slouží k určení jestli tam dopravní značka je nebo není a taky se používá k určení typu dopravní značky. Následující dvě práce využívají klasifikátor AdaBoost.

Autoři vycházejí ze svého výzkumu [1] detekce osob. Z analyzovali víc jak 40 detektorů použitých během posledních 10 let a na základě nasbíraných zkušeností vytvořili vlastní detektor, který předčil ostaní. A patří nyní k nejlepším řešení na vědecké úrovňi. Detektor je založen na klasifikátoru Adaboost, který je popsán v kapitole **??**. Klasifikátor je složen ze slabých učících se dvoudimenzionálnách stromů, kde každý uzel je jednoduchý rozhodující prvek, definovaný trojuhelníkovou oblastí, kanálem a prahováním.

K natrénování detektoru autoři použili Quad-Core AMD Opteron 8360 SE s 64GBytes RAM. Vstupní snímky byly zmenšeny na 28x28 pixelů a převedeny na šedotonový obraz. Jako datové sady byly použity Německé dopravní značky [12] a Belgické dopravní značky [19], které dohromady obsahují více jak pěttisíc dopravních značek. Detekovaní značek trvalo 45 minut a byl použit Intel Core i7 870 o 2.5 MHz společně s Nvidia GeForce GTX 470 GPU.

Úspěšnost řešení dosahovala 96 %-99 %. Metoda dosahuje nejlepších výsledků, ale při detekci a datových sadech se neuvažuje vliv nepříznivých povětrnostních podmínek např. sníh, déšť, mlha, noc.

Autoři této práce [15] trénují a rozpoznávají Japonské dopravní značky. Využívají barevný model HSV s využitím klasifikátoru AdaBoost a integralní obraz . K natrénování metody autoři použili 4125 značek , k detekci použili 9036 značek. Vstupní obrázky byly zmenšeny na rozměr 60 x 60 pixelů.

K rozpoznání dopravních značek bylo použito 210 značek, které autoři pořídili sami při rychlosti 40 až 70 km/h s rozlišením videa 640 x 480. Úspěšnost rozpoznání dosahovala v rozmězí 80%-97%.

2.3 Datové sady

V této sekci budou popsány datové sady, které byly použity nebo vytvořeny pro testování, trénování SVM a vyhodnocení navrhnutých experimentů. Nejprve bude popsaná datová sada GTSRB, kterou jsem jako první používal ve své práci, jelikož součástí byly vypočítané vektory descriptů. Následně bude popsána datová sada Belgických dopravních značek, ze které jsem si pro potřebu trénování SVM vytvořil vlastní negativní datovou sadu. A závěr kapitoly bude věnován vytvoření vlastní datové sadě s Evropských dopravních značek.

Obrázek 2.1: Datová sada GTSRB

Datová sada GTSRB

Datová sada GTSRB (**G**erman **T**raffic **S**ign **D**etection **B**enchmark) [18] je tvořena německými dopravními značkami z různých regionů Neměcka pořízených z autokamery. Snímky byly pořízeny beheěm dne, ale také ve večerních hodinách. Jednotlivé snímky značek jsou realizovány pomocí výřezů o velikosti mezi 15x15 až 250x250 pixelů. Dopravní značka není přesně umístěna ve středu snímku a okolí značky je tvořeno minimálně z 10% z celkového obsahu plochy obrázku. Obrázky jsou uloženy ve formátu *ppm*.

Sada obsahuje celkem 39 209 výřezů pro trénování rozdělených do 43 tříd podle typu značky. Snímky lze rozdělit na čtyři skupiny podle typu značky: zákazové, příkazové, výstražné a upravující přednost. Na obrázku 2.1 jsou zobrazeny snímky, které jsou rozmazané, odlišné velikostí a pořízené v různou denní dobu.

K dostupným výřezům jsou k dispozici také vypočítané vektory deskriptorů pro histogram orientovaných gradientů, který je popsán v kapitole [odkaz]. Snímky byly zmenšeny na velikost 40x40 pixelů. Velikost buňky byla nastavena na 5x5 pixelů, posunutí 5x pixelů v obou směrech a velikost výsledného vektoru descriptoru nastavena na 1568.

Součástí datové sady je anotace uložená ve formátu *csv*, která obsahuje tzn. *groundtruth*. Informace jsou odděleny pomocí ";"a jsou složeny z jména souboru, šířky snímku, výšky snímky, souřadnicemi dopravní značky a číslem zařazení do třídy 0 do 42.

Společně s trénovací datovou sadou je dodána i testovací sada, která obsahuje 12 569 snímků. Testovací sada dodržuje stejné rozvržení jako trénovací sada.

Datová sada BelgiumTSC

Datová sada BelgiumTSC (**B**elgium **T**raffic **S**ign **C**lassification **B**enchmark) [19] obsahuje 4 591 výřezů pro trénování a 2 534 výřezů pro testování. Snímky jsou rozděleny do 62 tříd podle typu značek. Datová sada dodržuje stejnou strukturu jako datová sada GTSRB, akorát neobsahuje vypočítané vektory destcriptorů.

Součástí datové sady je i sadu obrázků, které tvoří pozadí respektive, kde není dopravní značka. Sada obsahuje 20 550 snímků, které byly pořízeny pomocí kamery z auta. Snímky zachycují běžný silniční provoz a ulice. Dále jsou obsaženy snímky stromů, lidí, domů, reklamních ploch a další.

Z datové sady pozadí si vytvářím vlastní negativní datavou sadu pro trénování kaskády SVM o velikosti 40 000 výřezů a pro testování o velikosti 20 000 výřezů. K tomutu účelu je využit script *create_neg.py*, který z vybraného snímku vyřízně vždy 5 výřezů. Pozice výřezu ze snímku, je vždy náhodně generována a výsledný výřez má také náhodně generovanou velikost v rozmezí 50 až 300 pixelů. Z důvodu zabránění generování podobních hodnot, je daný rozsah rozdělený na pět částí, vždy s rozdílem 50 pixelů. A z každé vzniklé skupiny, je vždy pořízený jeden výřez. Každá pětice výřezů je vždy umístěna do jednoho adresáře.

Obrázek 2.2: Datová sada BelgiumTSC

Obrázek 2.3: Datová sada Evropských dopravních značek

Na obrázku 2.2 v horní řadě jsou zobrazeny mnou vytvořeny negativní snímky a ve spodní řadě jsou zobrazeny výřezy značek, které jsou použity pro trénování SVM.

Datová sada evropských dopravních značek

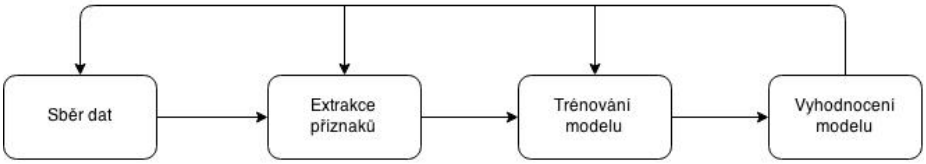
Jelikož není dostupná datová sada pro Českou Republiku, vytvořil jsem si vlastní datovou sadu z Evropských dopravních značek . Získané snímky tvoří jenom dopravní značku bez okolí. Ale nejprve pomocí scriptu *eu.py* jsou obrázky roztříděné do tříd podle vzoru datové sady GTSRB. Ke správnému roztřídění značek jsou implementovány metody ve třídě *Sort*. V dalším kroku se přidává k obrázkům pozadí, které bylo vygenerováno pomocí scriptu *create_neg.py*. Pro vložení dopravní značky do výřezu obrázku s okolím, je nutné nejdřív vytvořit masku značky pomocí funkcí *threshold* a *bitwise_not* z OpenCV [odkaz na kapitolu]. Pomocí funkce *add* z OpenCV je vzniklá maska přidána ke snímku s okolím. K takto vzniklému výřezu se vytváří i anotace ve stejném formátu jako v datové sadě GTSRB, která je popsána v sekci 2.3.

Na obrázku 2.3 jsou zobrazeny vytvořené výřezy dopravních značek. Značky jsou rozděleny celkem do 106 tříd a obahují celkem 2 332 snímků.

2.4 Procedura trénování a testování

Postup trénování a testováí je znázorněn na obrázku 2.4 . Řešení se skládá ze čtyř bloků - sběr dat, extrakce příznaků, trénování modelu a vyhodnocení. Na základě vyhodnocení natrénovaného modelu bude zdřejmé jak je model kvalitní či nikoliv. V případě neuspokokivých výsledků je vhodné se v procesu vrátit, na základě vyhodnocení upravit určitý krok. A následně zopakovat od toho kroku celý postup až po vyhodnocení. Z návrhu je patrné, že nalezení kvalitního modelu je časově náročné.

