VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV INTELIGENTNÍCH SYSTÉMŮ
DEPARTMENT OF INTELLIGENT SYSTEMS

ROZPOZNÁVÁNÍ CAPTCHA KÓDŮ

CAPTCHA CODE RECOGNITION

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **RADEK PAZDERKA**
AUTHOR

VEDOUCÍ PRÁCE **Doc. Ing. FRANTIŠEK ZBOŘIL, CSc.**
SUPERVISOR

BRNO 2017

Zadání bakalářské práce/19260/2016/xpazde14

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav inteligentních systémů Akademický rok 2016/2017

Zadání bakalářské práce

Řešitel: **Pazderka Radek**
Obor: Informační technologie
Téma: **Rozpoznávání CAPTCHA kódů**
Captcha Code Recognition
Kategorie: Umělá inteligence

Pokyny:

1. Seznamte se se způsoby rozpoznávání znaků v CAPTCHA kódech.
2. Navrhněte program, který ze zadané webové stránky stáhne všechny obrázky standardních formátů.
3. Z těchto obrázků shromážděte dvě sady obrázků s CAPTCHA kódy (tréninkovou a testovací).
4. Navrhněte a implementujte program pro rozpoznání zvoleného CAPTCHA kódu z tréninkové sady.
5. Otestujte tento program na rozpoznání CAPTCHA kódů na obrázcích z testovací sady a zhodnoťte získané výsledky.

Literatura:

- The Official CAPTCHA Site: http://www.captcha.net/
- Szeliski, R.: Algorithms and Applications, Springer, 2010

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

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

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

Vedoucí: **Zbořil František V., doc. Ing., CSc.**, UITS FIT VUT
Datum zadání: 1. listopadu 2016
Datum odevzdání: 17. května 2017

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

L.S.

doc. Dr. Ing. Petr Hanáček
vedoucí ústavu

Abstrakt

Tato bakalářská práce se zaměřuje na návrh a implementaci aplikace sloužící k rozpoznávání textových CAPTCHA kódů. Popisuje algoritmy pro předzpracování obrazu, segmentaci a následnou klasifikaci znaků. Pro klasifikaci byly použity dva různé přístupy. Jedná se o konvoluční neuronovou síť LeNet a histogramový klasifikátor využívající Pearsonova korelačního koeficientu. Následně se zvolené klasifikátory testují na různých CAPTCHA kódech a zjišťuje se výsledná úspěšnost při rozpoznávání.

Abstract

This bachelor thesis is dedicated to design and implementation of application, which's purpose is to recognize text CAPTCHA codes. It describes image processing algorithms, segmentation algorithms and character classification. Two different aproaches were used for classification. Convolution neural network LeNet and histogram classificator, which uses Pearson's correlation coefficient. Chosen classificators were tested on different CAPTCHA codes while finding out the success rate of recognition.

Klíčová slova

CAPTCHA kód, strojové učení, konvoluční neuronová síť, LeNet, Caffe, kaskádový klasifikátor, histogramový klasifikátor

Keywords

CAPTCHA code, machine learning, convolution neural network, Lenet, Caffe, cascade classifier, histogram classifier

Citace

PAZDERKA, Radek. *Rozpoznávání CAPTCHA kódů*. Brno, 2017. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Zbořil František.

Rozpoznávání CAPTCHA kódů

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Doc. Ing. Františka Zbořila, CSc. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Radek Pazderka
11. května 2017

Poděkování

Rád bych poděkoval mému vedoucímu panu Doc. Ing. Františkovi Zbořilovi, CSc. za jeho odbornou pomoc a vedení mé bakalářské práce.

Obsah

Úvod

Každý, kdo chce v dnešní době vytvořit webovou stránku, buď pro sebe, nebo pro někoho na zakázku, měl by dbát na její bezpečnost. Hrozbu představují nelegální programy, které se na webové stránce vydávají za běžného uživatele, ale na rozdíl od něho odesílají větší počet a libovolný typ požadavků na webovou stránku. Webová stránka zpravidla slepě odpovídá na každý požadavek, který jí přijde, a nezjistí, že se nejedná o člověka, ale o nelegální program. Webový útočník, který vytvoří tento program, se zajímá pouze o profit nebo získání důležitých informací o systému a tím celou webovou stránku může znehodnotit.

Jako obrana proti těmto webovým zločincům vznikl CAPTCHA kód [7]. Hlavní myšlenka CAPTCHA kódů byla rozlišit, zda se jedná o reálného uživatele nebo o program, který se za reálného uživatele pouze vydává. Ochrana pomocí CAPTCHA kódů spočívá v tom, že je po uživateli vyžadován přepis textu z obrázku nebo odpověď na jednoduchou otázku, popř. identifikace předmětu v obrázku. Na tento „triviální“ úkol člověk dokáže bez problémů odpovědět, ale pro program je tento úkol značně obtížnější. Postupem času se ale zvyšuje schopnost programů rozpoznávat větší množství druhů CAPTCHA kódů, proto je pro CAPTCHA kód důležité stále přicházet s novými bezpečnostními prvky, pomocí kterých se CAPTCHA kód stane na určitý čas pro programy nerozpoznatelný.

V této bakalářské práci se zaměříme na dva možné způsoby rozpoznání CAPTCHA kódů. První způsob rozpoznání je založen na podobnosti histogramů jednotlivých znaků. Druhý způsob je založen na klasifikaci znaků pomocí konvolučních neuronových sítí.

Protože existuje mnoho druhů textových CAPTCHA kódů a není možné získat všechny jejich datové sady, je nutné vytvořit aplikaci, která si natrénuje znaky z daných CAPTCHA kódů umístěných na webové stránce a následně je dokáže klasifikovat. Pro splnění tohoto účelu jsem vytvořil histogramový klasifikátor, který tento problém řeší.

Druhý způsob rozpoznávání textových CAPTCHA kódů je založen na klasifikaci znaků pomocí konvolučních neuronových sítí. Jedná se o složitější postup trénování, než je u histogramového klasifikátoru, ale úspěšnost správného rozpoznávání u tohoto klasifikátoru je značně vyšší.

V rámci této bakalářské práce byla vytvořena aplikace s grafickým uživatelským rozhraním, která podporuje oba způsoby rozpoznání.

Tato bakalářská práce je rozdělena do desíti kapitol. V první kapitole si uvedeme základní informace o CAPTCHA kódech. Popíšeme si jeho druhy a bezpečnostní prvky, které by po přidání do CAPTCHA kódu tvořily problém při rozpoznávání. Ve druhé kapitole se seznámíme s algoritmy sloužící k předzpracování obrazu CAPTCHA kódu. Jsou zde uvedeny algoritmy, které slouží k odstranění některých bezpečnostních prvků z CAPTCHA kódu. Ve třetí kapitole se seznámíme se segmentačními algoritmy, které mají za úkol oddělit jednotlivé znaky v CAPTCHA kódech. Ve čtvrté kapitole si uvedeme klasifikační algoritmy, které budou jednotlivé znaky zařazovat do tříd podle významu. Pátá kapitola se věnuje popisu datových sad, které jsou použité jako zdroj dat pro klasifikační algoritmy. V šesté

kapitole se seznámíme s návrhem aplikace, která se bude implementovat. Sedmá kapitola popisuje postup trénování jednotlivých klasifikátorů. V osmé kapitole si uvedeme popis implementace aplikace. V deváté kapitole otestujeme segmentační a klasifikační algoritmy použité v aplikaci na reálných CAPTCHA kódech a zhodnotíme výsledky testování. V závěrečné desáté kapitole zrekapitulujeme všechny výsledky, kterých bylo dosaženo v průběhu tvorby celé bakalářské práce.

Kapitola 1

CAPTCHA kód

V této kapitole se zaměříme na popis CAPTCHA kódů. Důkladněji si popíšeme textové CAPTCHA kódy a jejich bezpečnost, která hraje velkou roli při rozpoznávání. Uvedeme si také další druhy CAPTCHA kódů, které se v dnešní době používají. V závěru kapitoly se seznámíme s existujícími metodami, pomocí kterých lze CAPTCHA kódy rozpoznávat.

1.1 Co je to CAPTCHA kód

CAPTCHA neboli „Completely Automated Public Turing test to tell Computers and Humans Apart“ je Turingův test. Tento test se snaží odlišit člověka a robota na webových stránkách [6]. CAPTCHA kód byl vytvořen v roce 2000 týmem programátorů na Carnegie Mellon University. První CAPTCHA kód vznikl jako forma obrany proti automatickému vkládání URL do katalogu AltaVista. Dříve CAPTCHA kódy používaly přímé vykreslení textu do obrázku bez jakékoliv deformace písma, které se ale webovým útočníkům velmi jednoduše podařilo překonat. Musely se tedy začít využívat metody, které znesnadnily detekci znaků pro programy rozpoznávající CAPTCHA kódy, dále pouze „programy“, ale zachovaly čitelnost pro člověka. Mezi tyto metody patří např. deformace textu nebo částečné překrytí jednotlivých znaků. CAPTCHA kód může představovat problém pro zrakově postižené. Tento hendikep lze zmírnit využitím audio CAPTCHA kódu, který text přehraje.

CAPTCHA kód se musí stále zlepšovat, kvůli neustále se zdokonalujícím programům. Z důvodu zhoršující se schopnosti lidí přečíst deformované texty, se objevuje mnoho modifikací CAPTCHA kódu. Je tedy možné, že se místo opisování textu setkáme s rozpoznáním obrázků nebo např. s 3D CAPTCHA kódy.

1.2 Textové CAPTCHA kódy

Jedná se o základní druh CAPTCHA kódů. Po uživateli je požadováno opsání libovolně deformovaného textu. Tento druh je v dnešní době nejčastěji používaný na webových stránkách. Na Obrázku 1.1 je uveden textový CAPTCHA kód se vstupním polem, do kterého se opisuje text z obrázku.

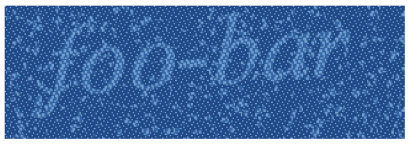
Obrázek 1.1: Základní CAPTCHA kód.

1.2.1 Bezpečnost textových CAPTCHA kódů

Pro programy je nejobtížnější segmentace jednotlivých znaků. Pokud CAPTCHA kód obsahuje mnoho bezpečnostních prvků, popsaných níže, programy ztrácí úspěšnost správného rozpoznání.

- **Pozadí textu**

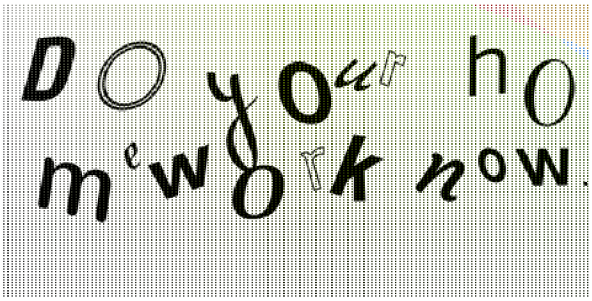
  Pokud CAPTCHA kód obsahuje text odstínově podobný svému pozadí, je pro programy obtížné určit polohu jednotlivých písmen. Lehko rozpoznatelné CAPTCHA kódy obsahují statické pozadí, které programy dokážou jednoduše odstranit. Na Obrázku 1.2 je příklad obtížně rozpoznatelného CAPTCHA kódu z důvodu zvoleného pozadí textu.



Obrázek 1.2: CAPTCHA kód s vhodným pozadím textu.

