Mendelova univerzita v Brně
Provozně ekonomická fakulta

Modul pro vyhledávání nevhodných obrázků

Diplomová práce

Vedoucí práce:
Ing. Ondřej Popelka, Ph.D.

Bc. Aleš Žurek

Brno 2015

Chtěl bych poděkovat všem lidem díky kterým tato diplomová práce vznikla. Hlavně bych pak chtěl poděkovat svému vedoucímu Ondřeji Popelkovi, který dodával cenné rady pro zpracování této diplomové práce, produktovému manažerovi Marku Vackovi s kterým byla diplomová práce dohodnuta ve společnosti Seznam.cz, a.s a Petru Bartůnkovi ze společnosti Seznam.cz, a.s. s kterým probíhala komunikace a konzultace. Dále bych chtěl poděkovat rodině a přátelům, kteří mě v této diplomové práci podporovali.

Čestné prohlášení

Prohlašuji, že jsem tuto práci: **Modul pro vyhledávání nevhodných obrázků** vypracoval samostatně a veškeré použité prameny a informace jsou uvedeny v seznamu použité literatury. Souhlasím, aby moje práce byla zveřejněna v souladu s § 47b zákona č. 111/1998 Sb., o vysokých školách ve znění pozdějších předpisů, a v souladu s platnou *Směrnicí o zveřejňování vysokoškolských závěrečných prací.*
Jsem si vědom, že se na moji práci vztahuje zákon č. 121/2000 Sb., autorský zákon, a že Mendelova univerzita v Brně má právo na uzavření licenční smlouvy a užití této práce jako školního díla podle § 60 odst. 1 Autorského zákona.
Dále se zavazuji, že před sepsáním licenční smlouvy o využití díla jinou osobou (subjektem) si vyžádám písemné stanovisko univerzity o tom, že předmětná licenční smlouva není v rozporu s oprávněnými zájmy univerzity, a zavazuji se uhradit případný příspěvek na úhradu nákladů spojených se vznikem díla, a to až do jejich skutečné výše.

V Brně dne 18. května 2015 .....

Abstract

Žurek, A. Module for explicit pictures searching, Brno, 2015.

This work is focused on classifying photos which are uploaded on dating service Lidé.cz. Pictures are classified into two categories based on whether they contain pornographic content or not. Convolutional neural networks are used for classification and these neural networks are taught by using Caffe framework. The results of this work fulfilled all requirements from Seznam.cz, a.s. company. Classification accuracy of the best model on created testing dataset with 5 643 photos was 93,64% and the time for classification of photography is low enough to perform classification in real time. The first part contains an analysis of the current approaches for image classification. The second part focuses on the analysis and draft of the solution and the third part describes the implementation of the solution and the testing of neural networks models.

Abstrakt

Žurek, A. Modul pro vyhledávání nevhodných obrázků, Brno, 2015.

Práce je zaměřena na klasifikování fotografií, které jsou nahrávány na seznamovací službu Lidé.cz. Fotografie jsou klasifikovány do kategorií závadné a nezávadné dle toho, jestli obsahují pornografický obsah. Pro klasifikaci je využito konvolučních neuronových sítí, které jsou učeny pomocí frameworku Caffe. Výsledek práce splnil všechny požadavky kladené společností Seznam.cz, a.s. Na vytvořeném datasetu s 5 643 fotografiemi bylo dosaženo přesnosti klasifikátoru 93,64% a doba klasifikace jedné fotografie je natolik nízká, aby se dala provádět v reálném čase. První část práce obsahuje analýzu současného stavu klasifikace fotografií. Druhá část je zaměřena na analýzu a návrh řešení a v třetí části je popsaná implementace řešení a průběh testování modelů neuronových sítí.