Prvním krokem pro detekci objektů v obraze je sběr dat. Příprava dat je časově nejnáročnější a nejobtížnější krok celého procesu. Současně je to ale krok, který má klíčový význam pro úspěch výsledné aplikace. Je vhodné mít dostatečně velkou datovou sadu hledaných ale i nehledaných objektů. V případě bude-li datový set tvořen např. z 95% pozitivních objektů a zbylých 5% negativních objektů. Bude výsledný systém při testování



Obrázek 2.4: Blokové schéma procedury trénování a vyhodnocení

dávat velmi dobré výsledky pro hledané objekty, ale negativní sadu může model klasifikovat jako hledané objekty. A při tak malé negativní datové sadě, se ve výsledku natrénovaný model může chybně považovat za kvalitní. K zabránění tomuto nežádoucímu stavu se snaží předejít pomocí různých metod vyhodnocení modelu, které jsou popsány v kapitole 5.1.

Ve své práci se nezabývým implementací extrakce příznaků. Nejprve jsem používal dodané descriptory v datové sadě a následně jsem využíval výstupu programu Jana Duška. Který decriptory histogramu orientovaných gradientů generoval pomocí knihovny OpenCV s nastavením: velikost bloku- 10x10 pixelů , velikost buňky- 5x5 pixelů, s posunitím bloku při normalizaci o 5 pixelů v obou směrech, spočtem 8 kanálů v buňce a normalizací L2Hys s hranicí 0,5.

Popis dalších kroků v procesu je popsán v následujícíh kapitolách.

Kapitola 3

Obrazové příznaky

V této kapitole budou popsány algoritmy pro zpracování obrazu a extrakci příznaků

Účelem extrakce příznaků je získání informace z obrazu. Získané hodnoty jsou pak vstupem klasifikátoru.

3.1 Předzpracování obrazu

V této sekci budou popsány 3 barevné modely, které se využívají při výběru oblastí pravděpodobného výskytu dopravní značky. Budou zde diskutovány jejich výhody a nevýhody.

Při psaní této kapitoly jsem vycházel z knihy zabývající se počítačovou grafikou [21] a zpracování obrazu [5].

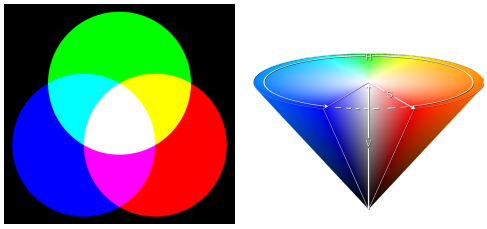
Model RGB

Barevný model RGB se často používá pro výběr oblastí provděpodobného výskytu značky. Model je složen ze tří složek - červené (**R**, red), zelené (**G**, green) a modré (**B**, blue), které udávají barvu jednoho bodu. Smícháním všech barevných složek vznikne bílá bárva. Další barevné kombinace lze získat adaptivním smícháním těchto základních barev, které lze vidět na obrázku odkaz. Tento model je jeden z nejpoužívanejších modelů a využívá na zobrazovacích zařízeních, jako LCD monitory, plasmové televize apod. Barevný model je reprezentován krychlí.

Nevýhodou tohoto modelu je jeho závislot na světelných podmínkách. To se nehodí při výběru oblasti. Snímky pořízené za různého počasí nebo v různou denní se budou lišit. Jedním z řešením je použítí jiného barevného modelu např. HSV, který bude popsán v následující kapitole 3.1, který odděluje informace o barvě od informace o jasové složce daného bodu.

Model HSV

Barevný model HSV (případně HSL) je dalším často využívaným modelem pro výběr oblastí s pravděpodobným výskytem dopravní značky. Tento model může být náhradou za model RGB. Umožnuje totiž oddělovat informace o barvě od informace o jasové složky daného bodu a může být odolný vůči změně světelných podmínek. Opět je barva bodu určena třemi složkami, ale mají zde jiný význam než u RGB modelu. Složky - barevný tón (**H**, hue), sytost (**S**, saturation) a jasová hodnota (**V**, value). Barevný tón označuje převládající spektrální barvu, sytost určuje příměs jiných barev a jas je dán množstvím bíleho světla.



(a) RGB (b) HSV

Obrázek 3.1: Ukázka modelu HSV a adaptivní míchaní modelu RGB

Nevýhodou tohoto modelu je, že musí být převeden z modelu RGB. Převod se provádí na základě rovnic 3.1

$$
\begin{gathered}
R, G, B \in < 0, 1 > \quad MAX = max(R, G, B), \quad MIN = min(R, G, B) \\
H = \begin{cases} (0 + \frac{G-B}{MAX-MIN} \times 60, & pokud \quad R = MAX \\ (2 + \frac{B-R}{MAX-MIN} \times 60, & pokud \quad G = MAX \\ (4 + \frac{R-G}{MAX-MIN} \times 60, & pokud \quad B = MAX \end{cases} \\
S = \frac{MAX-MIN}{MAX} \\
V = MAX
\end{gathered} \tag{3.1}
$$

Pro popis zobrazení barev v modelu HSV je zde použit šestiboký jehlan umístěný do souřadnicového systému takovým způsobem, že vrchol jehlanu je v počátku a osa jehlanu je shodná se svislou osou, která znázorňuje úrovně jasu. Barevný tón je definován jako velikost úhlu, který může nabývat hodnot 0-360°. Sytost a jas, které jsou umístěny na vodorovné ose, se mění v intervalu $< 0, 1 >$. Další nevýhodou toho modelu je, že přechod mezi bílou a černou barvou není plynulý a pohyb barevného tónu se neodehrává po kružnici, ale po šestiúhelníku. Změna barevného tónu není plynulá

Šedotónový obraz

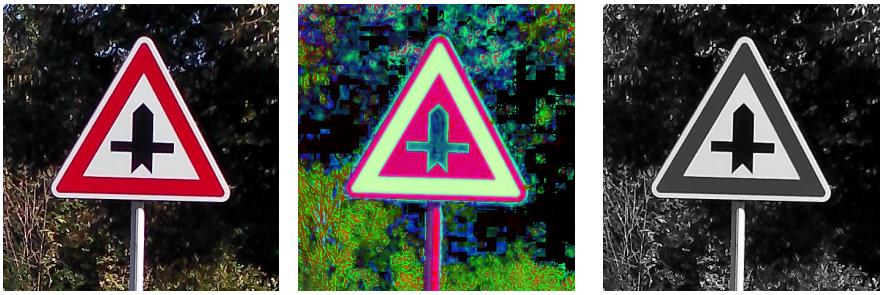
Z jedním důvodů převedení obrazu do odstínů šedi, může být, že se v šedotónovém obrazu hledají hrany, které jsou potřeba pro následné rozpoznávání objektů. Nebo dalším důvodem může být tvorba uměleckých fotografií. Barva bodu je reprezentována pouze jednou hodnotou a ne třemi jako u RGB modelu 3.1. Proto je často využíván při výběru oblasti pro svou jednoduchost. Lidské oko je různě citlivé na jednotlivé složky RGB a nelze tak sečíst jednotlivé složky, ale je nutný převod pomocí empirického vztahu:

$$f = 0.299R + 0.587G + 0.114B \tag{3.2}$$

Kde f je výsledná úroveň jasu v šedotónovém obraze a hodnoty R,G,B jsou hodnoty z modelu RGB. Na obrazku je ukázka převodu z modelu RGB na šedotónový obraz.

3.2 Haarovy příznaky

Haarovy příznaky též znány pod názvém Haarovy vlnky,které jsou založeny na rozdílu jasu mezi obdélníkovými oblastmi. Byly použity v práci Viala & Jones [20].



(a) RGB (b) HSV (c) Šedotónový obraz

Obrázek 3.2: Ukázka převodu barevných modelů

Hodnota příznaku je získána z obrazu s vypočítáním suma pixelů obrazu odpovídající bílé části. A suma pixelů v černé části. Tyto hodnoty jsou pak odsebe odečteny. Ten to postup je značně časově náročný a proto se používá integrální obraz 3.3.

3.3 Integrální obraz

Integrální obraz (summared area table) se používá jako rychlý a efektivní způsob výpočtu součtu hodnot v obdelníkové oblasti v daném obrazu. Integralní obraz se používá při posledním kroku extrakce příznaků, protože je extrakce často časově náročná kvůli častým přístupům do paměti. Tato metoda byla použita v práci [20] pro rychlou extrakci příznaků pro detekci obličejů.

Výpočet integrálního obrazu pro pixel o souřadnicích (x,y) se provede pomocí vzorečku 3.3. Jedná se tedy o sumu hodnot jednotlivých pixelů v obdélníku, který je podmnožinou původního obrázku.