- **Množina písmen**

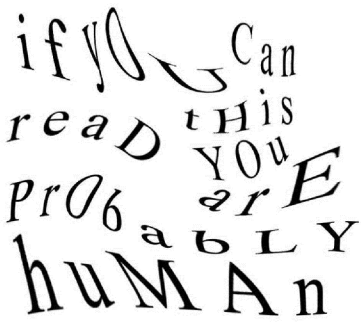
  Hrozbu pro programy představují CAPTCHA kódy, které obsahují všechna písmena, číslice, popř. i speciální znaky v různých variantách. Programy si pak musí natrénovat větší datovou sadu znaků a tím se může snížit výsledná úspěšnost rozpoznání. Lehce rozpoznatelné CAPTCHA kódy většinou neobsahují všechna písmena abecedy. Například obsahují pouze velká písmena a jen část abecedy. Některé mají i redukovanou řadu číslic. V případě nedostačujícího počtu typů znaků se velice usnadní práce programům při rozpoznání daného CAPTCHA kódu. Na Obrázku 1.3 je ukázka CAPTCHA kódu, který obsahuje velkou množinu písmen.



Obrázek 1.3: CAPTCHA kód s vhodnou množinou písmen.

- **Deformace**

  Vhodná deformace textu ztíží práci programu při klasifikaci znaků z CAPTCHA kódu. Deformují se hrany nebo i celý tvar znaku. V případě, že stejný znak po deformaci vypadá vždy jinak, způsobí velké snížení úspěšnosti rozpoznání výsledného CAPTCHA kódu. Na Obrázku 1.4 je uveden CAPTCHA kód s deformovaným textem.



Obrázek 1.4: CAPTCHA kód s deformací písmen.

- **Náhodné pootočení**

  Dalším prvkem snižujícím úspěšnost rozpoznání pro programy je rotace písmen. Tento prvek není příliš zásadní, protože programy dokážou jednoduchým algoritmem vyseparované znaky otočit zpět do správné pozice. Algoritmus vychází z toho, že alfanumerické znaky jsou správně postaveny v případě, když mají nejmenší možnou šířku.

- **Barva a velikost**

  Při změně barevné intenzity každého znaku se znemožní programům jednoduchý převod každého znaku na černý s bílým pozadím. Zavedením libovolné velikosti jednotlivých písmen se také zvýší nerozpoznatelnost daného CAPTCHA kódu. Programy pak musí u znaků modifikovat velikost a tím ztrácí kvalitu znaků i úspěšnost celého rozpoznání.

- **Překrývající se písmena**

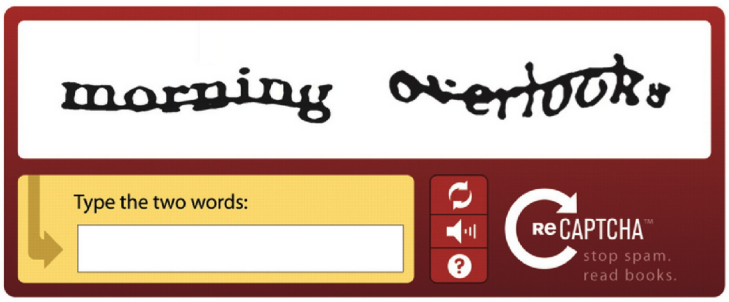
  Překrývání znaků v CAPTCHA kódu patří mezi nejsilnější bezpečnostní prvek. Programům se tím znemožní jednoduchá segmentace znaků. Na Obrázku 1.5 je demonstrována postupně se zlepšující bezpečnost CAPTCHA kódu v podobě překrývajících se znaků.

Obrázek 1.5: CAPTCHA kódy s překrývajícími se znaky.

1.2.2 Příklady textových CAPTCHA kódů

- **reCAPTCHA**

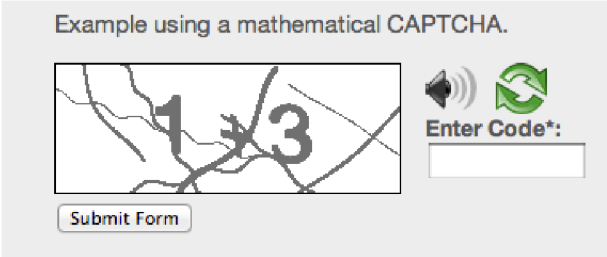
  Příkladem textového CAPTCHA kódu může být reCaptcha. Tento CAPTCHA kód sloužil k pomoci s digitalizací tištěných médií např. knih, časopisů atd. Nejprve se naskenoval text dokumentu a tento text se předal OCR [3] programu na analýzu. V případě, že OCR program nerozpoznal konkrétní slovo v textu, bylo toto slovo přidáno do CAPTCHA kódu společně se slovem, které bylo známé. Podle přepisu od uživatele se zjišťovalo, co obsahovalo neznámé slovo. Tento CAPTCHA kód se používal jako ochrana webových stránek a také jako pomocník při rozpoznávání nečitelných slov tištěných dokumentů. Na Obrázku 1.6 je uvedena ukázka reCaptcha kódu.



Obrázek 1.6: reCaptcha.

- **Řešení úloh**

  Tento druh CAPTCHA kódu vyžaduje nejen správné rozpoznání znaků, ale i logické vyřešení příkladu. Většina programů nemá implementované logické myšlení a schopnost řešení jednoduchých úkolů, což činí tento CAPTCHA kód obtížně rozpoznatelný. Mezi další varianty patří CAPTCHA kód, který vyžaduje odpověď na jednoduchou otázku: „Jaký je čtvrtý den v týdnu?“, „Kolik nohou má pes?“ atd. Obrázek 1.7 demonstruje CAPTCHA kód vyžadující logické uvažování.



Obrázek 1.7: CAPTCHA kód s matematickým příkladem.

- **Rozdělení CAPTCHA kódu pomocí CSS stylu**

  Tento CAPTCHA kód se vyznačuje tím, že zároveň zachovává čitelnost pro člověka a je obtížně rozpoznatelný programy. Principem je, že se CATPCHA kód rozdělí na dvě až tři části. Tyto části se uloží zvlášť do různých souborů a jednotlivé soubory se vloží do odlišných částí HTML kódu. Znovu se složí až pro uživatele. Program tak obdrží pouze jednotlivé části CAPTCHA kódu, ze kterých nezíská výsledný text. Po složení

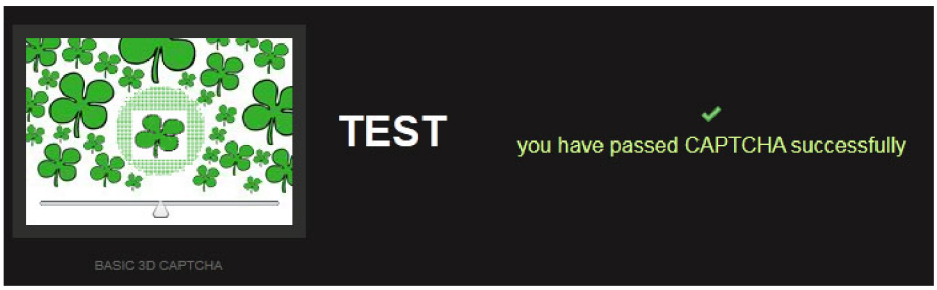
jednotlivých částí CAPTCHA kódu bude font i velikost písma pro uživatele čitelné, bez zbytečného přeškrtávání, naklánění nebo jiné deformace písma. CAPTCHA kód také nebude zabírat tolik prostoru na webové stránce, který se poté může využít pro další sdělování potřebnějších informací pro uživatele [22]. Na Obrázku 1.8 je uvedeno možné rozdělení daného CAPTCHA kódu.

Obrázek 1.8: CAPTCHA kód založený na CSS a HTML.

1.2.3 3D CAPTCHA kód

Tento CAPTCHA kód se skládá z Adobe flash [2] animace a z posuvného ukazatele. Při pohybu tímto ukazatelem se v určité části obrázku začnou otáčet trojúhelníky, na které byla tato část obrázku původně rozdělena. Uživatel má za úkol uvést posuvník do takové pozice, aby vznikl obrázek vizuálně podobný svému okolí. Při potvrzení se ověří, zda se uživatel nachází v toleranci. Tento inovativní druh CAPTCHA kódu je velice uživatelsky přívětivý.

Výhoda tohoto druhu CAPTCHA kódu spočívá v tom, že se může použít libovolný obrázek jako předloha. Programy si nebudou moci zaznamenat všechny možné kombinace obrázků. Tím se zvýší bezpečnost tohoto CAPTCHA kódu. Většina programů v dnešní době není vybavena mechanismem na stahování a následné spouštění Adobe flash aplikací z webových stránek, což zvyšuje nerozpoznatelnost tohoto CAPTCHA kódu. Na druhou stranu se od Adobe flash postupně ustupuje a nahrazuje se HTML5[1]. Časem se tento CAPTCHA kód může stát nepodporovaným na webových prohlížečích. Na Obrázku 1.9 je uvedena ukázka 3D CAPTCHA kódu.

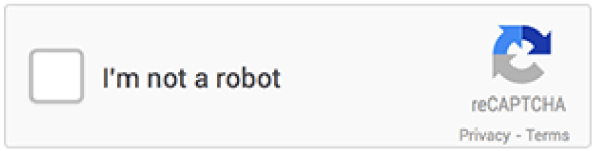


Obrázek 1.9: 3D CAPTCHA kód.

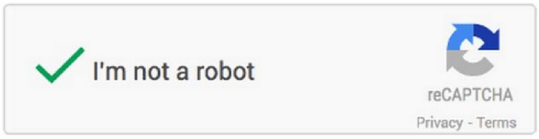
[1] Verze značkovacího jazyka HTML s multimediální podporou.

1.3 Google reCaptcha

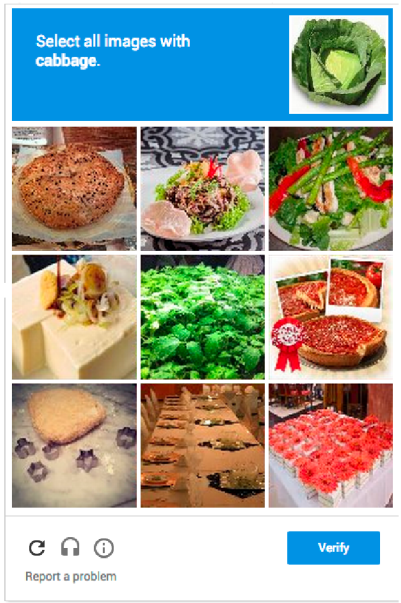
Služba reCaptcha od firmy Google je široce rozšířený CAPTCHA kód využívaný mnohými populárními webovými stránkami proti programům provádějící zločinné aktivity. Když uživatel navštíví webovou stránku chráněnou pomocí Google reCaptcha, uvidí jedno bílé zaškrtávací pole, s textem „I'm not a robot", Obrázek 1.10. Při zaškrtnutí políčka se spustí JavaScript, který sbíral informace o chování uživatele. Pokud prohlásí, že se jedná na 100 % o uživatele, povolí hned přístup a uživatel může pokračovat, viz Obrázek 1.11. V případě, že JavaScript nemá dostatek informací o aktivitě uživatele, požádá ho o identifikaci devíti obrázků, viz Obrázek 1.12. I na tento moderní druh CAPTCHA kódu již existuje metoda, která dokáže tento CAPTCHA kód rozpoznat. Stručný popis rozpoznání je uveden v Kapitole 1.4.2.



Obrázek 1.10: reCaptcha od firmy Google 1.



Obrázek 1.11: reCaptcha od firmy Google 2.



Obrázek 1.12: reCaptcha od firmy Google 3.

1.4 Existující přístupy pro rozpoznávání CAPTCHA kódů

1.4.1 Optical Character Recognition

Optical Character Recognition (OCR) slouží k převodu skenovaného textu do digitální podoby. Na základě této technologie vzniklo několik programů, které rozpoznávají CAPTCHA kódy. Jeden z nejlepších uváděných placených programů pro rozpoznávání CAPTCHA kódů je GSA Captcha Breaker, který umožňuje rozpoznání více než 600 druhů CAPTCHA kódů [1].

1.4.2 Rozpoznání Google reCaptcha

Rozpoznání tohoto CAPTCHA kódu spočívá v tom, že se jednotlivé obrázky vyhledají pomocí *images google*. Podle nalezených klíčových slov, názvů dvaceti nejlepších příspěvků a dalších informací, lze obrázek s velkou úspěšností správně klasifikovat. Pro zpracování informací nalezených pomocí *images google* a následné rozhodnutí, co se na obrázku nachází, se využívá neuronová síť [16].