Obsah

1 Úvod

Diplomová práce vznikla ve spolupráci se společností Seznam.cz, a.s., která bude dále v této práci označena pouze jako Seznam.cz. Tato společnost vlastní seznamovací službu Lidé.cz, kde se uživatelé registrují a nahrávají své fotografie. Jelikož je registrace bezplatná a uživatelský účet není potřeba před nahráváním fotek nijak ověřovat, tak dochází k tomu, že uživatelé se registrují pod falešnými jmény a nahrávají závadné fotografie, které jsou na tomto diskusním fóru nepřípustné a musí být odstraněny.

Pod závadnými fotografiemi se myslí fotografie, které porušují pravidla služby Lidé.cz. Zde patří obecně fotografie, které vyobrazují činnost týkající se výdělků, odměn či sexuálních služeb (a to i se skrytým podtextem), komerční, erotický, pornografický, vulgární, rasistický či nacistický obsah. Dále jsou to fotografie zobrazující násilí, sebepoškozování či poškozování jiné osoby, psychotropní látky případně flóru jenž je může připomínat.

Nejvíce problematické jsou nyní dle pohledu společnosti Seznam.cz fotografie erotického a pornografického charakteru, kterých by se rádi zbavili. Tato diplomová práce je zaměřena na pornografické fotografie a proto pod pojmem závadná fotografie bude myšlena právě fotografie obsahující pornografický obsah.

2 Cíl práce

Cílem práce je vytvořit modul komunikující s rozhraním, které se nyní používá ve společnosti Seznam.cz. Pomocí tohoto rozhraní by měl modul komunikovat a pro zaslané fotografie vyhodnocovat jestli se jedná o fotografie závadné (pornografického charakteru) nebo o slušné fotografie, které mohou být nahrány. Úmyslem je použít tento modul při nahrávání fotografií na seznamovací službu Lidé.cz. Při nahrání fotografie se vyhodnotí závadnost fotografie a pokud bude fotografie označena jako závadná, tak uživateli systém nepovolí danou fotografii uložit a zobrazit na stránce Lidé.cz.

Tento modul by měl převážně usnadnit práci administrátorů tím, že část fotografií klasifikuje sám a zbývající fotografie, kde si nebude modul jistý, předá administrátorům k ručnímu schválení. Tímto se sníží celkový počet fotografií, které musí administrátoři kontrolovat a dojde tak ke snížení nákladů společnosti Seznam.cz, které musí vynaložit na ruční schvalování fotografií.

3 Současný stav

Ve společnosti Seznam.cz nyní úspěšně (s relativně malou chybou) kategorizují webové stránky dle textového obsahu. Konkrétně takto například zjišťují jestli se na dané stránce nachází erotický obsah a dle toho upravují výsledky ve vyhledávači. Tuto kategorizaci stránek by chtěli rozšířit o identifikaci závadných webových stránek pomocí klasifikace fotografií. Zároveň by chtěli mít možnost klasifikovat fotografie na seznamovací službě Lidé.cz, kde potřebují rozdělit fotografie na fotografie nezávadné a fotografie závadné, které obsahují erotickou tématiku. Protože aktuálně se na seznamovací službě Lidé.cz schvalují profilové fotografie ručně a ostatní (ne profilové) fotografie se neschvalují vůbec a jsou pouze nahlašovány uživateli.

Klasifikace obrazu je netriviální proces. Fotografie se stroji jeví jako množina barevných bodů, které jsou zapsány jako čísla a postrádají jakýkoliv smysl nebo uspořádání. Pouze mozek umí zpracovat dané barevné body uspořádané v matici a získat z nich informace jako jsou tvary a prostorové vyjádření.

Tomuto problému se věnuje vědní obor zvaný počítačové vidění. Obecně počítačové vidění znamená transformaci dat z fotografie nebo videa a převedení je do nové reprezentace. Všechny takové transformace jsou pak prováděny za nějakým účelem. Příkladem těchto účelů může být detekce objektů z obrazových dat, odstranění určité barvy z obrazu, stabilizace video obrazu a mnohé další [7]. Obor počítačového vidění patří mezi technologie *Soft computingu*.

