

Katedra informatiky  
Přírodovědecká fakulta  
Univerzita Palackého v Olomouci

# DIPLOMOVÁ PRÁCE

Prolamování CAPTCHA zabezpečení



2019

Bc. Kamil Hanus

Vedoucí práce: Mgr. Martin  
Trnečka, Ph.D.

Studijní obor: Aplikovaná informatika,  
prezenční forma

## **Bibliografické údaje**

Autor:	Bc. Kamil Hanus
Název práce:	Prolamování CAPTCHA zabezpečení
Typ práce:	diplomová práce
Pracoviště:	Katedra informatiky, Přírodovědecká fakulta, Univerzita Palackého v Olomouci
Rok obhajoby:	2019
Studijní obor:	Aplikovaná informatika, prezenční forma
Vedoucí práce:	Mgr. Martin Trnečka, Ph.D.
Počet stran:	30
Přílohy:	1 CD/DVD
Jazyk práce:	český

## **Bibliographic info**

Author:	Bc. Kamil Hanus
Title:	Breaking the CAPTCHA
Thesis type:	master thesis
Department:	Department of Computer Science, Faculty of Science, Palacký University Olomouc
Year of defense:	2019
Study field:	Applied Computer Science, full-time form
Supervisor:	Mgr. Martin Trnečka, Ph.D.
Page count:	30
Supplements:	1 CD/DVD
Thesis language:	Czech

## **Anotace**

*Výsledkem práce je webová aplikace demonstrující vliv různých grafických operací na segmentaci symbolů z obrazu. Zároveň prolamuje známé druhy CAPTCHA kódů.*

## **Synopsis**

*Result of this thesis is web application presenting influence of different graphical operations to character segmentation from image. Application also breaks known types of CAPTCHA codes.*

**Klíčová slova:** strojové učení; prolamování CAPTCHA

**Keywords:** machine learning; breaking CAPTCHA

Děkuji trpělivým.

*Místopřísežně prohlašuji, že jsem celou práci včetně příloh vypracoval/a samostatně a za použití pouze zdrojů citovaných v textu práce a uvedených v seznamu literatury.*

datum odevzdání práce

podpis autora

# Obsah

<b>1</b>	<b>Úvod</b>	<b>7</b>
<b>2</b>	<b>Captcha</b>	<b>8</b>
2.1	Historie . . . . .	8
2.2	Druhy kódů . . . . .	8
2.3	Možnosti prolamování . . . . .	11
<b>3</b>	<b>Captcha Breaker</b>	<b>13</b>
3.1	Použité technologie . . . . .	14
3.2	Uživatelská část . . . . .	15
3.3	Administrační rozhraní . . . . .	16
3.4	Podporované grafické operace . . . . .	18
3.5	Extrakce symbolů . . . . .	22
3.6	Tvorba klasifikátoru . . . . .	23
3.7	Úspěšnost implementované metody prolamování . . . . .	23
3.8	Další vývoj . . . . .	24
	<b>Závěr</b>	<b>26</b>
	<b>Conclusions</b>	<b>27</b>
<b>A</b>	<b>Obsah přiloženého CD/DVD</b>	<b>28</b>
	<b>Seznam zkratk</b>	<b>29</b>
	<b>Literatura</b>	<b>30</b>

## Seznam obrázků

1	Příklad obrázku, které generovala služba reCAPTCHA (verze 1) obsahující text „Chomsky fake“.	9
2	Příklady CAPTCHA vyžadující označení specifických objektů.	10
3	Příklady CAPTCHA vyžadující minimální interakci s uživatelem.	10
4	Varianta GeeTest Slide používaná na webu <a href="https://binance.com">binance.com</a>	11
5	Rozpoznaný kód.	16
6	Konfigurace extrakce symbolů z datasetu.	18
7	Vlevo původní obrázek, vpravo po černobílá verze po transformaci.	19
8	Vlevo původní obrázek, vpravo po aplikaci operace oříznutí.	19
9	Vlevo původní obrázek, vpravo po aplikaci Otsova prahování.	19
10	Vlevo původní obrázek, vpravo po aplikaci inverze.	19
11	Vlevo původní obrázek, vpravo po aplikaci uživatelského prahování v rozsahu 0 – 50.	20
12	Vlevo původní obrázek, vpravo po odstranění spojitých ploch.	20
13	Vlevo morfologické otevření s jádrem tvaru čtverec, vpravo stejná operace s jádrem tvaru kříž. Symbol 5 je vlevo poškozený.	21
14	Vlevo původní obrázek, vpravo po aplikaci morfologické eroze.	21
15	Vlevo původní obrázek, vpravo po aplikaci morfologické dilatace.	22
16	Vlevo původní obrázek, vpravo po aplikaci morfologického uzavření.	22
17	Vlevo původní obrázek, vpravo po aplikaci morfologického otevření.	22

## Seznam tabulek

1	URL endpointy pro globální namespace	16
2	Argumenty operace oříznutí.	19
3	Argumenty uživatelského prahování.	20
4	Argumenty morfologických operací.	21
5	Argumenty operace škálování.	22
6	Úspěšnost rozpoznání jednotlivých symbolů CAPTCHA kódů metodou této navrženou v této práci.	24
7	Úspěšnost rozpoznání jednotlivých symbolů CAPTCHA kódů v [4].	24

# 1 Úvod

V posledních letech urazila technologie strojového učení a umělé inteligence velký posun vpřed. Ačkoliv historie vzniku termínu strojové učení sahá do 60. let 20. století, masová adaptace nastává až v posledních několika letech, kdy se lze s pojmy umělá inteligence či AI (z anglického *artificial intelligence*) <sup>1</sup> setkávat stále častěji. Můžeme se domnívat, že jistou spojitost s nárůstem zájmu o strojové učení má i zpřístupnění open-source nástrojů, umožňujících snadnou aplikaci příslušných algoritmů, široké veřejnosti <sup>2</sup>.