Kapitola 2

Předzpracování obrazu

V této kapitole se seznámíme s algoritmy, které předpřipraví CAPTCHA kód pro jednodušší a přesnější segmentaci znaků. Popíšeme si základní algoritmus prahování a Otsu prahování, které využijeme v této bakalářské práci.

2.1 Algoritmus prahování

Jedná se o základní algoritmus segmentace obrazu založený na hodnocení jasu každého pixelu [11]. Základem algoritmu je hodnota prahu, podle které se určuje, na jakou barvu se má daný pixel nastavit. Vzorec (2.1) popisuje základní výpočet hodnoty jednoho bodu obrazu pomocí algoritmu prahování. Tento vzorec se aplikuje na všechny body daného obrazu.

$$g(x) = \begin{cases} A & \text{pokud } x \geq T \\ B & jinak \end{cases} \tag{2.1}$$

kde:

- x značí vstupní hodnotu intenzity šedé.
- A a B jsou nové hodnoty jasu pro vstupní hodnotu x. Většinou se volí jako 0 a 255, neboli černá a bílá barva bodu obrazu.
- T značí hodnotu prahu.
- $g(x)$ vrací výslednou barvu bodu obrazu pro vstupní hodnotu x.

2.2 Gaussovo rozostření

Aplikováním Gaussova filtru na obraz získáme obraz rozostřený. Toto rozostření se provádí pomocí Gaussovy funkce. Používá se pro snížení šumu a detailů v obraze. Gaussovo rozostření použijeme jako předzpracování obrazu před Otsu metodou prahování.

2.3 Otsu metoda prahování

Otsu metoda prahování převádí obraz v odstínu šedi na binární. Algoritmus hledá optimální hodnotu prahu, kterým oddělí popředí a pozadí obrazu. Použijeme ho k oddělení textu v CAPTCHA kódu a jeho pozadí. Při hledání prahu iteruje přes všechny možné hodnoty prahu. Při každé iteraci se obraz rozdělí do dvou tříd - popředí a pozadí obrazu a hledá hodnotu prahu, kde je vážený součet $\sigma_W^2(T)$ těchto tříd minimální.

$$\sigma_W^2(T) = W_b(T)\sigma_b^2(T) + W_f(T)\sigma_f^2(T) \tag{2.2}$$

kde:

$$W_b(T) = \sum_{i=0}^{T-1} P(i) \tag{2.3}$$

$$W_f(T) = \sum_{i=T}^{N-1} P(i) \tag{2.4}$$

$$\sigma_b^2(T) = \frac{\sum_{i=0}^{T-1}(i - \mu_b(T))^2 P(i)}{W_b(T)} \tag{2.5}$$

$$\sigma_f^2(T) = \frac{\sum_{i=T}^{N-1}(i - \mu_f(T))^2 P(i)}{W_f(T)} \tag{2.6}$$

$$\mu_b(T) = \sum_{i=0}^{T-1} \frac{iP(i)}{W_b(T)} \tag{2.7}$$

$$\mu_f(T) = \sum_{i=T}^{N-1} \frac{iP(i)}{W_f(T)} \tag{2.8}$$

- T značí hodnotu prahu.
- N celkový počet intenzit šedé.
- $P(i)$ počet pixelů o dané intenzitě i.
- $W_b(T)\sigma_b^2(T)$ značí třídu pozadí obrazu.
- $W_f(T)\sigma_f^2(T)$ značí třídu popředí obrazu.

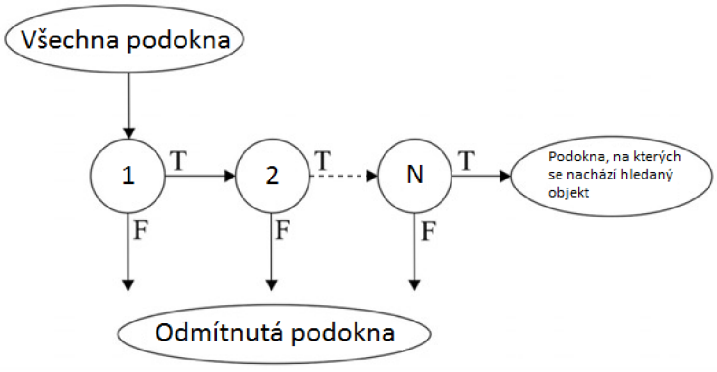
Kapitola 3

Segmentační algoritmy

V této kapitole se seznámíme se segmentačními algoritmy, které jsou použité v této bakalářské práci pro separaci znaků z textových CAPTCHA kódů. Jedná se o segmentaci pomocí kaskádového klasifikátoru, o algoritmus semínkového vyplnění a o histogramový segmentační algoritmus.

3.1 Kaskádový klasifikátor

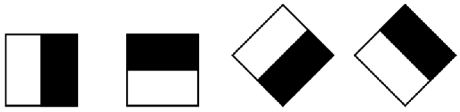
Kaskádový klasifikátor patří do knihovny OpenCV [14] a používá se pro detekci objektů v obraze. Jedná se o strojové učení založené na kaskádové funkci, která je natrénována z mnoha pozitivních a negativních vzorků. Mezi pozitivní vzorky patří obrázky obsahující hledaný objekt. Mezi negativní vzorky patří obrázky, na kterých se nenachází žádný hledaný objekt. Při detekci objektů v obraze kaskádový klasifikátor využívá posuvného okna (sliding window), aby získal podokna obrázku, na kterých se může nacházet hledaný objekt. Pro rozhodování, zda se v daném podokně nachází hledaný objekt, používá kaskádový klasifikátor slabé klasifikátory, které filtrují velké množství negativních podoken na základě příznaků. Tyto slabé klasifikátory jsou za sebou v kaskádě, jak je znázorněno na Obrázku 3.1. Pokud podokno projde všemi slabými klasifikátory, je velká pravděpodobnost, že se v daném podokně hledaný objekt nachází.



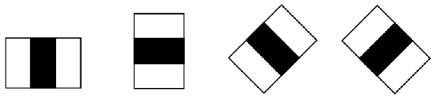
Obrázek 3.1: Schéma kaskádového klasifikátoru.

3.1.1 HAAR příznaky

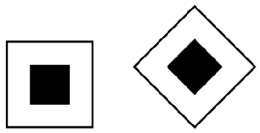
HAAR příznaky se skládají z černého a bílého obdélníku. Sousedící obdélníky označují změnu kontrastu v obrázku. Pomocí těchto příznaků můžeme detekovat např. hrany objektů, Obrázek 3.2, čáry, Obrázek 3.3 nebo osamocené body v objektu, Obrázek 3.4. Tyto příznaky jsou aplikovány na celý vstupní obraz. Příznaky se ale mohou zvětšovat, což znamená, že se HAAR příznak hrany z velikosti 1x2 pixelu může zvětšit až na velikost podokna. Základní velikost podokna je 24x24 pixelů. V tomto okně můžeme získat přes 162 000 příznaků[1]. U každého příznaku bychom museli počítat rozdíl sum pixelu pod bílým a černým obdélníkem zvlášť, což by způsobilo kritické zpomalení při trénování a detekci objektů. Řešením tohoto problému je vytvoření integrálního obrazu, popsaného v Kapitole 3.1.2.



Obrázek 3.2: HAAR příznaky hran.



Obrázek 3.3: HAAR příznaky čar.



Obrázek 3.4: HAAR příznaky středů.

3.1.2 Integrální obraz

Integrální obraz je použit při výpočtu jednotlivých HAAR příznaků. Nejprve se vytvoří kopie vstupního obrazu. Každý bod této kopie obsahuje sumu intenzit pixelů, které se vyskytují v obdélníku vstupního obrazu začínajícím v počátečním bodě [0;0] a končícím ve zvoleném bodě. Po výpočtu všech bodů se tato kopie může nazývat integrálním obrazem. Matematický zápis vzorce pro výpočet všech bodů v integrálním obraze je:

$$I_{\sum}(x,y) = \sum_{i=0}^{i \leq x} \sum_{j=0}^{i \leq y} I(i,j) \tag{3.1}$$

kde:

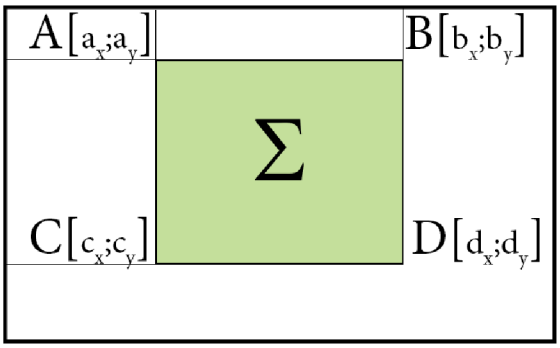
- x, y je pozice v integrálním obraze $I_{\sum}$.
- i, j je pozice ve vstupním obraze I.

[1] Počítáno z příznaků rovnoběžných s osou x.

Po vytvoření integrálního obrazu je velice jednoduché spočítat hodnotu HAAR příznaku. V případě, že počítáme sumu jednoho obdélníku, postačí nám pouze sečíst nebo odečíst čtyři hodnoty z integrálního obrazu:

$$Suma = I_{\sum}(d_x; d_y) + I_{\sum}(a_x; a_y) - I_{\sum}(b_x; b_y) - I_{\sum}(c_x; c_y) \tag{3.2}$$

Na Obrázku 3.5 je znázorněn integrální obraz a obdélník, ze kterého se počítá suma intenzit pixelů.



Obrázek 3.5: Integrální obraz.

3.1.3 LBP příznaky

LBP (Local Binary Pattern) příznaky neboli příznaky lokálních binárních vzorů se počítají pro každý bod vstupního obrazu podle vzorce (3.3). Nejdříve se převede obrázek do stupně šedi. Následně se provede prahování, popsané v Kapitole 2.1 s nastaveným prahem na hodnotu jasu středového bodu, dále se výsledná hodnota vynásobí váhou daného bodu. Pomocí tohoto vzorce lze detekovat LBP příznaky, které označují základní tvary v obraze.

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(g_p - g_c)2^p \tag{3.3}$$

kde:

- R označuje vzdálenost všech bodů od středového bodu.
- P označuje počet bodů ve vzdálenosti R.
- 2^p označuje váhu bodu na pozici p v LBP příznaku.
- g_c označuje hodnotu středového bodu v odstínu šedé.
- g_p označuje hodnotu bodu na pozici p v LBP příznaku v odstínu šedé.
- $s(x) = \begin{cases} 1 & \text{pokud } x \geq 0 \\ 0 & \text{pokud } x < 0 \end{cases}$

3.1.4 Porovnání HAAR a LBP příznaků

- **HAAR příznaky:**
  - (+) Vyšší úspěšnost detekce objektů v obraze.
  - (-) Je pomalejší při trénování kaskády a detekování objektů v obraze.
  - (-) Je citlivý na tvar objektu.
- **LBP příznaky:**
  - (+) Je rychlejší při trénování kaskády a detekování objektů v obraze.
  - (+) Invariantní vůči změnám intenzity bodů v obraze.
  - (-) Nižší úspěšnost detekce objektů v obraze.

Pro segmentaci znaků v CAPTCHA kódech jsem použil LBP příznaky.

3.2 Algoritmus semínkového vyplnění

Tento algoritmus slouží k separaci stejnobarevného tvaru v rastrovém obrázku. Na začátku se musí vybrat souřadnice bodu, od kterého se začne. Postupně se vybírají sousedé bodu stejné barvy a přenáší se do nového obrázku, čímž se oddělí jeden tvar od ostatních. Varianty tohoto algoritmu se liší podle výběru sousedních bodů. Jedná se o výběr čtyř sousedních bodů, což umožní oddělit i tvary, které se dotýkají rohy bodů. Nebo se jedná o výběr všech osmi sousedních bodů, což umožní označovat např. přímky, u kterých se body libovolně dotýkají. Tento algoritmus se často používá z důvodu jednoduché implementace a využitelnosti na libovolných obrázcích pro segmentaci všech jejich tvarů.