Tradičně od počítače očekáváme přesné hodnoty, jisté výsledky a žádnou možnost odchylky. Hlavní myšlenkou soft computingu je to, že přesné dodržování pravidel brání přirozenému učícímu vývoji. Protože možnost tolerovat nepřesnost a nejistotu patří k pozoruhodným schopnostem člověka. Díky tomu můžeme porozumět zkreslené řeči, rozluštit špatně čitelný rukopis, řídit vozidlo v nepřehledném provozu a obecně dělat racionální rozhodnutí v prostředí plném nejistoty a nepřesností [35].

V tomto případě budeme očekávat, že naučený klasifikátor bude schopen z dodaných fotografií vyčíst jisté souvislosti a naučí se rozpoznávat jak vypadá pornografická fotografie. Ze zadání vyplývá, že klasifikátor má třídit fotografie do dvou kategorií. Pokud se provádí klasifikace do dvou kategorií, tak se tomuto procesu říká binární klasifikace. Při binární klasifikaci do dvou kategorií lze výsledky popsat pomocí termínů true positive, true negative, false positive a false negative [29].

Tyto termíny budou použity v této diplomové práci a jejich význam je následující:

true positive
: Závadná fotografie, která byla správně klasifikovaná jako závadná.

true negative
: Nezávadná fotografie, která byla správně klasifikovaná jako nezávadná.

false positive
: Nezávadná fotografie, která byla špatně klasifikovaná jako závadná.

false negative
Závadná fotografie, která byla špatně klasifikovaná jako nezávadná.

Umění a obrazy v lidech vyvolávají subjektivní dojem. Tak je tomu i u erotických a pornografických fotografií. Co je nezávadná fotografie a co fotografie závadná? I pro lidi je toto rozhodnutí nejednoznačné. I kdyby existovala přesně definovaná pravidla jak označit fotografii za pornografickou, tak by byl problém určit, jestli tato fotka splňuje všechny pravidla na 100%. Stačí posunout oblečení o 1 centimetr níže než povolují pravidla nebo stačí vyfotit fotku stejné osoby pouze z jiného úhlu a najednou se náš pohled na závadnost fotografie může lišit.

Z toho plyne, že je žádoucí využít klasifikátor, který se dokáže přizpůsobit vstupním datům a dokáže klasifikovat fotografie, i když nejsou jednoduše zařaditelné.

3.1 Současný stav v Seznam.cz

3.1.1 Seznamovací služba Lidé.cz

Lidé.cz je moderní seznamovací služba a prostor pro poznávání nových lidí. Služba nabízí možnost procházet, prohlížet a oslovovat ostatní. Uživatelé se na službě prezentují svými profily, které tvoří nejoblíbenější část služby. Navazovat známosti je možné buď skrze soukromou poštu, nebo formou veřejné komunikace na diskuzích. Na službě je kladen důraz na jednoduchost a přehlednost, stejně jako na svižnou a pohodlnou komunikaci. Lidé.cz má za cíl usnadnit a zpříjemnit krásný proces poznávání budoucích známostí, přátel či partnerů [45]. Ukázka služby Lidé.cz je zobrazena na obrázku 1.