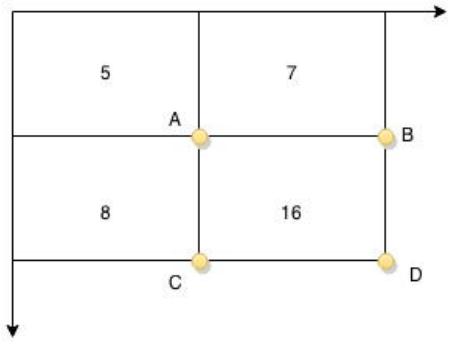
$$I(x,y) = \sum_{x' \leq x, y' \leq' y} i(x^{'}, y^{'}) \tag{3.3}$$

Integrální obraz lze spočítat jediným průchodem původního obrazu v konstantním čase. Složitost O(1). Výpočet hodnoty sumy zabere vždy pouze 4 přístupy do paměti. Například pro výpočet hodnoty sumy na obrazku stačí znát oblasti určené body A, B, C, D odpovídajícího integrálního obrazu. Pro výpočet stačí tři aritmetické operace sčítání a odčítání:

$$i(x^{'}, y^{'}) = s(A) + s(D) - s(B) - s(C) = 5 + 16 - 7 - 8 = 6 \tag{3.4}$$

3.4 Histogram orientovaných gradientů

Myšlenkou histogramu orientovaných gradientů je, že hledaný objekt v obraze, lze pomocí tvaru a vzhledu popsat intenzitou gradientu. A lze použít pro rozpoznávání objektů a klasifikaci. V případě kdy objekty nemění své natočení v obraze. Histogram gradientů lze počítat i z černobílých obrazů, ale tím se získá jenom jedna hodnota. A nebo lze počítat i pro barevný obraz, kde se počítá pro každou barevnou složku zvlášť. Obraz se rozdělí na malé oblasti - buňky a pro každou buňku je vypočítaný jednorozměrný histogram, který se počítá pro všechny pixely z buňky. Obraz před výpočty je vhodné znormalizovat. Shomaždováním informací z buňky, ale i z okolí vzniká tzv. blok.



Obrázek 3.3: Integralní obraz

Celý obraz je tedy rozdělen do bloků o stejné velikosti a vzniklé buňky jsou nositely hodnot. Při výpočtu histogramu s využitím kruhové nebo pravoúhlé obblasti je nutné rozdělit vypočítaný úhel na několik rozsahů. Velikostí rozsahů se může dodcílit a přijetí malého natočení klasifikovaného objektu.

Kapitola 4

Klasifikace a modely

V této kapitole budou popsány použité knihovny, postupy pro vytvoření trénovacího modelu a následně klasifikátory, které se nejčaštěji používají pro detekci dopravních značek. Závěr kapitoly bude věnován trénování SVM a implementaci výsledného programu.

Informace jsem čerpal z [17] a [6].

4.1 Použité knihovny

Pro implementaci mé práce jsem využil některé existující knihovny.

Knihovna OpenCV 2.4.10

Knihovna OpenCV je volně dostupná a multiplatformní knihovna pro práci s obrazem nebo s videem v reálném čase zameřená na algoritmy počítačového vidění. Dále poskytuje nástroje pro extrakci příznaků, rozpoznávání a klasifikaci nebo také podporu pro strojové učení - SVM, AdaBoost, neuronové sítě a další. Knihovna je implementovaná v jazyce C/C++ a poskytuje rozhraní pro jazyky C, C++, C#, Python, Java. Je distribuována pod BSD licencí.

Z možností knihovny jsem využil funkce pro načítaním a ukládání obrázku. Vytvoření výřezu z obrázku a funkce pro vytvoření masky obrázku.

Knihovna LibSVM 3.20

Další knihovou, kterou jsem používal, je knihovna LibSVM [4] verze 3.20, která je napsaná v jazyce C/C++ a je multiplatformní. Je vyvíjena na Národní Taiwanské univerzitě.Jedná se o knihovnu pro prácí se support vector machines. Knihovna implementuje kernel funkce - lineární, polynomiální, RBF a sigmoidální. Ve své práci jsem využíval RBF kernelu.

Knihovna poskytuje binární soubory *svm-train* a *svm-predic* pro použití z příkazové řádky. Ve své práci jsem je využíval pro hromadné trénovaní při hledání nejvhodněších parametrů. Dále jsem využíval některé funkce knihovny ve vlastním programu pro trénovaní modelu.

Více informací k LibSVM, lze získat v manuálu ke knihovně [3]

4.2 Vytvoření modelu

Při trénování modelu se většinou používá metoda učení s učitelem, kterou využívám ve své práci. Vychází se tedy z toho, že se učícímu algoritmu předají trénovací data, která jsou zařazena do odpovídajících tříd. Na testovacích datech se pak porovnává úspěšnost klasifikovaných hodnot z trénovacích dat. Metod testování je více, záleží jaká data se použií pro testování a jaká pro trénování:

- testování v celých trénovacích datech
- křížová validace (cross-validation)
- bootstrap
- leave-one-out
- testování na testovacích datech.

Testování na datech, které byly použity pro učení klasifikátoru mají nejmenší vypovídající schopnost o tom, jak budou hodnotné znalosti získané z klasifikování nových zkoumaných dat. Při trénování na datech může často dojít k přeučení (overfitting) a může se tento stav snadno přehlédnout.

Testovací metoda křížová validace zamezuje překrývání testovacích dat. Data se rozdělí na K částí, kde část $\frac{1}{K}$ se použije pro testování a zbylé data $1-\frac{1}{K}$ se použijí pro trénování. To se opakuje K-krát. Každá testovací část je použita právě jednou pro testování modelu vzniklého ze zbylých částí dat. Nejčastěji volená hodnota K bývá 10 nebo 30. Ve své práci využívám tu to metodu a kvůli časové náročnosti testovaní jsem zvolil hodnotu $K = 3$.

U metody bootstrap, která je podobná křížové validaci se mohou vybrané data pro učení použít vícekrát. Trénovací data se rozdělí v poměru 63,2% pro trénování a zbylých 36.8% pro testování

Metoda leave-one-out je taky podobná metodě křížové validace, ale narozdíl z dat o N vzorcích se vybere jenom jeden pro testování a zbytek $N-1$ se použije na trénování. Celý postup se provede N-krát. Tato metoda se obvykle používá na medicínských datech.

Testování na testovacích datech je nejvhodnějším způsobem ověření modelu. Problém můžu vzniknout při nedostatku dat.

Vytvoření trénovacího modelu

Pro vyhodnocení natrénovaného modelu využívám metodu křížové validace, která je popsaná v kapitole 4.2. Z časových důvodů vyhodnocení natrénovaných modelů jsem zvolil postup, který pomocí křížové validace s hodnotou $K = 3$ rozdělí testovací a trénovací soubory na tři části. Pro vytvoření trénovacího modelu pro binární klasifikátor je vytvořen script *create_bin_input.py* a pro potřeby modelu klasifikátoru do více tříd je vytvořen script *create_class_input.py*.

Script pro binární klasifikátor očekává adresář s pozitivními descriptory příznaků a adresář s negativními descriptory. V případě scriptu pro klasifikátor do více tříd je očekáván jedinný adresář, ve kterém budou už descriptory příznaků rozdělené do tříd. Rozdělení decriptorů příznaků na testovací a trénovací je implementován v metodě *split_cross*. Metodě je předány parametry, kde jsou uloženy absolutní cesty souborů příznaků, hodnota zvolené křížové validace a hodnota udávající jaká část souborů bude použita pro testování. Script je

umožnuje vytvořit ve správném formátu soubory poddle zvolené hodnoty křížové validace. Výstupem scriptů jsou modely pro trénování a testování, které se uloži na pevný disk.

Oba scripty nejprve upraví vektory decriptorů snímku na požadovaný tvar knihovnou LibSVM 4.1. Převod do požadovaného formátu obstarává metoda *prepare_data*, které jsou předány hodnoty decriptory příznaků jednoho snímku. Metoda postupně iteruje předané hodnoty a přidává k hodnotě její načtené pořadí oddělené ":". Tak to vzniklý řetězec, využívá metoda *format_file.py*, která ještě před řetezec přidává třídu zařazení daného vzorku. Třída zařazení může být ovlivněna, jestli se jedná o binární klasifikátor nebo klasifikátor do více tříd. V případě binárního klasifikátoru hodnota pro hledané objekty bude $+1$ a v případě nehledaného objektu -1. Tomuto postupu odpovídá metoda *one-vs.-one (OvO)*. U klasifikátoru do více tříd je použita metoda *one-vs.-rest (OvR)*, která spočívá v tom, že zvolená třída bude označena kladně a zbylé třídy záporně. Třídy jsou označeny podle počtu tříd v daném adresáři.

Součástí scriptů jsou i metody, které umožnují dodatečné přidaní negativních nebo pozitivních příznků do výsledných modelů.

4.3 Klasifikátory

Úkolem klasifikátoru je přiřadit daný objekt k odpovídající třídě. Protože nelze získat často dokonalý klasifikátor a tak obecnějším úkolem je určení pravděpodobnosti pro každou třídu. Výstupem klasifikátoru v úloze detekce dopravních značek je tedy informace udávající jestli vstupní výřez obrazu je dopravní značka či okolí. Nebo může představovat konkrétní typ dopravní značky.