Při segmentaci tvarů v CAPTCHA kódech pomocí algoritmu semínkového vyplnění je důležitou prerekvizitou binarizace CAPTCHA kódu, neboli převod CAPTCHA kódu na černobílý. K tomu slouží algoritmus prahování, Kapitola 2.1.

3.3 Histogramový segmentační algoritmus

Tento algoritmus slouží k segmentaci objektů z 2D obrazu. Nejprve převede 2D vstupní obraz do 1D. Výsledkem je 1D vektor sum intenzit pixelů z jednotlivých sloupců. Následně se tento 1D vektor rozděluje podle nejnižších hodnot. Indexy nalezených hodnot z 1D vektoru označují x souřadnice vertikálních řezů 2D vstupního obrazu.

Kapitola 4

Klasifikační algoritmy

V této kapitole si popíšeme klasifikační algoritmy, které jsou v této práci použity. Klasifikační algoritmy slouží ke klasifikaci objektů získaných ze segmentačních algoritmů. Konkrétně se jedná se o konvoluční neuronové sítě a histogramový klasifikátor. U konvolučních neuronových sítí si navíc popíšeme zvolenou konvoluční síť LeNet a framework Caffe. U histogramového klasifikátoru si uvedeme jeho princip a hlavní myšlenku.

4.1 Konvoluční neuronové sítě

Konvoluční neuronové sítě (Convolutional Neural Networks, dále jen CNN) jsou velmi populárním nástrojem pro detekci nebo klasifikaci objektů v obraze [9]. CNN využívají převážně konvolučních vrstev, které provádějí konvoluci svého vstupu a jádra, podle kterých získala CNN svůj název. CNN jsou tvořeny z navzájem propojených vrstev. Každá vrstva v CNN aplikuje filtr na svá vstupní data a výstup posílá do následující vrstvy. Pokud se jedná o poslední vrstvu, tak výstupem je číslo třídy, do které patří vstup, který jsme CNN na vstupu předali.

Trénování CNN spočívá v aktualizaci hodnot v konvolučních jádrech a využívá se metody backpropagation, neboli zpětné šíření chyby.

V této bakalářské práci jsem použil konvoluční síť LeNet (Kapitola 4.2), která byla navrhnuta pro klasifikaci znaku ze vstupního obrazu.

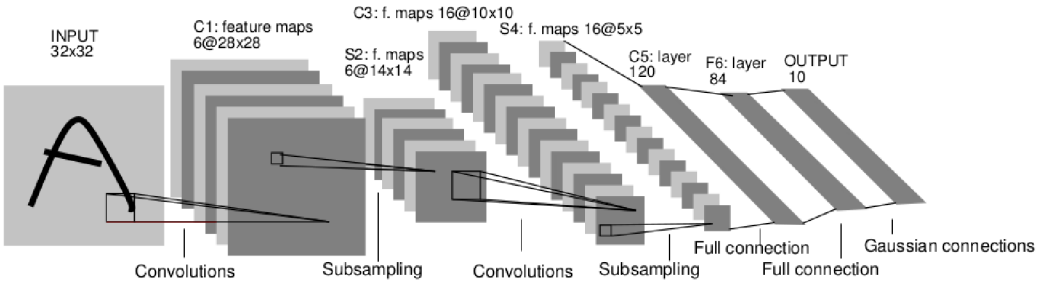
4.2 LeNet

LeNet je populární konvoluční síť využívaná ve frameworku Caffe, Kapitola 4.2.2. Je navržena pro klasifikaci znaku ze vstupního obrazu. Model sítě LeNet je ilustrován na Obrázku 4.1. Model prezentuje typy a propojení jednotlivých vrstev. Při klasifikaci datová vrstva (INPUT) obsahuje vstupní obraz se znakem. Postupně se na vstupní obraz aplikují uvedené vrstvy CNN[1] a výstupem je hodnota třídy, do které spadá vstupní obraz[2].

Při trénování datová vrstva obsahuje trénovací a testovací data. V každé iteraci při trénování procházejí vstupní data postupně uvedenými vrstvami a výstupem je hodnota přesnosti (accuracy) a ztrátovosti (loss). Kapitola 4.2.1 detailněji popisuje jednotlivé vrstvy sítě LeNet [15].

[1] Convolutions, Subsampling, Full connection, Gaussian connections

[2] V modelu sítě LeNet se jedná o hodnotu OUTPUT.

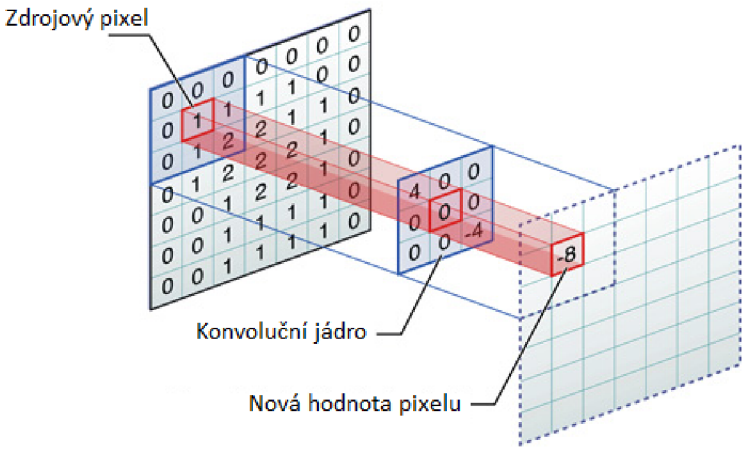


Obrázek 4.1: Model LeNet sítě.

4.2.1 Vrstvy sítě LeNet

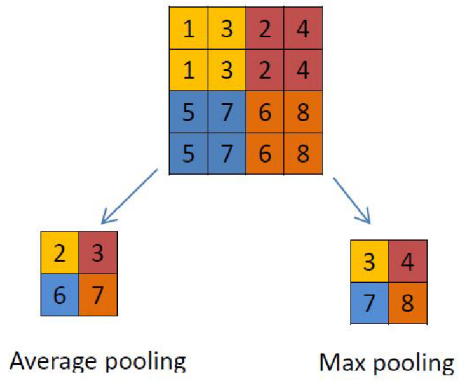
- **Datová vrstva** (Data layer) je první vrstvou v síti LeNet (INPUT). Tato vrstva přivádí vstupní data, která mohou pocházet z LevelDB nebo z LMDB databází nebo přímo z paměti. LMDB (Ligtning memory mapped database) je mimořádně rychlá a paměťově nenáročná databáze [17]. Do této databáze se převádějí vstupní data určená pro trénování a testování vlastní LeNet sítě.

- **Konvoluční vrstva** (Convolution layer) detekuje hrany objektů v určitém směru. Záleží na hodnotách ve zvoleném jádře. Konvoluce se aplikuje na všechny pixely vstupního obrazu. Konvoluce jednoho pixelu se provádí součtem součinů okolních pixelů včetně aktuálního a jádra konvoluce, jak je demonstrováno na Obrázku 4.2.



Obrázek 4.2: Princip konvoluce.

- **Sdružovací vrstva** (Pooling / Subsampling layer) slouží ke snížení rozlišení obrázku. Nejčastěji se používá Max pooling, který ze zadaných pixelů vybere maximální hodnotu, nebo Average pooling, který vypočítá průměrnou hodnotu zvolených pixelů. Obrázek 4.3 demonstruje princip této vrstvy.



Obrázek 4.3: Princip pooling / subsampling vrstvy.

- **Plně propojená vrstva** (Full connection layer). Jedná se o plně propojenou vrstvu, což znamená, že na vstup každého neuronu v této vrstvě je připojený každý neuron z vrstvy předcházející.
- **Vrstva přesnosti** (Accuracy layer). Tato vrstva se používá při trénování CNN. Nenachází se v modelu sítě LeNet, Obrázek 4.1. Vrací procentuální úspěšnost klasifikace vzorků ze vstupních vzorků. Počet vzorků na vstupu udává datová vrstva.
- **Ztrátová vrstva** (Loss layer). Tato vrstva se také používá pouze při trénování CNN. Nenachází se v modelu sítě LeNet, Obrázek 4.1. Obsahuje ztrátovou funkci, která určuje nepřesnost v nastavení vah sítě. Při trénování CNN je důležité najít ideální nastavení vah, při kterých bude výstup ztrátové funkce minimální.

4.2.2 Framework Caffe

V této bakalářské práci framework Caffe využijeme pro trénování a testování LeNet sítě. Obecně framework Caffe je open-source nástroj určený pro trénování a testování hlubokých CNN. Klade důraz na rychlost a modularitu. Tento oblíbený framework byl vytvořen od Berkeley Vision and Learning Center (BVLC) [21]. Caffe obsahuje knihovnu napsanou v C++, která disponuje rozhraním pro Python a MATLAB. Modely a optimalizace sítí jsou definovány pomocí konfiguračních souborů, které jsou ve formátu Google Protobuf [10]. Při trénování a testování CNN není vyžadováno žádného programování. Pro trénování a testování vlastní CNN ve frameworku Caffe je zapotřebí nastavit tři konfigurační soubory:

- **solver.prototxt** je hlavní konfigurační soubor frameworku Caffe. V tomto konfiguračním souboru se nastavují základní parametry pro trénování sítě. Nastavuje se zde např. cesta k souboru train_test.protobuf, počet iterací, rychlosti trénování a testování celého modelu sítě. Důležitým parametrem je také solver_mode, který udává, zda se má model sítě trénovat pomocí CPU nebo GPU.
- **train_test.prototxt** obsahuje konkrétní síť určenou k trénování modelu. Tento konfigurační soubor obsahuje jednotlivé vrstvy sítě. V našem případě se jedná o konvoluční síť LeNet 4.2.
- **deploy.prototxt** je určen pro konfiguraci výsledné predikce objektů v obraze.

V Caffe data komunikují a ukládají se pomocí tzv. blobs, které tvoří každou vrstvu [20]. Jinak řečeno, blob je základní stavební jednotka Caffe sítí. Blob je tvořený N rozměrnou strukturou pro ukládání informací, jako jsou informace o vstupních datech, parametry modelu nebo derivace pro optimalizaci. Např. 4 rozměrný blob NxKxHxW ukládá informace o N (počtu zpracovaných obrazových datech v jednom průchodu), K (počtu kanálů obrázku) H a W (šířce a výšce vstupního obrázku).

4.3 Histogramový klasifikátor

Klasifikace se provádí pomocí podobnosti histogramů. Vždy se porovnává histogram klasifikovaného znaku z CAPTCHA kódu a histogram znaku z trénovací sady. Základem pro klasifikaci je vzorec pro porovnání dvou histogramů se stejným počtem sloupců[3] z knihovny OpenCV [13]. Algoritmus klasifikace jsem však modifikoval tak, aby byl schopen ohodnotit shodnost dvou histogramů obsahující libovolný počet sloupců. Modifikace algoritmu spočívá v umělém rozšíření histogramů o nové sloupce tak, aby vznikly dva histogramy obsahující stejný počet sloupců. Algoritmus si však pamatuje, který histogram původně obsahoval méně sloupců a tento histogram posouvá pod původně mohutnějším histogramem. Při každém posunutí se spočítá podobnost histogramů pomocí vzorce (4.1). Po vyzkoušení všech možností se vybere možnost s nejvyšším skóre podobnosti.

$$d(H_1, H_2) = \frac{\sum_{i=0}^{I}(H_1(i) - \overline{H_1})(H_2(i) - \overline{H_2})}{\sqrt{\sum_{i=0}^{I}(H_1(i) - \overline{H_1})^2 \sum_{i=0}^{I}(H_2(i) - \overline{H_2})^2}} \tag{4.1}$$

kde:

- $\overline{H_k}$ vyjadřuje průměr všech hodnot v daném histogramu:
  - $\overline{H_k} = \frac{1}{N}\sum_{j=0}^{N} H_k(j)$.
  - $k = 1, 2$.
  - N udává počet sloupců v histogramu.
- I udává počet sloupců histogramů H_1 a H_2.
- $H_{1,2}(i)$ udává i-tý sloupec v daném histogramu.
- $d(H_1, H_2)$ je Pearsonův korelační koeficient. Vrací skóre podobnosti dvou stejně širokých histogramů, kde $d \in < -1, 1 >$.