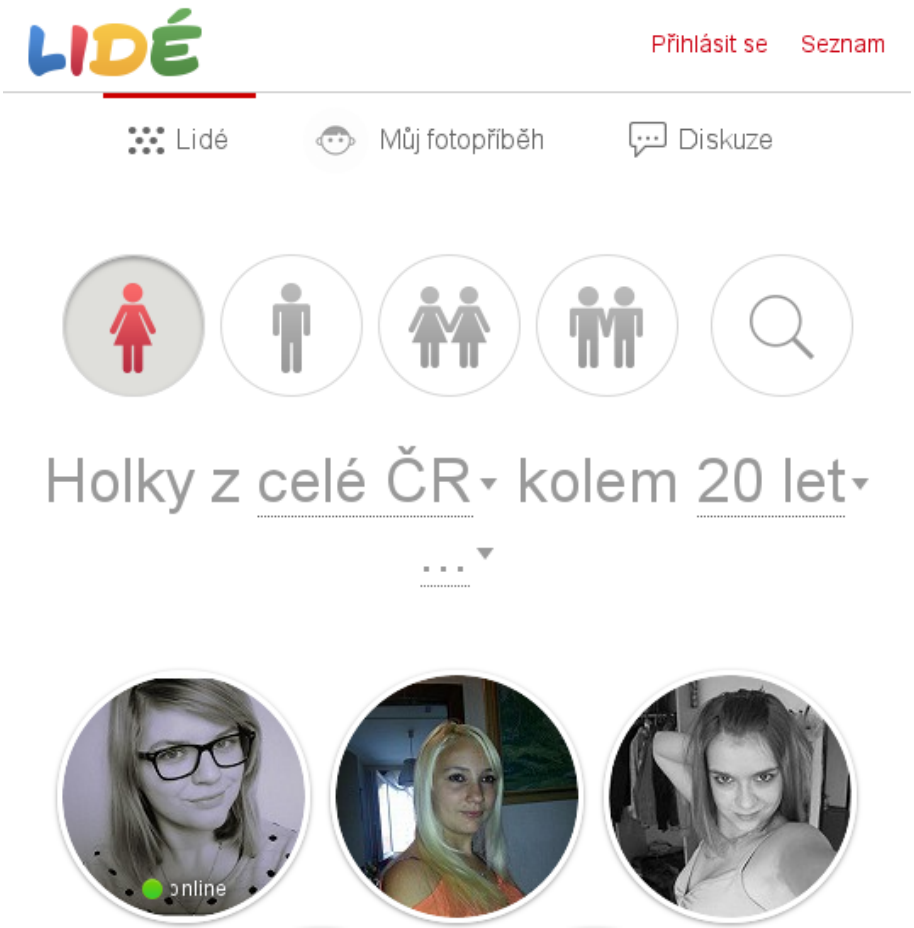
3.1.2 Ruční schvalování profilových fotek

Seznamovací služba Lidé.cz obsahuje skupinu pravidel, které musí profilová fotografie splňovat. Pokud uživatel nahrává profilovou fotografii, tak musí počkat až mu bude daná fotografie schválená někým z pracovníků společnosti Seznam.cz, kteří mají na starosti kontrolu a schvalování uživatelských fotografií. Toto schvalování může trvat někdy i několik hodin. Záleží v jakou denní dobu byla fotografie nahrána.

Výhodou je, že všechny fotografie jsou zkontrolovány člověkem a není možná chyba algoritmu nebo zvolené programové metody. Nevýhodou je velmi dlouhá doba schvalování fotografií a z dlouhodobého hlediska vysoké náklady, které se odvíjí z platu daných zaměstnanců, kteří musí kontrolovat fotografie.

3.1.3 Nahlašování závadných fotografií

Ostatní fotografie na profilu uživatele neprocházejí procesem schvalování a uživatelé, tak mohou nahrát v podstatě libovolnou fotografii. To se ukazuje jako problém, protože uživatelé nahrávají fotografie porušující pravidla seznamovací služby Lidé.cz. Aktuálně jedinou možností jak smazat urážlivou nebo jinak závadnou fotografii je



Obrázek 1: Seznamovací služba Lidé.cz

možnost jako uživatel nahlásit profil správcům webu a ti musí danou fotografii prohlédnout a provést příslušnou akci. To znamená buď fotografii smažou, nebo budou ignorovat dané nahlášení. Smazání závadné fotografie může být mnohdy problematické, protože při nahlášení profilu musí administrátor projít všechny fotky na profilu a závadné fotografie odebrat. Někteří uživatelé mají na profilu stovky fotografií a to administrátorskou práci, kterou je potřeba provést na nahlášeném profilu, značně ztěžuje. Důsledkem je, že administrátorovi trvá zbytečně dlouho odstranit závadnou fotografii což má za následek delší prodlevu než bude závadná fotografie odebrána a vyšší cenu, kterou společnost Seznam.cz musí vynaložit na práci administrátorů, kteří provádějí kontrolu fotografií.