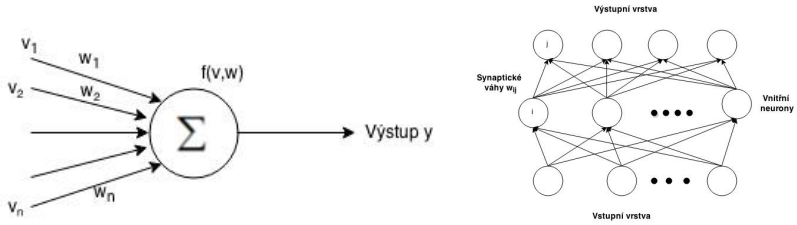
Neuronove sítě

Neuronové sítě jsou inspirovány biologickými neuronovými sítěmi. Základním prvkem je neuron na obrázku[odkaz na obrazek] je jeho model, který navrhli McCulloch a Pitss. Model obsahuje několik vstupů, které jsou ohodnoceny váhami a jedinný výstup. Činnost neuronu lze popsat matematicky. Vstupy jsou označeny $v_1 - v_n$ a váhy $w_1 - w_n$ a práh w_0. Bude-li potenciál , který se spočítá podle rovnice 4.1, větší než práh w_0 tak neuron vyšle signál s hodnotou 1 a v opačném případě s hodnotou 0. Neuron je tím to schopný rozdělit prostor možných řešení pouze na dva poloprostory. Ve složitějších úlohách je to to řešení nedostatečné a proto se jednotlivé neurony uspořádávají do sítí neuronů. Neuronová síť je schopná řešit nelineární úlohy.

$$P \quad = \sum_{i=1}^{n} v_i w_i \tag{4.1}$$

Nejrozšířenější způsob propojení neuronů jsou tzv. vícevrstvé sítě - několik vrstev neuronů je vzájemně propojených každý s každým. Příklad je uveden na obrázku[odkaz na obrazek]. Z obrázku je patrné, že obsahuje minimálně tři vrstvy neuronů- vstupní, výstupní a alespoň jednu vnitřní vrstvu. Vstupní vrstvě jsou předány signály, které přeposílá na všechny neurony. Následně se signály zpracovávají jako v předešlém kroku - sečtou se váhové vstupy a bude-li potenciál dost velký tak neuron vyšle signál. Výstupní vrstva předává informavi o rozhodnutí sítě a případně rozdělení do tříd.

Během učení neuronové sítě může dojít k uváznutí v lokálním minimu. Tento stav vzniká, když se siť nepodaří naučit s dostatečně malou chybou a v takovém případě se dosáhne určité hodnoty,která nadále přestane klesat. Řešením uváznutí v lokálním minumu je vhodonou



(a) Model umělého neuronu (b) Neuronová síť

Obrázek 4.1: Neuronové sítě

volbou parametrů nebo vhodným počátečním nastavením vah. I přes tady tento nežádoucí stav se neuronové sitě používají často rozpoznávání dopravních značek.

Boosting

Meto je založena na principu kombinování slabých klasifikátorů s horší úšpěšností do jednoho výsledného klalitního klasifikátoru. Slabý klasifikátor může být reprezentován rozhodovacím stromem, perceptronem a další. Omezujícím kritériem výběru je aby byla chybovost klasifikátoru menší než 0,5. Výsledný klasifikátor $H(x)$ 4.2 lze matematicky zapsat jako lineární kombinací slabých klasifikátorů $h_t(x)$. Výsledný klasifikátor lineární není.

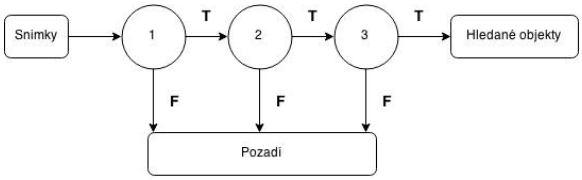
$$H(x) = sign(\sum_{t=1}^{T} \alpha_t h_i())$$

V součastnosti nejpoužívanější variantou metody boosting je AdaBoost (**ada**ptive **boost**ing). Metoda umožnuje při návhu přidávat slabé žáky tak dlouho, dokud není splněna optimální hodnota klasifikační chyby. Každý trénovací vzorek dostane přidělenou váhu, která určí pravděpodobnost zařazení do trénovací množiny pro jednotlivé klasifikátory. Je-li vzorek klasifikován přesně, jeho opětovné zařazení do klasifikace klesá a v opačném případě roste. AdaBoost se zaměřuje na obtížné vzorky. Jeho funkce je definována následovně:

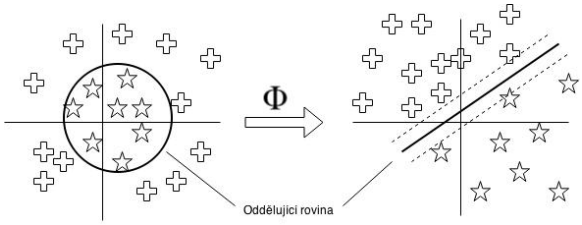
- Při inicializaci dostanou všechny vzorky stejné hodnoty vah. V každém iteračním kroku k se natrénuje klasifikátor C_k
- V následujícím kroku je pro klasifikátor C_k proveden výpočet váhy klasifikátoru α_k. Hodota váhy je závislá na trénovací chybě klasifikátoru.
- Pro chybně klasifikované vzorky se hodnota váhy zvýší a při správné klasifikaci se hodnota sníží. Z_k je normalizační konstanta.

Kaskáda klasifikátorů

Myšlenka kaskádních klasifikátorů je mít sekvenčně seřezené slabé klasifikátory do jedinného silného klasifikátoru. Např. práce Viola & Jones [20] na detekci obličejů je založena na skenování výřezů pomocí posuvného okna. Tím bude k detekci velká část vzorků, které nebudou patřit do hledaných tříd. Viola & Jones si uvědomili, že rychlost detektoru závisí na rychlosti klasifikace pozadí. Pomocí algoritmu AdaBoost je sestavena soustava slabých klasifikátorů, kterým byla nastavena nejvhodnější váha a výsledkem je pak kaskádový klasifikátor. Velká část výřezů je tak v každé úrovni kaskády zahozena a část výřezů, která je



Obrázek 4.2: Kaskáda klasifikátorů



Obrázek 4.3: SVM- transformace prostoru

klasifikována kladně, projde do další úrovně zpracování. Výřez, který projde až na konec kaskády je s velkou pravděpodobností hledaný obličej.

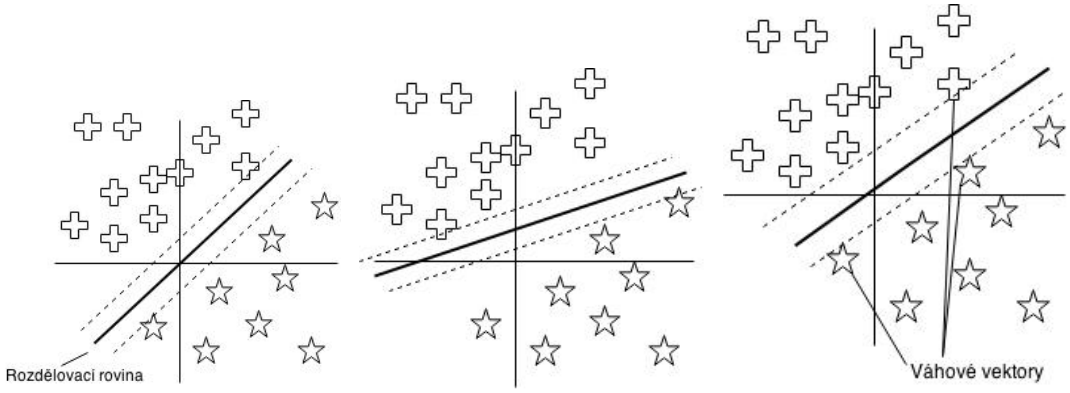
Na obrázku 4.2 je zobrazena architektura kaskády. Počáteční klasifikátory vyřadí velkou část negativních vzorků a pozitivní vyorky postupují k dalšímu zpracování. Většina negativních vzorků je vyřazena v počátečních úrovních kaskády a tím může v následujících fázích stoupnout výpočetní složitost klasifikace. Do poslední úrovně kaskády se dostane už jenom málo vzorků a celká klasifikace je velmi rychlá. Výstupem posleního klasifikátoru v kaskádě je pak hledaný objekt.

4.4 Support Vector Machines

Support Vector Machines (SVM) je klasifikátor, který se často používá pro klasifikaci kvůli jeho velké přesnosti a schopností pracovat s vícedimensionálními daty. Tuto metodu používám ve své práci. SVM patří do kategorie tzv. jádrových algoritmů (kernel machines) [11], které poskytují efektivní algoritmy pro nalezení lineární hranice a zároveň jsou schopny zpracovávat vysoce složité nelineární funkce. Jedním se základních principů je transformace nelineárního vstupního prostoru do jiného, vícedimensionálního, kde již lze od sebe lineárlně oddělit třídy. Tato myšlenka je v podstatě jednoduchá. Na obrázku 4.3 v levo ve dvourozměrném prostoru jsou dvě třídy, oddělené nelineární kružnící,které jsou transformovány do jiného prostoru (future space) pomocí nelineární funkce Φ. Vzniký prostor na obrázku 4.3 v pravo je vícedimensionální, který už umožnuje oddělit obě třídy lineární rovinou.

Základní funkcí SVM je nalezení oddělující roviny mezi dvěma třídami dat . K nalezení roviny pomáhají podpůrné vektory (support vectors) 4.4c, které leží na okraji prázdné oblasti a přímo ovlivnují řešení. Čím budou mít větší vzdálenost od oddělující roviny tím zlepší výsledný klasifikátor. Kdyby se ostatní data vypustila z trénovacího setu, výsledek by se nezměnil. Na obrázku 4.4a a 4.4b jsou znázorněny neoptimální rozdělení rovinou a na obrázku 4.4c je zobrrazeno optimální rozdělení.