[3] Počet sloupců v histogramu závisí na šířce znaku.

Kapitola 5

Datové sady

V této kapitole si popíšeme datové sady pro jednotlivé klasifikáry. Bude se jednat o datovou sadu pro kaskádový klasifikátor, CNN a histogramový klasifikátor.

5.1 Datová sada pro kaskádový klasifikátor

Pro vytvoření datové sady pro kaskádový klasifikár budeme potřebovat datovou sadu pozitivních a negativních obrázků.

5.1.1 Pozitivní datová sada

Pozitivní datová sada se skládá ze 180 000 obrázků alfanumerických znaků. Každý tento znak je reprezentován 5 000 vzorky, ve kterých 2 500 obsahuje znak s bílým pozadím a 2 500 obsahuje znak s pozadím reálného CAPTCHA kódu. Pro získání všech znaků jsem vytvořil skript, který každý znak vyřízl. Ukázka na Obrázku 5.1.

Obrázek 5.1: Ukázka pozitivní datové sady.

5.1.2 Negativní datová sada

Negativní datová sada se skládá z 10 255 obrázků obsahující náhodný obsah [18] [5]. Vybíral jsem jen ty obrázky, které neobsahovaly žádná písmena ani žádné číslice.

Do negativního datové sady jsem přidal ještě 2 000 obrázků s pozadím CAPTCHA kódu. Přidal jsem je z důvodu zvýšení úspěšnosti detekce alfanumerických znaků v jednotlivých CAPTCHA kódech. Ukázka negativní datové sady pro kaskádový klasifikátor je na Obrázku 5.2.

Obrázek 5.2: Ukázka negativní datové sady.

5.2 Datová sada pro CNN

Datová sada pro CNN obsahuje 224 333 obrázků alfanumerických znaků. Z těchto obrázků je 180 000 shodných s pozitivní datovou sadou uvedenou v Kapitole 5.1, 36 567 obrázků znaků napsaných různými fonty, 1 979 obrázků ručně psaných znaků a 5 787 obrázků obsahující vyseparované znaky z fotek. Různé formáty datových sad jsem vybíral proto, aby CNN byla schopná klasifikovat co nejvyšší možné množství alfanumerických znaků [8]. Na stažené datové sady jsem musel aplikoval skript, který jednotlivé znaky ořízl. Na Obrázku 5.3 je uvedena ukázka dvou vzorků z každé datové sady.

Obrázek 5.3: Ukázka datové sady pro CNN.

5.3 Datová sada pro histogramový klasifikátor

Datová sada pro histogramový klasifikátor se skládá z několika CAPTCHA kódů, ve kterých se vyskytují všechny alfanumerické znaky, které jsou v daném druhu CAPTCHA kódu použity. Při vytváření této datové sady je důležité dbát na to, aby se znaky v CAPTCHA kódu nepřekrývaly. Pokud by tomu tak bylo, aplikace by znaky z daného CAPTCHA kódu nemohla použít pro trénování a celý CAPTCHA kód by přeskočila.

Kapitola 6

Návrh aplikace

V této kapitole se seznámíme s návrhem aplikace, která je v rámci této bakalářské práce implementována. Jedná se o aplikaci s grafickým uživatelským rozhraním (Graphical User Interface, dále jen GUI). Tato aplikace umožňuje rozpoznávání CAPTCHA kódů pomocí dvou klasifikátorů, ze kterých si uživatel bude moci vybrat. Konkrétně se jedná o rozpoznávání CAPTCHA kódů pomocí histogramového klasifikátoru a CNN.

Při výběru histogramového klasifikátoru aplikace umožňuje vytvořit si vlastní histogramový klasifikátor, který je schopen následně rozpoznávat daný CAPTCHA kód. Pro vytvoření vlastního klasifikátoru je zapotřebí mít připravenou datovou sadu CAPTCHA kódů. Tuto datovou sadu CAPTCHA kódů je možné stáhnout z webových stránek, na kterých se daný CAPTCHA kód vyskytuje nebo je možné danou datovou sadu CAPTCHA kódů načíst ze zvoleného adresáře. Pomocí získané datové sady CAPTCHA kódů se vytvoří soubor, pomocí kterého je možné klasifikovat natrénované znaky.

Při výběru CNN aplikace očekává soubory popisující CNN, pomocí kterých se budou rozpoznávat znaky z CAPTCHA kódů. Všechny potřebné soubory jsou přiložené u aplikace.

Po nastavení klasifikátorů je možné rozpoznávat daný CAPTCHA kód. Aplikace umožňuje rozpoznávat CAPTCHA kódy jednotlivě, hromadně s možností testování úspěšnosti rozpoznávání nebo se zobrazením jednotlivých fází rozpoznávání, které umožní jednoduché odhalení příčiny neúspěšného rozpoznání daného CAPTCHA kódu.

Kapitola 7

Trénování klasifikátorů

V této kapitole se seznámíme s trénováním vlastních klasifikátorů. Nejprve si uvedeme postup trénování kaskádového klasifikátoru. Dále se seznámíme s postupem trénování CNN pomocí frameworku Caffe a s konfigurací modelu sítě LeNet. V poslední řadě se seznámíme s trénováním histogramového klasifikátoru.

7.1 Trénování kaskádového klasifikátoru

Trénování kaskádového klasifikátoru se rozděluje do dvou částí. V první části se musí vytvořit soubory popisující pozitivní a negativní datové sady kaskádového klasifikátoru, uvedené v Kapitole 5.1. Dále se musí pozitivní datová sada převést do vektorového souboru. K tomuto převodu se používá binární soubor *opencv_createsamples.exe* z knihovny OpenCV. Ve druhé části se použijí připravená data k trénování vlastního kaskádového klasifikátoru. Výstupem trénování je soubor, pomocí kterého lze detekovat natrénované objekty v obraze.

Postup přípravy dat pro trénování:

- Vytvoříme textový soubor, který bude obsahovat seznam cest k negativním obrázkům.
- Vytvoříme informační soubor (*.info) pro pozitivní datovou sadu, ve kterém bude na každém řádku napsaná cesta k souboru, počet výřezů a souřadnice výřezu detekovaného objektu. Příklad jednoho řádku: *path_to_image/image_name.jpg 1 0 0 30 50*
- Vytvoříme vektorový soubor z pozitivní datové sady a z informačního souboru. Použijeme příkaz:

  opencv_createsamples.exe –info **A** –num **B** –w **C** –h **D** -vec **E**

  kde:

  - A udává cestu k informačnímu souboru.
  - B udává celkový počet pozitivních obrázků.
  - C udává šířku vzorků.
  - D udává výšku vzorků.
  - E udává cestu k výstupnímu vektorovému souboru.

Pro trénování použijeme binární soubor *opencv_traincascade.exe* s parametry:

opencv_traincascade.exe –data **A** –vec **B** –bg **C** –numPos **D** –numNeg **E** –numStages **F** –w **G** –h **H** –featureType LBP -minHitRate **I** -maxFalseAlarmRate **J**
kde:

- A udává cestu ke složce, do které se mají ukládat výstupní data.
- B udává cestu k vektorovému souboru vytvořený v první části.
- C udává cestu k textovému souboru obsahující negativní obrázky.
- D udává počet pozitivních obrázků, které mají vstupovat do každé fáze trénování. Toto číslo musí být menší než celkový počet pozitivních obrázků.
- E udává počet negativních obrázků, které mají vstupovat do každé fáze trénování. Toto číslo musí být menší než celkový počet negativních obrázků.
- F udává počet trénovacích fází.
- G udává výšku vzorků. Měla by být stejná jako v první části.
- H udává šířku vzorků. Měla by být stejná jako v první části.
- I procentuálně označuje počet správně klasifikovaných **pozitivních** obrázků.
- J procentuálně označuje počet nesprávně klasifikovaných **negativních** obrázků jako **pozitivních**.

Pro natrénování vlastního kaskádového klasfikátoru jsem použil tyto hodnoty:

numPos	**numNeg**	**numStages**	**w**	**h**	**minHitRate**	**maxFalseAlarmRate**
20 000	7 000	12	19	22	0.999	0.4

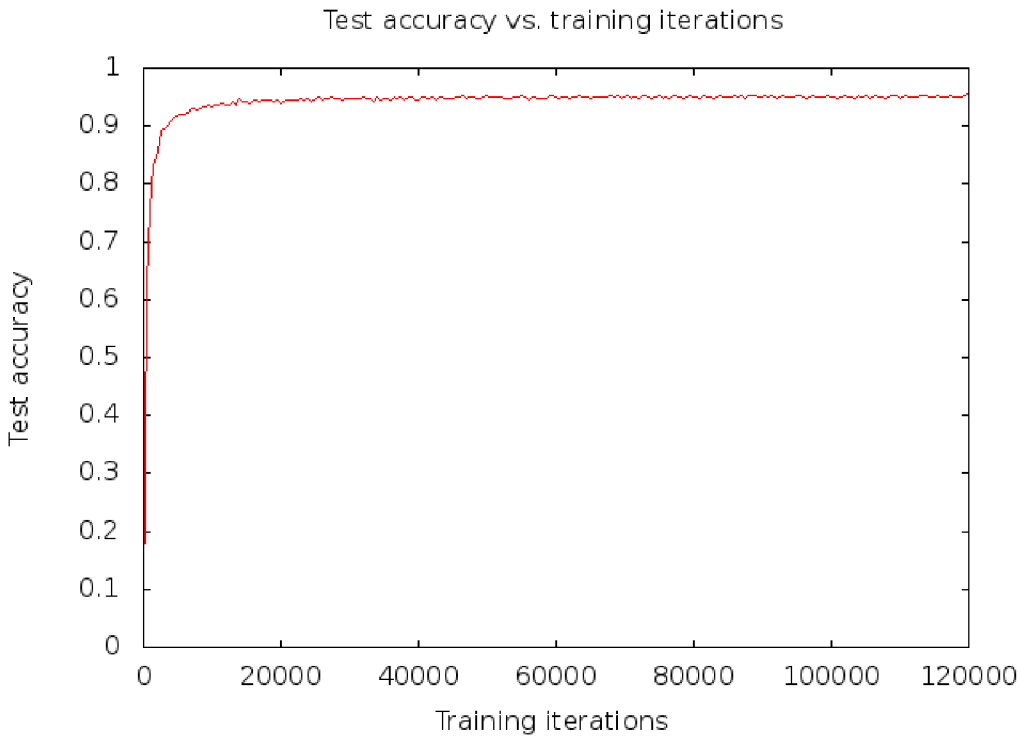
Tabulka 7.1: Hodnoty parametrů pro trénování kaskádového klasifikátoru.