Cílem této diplomové práce je rozšíření autorových znalostí v oblasti umělé inteligence. Jako vhodný problém na kterém lze demonstrovat potenciál strojového učení se naskýtá prolamování CAPTCHA zabezpečení. Aktuálně dostupné nástroje strojového učení poskytují elegantní způsob jak prolamovat klasické CAPTCHA zabezpečení. Zřejmě proto lze pozorovat vývoj nových typů CAPTCHA zabezpečení, které využívají některé prvky umělé inteligence. Diplomovou práci lze rozdělit do dvou částí – první se zabývá samotným problémem CAPTCHA zabezpečení, jeho historií a možným prolamováním. Druhá popisuje aplikaci nazvanou CaptchaBreaker, která popisuje možnost jak lze v rozumném čase řešit problém dekodování některých typů CAPTCHA kódů.

---

<sup>1</sup>[https://trends.google.com/trends/explore?date=all&q=%2Fm%2F01hyh\\_](https://trends.google.com/trends/explore?date=all&q=%2Fm%2F01hyh_)

<sup>2</sup>[https://trends.google.com/trends/explore?date=all&q=%2Fm%2F01hyh\\_,%2Fg%2F11gd3905v1,%2Fg%2F11bwp1s2k3,%2Fg%2F11c1r2rvnp](https://trends.google.com/trends/explore?date=all&q=%2Fm%2F01hyh_,%2Fg%2F11gd3905v1,%2Fg%2F11bwp1s2k3,%2Fg%2F11c1r2rvnp)

## 2 Captcha

### 2.1 Historie

Za akronymem CAPTCHA je skryt anglický text „**C**ompletely **A**utomated **P**ublic **T**uring test to tell **C**omputers and **H**umans **A**part“. Obecně je principem testu vygenerování určitého typu dotazu a následné ověření odpovědi. Typicky se technologie využívá ve formulářích webových stránek. Na základě verifikace odpovědi je rozhodnuto, zda systém komunikuje s reálným uživatelem nebo robotem. Některé zdroje <sup>3</sup> uvádějí pouze přepis textu z deformovaného obrazu do vstupního pole, což lze považovat za zavádějící, jelikož existuje několik dalších variant.

Důvod pro aplikaci takové ochrany je prostý. Pokud je umožněno na web zaslat libovolné požadavky bez omezení (například příspěvky do diskuzních fór, zakládání uživatelských účtů, pokusy o přihlášení, atd.), lze očekávat využití příležitosti útočníkem, který začne systém zahlcovat s úmyslem zamezit přístupnosti či šíření spamu. Ovšem je nutné mít na paměti, že nutnost prokazovat svou autentičnost je pro uživatele snížením komfortu. Proto se objevují technologie které se snaží omezit interakci s uživatelem na minimum.

Zajímavým použitím technologie byla digitalizace archivu článků deníku The New York Times či literárních děl společností Google [1]. Slova, která nebylo možné strojově rozpoznat s dostatečnou přesností byla dekodována s použitím nástroje reCAPTCHA. Metoda spočívala vždy v zobrazení obrázků dvou slov v kódu, přičemž u jednoho z nich byl známý jeho obsah. Pokud uživatel zadal správně známou část, o druhé také uvažovalo že je zadaná správně. Samozřejmě takové řešení by bylo příliš naivní a proto lze usuzovat, že za popsáním procesem existuje ještě další ověřovací mechanismus, který porovnává odpovědi uživatelů pro konkrétní slova a přiřadí jim váhu dle jejich výskytu.

### 2.2 Druhy kódů

#### Leetspeak

Na úvod je vhodné zmínit tzv. *leetspeak*<sup>4</sup>. V raných fázích internetu se jednalo o přepis znaků, které s užitím jisté míry představivosti tvořily smysluplný text (např. *N04m* jako ekvivalent slova *Noam*), avšak pro útočníka bez znalosti generujícího mechanismu bylo obtížné automatizovat vyplňování kontrolních otázek, respektive sledovat klíčová slova v komunikaci. Ochranu pomocí *leetspeak* bylo možné potkat např. na některých IRC <sup>5</sup> kanálech.

#### Jednoduchá otázka

Velmi triviální metodou, kterak lze implementovat CAPTCHA zabezpečení, je použít jednoduchou logickou otázku. Jedná se například o dotazy typu *Jaký je*

<sup>3</sup><https://cs.wikipedia.org/wiki/CAPTCHA>

<sup>4</sup><https://en.wikipedia.org/wiki/Leet>

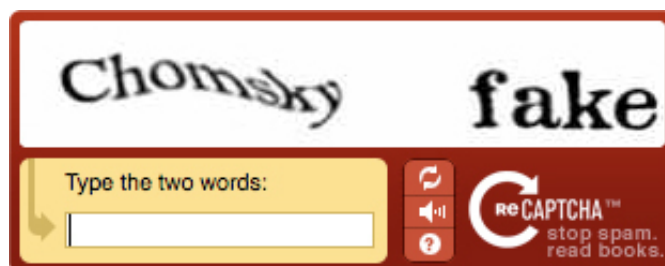
<sup>5</sup>[https://en.wikipedia.org/wiki/Internet\\_Relay\\_Chat](https://en.wikipedia.org/wiki/Internet_Relay_Chat)



*dnes den?*, respektive *Kolik je 4 krát 6?*. Jako odpověď je očekáván jednoslovný název dne v týdnu, respektive celočíselná odpověď. Reálné využití popisuje [2]. V případě, že není použit sofistikovaný nástroj generující nové druhy otázek je řešení tohoto zabezpečení triviální. Vyvine-li útočník dostatečné úsilí, po několika vygenerování různých otázek lze popsat gramatiku CAPTCHA schématu. Naivním řešením je zkonstruovat mapovací tabulku jednotlivých slov na funkce, se navzájem aplikují. Vzhledem ke konečnosti generovaných výrazů je však možné, že vytvoření formální gramatiky popisující jazyk CAPTCHA schématu a následné vytvoření překladače by bylo robustnějším řešením.

### Zkreslený text

Stále běžně se vyskytujícím typem CAPTCHA zabezpečení je přepis textu z deformovaného obrazu uživatelem. Předpokládá se, že bot není schopný vlivem zkreslení provést extrakci jednotlivých symbolů z obrazu. Ztížit čtení obrazu je možné mnoha způsoby - rozostřením textu, spojením jednotlivých znaků, vložením zkreslujících křivek, vynecháním malých částí znaků, atd. V další kapitole se diplomová práce zaměřuje právě na možnost prolomení typů CAPTCHA využívající zkreslování textu v obraze.