Více informací o SVM, lze získat v článku [13].



(a) Neoptimální rozdělení (b) Neoptimální rozdělení (c) Optimální rozdělení

Obrázek 4.4: SVM - odělující rovina

4.5 Implementace

V této části kapitoly se budu věnovat popisu trénování modelů pro kaskádu SVM. A na závěr popisu implementaci výsledné aplikace.

Návh kaskády SVM

Při návrhu klasifikátoru jsem vycházl z myšlenky práce Viola & Jones [20], kde autoři vytvořili kaskádu slabých klasifikátorů. A v každém kroku procesu se snažili, vždy eliminovat nejvíc negativních snímků a až ve výsledku jim zůstanou jenom pozitivní objekty. Na základě této myšlenky jsem si vytvořil binární klasifikátor, který je prvním stupněm kaskády a jeho účelem je eliminovat negativní vzorky a předat dál hledané objekty druhému klasifikátoru, který klasifikuje vzorky do tříd a tvoří tak druhý a konečný stupeň kaskády.

Při testování kaskády klasifikátorů jsem si všiml, že binární klasifikátor označí pár vzorků jako falešně pozitivní a klasifikátor složený jenom z dopravních značek je označí jako dopravní značky s konkétní třídou.

Trénování kaskády SVM

V této sekci bude popsán popstup trénování a volba vhodných parametrů. Z návrhu je patrné, že se kaskáda skládá ze dvou klasifikátorů, které využívají odlišné trénovací modely.

Z počátku jsem používal binární soubory dostupné v knihovně LibSVM a to pro trénování konkrétně *svm-train* a pro klasifikaci *svm-predic*. Binární soubory očekávaly na vstupu vytvořené modely ve správném formátu požadované knihovnou LibSVM. Popis vytvoření modelů pro kontrétní klasifikátory je popsán v kapitole 4.2. Využíval jsem jej i pro hromadné trénovaní modelů, při hledání optimálních parametrů.

Později jsem si implementoval vlastní program *trainSign* pro trénování využívající funkce z *svm.cpp* a *svm.h* dostupné z knihovny. Program je implementovaný v jazyce C++ a při spuštění očekává trénovací model. Načtě trénovací model, zjistí se délka příznakových descriptů a počet vzorků. Podle získaných hodot se alokuje pamět ve struktuře *svm_model* a pomocí automatu jsou do ní ukládaní hodnoty tříd a hodnoty descritorů. Po načtení souboru je struktura předaná funkci *train* z knihovny, která vrací natrénovany model a pro uložení modelu na pevný disk je volaná funkce *save_to_file*.

Vyzkoušel jsem přesnost klasifikace natrénovaných modelů s využitím lineárního kernelu a kernelu RBF. Ukázalo se, že klasifikace pomocí RBF kernelu je přesnější, ale proces trénovaní a klasifikování je pomalejší. Ve své práci požaduji více přesnost a tak jsem zvolil kernel RBF.

Na základě testování jsem zjistil, že klasifikace s základními parametry při trénovaní není ideální. Velká část pozitivních vzorků v případě binárního klasifikátoru byla označena jako falešně negativní. Po prostudování manuálu ke knihovně LibSVM jsem zjistil, že zlepšit výsledky klasifikace mohu pomocí paramatrů *-c* a *-g*. Parametr *-c* může být měněn od hodnoty 2^{-5} až po hodnotu 2^{15}. Používá se k posunutí oddělovací roviny mezi třídami dat. Čím víc bude hodnota parametru c větší, tak tím víc se rozhodovací rovina přikloní k menší skupině třídy. V případě malé hodnoty se bude přiklánět k větší skupině. Defaultní hodnota je na stavena na 1. Parametr *-g* může být měněn od hodnoty 2^{-15} až po hodnotu 2^{3}. Je to konstanta pro funkci kernelu. Defaultně byvá nastavena jako:

$$g = \frac{1}{delka\ vektoru\ descriptoru} \tag{4.2}$$

Podle článku [1], ve kterém se používá stejná datová sada jako v mé práci, jsem volil parametr $c = 10$. Výsledky byly mnohem lepší. Experimentoval jsem s různými hodnotami c a g a na základě článku, kde používají opět stejnou datovou sadu, jsem volil hodnoty $c = 10$ a $g = 0.01$. Výsledky byly nepatrně lepší. Takto zíkané parametry jsou platné jenon pro klasifikátor do více tříd. Protože ve své práci používám i binární klasifikátor s vlastní negativní datovou sadou, tak jsem si vytvořil script pro nalezení optimálních parametrů. Na základě experimentování a článku [kaskada], jsem měnil hodnotu c od 1 do 20. A hodnotu g od 1 do 0.00781, kterou jsem získal vždy jako polovinu hodnoty z předešlého kroku. A z vyhodnocení byly získány nejlepší parametry $c = 14$ a $g = 0.00391$. Hledání parametrů je implementováno ve scriptu *train.py*, který využívá binární soubory pro trénovaní a klasifikaci.

Vyzkoušel jsem i několik modelů sestaných s různou velikostí datové sady. Testoval jsem model s relativně malou datovou sadou přibližně 2 000 snímků, střední 20 000 snímků a výslednou se 40 000 snímků. Se vzrůstajícím počtem snímků v negativní sade se i zlepšovala přesnost klasifikace, ale také se i zvětšovala velikost modelu. Velikost 40 000 snímků negativní datové sady je vytvořen na základě čtení manuálu k LibSVM, kde je doporučeno mít vyváženou datovou sadu. V případě nevyvážené datové sady, je možné nastavit váhové hodnoty pro rozhodování.

Zkoušel jsem použít pro natrénování modelu i SVM z OpenCV 4.1, které je založené na LibSVM. Výsledky z testováni byly nepatrně horší se stejnýma parametrama. Důvodem je, že OpenCV používá starší verzi LibSVM. A natrénované modely nejsou vzájemně kompatibilní tak jsem to to řešení ve své práci nepoužil.

Výsledná aplikace

Tato sekce bude věnována nejprve popisu implementace kaskády SVM a v závěru popisu modelu, který jsem předal Honzovi Duškovi.

Výsledná aplikace je implementována v *classifySign* a je napsaná v jazyce C++. Jsou realizovány dva moduly pro práci s binárním klasifikátorem a s klasifikátorem do tříd. Výstupem je buď soubour klasifikovaných hodnot a nebo slovní pojmenování dopravních značek v českém jazyce, který je platný jenom pro datovou sadu německých dopravních značek 2.3.

Program očekává testovací soubor ve formátu platný pro LibSVM. Testovací soubor můžu obsahovat i jedinný vzorek. Načtou natrénované modely pro oba stupně kaskády. Následně se čte z testovacího souboru jenom vždy jeden řádak a hodnoty jsou uloženy do struktury *svm_node*. Následně je volaná funkce pro klasifikování z *svm.cpp*, která vrací *double* hodnotu a po vyhodnocení jestli je to hledaný objekt se předá *svm_node* druhému klasifikátoru na vyhodnocení. A získaná hodnota třídy je zapsána do souboru. V případě, kdy binární klsifikátor označí vzorek jako negativní, tak struktura se nepředává dál a čte se další testovací vzorek z testovacího souboru.

Jelikož kolega nemohl ve své práci využít natrénovaný model provedl jsem přepočet z modelu pro binární klasifikátor. Program je implementovany v *extractModel* a pomocí jazyky C++ a využívá knihovní funkce z knihovny LibSVM. Program načte model do struktury *svm_model* pomocí funkce *loadModelFromFile()* a získá tak potřebné hodnoty konstant. Následně se hodnoty modelu se stejným pořadím vynásobí mezi sebou s konstantou α, která vyjadřuje hodnotu konkrétní třídy zařazení vzorku a získané výsledky jsou postupně ukládány do pomocného vektoru. Délka vektoru je daná delkou vektoru descriptorů na základě, kterých byl natrénovaný model. Po provedení výpočtu je obsah v pomocném vektoru uložen do souboru na pevný disk.

Kapitola 5

Experimenty a vyhodnocení

V této kapitole se budu věnovat vyhodnocení výstupu klasifikátoru dopravních značek, který byl implementovaný v kapitole 4.5 a také vyhodnocení natrénovaných modelů. Pro vyhodnocení klasifikovaných objektů je použita charakteritika $F - measure$, která je popsána v kapitole 5.1. V první části kapitoly budou popsány metriky pro vyhodnocení modelu, následně vyhodnocení a závěr kapitoly bude věnovaný experimentům.

Pro natrénování a vyhodnocení klasifikátoru jsem využíval převedené snímky na příznaky pomocí práce Jana Duška [7].

5.1 Metriky pro vyhodnocení kvality modelu

V této kapitole budou popsány metriky, které se používají k vyhodnocení natrénovaného modelu. Při psané této kapitoly jsem čerpal informace z [10] a [2].