7.2 Trénování CNN

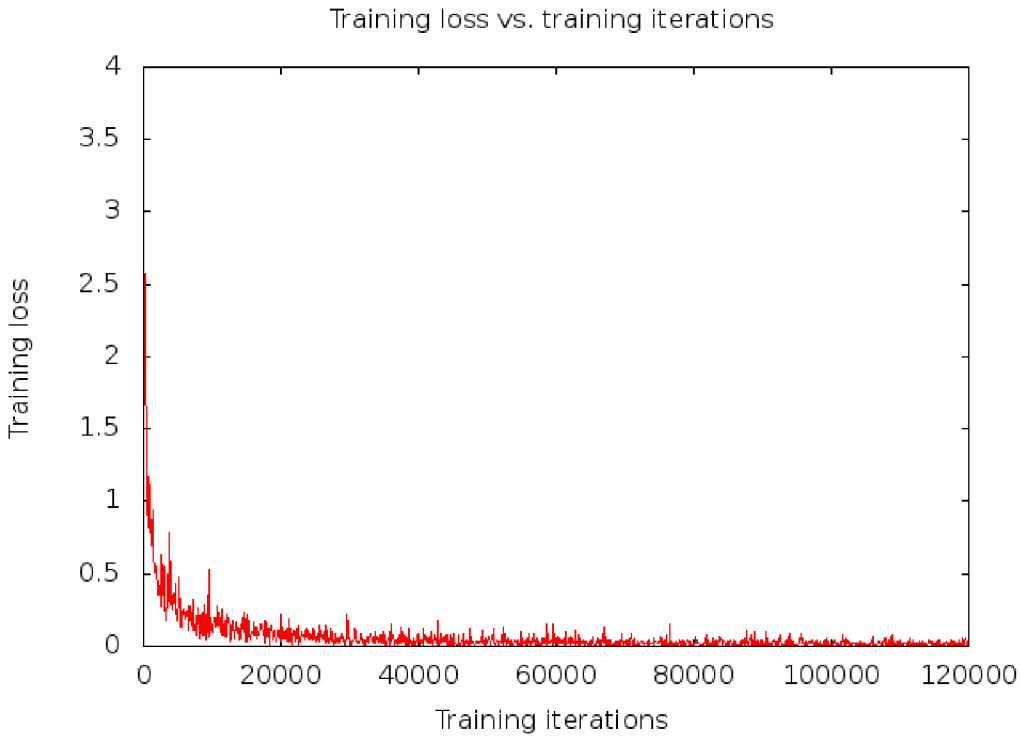
Trénování CNN se rozděluje do několika částí. V první části se musí vstupní datová sada, Kapitola 5.2, rozdělit na testovací a trénovací datovou sadu v doporučeném poměru 1:5. Tyto dvě datové sady se musí transformovat do LMDB databází. Dále se musí pomocí transformovaných datových sad vytvořit soubor, který udává váhy bodů v datových sadách. V další části se pomocí souborů solver.prototxt, train_test.prototxt a frameworku Caffe začne trénovat CNN. Trénování trvalo 9 hodin, 7 minut a 59 sekund[1]. Výstupem trénování CNN jsou dva grafy. Na Obrázku 7.1 je uveden graf závislosti přesnosti na iteracích CNN. Podle tohoto grafu můžeme zjistit, že se úspěšnost CNN od hranice 20 000 iterací nemění a stále se drží okolo 95 %. Na Obrázku 7.2 je uveden graf závislosti výstupu ztrátové funkce na iteracích CNN. Jak je uvedeno v Kapitole 4.2, cílem je nalézt takovou iteraci, ve které je výstup ztrátové funkce minimální.

Do aplikace jsem vybral caffemodel s iterací 55 000, který má 95,22% přesnost.

[1]Na CPU Intel Core(TM) i7-3630QM, 2.4GHz



Obrázek 7.1: Graf závislosti přesnosti CNN na trénovacích iterací.



Obrázek 7.2: Graf závislosti ztrátové funkce na trénovacích iterací.

7.2.1 Konfigurace konvoluční sítě LeNet

Konfigurovat síť LeNet lze pomocí tří konfiguračních souborů popsaných v Kapitole 4.2.2. Uvedeme si zde nejdůležitější konfigurace sítě LeNet.

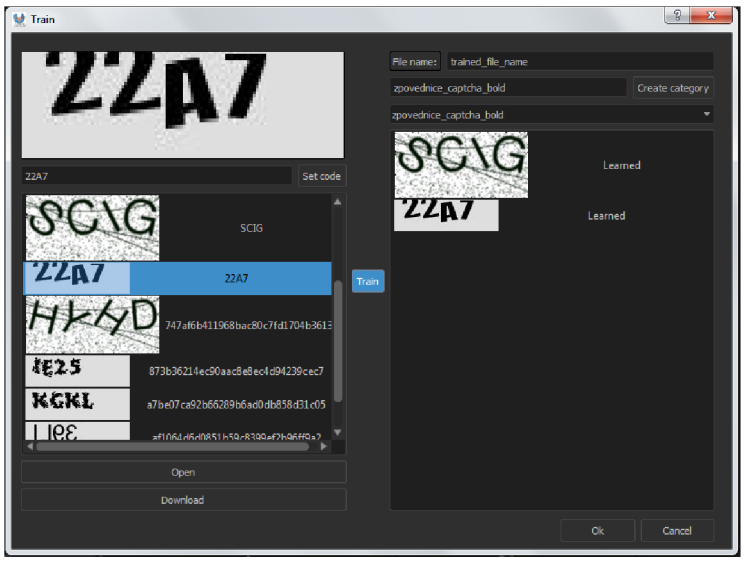
V souboru **solver.prototxt** jsem upravil počet iterací na 120 000, snížil jsem rychlost trénování[2] z důvodu získání přesnějšího modelu sítě a nastavil jsem trénování na CPU.

V souboru **train_test.prototxt** jsem nastavil cesty ke vstupním datovým sadám transformovaných v LMDB a nastavil jsem počet výstupních tříd na 36[3].

V souboru **deploy.prototxt** jsem nastavil počet výstupních tříd na 36, jako v předchozím konfiguračním souboru. Nastavil jsem parametry popisující vstupní data - jedná se o velikost a počet kanálů obrázku. Dále jsem odstranil poslední vrstvu Softmax z důvodu toho, že do této vrstvy při klasifikaci vstupuje skóre každé třídy a tato vrstva nastavuje třídu s nejlepším skóre na hodnotu 1 a ostatní třídy na hodnotu 0. V aplikaci budeme pracovat se skóre jednotlivých tříd, tím pádem tuto poslední vrstvu nepotřebujeme.

7.3 Trénování histogramového klasifikátoru

Trénování histogramového klasifikátoru probíhá ve třech fázích. V první fázi se musí načíst datová sada CAPTCHA kódů do aplikace. Každý CAPTCHA kód z datové sady by měl být pojmenovaný tak, aby jeho název odpovídal textu na CAPTCHA kódu. V případě, když jednotlivé CAPTCHA kódy nejsou korektně pojmenované, aplikace poskytuje rozhraní pro přejmenování a následné natrénování všech CAPTCHA kódů z datových sad. V druhé fázi může uživatel zvýšit rychlost a přesnost klasifikace tím, že CAPTCHA kódy rozdělí do skupin podle druhů CAPTCHA kódů. Ve třetí fázi užvatel vytvoří název pro soubor s trénovacími daty a znaky ze zvolených CAPTCHA kódů nechá natrénovat. Na Obrázku 7.3 je uvedena ukázka GUI pro trénování histogramového klasifikátoru. Lze vidět datovou sadu CAPTCHA kódů, ze kterého uživatel první dva CAPTCHA kódy nastavil a následně je natrénoval.



Obrázek 7.3: GUI pro trénování histogramového klasifikátoru.

[2]Parametr *base_lr* na hodnotu 0.001

[3]Jedná se o počet všech alfanumerických znaků

Kapitola 8

Popis implementace aplikace

V této kapitole se seznámíme s implementací celé aplikace vytvářené v rámci bakalářské práce. Aplikace se skládá z GUI aplikace napsané v C++ s využitím knihovny Qt a ze dvou Python skriptů. První Python skript stahuje obrázky z webových stránek a pomocí druhého skriptu lze rozpoznávat CAPTCHA kódy s využitím CNN.

8.1 Rozpoznávání CAPTCHA kódů pomocí histogramového klasifikátoru

Rozpoznávání CAPTCHA kódů pomocí histogramového klasifikátoru je implementované v jazyce C++. Výběr kompilovaného jazyka C++ je z důvodu vysoké rychlosti, která je při rozpoznávání CAPTCHA kódů důležitá.

Implementace probíhala ve vývojovém prostředí Qt Creatoru. Qt Creator byl zvolen kvůli budoucí práci na grafickém rozhraní.

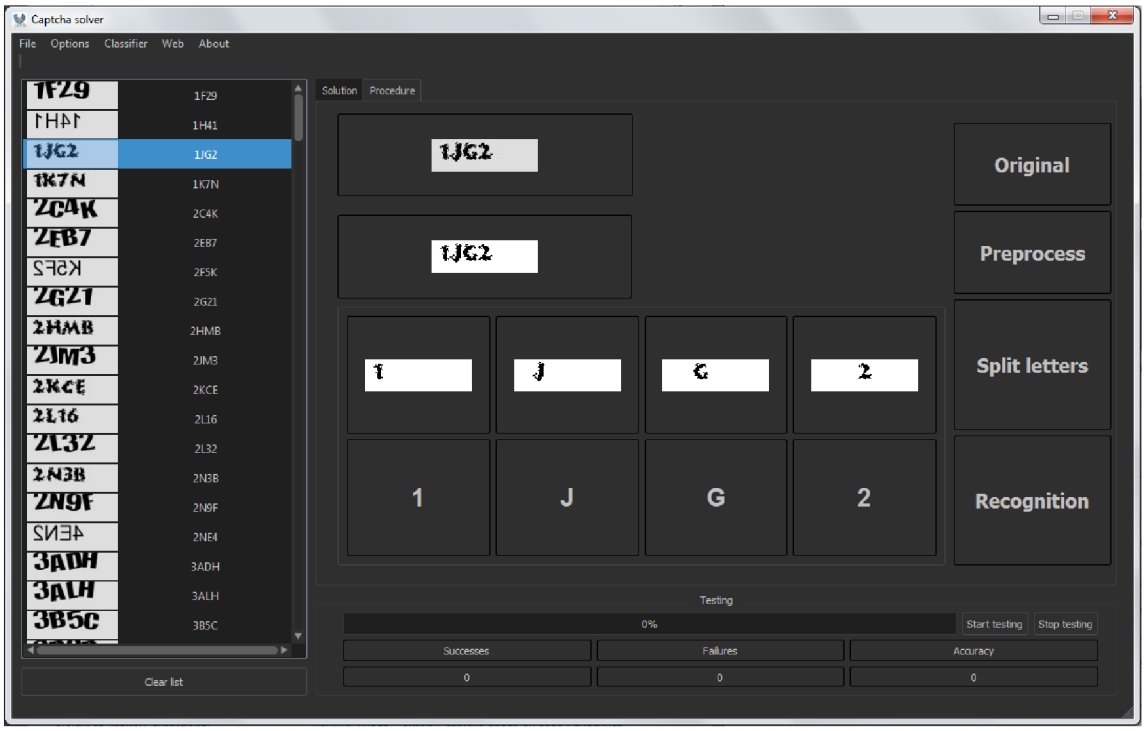
Implementace rozpoznání CAPTCHA kódů pomocí histogramového klasifikátoru se rozděluje do tří fází. Jedná se o fáze předzpracování, segmentace a klasifikace. V první fázi předzpracování se na CAPTCHA kód aplikuje algoritmus prahování, Kapitola 2.1. Ve druhé fázi se pomocí algoritmu semínkového vyplnění získají všechny tvary, Kapitola 3.2. Tyto tvary se filtrují na základě jejich velikostí. Vybírají se jen ty, které obsahují alespoň určitý počet pixelů. Po filtraci se každý tvar transformuje do horizontálního histogramu[1]. Pokud šířka histogramu několikanásobně převyšuje průměrnou šířku z natrénovaných histogramů, Kapitola 7.3, je velká pravděpodobnost, že se jedná o spojené znaky. V tomto případě se musí nalézt nejvhodnější řez k oddělení spojených znaků, Kapitola 8.1.1. Následně se histogramy se předají do poslední fáze klasifikace. Klasifikace využívá histogramového klasifikátoru, Kapitola 4.3. Pro svou práci potřebuje mít dostupný natrénovaný soubor s názvy znaků a jejich histogramů. Pokud není dostupný, je možné si ho vytvořit, Kapitola 7.3.