Výhodou je, že se žádné fotografie nemusí kontrolovat při nahrávaní, ale jsou nahlášeny až když danou fotografii někdo nalezne. Nevýhodou je, že o závadných fotografiích se neví dokud je někdo nenahlásí a také je problémem následná prodleva mezi nahlášením a smazáním, protože fotografie musí být opět zkontrolována člověkem, jestli je skutečně závadná, jak někdo nahlásil.

3.1.4 Požadavky společnosti Seznam.cz

Společnost Seznam.cz si klade jisté požadavky, které je nutné splnit, aby bylo reálné uvažovat o integraci modulu pro zpracování nevhodných fotografií do služby Lidé.cz.

Mezi tyto požadavky patří:

- Real-time klasifikace fotografií. Výsledek klasifikace musí být vyhotoven v čase menším než 1 sekunda.
- Přesnost klasifikátoru musí být vyšší než 90%.
- Počet závadných fotografií, které projdou klasifikátorem musí být co nejmenší i na úkor zvýšení počtu nezávadných fotografií, které budou označeny jako závadné.

3.2 Současný stav ve světě

3.2.1 Testovací data

Testovací data neboli takzvané *datasety* vzniky pro lepší možnost srovnání algoritmů. Pro klasifikaci fotografií existuje velké množství *datasetů* a mnohdy jsou součástí různých soutěží. *Dataset*, který by vyhovoval potřebám společnosti Seznam.cz a obsahoval by dvě kategorie, kde by byly fotografie označeny jako pornografické nebo slušné, nebyl nalezen a proto je nutné si jej vytvořit z fotografií, které jsou na službě Lidé.cz k dispozici. Algoritmy, které budou popsány v následující kapitole, nelze objektivně mezi sebou porovnat, protože každý z nich je učen na jiných datasetech a proto se udávané přesnosti jednotlivých algoritmů mohou značně lišit.

3.2.2 Ruční schvalování

Tento způsob klasifikace fotografií dosahuje nejlepších možných výsledků, protože fotografie klasifikují lidé. Nevýhodou této metody jsou z dlouhodobého hlediska vysoké náklady a dlouhá doba klasifikace jedné fotografie. V rozsáhlých internetových aplikacích se nevyplatí si pro tento způsob najímat konkrétní pracovníky do kanceláří, protože takovým zaměstnancům je vyplácená mzda a navíc je ještě potřeba obstarávat kancelářský prostor pro tyto zaměstnance.

Lepší variantou je využít speciálních agentur, které mají velký počet zaměstnanců a specializují se na takovéto úkony. Případně využít internetových služeb jako je Amazon Mechanical Turk [1]. Zde se lidé registrují a pracují na jednoduchých úkolech jako je opisování textu z obrázků a podobně. V této variantě si společnost založí profil, nabije si kredit a z tohoto kreditu se pak vyplácí peníze lidem, kteří dané úkoly plní. Výhodou této služby je velký počet uživatelů, kteří jsou připravení vykonávat úkoly. Lze tak značně snížit čas na schvalování fotografií z hodin na minuty. Další problém, který tato služba odstraňuje, je problém s nahráváním fotografií mimo pracovní dobu, kdy zaměstnanci ve společnosti Seznam.cz nepracují.

Díky velké množině lidí z celého světa se najde vždy někdo, kdo má možnost danou práci vykonávat, protože se nachází v jiném časovém pásmu. Je tak možné efektivně zpracovávat požadavky po celý den. Nevýhodou tedy pouze zůstává relativně vyšší cena za zpracovanou fotografii v porovnání s cenou za zpracovanou fotografii počítačem.