Obrázek 1: Příklad obrázku, které generovala služba reCAPTCHA (verze 1) obsahující text „Chomsky fake“.

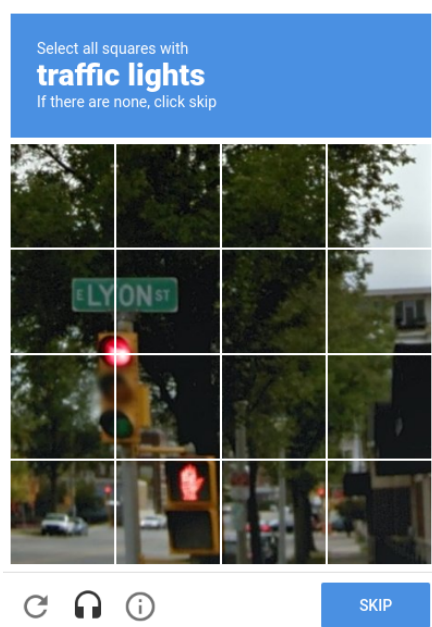
### Identifikace objektů v obraze

Uživatelsky komfortnějším přístupem, jak rozlišit člověka od počítače, je zobrazit uživateli množinu různých obrázků a požadovat výběr pouze těch, které obsahují nějaký předmět. Zaměříme-li se na technologii Google reCaptcha<sup>6</sup>, lze pozorovat dva přístupy k tomuto problému. První variantou je zobrazit uživateli 9 obrázků, ze kterých je například vyžadován výběr pouze těch obsahující dopravní značky. Druhý přístup je segmentace obrazu do několika částí a vyžadování výběru pouze těch, které obsahují určitý objekt (například auto). Využití podobného přístupu se také využívá k monetizaci obsahu majitelů webů, jelikož existují technologie používající CAPTCHA jako inzertní plochu<sup>7</sup>, respektive poskytují nástroj na

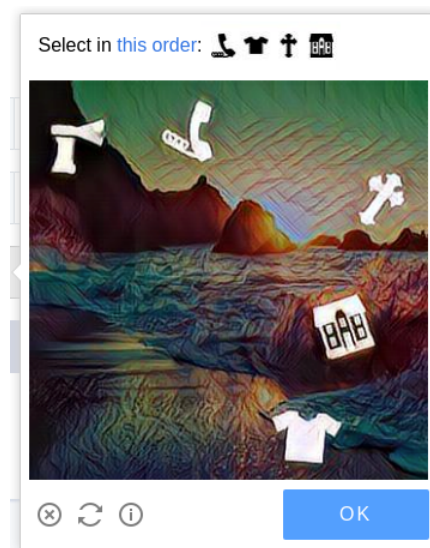
<sup>6</sup><https://www.google.com/recaptcha/>

<sup>7</sup><http://confidentcaptcha.com/>

kategorizaci velkých datasetů uživateli <sup>8</sup>.



(a) Výběr dlaždic obsahujících semafor.



(b) Výběr objektů v daném pořadí.

Obrázek 2: Příklady CAPTCHA vyžadující označení specifických objektů.

## noCAPTCHA

Ve snaze snížit interakci s uživatelem na minimum a tím i zvýšit komfort užívání webových stránek byly vyvinuty tzv. *noCAPTCHA* typy. Ukázkou takového přístupu je *No CAPTCHA reCAPTCHA* či *GeeTest Click*. V případě zmíněných typů musí návštěvník pouze kliknout na tlačítko. Veškerá verifikace uživatele je poté provedena na pozadí s využitím analýzy různých příznaků – cesta myši, doba kliku, otisk prohlížeče, atd. V případě nedostatečné jistoty je zobrazena složitější výzva.



(a) No CAPTCHA reCAPTCHA



(b) GeeTest Click

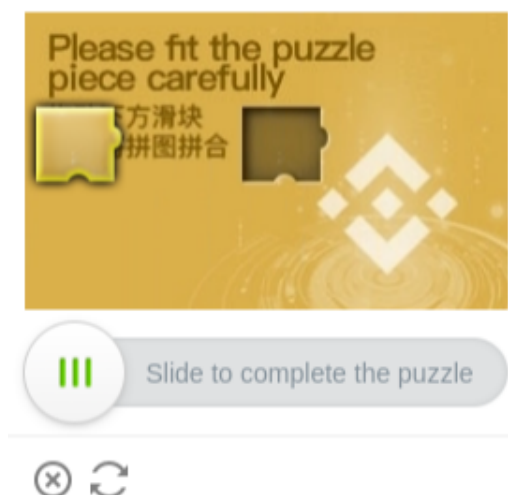
Obrázek 3: Příklady CAPTCHA vyžadující minimální interakci s uživatelem.

## Skládání puzzle

Další variantou CAPTCHA technologie pracující s obrazem nazvěme skládáním puzzle (společnost *GeeTest* <sup>4</sup> používá anglický název „slide“). Přístup spočívá

<sup>8</sup><https://hcaptcha.com/>

v přesunu části obrazu ve tvaru dílku puzzle zpět na své původní místo pomocí posuvníku. Během procesu ověření je sledováno několik faktorů ovlivňujících vyhodnocovací proces, stejně jako v předchozím uvedeném typu.



Obrázek 4: Varianta GeeTest Slide používaná na webu [binance.com](https://binance.com)

## Audio CAPTCHA

Zabezpečení pomocí textové/obrazové verifikace má nevýhodu v obtížném vyplňování osobami se zrakovým postižením. Právě proto je vhodné takovým uživatelům nabídnout alternativní možnost (ikonu obecně užívanou pro audio lze nalézt v 1 nebo 2a ) ověření, aby mohli využívat chráněné služby. Často se tomu děje pomocí tzv. audio CAPTCHA, kdy jsou v nahrávce zakódovány znaky abecedy a uživatel je vyzván k jejich přepisu. V audionahrávkách lze nalézt jisté metody zkreslující zvuk, aby bylo možné předejít automatizovanému zpracování.