Aplikace umožňuje v každé fázi si uložit CAPTCHA kód s aplikovanými změnami do dočasné složky. Při zobrazení jednotlivých fází si aplikace v dočasné složce vytvoří novou složku pojmenovanou md5 hashem získaného z originálního CAPTCHA kódu a do této složky nahraje všechny potřebné obrázky určené k zobrazení. Na Obrázku 8.1 je ukázka rozpoznání CAPTCHA kódu se zobrazením všech fází rozpoznání. Pokud se zvolí rozpoznávání bez zobrazení fází, tak se žádné dočasné soubory neukládají a jen se rozpoznávají CAPTCHA kódy.

[1] Součet všech černých pixelů v jednotlivých sloupcích.

8.1.1 Algoritmus pro oddělení spojených znaků

Algoritmus přijímá jeden histogram obsahující více než jeden znak. Algoritmus na začátku odhadne místa vertikálních řezů vstupním histogramem podle jeho šířky. V okolí těchto míst algoritmus vyhledává tři vertikální sloupce, které obsahují nejvyšší hodnotu v histogramu[2]. Podle nalezených sloupců se vstupní histogram rozdělí a jednotlivé části pokračují do klasifikačního algoritmu.



Obrázek 8.1: GUI zobrazení rozpoznání CAPTCHA kódu krok za krokem pomocí histogramového klasifikátoru.

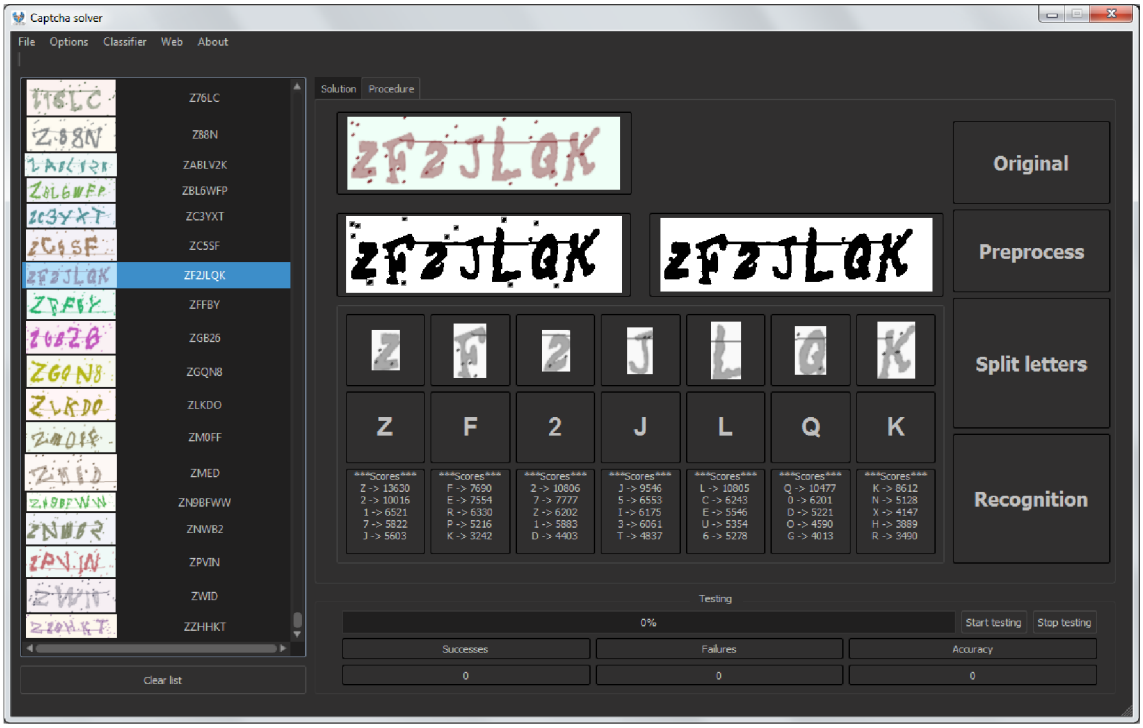
8.2 Rozpoznávání CAPTCHA kódů pomocí CNN

Rozpoznávání CAPTCHA kódů pomocí CNN má na starost Python skript, který převezme cesty ke čtyřem důležitým souborům pro rozpoznávání CAPTCHA kódů pomocí CNN. Jedná se o deploy soubor, caffemodel sítě LeNet, imagenet_mean soubor datatové sady a soubor s popisky jednotlivých tříd. Podobně jako v předchozí Kapitole 8.1 se rozpoznání CAPTCHA kódu pomocí CNN rozděluje do tří fází. Ve fázi předzpracování se na obrázek aplikuje Gaussovo rozostření, Kapitola 2.2, a následně se aplikuje Otsu metoda prahování, Kapitola 2.3, pomocí které binarizujeme obraz o libovolné barevné intenzitě textu. Po binarizaci mohou v obraze zůstat osamocené černé fleky, které je nutné odstranit. K tomu slouží metoda findContours z knihovny OpenCV. Metoda vyhledá osamocené objekty, které lze následně odstranit. Z předpřipraveného obrazu lze vytvořit horizontální histogram, který vstupuje do další fáze segmentace. Segmentace má za úkol převzít histogram z předchozí fáze a určit souřadnice hran znaků. Podle těchto souřadnic se originální CAPTCHA kód

[2]Nejvyšší hodnota značí nejvíce bílých pixelů v daném sloupci.

rozřeže a tyto fragmenty uloží do dočasné složky. Poslední fáze klasifikace převezme fragmenty vyprodukované z předchozí fáze a zavolá nad nimi CNN pro klasifikaci znaků. CNN vrátí list znaků, které se nejvíce podobají znaku ze vstupního obrazu. Podle nejlépe ohodnoceného znaku z CNN se rozhoduje, jaký znak se nachází na daném obrázku.

Aplikace umožňuje zobrazení výsledků jednotlivých fází v GUI, Obrázek 8.2, nebo je možné rozpoznávat CAPTCHA kódy bez zobrazení jednotlivých fází.



Obrázek 8.2: GUI zobrazení rozpoznání CAPTCHA kódu krok za krokem pomocí CNN.

8.3 Stahování obrázků z webových stránek

Stahování CAPTCHA kódů z webových stránek má na starost skript, který je napsaný ve skriptovacím jazyce Python, verze 3.4. Výběr skriptovacího jazyka Python je z důvodu jednoduché implementace HTTP a HTTPS komunikace.

Celá implementace Python skriptu probíhala ve vývojovém prostředí Microsoft Visual Studio 2015. Visual Studio disponuje kvalitním ladícím nástrojem, který je při implementaci velice důležitý.

Tento skript převezme URL (Uniform Resource Locator) webové stránky, ze které se budou stahovat všechny obrázky včetně CAPTCHA kódu a parametry určující filtry na stahované obrázky. Poté se skript pokusí na zadanou URL adresu připojit. Pokud je URL adresa zadaná správně a skript se připojí, vrátí informace o připojení a data z požadované stránky. Pomocí dat stažených z webové stránky se získají URL adresy všech obrázků. Před stažením každého obrázku ze seznamu URL se skript podívá na filtr na obrázky, který byl zadán v parametrech. Filtr může být nastaven na typ, velikost obrázku nebo na konkrétní URL. Při úspěšném stažení se obrázek uloží do dočasné složky a následně se zobrazí v GUI.

Při požadavku na stažení celé datové sady CAPTCHA kódů, musí být nastaven filtr na konkrétní URL obrázku, který představuje CAPTCHA kód. Skript začne ve smyčce

aktualizovat webovou stránku a stahovat pouze zvolený CAPTCHA kód, který je vždy po aktualizaci stránky jiný. Tím je možné si nastahovat libovolné množství CAPTCHA kódů. Toto stahování běží v separátním vlákně z důvodu zamezení zamrznutí GUI aplikace. Na Obrázku 8.3 je ukázka GUI dialogového okna pro stahování datových sad CAPTCHA kódů z webových stránek.

Obrázek 8.3: GUI pro stahování datových sad z webových stránek.

Kapitola 9

Testování

V této kapitole si uvedeme testy aplikace prováděné na CAPTCHA kódech. Budeme se orientovat podle oficiální 1% hranice úspěšnosti rozpoznání CAPTCHA kódu [4]. Pokud testy zobrazí vyšší úspěšnost jak 1 %, můžeme hovořit o tom, že naše aplikace byla s rozpoznáváním úspěšná. Dále si otestujeme schopnost aplikace stahovat datové sady z webových stránek. V závěru kapitoly zhodnotíme dosažené výsledky.

9.1 Testování aplikace na CAPTCHA kódech

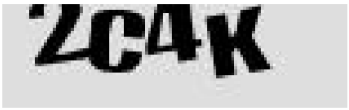
- **Testování histogramového klasifikátoru**

  Histogramový klasifikátor jsem testoval na CAPTCHA kódech v odstínu šedé. CAPTCHA kód se rozděluje do tří skupin. Jedná se o typ s deformovanými hranami (Obrázek 9.1a), zrcadlově otočenými písmeny (Obrázek 9.1b) a tučně psaným textem (Obrázek 9.1c).

  Testování proběhlo na 200 náhodně stažených CAPTCHA kódech. Testy ukázaly, že histogramový klasifikátor dokázal rozpoznat všechny CAPTCHA kódy se 100% úspěšností.

(a) CATPCHA kód s deformovanými hranami.

(b) CAPTCHA kód se zrcadlově otočeným písmem.



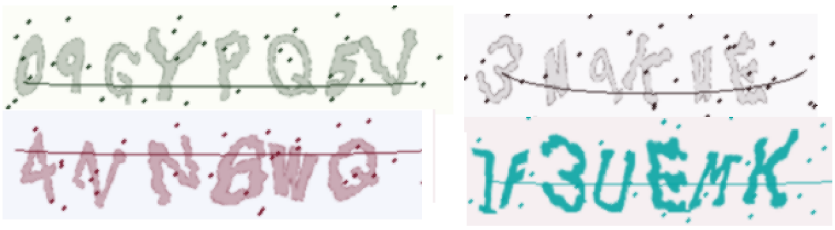
(c) CAPTCHA kód s tučným písmem.

Obrázek 9.1: Rozpoznávaný CAPTCHA kód v odstínu šedé

- **Testování CNN**

  Testování CNN proběhlo na CAPTCHA kódech vygenerovaných pomocí oficiálního Python generátoru CAPTCHA kódů [19]. Tento generátor využívá mnoho webových stránek z důvodu možnosti vygenerování CAPTCHA kódu z vlastních fontů písma a tím si vytvořit unikátní CAPTCHA kód. Tento generátor u CAPTCHA kódu deformuje celé znaky, mění jim velikost, úhel postavení, barvu a její intenzitu. Jako další bezpečnostní prvek používá vodorovnou čáru a rušivý šum v pozadí. Na Obrázku 9.2 je ukázka použitého CAPTCHA kódu.

Při vytváření vlastního CAPTCHA kódu jsem použil různých 8 fontů písma.



Obrázek 9.2: Ukázka CAPTCHA kódů z testovací sady pro CNN.

Testování proběhlo na 600 náhodně vytvořených CAPTCHA kódech, ve kterých je proměnlivý počet znaků. Nastavil jsem rozmezí 4-8 znaků na CAPTCHA kód. Aplikace dosáhla 53,5% úspěšnosti, což několikanásobně překonala minimální 1% úspěšnost.

9.2 Testování kaskádového klasifikátoru na CAPTCHA kódech

Kaskádový klasifikátor byl testovaný pro segmentaci znaků z totožného CAPTCHA kódu jako v předchozí Kapitole 9.1. I když je uváděno, že kaskádový klasifikátor se používá k detekci libovolných objektů v obraze [12], na CAPTCHA kódy vhodný není. Je to způsobeno dvěma základními důvody. Prvním důvodem je to, že CAPTCHA kód obsahuje znaky, u kterých má každý znak jiný poměr stran, což je veliký problém pro kaskádový klasifikátor. Při trénování se musí zvolit pouze jeden poměr stran, viz. Kapitola 7.1. Tím pádem nedokáže např. detekovat písmeno "I"a zároveň široké písmeno "W". Byl by vhodný pro CAPTCHA kódy, které mají všechny znaky ve stejném poměru šířky a výšky. Druhým důvodem je, že kaskádový klasifikátor má vysokou procentuální hodnotu FA(False Alarm)[1], což je velký problém pro klasifikaci znaků pomocí naší natrénované CNN, která nedokáže rozlišit, zda se jedná o znak nebo ne, Kapitola 7.2.