Matice záměn

Matice záměn (confusion matrix) je užitečný nástroj k určení jak dobře nebo s jakou přesností klasifikátor rozpoznal testovací data a zařadil do tříd. Matice záměn viz tabulka 5.1, je to čtvercová matice o velikosti daných počtem tříd, do kterých se provádí klasifikace. V matici řádky představují třídy ve kterých je správná klasifikace a sloupce značí klasifikované třídy pomocí natrénovaného modelu. Tabulka 5.1 zobrazuje případ, kdy klasifikátor data zařazuje do dvou tříd, pozitivní a negativní.

Vysvětlení pojmů, které se vyskytují v tabulce:

- True Positive (TP) - vzorek patří do pozitivní třídy a je klasifikátorem označený správně pozitivní
- False Negative (FN) - vzorek patří do pozitivní třídy a je klasifikátorem označený chybně negativní
- True Negative (TN) - vzorek patří do negativní třídy a je klasifikátorem označený správně negativní
- False Positive (FP) - vzorek patří do negativní třídy a je klasifikátorem označený chybně pozitivní

Hodnoty TP a TN udávají jak klasifikátor správně klasifikoval data a zatímco hodnoty FP a FN udávají chybnou klasifikaci. V tabulce mohou být navíc řádky a sloupce, pro

celkové součty. V tabulce mohou být ješte uvedeny řádky a sloupce pro celkový počet klasifikovatelných tříd. Kde P a N udávájí celkový počet správně klasifikovaných vzorků a to pozitivně nebo negativně. P' a N' opět udávají počet, ale klasifikované vzorky modelem a to pozitivně nebo negativně.

	Klasifikace modelem		
Správná klasifikace	Pozitivní	Negativní	Celkem
Pozitivní	TP	FN	P
Negativní	FP	TN	N
Celkem	P'	N'	P + N

Tabulka 5.1: Matice záměn

Na základě matice záměn lze spočítat různé charakteristiky k určení vyhodnocení modelu:

Úspěšnost (accuracy)

$$ACC = \frac{TP + TN}{P + N} \tag{5.1}$$

Úspěšnost vrací hodnotu v procentech jak klasifikátor správně klasifikoval testovací data. Metoda je závislá na složení testovacích datech, které snadno můžou vést k chybnému vyhodnocení.

Chyba (error)

$$ERR = \frac{FP + FN}{P + N} \tag{5.2}$$

Celková chyba vyjadřuje relativní počet chybných rozhodnotí systému. Společně s úspěšností slouží jako nejjednodušší charakteristika vyhodnocení.

Úplnost (Senzivita)

$$SENS = \frac{TP}{P} = \frac{TP}{TP + FN} \tag{5.3}$$

Úplnost je charakteristika, která říká kolik z hledaných objektů bylo úspěšně nalezeno.

Specificita (Specificity)

$$SPEC = \frac{TN}{N} = \frac{TN}{TN + FP} \tag{5.4}$$

Specificita je chrakteristikou, která určuje jak byly správně klasifikované nehledané objekty. Není doporučeno používat ji jako jedinnou charakteritiku. Mohlo by to vést k chybným závěrům o kvalitě systému.Je to tedy poměr správně negativních vzorků oproti všech negativně označených vzorků.

Přestnost (Precision)

$$PREC = \frac{TP}{TP + FP} \tag{5.5}$$

Přestnost je charakteristika určující jak klasifikátor správně klasifikoval hledané objekty. Není doporučeno používat ji jako jedinnou charakteritiku. Mohlo by to vést k chybným

závěrům o kvalitě systému. Je to tedy poměr správně pozitivních vzorků oproti všech pozitivně označených vzorků.

Efektivita (Efficiency)

$$EFF = \frac{SENS + SPEC}{2} \tag{5.6}$$

Efektivita reprezentuje průměr mezi senzivitou a specifitou a výsledek vrací v procentech. Optimální hodnota je 100%, ale toho se zřítka dosáhne.

F-míra (F-measure)

$$F = \frac{2 * presnost * uplnost}{2 * presnost + uplnost} = \frac{2TP}{2TP + FP + FN} \tag{5.7}$$

F-míra je sourhná charakteristika přestnosti klasifikace. Nejlepší hodnota vyhodnocení dosahuje 1 a nejhorší 0.

5.2 ROC křivka

Pracovní charakteristika přijímače (**R**eceiver **O**perating **C**haracteristic) je užitečný visuální nástroj pro porovnání dvou modelů. ROC křivka vznikla během druhé světové války pro analyzování radarových snímků. Tato křivka dává do souvislosti vztah mezi specificitou 5.4 a senzitivitou 5.3 pro všechny hodnoty prahu θ . Tedy křivka dává do poměru správně klasifikované pozitivní vzorky a chybně klasifikované negativní vzorky jako pozitivní.

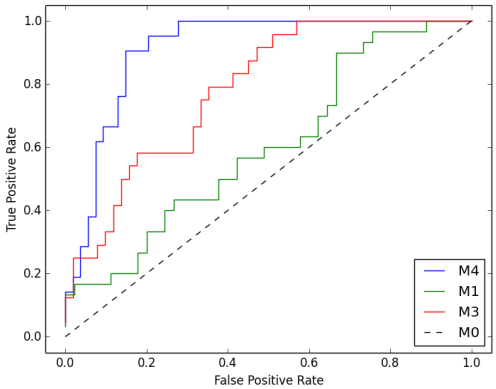
Graf ROC křivky

Grafem ROC křivky je dvourozměrný graf, kde na ose x je zobrazena pravděpodobnost chybné klasifikace negativních vzorků a na ose y pravděpodobnost správné klasifikace pozitivních vzorků. Pro každou hodnotu prahu θ, lze získat jeden bod na ROC křivce. Pro realizaci Roc křivky jsem výcházel z [9].

Na obrázku 5.1 je znázorněna ROC křivka se čtyřmi průběhy. Kde M0 zobrazuje průběh náhodného prediktoru. Bude-li průběh pod touto diagonálou tak je to nevyhovující stav a řešením je převrátit výsledky klasifikátoru. Z grafu je patrné, že ROC křivka vždy prochází počátečním bodem (0,0) a koncovým bodem (1,1). Průběh křivky je neklesající. Průběh M3 je považován za normální a bude-li se průběh křivky blížit průběhu M4, jde o ideální stav. Kdy klasifikátor zařadil většinu vzorků správně do tříd. Ideální křivka začíná v bode (0,0), pokračuje do bodu (0,1) a následně do bodu (1,1).

velikost AUC	hodnocení
0.5 - 0.6	nedostatečně
0.6 - 0.7	dostatečně
0.7 - 0.8	dobře
0.8 - 0.9	velmi dobře
0,9 - 1.0	výborně

Tabulka 5.2: Tabulka hodnot AUC



Obrázek 5.1: ROC křivka

Plocha pod ROC křivkou - AUC

Plocha pod ROC křivkou (**A**rea **U**nder the ROC **C**urve) slouží k porovnání ROC křivek a to buť vizuálně a nebo početně. Hodnota bude vždy mezi $< 0; 1 >$. Z poznatku o náhodném prediktoru, který nebude klesat pod 0,5 , bude výsledná hodnota mezi $0.5 - 1$. AUC vyjadřuje kromě plochy pod křivkou také míru uspořádanosti vzorků.

Na tabulce 5.2 je zobrazení ohodnocení testu podle plochy pod křivkou:

Jedným z možných použití AUC je pravděpodobnost jak vybraný vzorek byl klasifikovaný do tříd.

Implementace ROC křivky

Ve své práci jsem implementoval vlastní ROC křivku zdůvodu, že dostupná existující řešení podporovaly jenom binární klasifikátor a ne klasifikátor do více tříd. Nejprve ve své práci jsem používal script na vytvoření ROC křivky od autorů knihovny LibSvm [odkaz]. Který byl implementovaný v jazyce Python a při každém spuštení docházelo k trénování a klasifikování a až pak následně na základě získaných vlastních hodnot se vytvořila ROC křivka. To mě později přestalo vyhovovat a nazákladě toho scriptu jsem si vytvořil vlastní script *binary_roc.py* pro binární klasifikátor, který už neprováděl trénování a klasifikaci a využívá už klasifikované hodnoty. Script očekává model, který byl použit na testování a soubor, který obsahuje klasifikované hodnoty. Nejprve se získají hodnoty označující třídy z testovaného modelu a následně se načtou klasifikované třídy. Získané hodnoty jsou pak předané funkci *plot_roc()*, která pro vykreslení křivky provede kroky:

- Získá se počet P pozitivních tříd a počet N všech negativních tříd
- Do pomocného pole se uloží testované a klasifikované vzorky. Pole je pak setříděno setupně podle klasifikovaných hodnot
- Setříděné pole se prochází a počítá se matice záměn, která je popsána v kapitole 5.1, pro hodnoty TP a TN. Výpočet senzivity je ukládan do pole $tpr = TP/P$ a výpočet specifity je ukládán do pole $fpr = TN/N$.
- Výpočet plochy pod křivkou
- Vykreslení křivky na základě hodnot v tpr a fpr

Takto popsaný algoritmus není efektivní a používal jsem jej ze začátku. Algoritmus jsem upravil, že už není potřeba počítat počet všech pozitivních i negativních tříd v testovacím modelu a při počítaní matice záměn jsem zahrnul i výpočet pro hodnoty FP a FN. Na základě získaných hodnot TP, TN, FP a FN je vždy pro každý krok spočítaná senzivita podle rovnice 5.3 a specifita podle rovnice 5.4.