## 2.3 Možnosti prolamování

### Nalezení chyby v implementaci

Triviálním způsobem prolomení CAPTCHA kódů je využít chyby v implementaci generování výzev. Uživatel, který si říká Ak1T4, popisuje <sup>9</sup> vložení zahashované hodnoty odpovědi do formuláře spolu s typem obrázkové CAPTCHA. Využitím vhodných nástrojů a rainbow tabulek (tabulka obsahující dva sloupce - plain text a příslušnou hash) je možné s odpovídajícím hardware nalézt rychle správnou odpověď.

<sup>9</sup><https://medium.com/bugbountywriteup/bypassing-captcha-like-a-boss-d0edcc3a1c1>

## Outsourcing rozpoznání

Stále používanou variantou automatického rozpoznávání CAPTCHA kódů je pronájem služeb třetích stran<sup>10,11</sup>. Jde o webové služby, které typicky prostřednictvím svého API nabízejí placejícím zákazníkům rozpoznání reálným pracovníkem. Výhodou rozpoznání člověkem je adaptace na možné změny v typu zabezpečení zaručující zachování míry úspěšnosti. Avšak morálně pochybným faktem je, že tyto služby využívají pracovníky z rozvojových zemí (například Venezuely, Indonésie, Vietnamu, ...), kteří jsou ohodnoceni velmi nízkou mzdou (hodinová sazba se uvádí v rozpětí 0.2 USD až 0.8 USD).

## Strojové učení

V závěru se dostáváme k metodám, které jsou z informatického pohledu zajímavé. Díky pokroku oboru strojového učení a kvalitě softwaru umožňujícího s technikami oboru pracovat je možné nechat vykonat práci rozpoznání CAPTCHA, kterou prováděl v předchozím odstavci lidský pracovník, strojem. Techniky se zejména liší v přístupu, jaké druhy kódů lze rozpoznávat. Zatímco [3] se snaží o vytvoření univerzálního algoritmu prolamujícího co největší množství různých CAPTCHA kódů, [4] naopak přistupuje k problému minimalisticky a jednoduchou metodou rozpoznává konkrétní typy kódů.

---

<sup>10</sup><https://anti-captcha.com>

<sup>11</sup><https://2captcha.com>

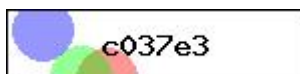
### 3 Captcha Breaker

Cílem diplomové práce je kromě stručného úvodu do problematiky CAPTCHA zabezpečení také demonstrace možnosti jeho prolamování. Jako vhodný typ se jeví rozpoznání zkresleného textu v obraze. Jelikož hlavní inspirace pocházela z [4], některé typy CAPTCHA vybrané k prolomení byly zvoleny shodně, aby bylo možné porovnat úspěšnost odlišných postupů rozpoznání extrahovaných symbolů. Aplikace demonstruje prolamování na typech CAPTCHA vyskytujících se na webech:

**adiseet.mfcr.cz** Daňový portál Finanční správy České republiky.



**kamody.cz** Zástupce typického internetového e-shopu.



**mojedatovaschranka.cz** Portál určený pro elektronickou komunikaci s úřady.<sup>12</sup>



**telerik.com** Framework pro platformu ASP.NET.



**ulozto.cz** Oblíbená služba pro sdílení souborů.



Jelikož je uživatelská příprava datasetu poměrně náročný úkol, kromě zmíněného algoritmu byla vytvořena webová aplikace, která umožňuje spustit průvodce tvorby datasetu nebo klasifikátoru. Zároveň je na stejném webu možné zkusit rozluštit známé typy CAPTCHA schéma.

<sup>12</sup>V průběhu tvorby diplomové práce došlo k zásadní aktualizaci webové aplikace <https://mojedatovaschranka.cz> a původní CAPTCHA zabezpečení bylo nahrazeno technologií Google reCAPTCHA.

## 3.1 Použité technologie

### Python

Populární interpretovaný jazyk podporující různá programovací paradigmaty. V současnosti se řadí mezi nejpopulárnější programovací jazyky <sup>13</sup>, zároveň v oblasti strojového učení poskytuje všechny potřebné knihovny. Nevýhodou jazyka je obtížnost paralelního zpracování dat, což je řešeno knihovnami třetích stran.

### Flask

Flask je microframework určený pro tvorbu webových aplikací napsaný v jazyce Python. Samotné jádro frameworku v základu obsahuje s nadsázkou pouze nástroje pro směrování požadavků a šablonovací systém Jinja. Vyžadujeme-li funkcionalitu navíc (například komunikaci s DB, ORM mapování, validaci formulářů nebo autentizaci požadavků), je nutné doinstalovat patřičný modul. Framework se tedy snaží být minimalistický a je pouze na uvážení vývojáře, jaké moduly k jádru frameworku dodá.

### PostgreSQL

Pro perzistentní uložení informací bylo nutné zvolit některý z relačních databázových systémů. V aplikaci bylo zvoleno PostgreSQL, zejména kvůli jeho popularitě a podpoře JSON datového typu.

### Celery

Důležitým požadavkem pro některé aplikace je možnost vykonávat některé operace asynchronně. K tomu lze v případě Flasku, respektive Pythonu obecně, použít knihovnu Celery. Jedná se distribuovaný systém sloužící pro zpracování zasílaných zpráv, které předávány skrze nějakého z podporovaných prostředníků, tzv. brokerů. I když primárně míří na zpracování v reálném čase, podporuje například zařazení obdržených zpráv do fronty a jejich sekvenční zpracování.

Knihovna umožňuje spustit libovolnou Python metodu jako úlohu v separátním vlákně, je-li obalena tzv. dekorátorem `@celery.task`. V aplikaci je takové chování využito k předání informací o klasifikátoru, který je potřeba natrénovat.

### RabbitMQ

Jako broker pro komunikaci s Celery byl použit RabbitMQ. Jedná se o úložiště pracující primárně v operační paměti, nicméně lze jej nakonfigurovat tak, aby ukládal data na pevný disk a po restartu zařízení byl schopen obnovit svůj stav.