9.3 Testování stažení datových sad CAPTCHA kódů z webových stránek

Testování schopnosti aplikace stahovat CAPTCHA kódy z webových stránek proběhlo na náhodně vybraných webových stránkách obsahující textové CAPTCHA kódy. Seznam všech testovaných webových stránek je uveden v Příloze B.

Testovaly se webové stránky, které měly CAPTCHA kódy vložené jako obrázek[2], jako generátor CAPTCHA kódů[3] nebo měly CAPTCHA kód zakódovaný pomocí Base64[4]. U všech uvedených odkazů aplikace neměla problém při stažení datových sad CAPTCHA kódů.

[1] Procentuálně označuje počet nesprávně klasifikovaných negativních obrázků jako pozitivních.
[2] např. <img alt=""src="img/domexample2.jpg"... >
[3] např. <img src="captcha_more.php"... >
[4] např. <img src=" ... >

9.4 Zhodnocení testování

Výhoda histogramového klasifikátoru spočívá ve vysoké rychlosti klasifikace, která se pohybuje okolo 2 ms. Naopak nevýhodou je jeho nízká úspěšnost při rozpoznávání CAPTCHA kódů obsahující mnoho obtížně rozpoznatelných prvků z Kapitoly 1.2.1.

Výhoda CNN klasifikátoru spočívá v možnosti rozpoznávání více druhů textových CAPTCHA kódů. CNN podporuje také rozpoznávání ručně psaných znaků. Menší nevýhodou je rychlost rozpoznávání jednoho CAPTCHA kódu, která se pohybuje v průměru okolo 2 vteřin.

Kapitola 10

Závěr

Hlavním cílem této bakalářské práce byla implementace aplikace pro rozpoznávání textových CAPTCHA kódů. Uvedli jsme si teoretický popis algoritmů pro předzpracování obrazu, segmentaci znaků z obrazu a následnou klasifikaci jednotlivých znaků. Popsali jsme si datové sady obrázků, které jsme využily pro trénování všech použitých klasifikátorů. Uvedli jsme si postup trénování a testování klasifikátorů. Při testování histogramového klasifikátoru na zvoleném CAPTCHA kódu jsme získali 100% úspěšnost. Při testování CNN na obtížnějším CAPTCHA kódu jsme obdrželi 53,5% úspěšnost. Testovali jsme také kaskádový klasifikátor na CAPTCHA kódech. Testy ukázaly, že obecně není vhodný pro segmentaci znaků v CAPTCHA kódech. Také jsme testovali aplikaci na stahování datových sad CAPTCHA kódů z webových stránek. Testy neuvedly žádnou webovou stránku, ze které by se datové sady nedaly stáhnout.

10.1 Přínos

Výhoda histogramového klasifikátoru spočívá ve vysoké rychlosti trénování a klasifikace znaků, která se pohybuje v jednotkách milisekund. Další velkou výhodou je možnost stažení datové sady CAPTCHA kódů přímo z webové stránky, na které se daný CAPTCHA kód nachází. Pomocí této nově stažené datové sady si uživatel může vytvořit klasifikátor, který bude daný CAPTCHA kód rozpoznávat. Rozpoznávat se dají i CAPTCHA kódy, které mají spojené znaky. Na tento případ je histogramový klasifikátor připravený a podle natrénovaných znaků pozná, že jsou 2 nebo více znaků spojených a pokusí se je rozdělit a následně klasifikovat. Naopak nevýhodou je jeho nízká úspěšnost při rozpoznávání CAPTCHA kódů obsahující špatně čitelné znaky. V tomto případě je možné v aplikaci změnit klasifikaci na CNN.

Výhoda CNN klasifikátoru spočívá v možnosti rozpoznávání málo čitelných textových CAPTCHA kódů. CNN umožňuje rozpoznat i takové znaky, u kterých ani člověk nedokáže s jistotou říci, o jaký znak se jedná[1]. CNN podporuje také rozpoznávání ručně psaných znaků. Menší nevýhodou je rychlost rozpoznávání jednoho CAPTCHA kódu, která se pohybuje v průměru okolo 2 vteřin.

[1] Nejčastěji se jedná o znaky: 0-O, 2-Z, 5-6, apod.

10.2 Náměry na budoucí rozšíření

Aplikace by se mohla rozšířit o R-CNN (neboli Region-based Convolutional Neural Networks). Jedná se o systém sloužící k detekci a lokalizaci objektů s využitím vlastností z CNN. Tento systém by se mohl natrénovat na detekci a lokalizaci znaků v textových CAPTCHA kódech.

Literatura

[1] 7 Best CAPTCHA Solvers: 7 Best CAPTCHA Solvers. 2017, [Online; navštíveno 16-04-2017].
URL
https://www.slideshare.net/marriagenamechange/7-best-captcha-solvers

[2] Balkan, A.; Dura, J.; Eden, A.; aj.: *Introduction to Flash 3D*. Berkeley, CA: Apress, 2003, ISBN 978-1-4302-0814-3, s. 1–14, doi:10.1007/978-1-4302-0814-3_1.
URL http://dx.doi.org/10.1007/978-1-4302-0814-3_1

[3] Bennamoun, M.; Mamic, G. J.: *Object Recognition: Fundamentals and Case Studies*, kapitola Optical Character Recognition. London: Springer London, 2002, ISBN 978-1-4471-3722-1, s. 199–220, doi:10.1007/978-1-4471-3722-1_5.
URL http://dx.doi.org/10.1007/978-1-4471-3722-1_5

[4] Bursztein, E.; Martin, M.; Mitchell, J.: Text-based CAPTCHA Strengths and Weaknesses. In *Proceedings of the 18th ACM Conference on Computer and Communications Security*, CCS '11, New York, NY, USA: ACM, 2011, ISBN 978-1-4503-0948-6, s. 125–138, doi:10.1145/2046707.2046724.
URL http://doi.acm.org/10.1145/2046707.2046724

[5] Caltech 101: COMPUTATIONAL VISION AT CALTECH. 2017, [Online; navštíveno 16-04-2017].
URL http://www.vision.caltech.edu/Image_Datasets/Caltech101/

[6] CAPTCHA: The Official CAPTCHA Site. 2017, [Online; navštíveno 16-04-2017].
URL http://www.captcha.net/

[7] Chandavale, A. A.; Sapkal, A.: *Recent Trends in Computer Networks and Distributed Systems Security: International Conference, SNDS 2012, Trivandrum, India, October 11-12, 2012. Proceedings*, kapitola Security Analysis of CAPTCHA. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, ISBN 978-3-642-34135-9, s. 97–109, doi:10.1007/978-3-642-34135-9_10.
URL http://dx.doi.org/10.1007/978-3-642-34135-9_10

[8] Chars74K: The Chars74K image dataset - Character Recognition in Natural Images. 2017, [Online; navštíveno 16-04-2017].
URL http://www.ee.surrey.ac.uk/CVSSP/demos/chars74k/

[9] CS231n: CS231n Convolutional Neural Networks for Visual Recognition. 2017, [Online; navštíveno 16-04-2017].
URL http://cs231n.github.io/convolutional-networks/

[10] Google: Developer Guide Protocol Buffers. 2017, [Online; navštíveno 16-04-2017].
URL https://developers.google.com/protocol-buffers/docs/overview

[11] Harrabi, R.; Ben Braiek, E.: Color image segmentation using multi-level thresholding approach and data fusion techniques: application in the breast cancer cells images. *EURASIP Journal on Image and Video Processing*, ročník 2012, č. 1, 2012: s. 1–11, ISSN 1687-5281, doi:10.1186/1687-5281-2012-11.
URL http://dx.doi.org/10.1186/1687-5281-2012-11

[12] OpenCV: Cascade Classification. 2017, [Online; navštíveno 16-04-2017].
URL http://docs.opencv.org/2.4/modules/objdetect/doc/cascade_classification.html

[13] OpenCV: Histograms OpenCV 2.4.12.0 documentation. 2017, [Online; navštíveno 16-04-2017].
URL http://docs.opencv.org/2.4/doc/tutorials/imgproc/histograms/histogram_comparison/histogram_comparison.html

[14] OpenCV: OpenCV library. 2017, [Online; navštíveno 16-04-2017].
URL http://opencv.org/about.html

[15] ShadowThink: Caffe MNIST tutorial-LeNet. 2017, [Online; navštíveno 16-04-2017].
URL https://shadowthink.com/blog/tech/2016/08/28/Caffe-MNIST-tutorial

[16] Suphannee Sivakorn and Jason Polakis and Angelos D. Keromytis: I'm not a human: Breaking the Google reCaptcha. 2017, [Online; navštíveno 16-04-2017].
URL https://www.blackhat.com/docs/asia-16/materials/asia-16-Sivakorn-Im-Not-a-Human-Breaking-the-Google-reCAPTCHA-wp.pdf

[17] Symas: Symas Lightning Memory-mapped Database. 2017, [Online; navštíveno 16-04-2017].
URL https://symas.com/offerings/lightning-memory-mapped-database/

[18] University of Bonn: Computer Science VI, Autonomous Intelligent Systems. 2017, [Online; navštíveno 16-04-2017].
URL http://www.ais.uni-bonn.de/download/datasets.html

[19] Yang, H.: Python captcha. 2017, [Online; navštíveno 16-04-2017].
URL https://pypi.python.org/pypi/captcha/0.1.1

[20] Yangqing Jia: Blobs, Layers, and Nets: anatomy of a Caffe model. 2017, [Online; navštíveno 16-04-2017].
URL http://caffe.berkeleyvision.org/tutorial/net_layer_blob.html

[21] Yangqing Jia: Caffe, Deep Learning Framework. 2017, [Online; navštíveno 16-04-2017].
URL http://caffe.berkeleyvision.org/

[22] Zdeněk Černín: Captcha - jak znechutit roboty a neodradit uživatele. 2017, [Online; navštíveno 16-04-2017].
URL http://www.cognito.cz/technologie/captcha-jak-znechutit-roboty-a-neodradit-uzivatele/

Příloha A

Obsah DVD

- **compiled** - složka obsahuje zkompilovaný program se všemi potřebnými knihovnami
- **src** - složka obsahuje zdrojové kódy všech aplikací
- **src_latex** - složka obsahuje zdrojové kódy písemné práce
- **data** - složka obsahuje:
  - testovací a trénovací sady pro použité klasifikátory
  - předtrénované soubory pro klasifikaci
  - video tutorial
- **manual.txt** - soubor, obsahující základní informace pro spuštění aplikace

Příloha B

Testované webové stránky na stažení CAPTCHA datové sady

- http://jafty.com/captcha/test.php
- http://www.jobmaster.cz/include/captcha/captcha.php?.png
- http://www.sladkymeda.cz/uzivatel/registrace
- http://www.zpovednice.cz/captcha_more.php
- http://www.zpovednice.cz/vlastnizp.php
- https://www.vmworld.com/myvmworld-create!input.jspa
- https://www.o2.cz/osobni/319895-captcha/
- http://tv.nova.cz/registrace
- http://voyo.nova.cz/bin/captcha_image.php?form_field_name=voyoUserRegCap
- https://www.workaway.info/pages/captcha.php?r=6404130
- http://contest.mmosite.com/index.php/activity/sendkey/neverwinter_gift_pack_giveaway
- https://muj.idnes.cz/Registrace.aspx
- http://www.fitnesstv.cz/jak-si-vytvorit-vlastni-sestavu-videi
- http://www.converthub.com/
- http://www.kampomaturite.cz/prava/
- http://smite.clanweb.eu/registration/
- https://my.vmware.com/web/vmware/registration
- http://www.promedica-praha.cz/new-registration.html
- http://www.jdipracovat.cz/velka-prazdninova-fotosoutez-hlasovani