Další výhodou ROC křivky, kromě zobrazení kvality natrénovaného modelu klasifikátoru, je že pomocí prahu θ se může ovlivňovat citlivost klasifikátoru a zobrazit jeho ROC křivka, aniž by se musel znovu natrénovat. Pomocí prahu θ rozhodování, se budou měnit hodnoty TP, TN, FP a FN a tím pádem i ROC křivka. Na základě toho poznatku, jsem upravil výstup klasifikátoru, aby místo tříd zařazení vracel míru zařazení mezi 0 až 1. Kde hodnota 1 značí, že daný vzorek patří do konkrétní třídy. Následně byl upravený script, aby ty to hodnoty načetl. Proces vykreslení křivky je stejný jako v předešlém kroku.

Pro klasifikátor do více tříd je implementovaná ROC křivka v samostatném scriptu *multiclass_roc.py*. Výpočet křivky se provádí stejným způsobem jako pro binární klasifikátor, ale v tom to případě se vykreslí n křivek podle počtu tříd. Script při načítaní hodnot si uloží hodnoty tříd a pak jsou následně procházeny postupně. Zkoumaná třída se vždy označí za kladnou a ostaní třídy na záporné, přičemž hodnoty tříd jim zůstanou. Takto upravé hodnoty tříd jsou ukládány do pole, které je setřítěno a provede se stejné vyhodnocení ROC křivky jako v předešlém kroku.

Ve své práci využívám ROC křivky s prahem pro vyhodnocení natrénovaného modelu a pak ROC křivky se stupem s klasikovanou třídou z výstupu kaskády klasifikátorů.

5.3 Vyhodnocení

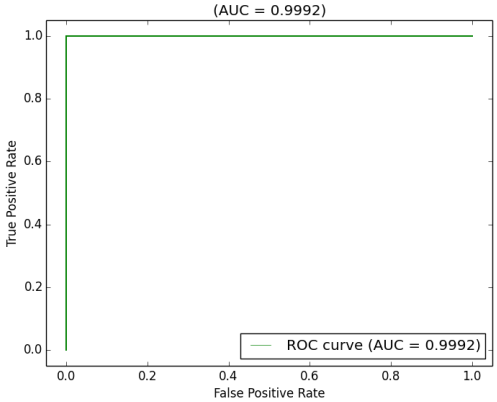
Tato část kapitoly se bude zabývat vyhodnocením modelů vytvořených ze třech datových sad německé, belgické a evropské. Výsledky jsou měřeny pro tři klasifikátory. Kde *klasifikátor1* značí binární klasifikátor, který hledaný objekt předává dál dalšímu klasifikátoru. *Klasifikátor2* klasifikuje předaný objekt od binárního klasifikátoru do tříd. Je výhradně vytvořený jenom ze značek. Jelikož binární klasifikátor není dokonalý, tak *klasifikátor3* je tvořený třídami dopravních značek a jednou třídou s pozadím. Jeho úkolem je falešné pozitvní objekt klasifikovat jako pozadí. Výslední hodnoty klasifikátoru 1 a 2 jsou dány průměrem všech tříd

Trénování a vyhodnocení bylo provedeno na notebooku Lenovo E530 s procesorem Intel Pentium B970 @ 2.30GHz a pamětí 8GB.

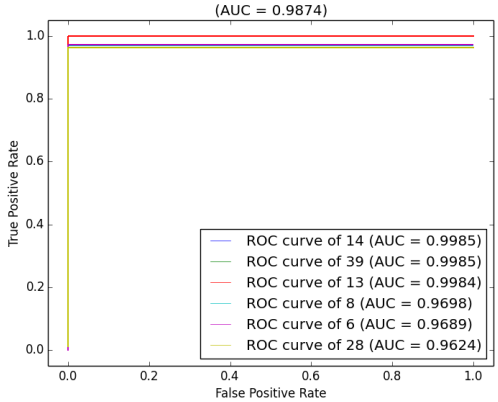
Vyhodnocení modelu německých dopravních značek

Pro natrénování modelu neměckých dopravních značek byla použita datová sada 2.3. pro binární *klasifikátor1* je využita trénovací sada přibližně o 40 000 snímcích, kterým je přidána negativní datová sada o 40 000 snímcích. *Klasifikátor2* do více tříd je natrénovaný na značkách a stejném počtu rozdělených do 43 tříd. A v případě *klasifikátoru3* model natrénovaný stejně jako předchozí a přidaná jedna třída s počtem 3 000 snímků okol9. Všechny natrénované modely do více tříd byly natrénované s parametrem *-c 10* a *-g 0.01* a binární klasifikátor s parametrem *-c 14* a *-g 0.015625*.

Na obrázku 5.2a je znázorněná ROC křivka binárního klasifikátoru a na obrázku 5.2b je ROC křivka *klasifikátoru2* do více tříd. Kde jsou zobrazeny 3 nejlepší průběhy a 3 nejhorší průběhy křivek.



(a) Klasifikátor1



(b) Klasifikátor2

Obrázek 5.2: ROC křivka trénovacího souboru vytvořeného křížovou validací

V tabulce 5.4 jsou zobrazeny výsledky klasifikace na testovacím modelu vytvořený křížovou validací:

Počet	*Klasifikátor 1*	*Klasifikátor 2*	*Klasifikátor 3*
26 442	99,91%	98,91 %	% 98,90

Tabulka 5.3: Úspěšnost na testovacím souboru vytvořený pomocí křížové validace.

Součástí datové sady je i testovací sada, která obsahuje přibližně 12 600 souborů značek, ke které jsem přidal přibližně 20 000 snímků z okolí. Takto vzniká testovací sada pak byla klasifikavána. Na následující tabulce jsou uvedeny dostupné výsledky.

Počet	*Klasifikátor 1*	*Klasifikátor 2*	*Klasifikátor 3*
32 608	99,97%	99,12 %	99,18 %

Tabulka 5.4: Úspěšnost na testovacím souboru vytvořený pomocí křížové validace.

Vyhodnocení modelu belgických dopravních značek

Pro vyhodnocení modelu belgických dopravních značek byla použita datová sada 2.3. *Klasifikátor1* je složen z přibližně 4 500 snímků dopravních značek pro trénování a z přibližně 5 000 snímků pozadí. *Klasifikátor2* je natrénovaný jenom pomocí značek rozdělených do 62 tříd. *Klasifikátor3* je natrénovaný stejně jako *klasifikátor2*, ale je zde přidaná jedna třída navíc s počtem 1 000 snímků pozadí. Všechny natrénované modely byly natrénované s parametrem *-c 10* a *-g 0.001*.

Na tabulce 5.5 jsou zobrazeny výsledky klasifikace na testovacím modelu vytvořený křížovou validací:

Součástí datové sady je i přibližně 2 500 snímků pro trénování, které jsem přidal přibližně 3 000 snímků pro pozadí. Klasifikované hodnoty jsem porovnával s anotacemi a výsledky testovaní jsou zobrazeny v následující tabulce:

Počet	*Klasifikátor 1*	*Klasifikátor 2*	*Klasifikátor 3*
3 193	99,47%	85,70 %	85,88 %

Tabulka 5.5: Úspěšnost na testovacím souboru vytvořený pomocí křížové validace.

Počet	*Klasifikátor 1*	*Klasifikátor 2*	*Klasifikátor 3*
5 526	99,76%	77,26%	77,26získaných%

Vyhodnocení modelu evropských dopravních značek

Pro vyhodnocení modelu evropských dopravních značek byla použita datová sada 2.3. Pro trénování binárního *klasifikátoru1* byla použita datová sada značek přibližně o 2 300 souborů, kterým jsem přidal přibližně 3 000 snímků s okolím. Všechny natrénované modely byly natrénované s parametrem *-c 10*.

Bohužel součástí není testovací sada a následující tabulka zobrazuje výsledky testovacího souboru vytvořený křížovou validací.

Počet	*Klasifikátor 1*	*Klasifikátor 2*
1 777	99,34%	69,69%

Vyhodnocení výsledků

Z naměřených hodnot pomocí *F-measure* je patrné, že přesnost klasifikace je ovlivňována velikostí datové sady. Natrénovaný model pro německé dopravní značky 5.3 se složením 40 000 snímků pozitvních objektů a 40 000 negativních snímků dosahoval úspěšnosti až 99,12 %. Oproti tomu natrénovaný model pro evropské dopravní značky s datovouo sadou 40x menší dosahoval úspěšnosti ani ne 70 %. Všechny binární klasifikátory mají vysokou úspěšnost, to je dáno větší datovou sadou při trénovaní. Naproti tomu klasifikátory do tříd pro belgické a evropske značky mají horší výsledky, neboť u belgických značek je v průměru 75 značek na třídu a u evropských značek 22 snímků na třídu. Na základě výsledků je to nedostatečné. Model pro nemecke dopravni znacky obsahuje 918 snímků na třídu.

Dále na základě výsledků lze vypozorovat, že *klasifikátor2* spávně klasifikoval vzorky, které binární klasifikátor označil chybně jako falešné pozitivní.