---

<sup>13</sup><https://insights.stackoverflow.com/survey/2019>

## PyTorch

Open-source knihovna zaměřená na strojové učení v jazyce Python. Byť se jedná o volbu méně populární než například TensorFlow<sup>14</sup>, pro potřeby vyvinuté aplikace je plně dostačující.

## jQuery

jQuery je již několik let nejpoužívanější javascriptovou knihovnou<sup>15</sup> vůbec. Mezi její silné stránky patří snadná manipulace s DOM nebo obsluha AJAX requestů. Těchto vlastností využívá zejména průvodce tvorbou datasetu v administrační části aplikace.

## Bootstrap

Spolu s jQuery tvoří CSS knihovna Bootstrap základ velkého množství webů. Programátor je s jejím použitím odstíněn od nutnosti stylovat prvky webové stránky a je možné se více soustředit na samotný vývoj backendu.

## 3.2 Uživatelská část

Část aplikace dostupná neautorizovanému uživateli obsahuje výčet známých typů CAPTCHA kódů, které je možné prolomit spolu s formulářem pro nahrání uživateleova obrázku. Po výběru obrázku k prolomení a příslušného klasifikátoru je možné zaslat dotaz na server, který z obrázku dekoduje znaky pomocí vybraného klasifikátoru, což je řešeno asynchronním dotazem na backend aplikace. Nastane-li chyba, je uživateli zobrazena příslušná hláška. V případě úspěšného rozpoznání pouze obsah dekodovaný ze zasláního obrázku.

Zasílání dotaz na server obsahuje pouze dva parametry - ID klasifikátoru a obrázek zakódovaný do Base64. Jakmile server obdrží požadavek, nejprve zkontroluje existenci klasifikátoru. Pokud přijal špatnou hodnotu, informuje o takové situaci uživatele chybovou odpovědí. Následně jsou z vstupního obrázku extrahovány jednotlivé symboly dle konfigurace datasetu zvoleného klasifikátoru. V posledním kroku jsou extrahované symboly předány klasifikátoru k rozpoznání. V odpovědi je poté předána dekodovaná hodnota.

---


<sup>14</sup><https://towardsdatascience.com/deep-learning-framework-power-scores-2018-23607ddf297a>

<sup>15</sup>[https://w3techs.com/technologies/overview/javascript\\_library/all](https://w3techs.com/technologies/overview/javascript_library/all)

# CaptchaBreaker


Select image

15964.png




☒

eet




☐

radcaptcha



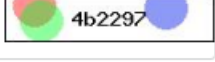
☐

ulozto




☐

kamody



☐

datovka



Submit

Recognized code is 15964.

Obrázek 5: Rozpoznaný kód.

```
{  
  "message":  
    "Unknown classifier",  
  "status":  
    "error"  
}
```

Chybová odpověď

```
{  
  "message":  
    "15964",  
  "status":  
    "success"  
}
```

Odpověď v případě rozpoznání znaků

URL	metoda	význam
/	GET	domovská stránka uživatelské části
/decode/	POST	AJAX dekodování CAPTCHA

Tabulka 1: URL endpointy pro globální namespace

## 3.3 Administrační rozhraní

Část webu dostupná pouze administrátorovi je dostupná na URL `/dashboard`. Všechny dotazy přicházející na adresy v tomto jmenném prostoru musejí být autentizovány. Jelikož jde pouze o demonstrační webovou aplikaci, není zde řešen přístup několika autorizovanými uživateli, případně různé víceúrovňové kontroly přístupu. Přihlašovací údaje jsou kontrolovány vůči konstantám `APP_USERNAME`,



respektive APP\_PASSWORD v konfiguračním souboru aplikace. Bezpečnost takového přístupu lze dále zvýšit například restrikcí IP adres u dotazů přicházející na chráněné endpointy v konfiguraci použitého aplikačního serveru.

### Nástěnka **/dashboard/overview/**

Na nástěnce jsou zobrazeny statistiky příchozích dotazů na rozpoznání obsahu CAPTCHA obrázků a počtu datasetů, resp. klasifikátorů. Existují-li navíc klasifikátory, které se momentálně trénují, jsou na nástěnce zobrazeny informace o stavu procesu. Kromě názvu klasifikátoru je zobrazeno pořadí aktuální iterace, respektive informace o fázi verifikace. Data o trénovaných klasifikátorech jsou pravidelně aktualizována v intervalu 5 vteřin pomocí AJAX dotazů. Po natrénování klasifikátoru je informace i něm z nástěnky odstraněna.

### Nahrání datasetu **/dashboard/datasets/new/**

Proces nahrání datasetu je zahrnut v jednoduchém průvodci. Administrátor je nejprve vyzván k vybrání ZIP archivu se vzorovými CAPTCHA obrázky. Následně je mu umožněno přiřadit jednotlivým obrázkům texty, které jsou v nich obsaženy. V případě vynechání tohoto kroku se význam textů určuje podle názvu souborů. V posledním kroku je konfigurace grafických operací (obrázek 6), které zkonvertují vstupní obraz do podoby vhodné pro extrahování symbolů. Jakmile je administrátor spokojený s výsledkem, odešle dataset na server a je přesměrován na stránku s jeho detaily.

### Zobrazení datasetu **/dashboard/datasets/:id/**

Stručný přehled s informacemi o vytvořeném datasetu. V horní části stránky jsou zobrazeny informace o času vytvoření, obsažených znacích a celkovém počtu rozpoznaných symbolů. Následuje výpis nahraných obrázků spolu se zobrazením extrahovaných symbolů. V poslední části je vypsána konfigurace extrakce ve formátu JSON.

### Tvorba klasifikátoru **/dashboard/classifiers/new/**

Formulář konfigurace klasifikátoru je parametrizován šesti vstupy - názvem klasifikátoru, zdrojovým datasetem, maximálním počtem iterací, velikostí *learning rate*, velikostí *momentum* a případným nastavením křížové validace. Po odeslání požadavku na vytvoření klasifikátoru je samotná fáze trénování spuštěna v asynchronním procesu na pozadí a administrátor je přesměrován na nástěnku, kde vidí průběžně aktualizované informace o trénování klasifikátoru.

### Detaily klasifikátoru **/dashboard/classifiers/:id/**

Stránka zobrazuje detaily konfigurace klasifikátoru spolu s informací o jeho konfiguraci. Po natrénování klasifikátoru je možné získat informace o hodnotách

# CaptchaBreaker

[logout](#)

[Overview](#)
[Datasets](#)
[Classifiers](#)

## New dataset

### Character extraction

Example image

Characters in captcha

#### Available operations

- Area Filter
- Crop
- Custom threshold
- Grayscale
- Inverse
- Morphological closing
- Morphological dilation
- Morphological erosion
- Morphological opening
- Otsu's threshold
- Scale

#### Selected operations

- Grayscale
- Otsu's threshold
- Inverse
- Morphological opening
 

kernel\_size:

iterations:

kernel\_shape:

#### Previews

Original

Grayscale

ThresholdOtsu

Inverse

Opening

Extracted

Made with by Kamil Hanus.  
[dashboard](#)

Obrázek 6: Konfigurace extrakce symbolů z datasetu.

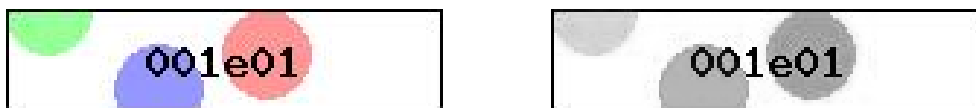
ztrátové funkce, respektive úspěšnosti křížové validace.

### 3.4 Podporované grafické operace

Aplikace momentálně podporuje několik grafických operací, které slouží k přípravě obrazu, ze kterého jsou extrahovány jednotlivé symboly. Jejich použití je nutné vzhledem k možnému použití metod, které zkreslují znaky zanesené do obrazu. Architektura aplikace je navržena tak, že jsou jednotlivé třídy grafických operací automaticky převoditelné do HTML kódu včetně svých argumentů. Zároveň implementují vlastní (de)serializaci, což umožňuje snadnou rozšiřitelnost webového průvodce tvorbou datasetu.

#### Stupně šedi (Grayscale)

Operace převádějící obraz z RGB spektra do stupňů šedi.



Obrázek 7: Vlevo původní obrázek, vpravo po černobílé verze po transformaci.

### Oříznutí (Crop)

V některých případech obsahuje vstupní obraz rámeček, který bez odstranění způsobuje selhání algoritmu nalezení největších ploch. Z toho důvodu byla přidána možnost oříznout vstupní obraz rovnoměrně ze všech stran.

Název argumentu	typ	význam
size	Integer	velikost ořezu

Tabulka 2: Argumenty operace oříznutí.



Obrázek 8: Vlevo původní obrázek, vpravo po aplikaci operace oříznutí.

### Otsovo prahování (Otsu's threshold)

Otsovo prahování se používá k automatickému převodu obrazu z odstínů šedi do binární černobílé podoby. Interně Otsovo prahování analyzuje histogram vstupního obrazu, dle kterého určuje následné rozdělení na popředí/pozadí obrazu.



Obrázek 9: Vlevo původní obrázek, vpravo po aplikaci Otsova prahování.

### Inverze (Inverse)

Inverze v aplikaci slouží k záměně popředí/pozadí obrazu. V některých případech užití Otsova prahování totiž dojde k jevu, kdy je část obrazu, která je hledaná v segmentačním algoritmu, označena jako pozadí. Jelikož je na vstupu očekávaný binární obraz, operace je implementována jako bitová negace.

**366bcd**

**366bcd**

Obrázek 10: Vlevo původní obrázek, vpravo po aplikaci inverze.

### Uživatelské prahování (Custom threshold)

Operace uživatelského prahování slouží k definování rozsahu hodnot pixelů, které jsou následně označeny jako popředí/pozadí a podle kterých je vstupní obraz převede do binární podoby. Operace je užitečná u typů CAPTCHA schémat, kdy je potřeba zachovat pouze části obrazu určité barvy. V případě použití Otsova prahování může totiž dojít k zachování nežádoucího šumu.

Název argumentu	typ	význam
lower_bound	Integer	spodní hranice hodnot popředí
upper_bound	Integer	horní hranice hodnot popředí

Tabulka 3: Argumenty uživatelského prahování.



Obrázek 11: Vlevo původní obrázek, vpravo po aplikaci uživatelského prahování v rozsahu 0 – 50.

### Odstranění spojitých ploch (Area Filter)

Jedná se o pomocnou operaci jejímž účelem je komfortnější obsluha průvodce tvorby datasetu. Operace odstraní plochy obrazu, které mají svou plochu menší než je uživatelem definovaná velikost, případně je jejich výška menší než definované velikost. Důsledkem je odfiltrování šumu z obrazu bez nutnosti použití morfologických operací, které by mohly s příliš vysokými argumenty nevhodně poškodit obraz.



Obrázek 12: Vlevo původní obrázek, vpravo po odstranění spojitých ploch.

### Morfologické transformace

Před samotným výčtem morfologických operací je vhodné zmínit, o jaký proces se jedná. Morfologická operace očekává na vstupu dva argumenty – obrázek a strukturální element (také nazývaný jádro). Pro zjednodušení uvažujme pouze binární obrázky, ačkoliv lze operace zobecnit také na barevné. Jádro i obrázek lze vyjádřit jako matice (v aplikaci je vždy použita čtvercová matice) s hodnotami z množiny 0, 1. Jádro je posouváno nad jednotlivými pixely vstupního obrázku, jejíž hodnotu mění dle dále popsanych vlastností. Samotné jádro je možné určit

několika tvary – jedná se o a) kříž b) kruh nebo c) čtverec. Všechny varianty jádra velikosti  $5 \times 5$  jsou vyobrazeny níže.

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Zároveň mají všechny morfologické operace shodné argumenty.

Název argumentu	typ	význam
kernel_size	Integer	velikost jádra $2 * kernel\_size + 1$
iterations	Integer	počet iterací operace
kernel_shape	ENUM	tvar jádra

Tabulka 4: Argumenty morfologických operací.



Obrázek 13: Vlevo morfologické otevření s jádrem tvaru čtverec, vpravo stejná operace s jádrem tvaru kříž. Symbol 5 je vlevo poškozený.