5.4 Experimenty

V této části kapitoly budou provedeny a popsány experimenty.

Experiment č.1 - Klasifikace jedinným modelem

Cílem toho experimentu je ověření,že klasifikace pomocí kaskády klasifikátrů dosahuje lepších výsledků než řešení implementované v jednom modelu. Model je složený z datového setu německých dopravních značek 2.3 přibližně o 40 000 snímcích , kterému je ještě přidá jedna třída z negativních snímků o velikosti přibližně 40 000. Model je natrénovaný pomocí parametru *-c 10* a *-g 0.001*.

Na tabulce 5.6 jsou zobrazeny výsledky klasifikace na testovacím modelu vytvořený křížovou validací:

Počet	*Klasifikátor 1*
26 442	96,70%

Tabulka 5.6: Úspěšnost na testovacím souboru vytvořený pomocí křížové validace.

Očekávánány vysledek je mírně horší než při použití kaskády klasifikátorů 5.4. Horší výsledek je získám tím, že v datové sadě dopravních značek jsou snímky rozmazené, pořízené za šera, v mlze. A binární klasifikátor ma jednu třídu komplet sestavým z naček a tak se na nich může lépe naučit než teto model.

Experiment č.2 - Lineární klasifikátor

Cílem toho experimentu je ověření výhod použitého RBF kernelu místo lineárního kernelu. Opět je použita datová sada s německými dopravními značkami 2.3. Pro vyhodncení je použit model pro klasifikátor do více tříd. Na tabulce 5.7 jsou zobrazeny výsledky klasifikace na testovacím modelu vytvořený křížovou validací:

Počet	*Klasifikátor 1*
13 069	98,26%

Tabulka 5.7: Úspěšnost na testovacím souboru vytvořený pomocí křížové validace.

S porování s tabulkou 5.4 jsou získané hodnoty pomocí lineární kernelu mírně horší.

Kapitola 6

Závěr

Cílem bakalářské práce je navrhout a implementovat systém pro klasifikaci dopravních značek. Tento úkol se mě podařilo splňit. Jak je vidět v tabulce 5.4, systém dosahuje přesnosti 99% pro danou datovou sadu německých dopravních značek. Pořídil jsem si datové sady pro trénování kaskády klasifikátorů. Implementoval jsem scripty pro vytvoření modelů a podpůrné nástroje pro vyhodnocení natrénvaných modelů.

Během implementace jsem prostudoval dostupné řešení detekce dopravních značek a nazákladě toho jsem vytvořil vlastní systém pro klasifikaci. Systém umí klasifikovat dopravní značku z výřezu. Vytvořil jsem si vlastní datovou sadu a nad získanými datovými sadamy bylo provedeno vyhodnocení systému.

Možným rozšířením práce by mohla být implementace extrace příznaků. Nebo implementace detektoru a mít tak možnost detekovat a klasifikovat dopravní značky v reálném čase. Dalším možným řešením může být rozšíření datových sad a klasifikovat tak přesněji nebo i nové typy značek. Dalším možným rozšířením může být použítí CUDA nebo OpenCL pro paralerní výpočty, pomocí kterých by bylo urychlěno trénování a klasifikace.

Literatura

[1] Benenson, R.; Omran, M.; Hosang, J.; aj.: Ten years of pedestrian detection, what have we learned? In *ECCV, CVRSUAD workshop*, 2014.

[2] Berka, P.: *Dobývání znalostí z databází*. Praha: Academia, vyd. 1. vydání, 2003, ISBN 8020010629.

[3] Chang, C.-C.; Lin, C.-J.: A Library for Support Vector Machines. 2001, dostupné na http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[4] Chang, C.-C.; Lin, C.-J.: LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, ročník 2, 2011: s. 27:1–27:27, software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[5] Dobeš, M.: *Zpracování obrazu a algoritmy v C#*. Praha: BEN - technická literatura, první vydání, 2008, ISBN 9788073002336.

[6] Duda, R. O.; Stork, D. G.; Hart, P. E.: *Pattern classification*. New York, N.Y: John Wiley & Sons, druhé vydání, c2001, ISBN 0471056693.

[7] Dušek, J.: *Detekce dopravních značek z kamery ve vozidle*. Bakalářská práce, FIT VUT v Brně, 2015.

[8] de la Escalera, A.; Armingol, J.; Mata, M.: Traffic sign recognition and analysis for intelligent vehicles. *Universidad Carlos III de Madrid*.
URL <http://orff.uc3m.es/bitstream/handle/10016/7089/traffic_escalera_IVC_2003_ps.pdf?sequence=1>

[9] Fawcett, T.: ROC Graphs. 2004.
URL http://home.comcast.net/~tom.fawcett/public_html/papers/ROC101.pdf

[10] Han, J.; Kamber, M.; Pei, J.: *Data mining*. Boston: Elsevier, třetí vydání, c2012, ISBN 9780123814791.

[11] Hofmann, T.; Schölkopf, B.; Smola, A. J.: Kernel methods in machine learning: s. 1171–1220.

[12] Houben, S.; Stallkamp, J.; Salmen, J.; aj.: Detection of Traffic Signs in Real-World Images: The German Traffic Sign Detection Benchmark. In *International Joint Conference on Neural Networks*, 1288, 2013.

[13] Ivanciuc, O.: Applications of Support Vector Machines in Chemistry. *Applications of Support Vector Machines in Chemistry*, ročník 23, 2007: s. 291–400.

URL
http://www.ivanciuc.org/Files/Reprint/Ivanciuc_SVM_CCR_2007_23_291.pdf

[14] Moutarde, F.; Bargeton, A.; Herbin, A.; aj.: Modular traffic signs recognition aplied to on-vehicle real-time visual detection of americian and european speed limit signs. URL <http://arxiv.org/pdf/0910.1295.pdf>

[15] Ruta, A.; Li, Y.; Liu, X.: Real-time traffic sign recognition from video by class-specific discriminative features. *Elsevier*, 2009: s. 416–430. URL <http://people.brunel.ac.uk/~csstyyl/papers/pr2010.pdf>

[16] Shojania, H.: Real-time traffic sign detection. 2003, dostupné z http://hassan.shojania.com/pdf/TrafficSignDetection-Paper.pdf. URL <http://hassan.shojania.com/pdf/TrafficSignDetection-Paper.pdf>

[17] SONKA, M.; HLAVAC, V.; BOYLE, R.: *Image processing, analysis, and machine vision*. New York: PWS Publishing, druhé vydání, 1999, ISBN 0-534-95393-X.

[18] Stallkamp, J.; Schlipsing, M.; Salmen, J.; aj.: The German Traffic Sign Recognition Benchmark: A multi-class classification competition. In *IEEE International Joint Conference on Neural Networks*, 2011, s. 1453–1460.

[19] Timofte, R.; Zimmermann, K.; Van Gool, L.: Journal of Machine Vision and Applications (MVA 2011). In *Multi-view traffic sign detection, recognition and 3D localisation*, December 2011.

[20] Viola, P.; Jones, M. J.: Robust Real-Time Face Detection. 2004. URL http://www.vision.caltech.edu/html-files/EE148-2005-Spring/pprs/viola04ijcv.pdf

[21] ŽÁRA, J.; BENEŠ, B.; SOCHOR, J.; aj.: *Moderní počítačová grafika*. Brno: Computer Press, druhé vydání, 2004, ISBN 80-251-0454-0.

Příloha A

Obsah DVD

K mé práci je přiloženo jedno DVD, které obsahuje zdrojové kódy, plakát, zprávu a datové sady. Adresářová struktura přiloženého DVD:

- Aplikace
  - bin - přeložený projekt
  - src - zdrojové soubory
  - scripty - vytvořené modelů pro trénovaní, vyhodnocení natrénovaných modelů, práce s datovými sadami
- DataSet - obsahuje obsahuje ziskane a vmé práci vytvořené sady
- Modely - natrénované modely, ktré se mohou použít pro klasifikaci
- Plakat - obsahuje plakát
- Zpráva

Příloha B

Plakát

DETEKCE DOPRAVNÍCH ZNAČEK

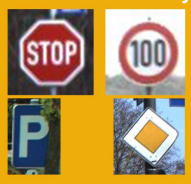
AUTOR: MICHAL JURČA VEDOUCÍ: Ing. VÍTĚZSLAV BERAN, Phd.

Cíl: klasifikace dopravních značek na základě dostupných datových sad.

Datové sady

Vlastní negativní sada obsahující cca 60 tisíc snímků okolí bez značek.

Německé značky



Pro každý typ značky:
cca 918 snímků

Úspěšnost klasifikace: 98,91 %

43 typů značek

Evropské značky

Pro každý typ značky:
cca 22 snímků

Úspěšnost klasifikace: 77,26 %

106 typů značek

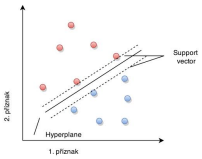
Belgické značky

Pro každý typ značky:
cca 75 snímků

Úspěšnost klasifikace: 85,70 %

62 typů značek

Klasifikace



Support vector machine
Kaskáda klasifikátorů
LibSVM