### Morfologická eroze (Morphological erosion)

Hodnota pixelu pod středem jádra zůstává zachována, jsou-li všechny pixely pod nenulovými body jádra také nenulové. V tomto případě dochází k odfiltrování šumu, vyhlazení hran a odstranění hrany obrazu.



Obrázek 14: Vlevo původní obrázek, vpravo po aplikaci morfologické eroze.

### Morfologická dilatace (Morphological dilation)

Dilatace představuje opačnou operaci k erozi. Hodnota pixelu pod středem jádra zůstává zachována, je-li alespoň jeden pixel pod nenulovými body jádra také nenulový. Dochází k rozšíření hran obrazu, případně zaplnění mezery mezi blízkými částmi obrazu.



Obrázek 15: Vlevo původní obrázek, vpravo po aplikaci morfologické dilatace.

### Morfologické uzavření (Morphological closing)

Dilatace následovaná erozí. Nejprve jsou okraje obrazu rozšířeny a zaplněny případné mezery, poté obraz eroduje a zmenší okraje na původní velikost.



Obrázek 16: Vlevo původní obrázek, vpravo po aplikaci morfologického uzavření.

### Morfologické otevření (Morphological opening)

V posledním případě jde o erozi následovanou dilatací. Po odfiltrování šumu a vyhlazení hran okraje obrazu dilatují do původních hodnot.



Obrázek 17: Vlevo původní obrázek, vpravo po aplikaci morfologického otevření.

### Škálování (Scale)

Operace zvětšující obraz celočíselným koeficientem. Jelikož mají morfologické operace v některých situacích na obrázcích s příliš nízkým rozlišením příliš destruktivní charakter, je vhodné vstupní obrázek zvětšit a provést několik iterací dané operace.

Název argumentu	typ	význam
scale_ratio	Integer	koeficient škálování

Tabulka 5: Argumenty operace škálování.

## 3.5 Extrakce symbolů

Extrakce symbolů se provádí nad obrázky z uživatelem nahraného datasetu v podobě ZIP archivu. Podporovány jsou všechny obrazové formáty, které podporuje knihovna OpenCV. Ke spuštění extrakce je také potřeba informace o počtu

hledaných symbolů v obraze a konfigurace zpracování obrazu. Volitelně je nahrán také slovník přiřazující jednotlivým souborům význam symbolů v nich uložených. Výsledkem jsou extrahované jednotlivé znaky do unifikovaného rozměru  $20 \times 20$  pixelů, které se následně ukládají do databáze.

### Možné kolize

Během procesu extrakce symbolů jsou v obraze hledány spojitě plochy pixelů, které mohou představovat symbol CAPTCHA kódu. Ne vždy je však ihned nalezeno požadované množství takových oblastí.

- **Byl nalezen velký počet spojitých ploch.** Proces předzpracování obrazu nemusí v některých případech vyčistit obraz v žádoucí kvalitě a proto může být nalezeno více ploch, spojitých ploch obrazu, které lze považovat za symbol. V takovém případě se pro každou takovou oblast vypočítá její velikost a zachová se pouze  $n$  největších, kde  $n$  značí počet hledaných symbolů.
- **Byl nalezen malý počet spojitých ploch.** Některé CAPTCHA obrázky mohou být natolik deformovány, že se symboly v nich obsažené navzájem prolínají. Takovou vadu lze například odstranit spočítáním tzv. lokálních minim v jednotlivých spojitých plochách. Každé takové lokální minimum se označí hodnotou určující rovnoměrnost rozdělení obsaženého symbolu. Dokud není určeno  $n$  oblastí obsahující symbol, vybere se a rozdělí taková spojitá plocha, jejíž ohodnocení rozdělení je maximální.

## 3.6 Tvorba klasifikátoru

Klasifikátorem se rozumí algoritmus, jehož účelem je přiřadit vstupnímu prvku správnou kategorii na základě trénovacích dat, se kterými byl algoritmus trénován. Konkrétně v případě rozpoznání symbolů v navrženém algoritmu se jedná o přiřazení indexu v poli známých symbolů datasetu. Klasifikátor může být implementován různými způsoby. Vyvinutá aplikace používá konvoluční neuronovou síť, jejíž architektura je patrná z příslušné třídy.

Uživatel může v konfigurátoru klasifikátoru nastavit parametry omezující počet iterací, určit velikost *learning rate*, *momentum* a případně zapnout vykonání křížové validace. Doba trénování klasifikátoru se odvíjí od použitého hardware, lze však konstatovat, že vytvořit klasifikátor pro použité datasety trvá maximálně několik minut.

## 3.7 Úspěšnost implementované metody prolamování

Pro ověření úspěšnosti navržené metody prolamování CAPTCHA kódů bylo z webů uvedených v úvodu kapitoly staženo a manuálně označeno 400 různých obrázků, ze kterých byly vytvořeny datasety. Množství trénovacích dat se může

na první pohled zdát být malé v porovnání s populárním datasetem MNIST, avšak podle výsledků křížové validace je pro účely diplomové práce postačující.

Název webu	Úspěšnost
adiseet.mfcr.cz	0.9610
kamody.cz	0.9983
mojedatovaschranka.cz	0.9915
telerik.com	0.7705
ulozto.cz	0.3568

Tabulka 6: Úspěšnost rozpoznání jednotlivých symbolů CAPTCHA kódů metodou této navrženou v této práci.

Porovnáme-li úspěšnosti rozpoznání symbolů v tabulce 6 oproti tabulce 7, lze pozorovat mírný nárůst úspěšnosti prolamování, resp. klasifikace konvoluční neuronovou sítí oproti algoritmu k-NN.

Název webu	Úspěšnost
mojedatovaschranka.cz	0.82
ulozto.cz	0.14

Tabulka 7: Úspěšnost rozpoznání jednotlivých symbolů CAPTCHA kódů v [4].

Zároveň lze pozorovat patrný význam aplikace zkreslujících křivek na popředí CAPTCHA typu použitého na webu <https://adiseet.mfcr.cz>. V porovnání s obrázky použitými na webu <https://mojedatovaschranka.cz>, kde je CAPTCHA typ až na umístění zkreslujících křivek shodný, dochází k poklesu úspěšnosti rozpoznání přibližně o tři procenta.

### 3.8 Další vývoj

Ultimativní nástroj pro lámání CAPTCHA kódů, byť pouze obrázkových, nebyl vytvořen. Implementace následujících funkcionalit, zejména v administračním rozhraní, by výrazně zlepšila komfort užívání a zřejmě i praktičnost systému jako celku.

#### Upozornění na chybně klasifikované znaky

Na stránce zobrazující detaily klasifikátoru se naskýtá možnost zobrazit upozornění na možné chybně klasifikované symboly. Taková chyba může nastat zejména v následujících třech případech:

- Příliš podobné znaky (číslíce 0 versus písmeno O).
- Algoritmus pro extrakci symbolů na výstupu nevrátí celý znak, ale například pouze jeho část.



- Symbol byl ručně chybně označen a klasifikátor jej rozpoznává správně.

### **Změna označení již nahraných obrázků**

Během procesu vytváření datasetu musí administrátor nahrát ZIP archiv obsahující obrázky jejichž název odpovídá CAPTCHA kódu, nebo je procesem označení obrázků proveden před uploadem na server. V obou případech však závisí jen a pouze na lidském faktoru, zda bude obrázku přiřazen správný nebo chybný text. Z toho důvodu je vhodné umožnit změnu označení obrázku, resp. samotného symbolu.

### **Odstranění některých znaků**

V některých případech není možné segmentovat jednotlivé symboly z datasetu se stoprocentní úspěšností. Aby se trénovaná neuronová síť neučila na chybně označených datech, je vhodné umožnit odstranit z datasetu ty symboly, které byly špatně extrahovány.

### **Volba z více klasifikátorů**

V současné době systém používá jako klasifikátor konvoluční neuronovou síť. Takový klasifikátor nemusí vyhovovat všem rozpoznávaným typům CAPTCHA obrázků. Rovněž pro účely porovnání úspěšnosti různých klasifikátorů je rozumné přidat možnost volby.

### **Klasifikátor používající více datasetů**

K dalšímu zlepšení rozpoznávání by mohlo také vést trénování klasifikátoru znaky z více zdrojů. Kromě primárního datasetu, který udává i proces extrakce symbolů, by k jeho množině znaků byly přidány ty symboly s odpovídající hodnotou ze zvolených datasetů. Tím by se docílilo rozpoznání transformací, které v trénovací množině symbolů nebyly prve zahrnuty.

## Závěr

Vyvinutá webová aplikace úspěšně prolamuje některé typy známých CAPTCHA kódů. Existuje však mnoho prostoru pro další vývoj, který by usnadnil rozpoznávání dalších typů zabezpečení. Retrospektivně je také vhodné zmínit, že volba frameworku Flask se zdá být velice nešťastná vzhledem k ne příliš pohodlné práci s objektově-relačním mapováním nebo tvorbě administračního rozhraní. Přepsání webové aplikace například do Ruby on Rails se zachováním části obstarávající strojové učení v jazyce Python by mohlo být optimálnější pro snadnou udržitelnost aplikace.

## Conclusions

Developed web application successfully breaks certain types of known CAPTCHA codes. However there is many possibilities of further development, which would make recongizing of other security types easier. It is also worth mentioning that retrospectively the choice of the Flask framework seems to be very unfortune due to not so comfortable work with object-relational mapping or creation of administration interface. Rewriting the code base of web application e.g. into Ruby on Rails with retaining machine learning part in Python language would be more feasible because of better sustainability of the application.

## A Obsah přiloženého CD/DVD

### **doc/**

Text práce ve formátu PDF, vytvořený s použitím závazného stylu KI PřF UP v Olomouci pro závěrečné práce, včetně všech příloh, a všechny soubory potřebné pro bezproblémové vygenerování PDF dokumentu textu (v ZIP archivu), tj. zdrojový text textu, vložené obrázky, apod.

### **src/**

Kompletní zdrojové texty webové aplikace CAPTCHABREAKER se všemi potřebnými (příp. převzatými) knihovnami. Dostupné také z <https://github.com/arthurwozniak/CaptchaBreaker>.

U veškerých cizích převzatých materiálů obsažených na CD/DVD jejich zahrnutí dovolují podmínky pro jejich šíření nebo přiložený souhlas držitele copyrightu. Pro všechny použité (a citované) materiály, u kterých toto není splněno a nejsou tak obsaženy na CD/DVD, je uveden jejich zdroj (např. webová adresa) v bibliografii nebo textu práce nebo v souboru `readme.txt`.



## Literatura

- [1] TECHCRUCH. *reCAPTCHA: Using Captchas To Digitize Books* / TechCruch. 2019. Dostupný také z: <https://techcrunch.com/2007/09/16/recaptcha-using-captchas-to-digitize-books/>.
- [2] TULEY, Rob. *TextCaptcha: textual CAPTCHA challenges*. 2019. Dostupný také z: <http://textcaptcha.com/>.
- [3] BURSZTEIN, Elie; AIGRAIN, Jonathan; MOSCICKI, Angelika; MITCHELL, John C. The End is Nigh: Generic Solving of Text-based CAPTCHAs. In. *WOOT'14 Proceedings of the 8th USENIX conference on Offensive Technologies*. 2014. Dostupný také z: <https://www.elie.net/publication/the-end-is-nigh-generic-solving-of-text-based-captchas>.
- [4] KOPP, Martin; PISTORA, Matous; HOLENA, Martin. How to Mimic Humans, Guide for Computers. In. *ITAT*. 2016.
- [5] PALLETS. *Welcome to Flask - Flask Documentation (1.1.x)*. 2019. Dostupný také z: <https://flask.palletsprojects.com/en/1.1.x/>.
- [6] PYTORCH. *Welcome to PyTorch Tutorials – PyTorch Tutorials 1.2.0 documentation*. 2019. Dostupný také z: <https://pytorch.org/tutorials/index.html>.
- [7] OPENCV. *OpenCV: OpenCV Tutorials*. 2019. Dostupný také z: [https://docs.opencv.org/master/d9/df8/tutorial\\_root.html](https://docs.opencv.org/master/d9/df8/tutorial_root.html).