3.2.3 Detekce procentuálního zastoupení kůže

Detekce tónů kůže je základní a jednoduchá varianta pro odhalování erotických fotografií. Jejím problémem je velká nepřesnost, která je způsobena faktem, že se na fotce může vyskytovat velké množství barvy podobné kůži, ale přesto se o kůži nejedná. Příkladem může být fotka s lehce oranžovou stěnou, která se bude jevit jako opálená kůže. Vizuální ukázka výstupu detekce kůže je zobrazena na obrázku 2. Tato metoda je vhodná pro menší stránky, kde stačí mít základní filtr pro odhalování erotických fotografií a nebude vadit, že uživateli bude zamezeno nahrávání některých jeho slušných fotografií. Pro seznamovací službu velikosti Lidé.cz se tato metoda nehodí. Uživatelé by mohli být frustrováni, že jejich fotky jsou často označeny za závadné a mohli by se rozhodnout přestat tuto seznamovací službu používat, což by znamenalo značný úbytek uživatelů a tudíž menší výdělky pro společnost Seznam.cz.

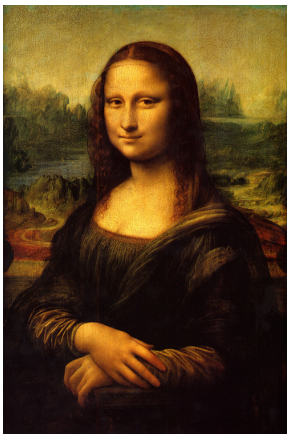
Obrázek 2: Vizualizace detekce kůže

Algoritmy pro zjišťování procentuálního zastoupení barvy kůže jsou mnohdy doplněny o detekci tváře na fotografii. Je to tím, že fotografie obličeje obsahují velké množství kůže, tak jsou často tyto fotografie vyhodnoceny jako erotické, i když nahotu neobsahují. Díky detekci obličeje lze jistou část fotografie ignorovat a kůži v této části fotografie nezapočítávat do celkového procentuálního zastoupení barvy kůže ve fotografii. Jelikož jedním z pravidel na seznamovací služby Lidé.cz je, že profilová fotografie musí obsahovat jasně viditelný obličej uživatele, tak pro případné použití, by se musel použít algoritmus, který již obsahuje detekci obličeje. Tím by se podařilo snížit počet *false positive* vyhodnocení, protože bychom vyhodnocovali pouze tu část fotografie, která neobsahuje obličej.

Výhodou této metody je její jednoduchá implementace a rychlá kontrola fotografie. Nevýhodou je již zmíněné velké množství *false positive* vyhodnocení. Další nevýhodou této metody je nepoužitelnost na fotky, které mají upravené tóny barev. Příkladem mohou být desaturované nebo sépiové fotografie. Na těchto fotografiích se změní barevný tón kůže a fotografie se vyhodnotí jako nezávadné [52].

3.2.4 PORNsweeper

PORNsweeper je zásuvný modul do aplikace MIMEsweeper [10]. Tato implementace využívá pro klasifikaci fotografií algoritmus detekce procentuálního zastoupení kůže s rozpoznáním obličeje.



Obrázek 3: Ukázka pornografické fotografie dle PORNsweeper

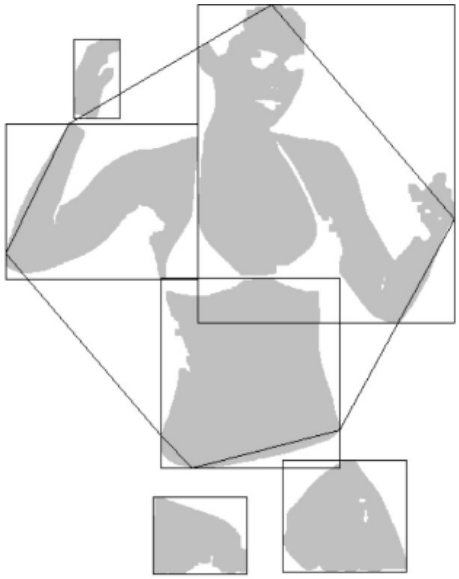
Nepřesnost této metody lze vyčíst z výsledku testů, které byly prováděny na malých množinách vstupních dat, kdy byla simulována data, která nahrává jeden uživatel. Dosažené výsledky byly velmi znepokojivé, protože velká část fotografií byla označena za pornografické, i když pornografii neobsahovaly. Příkladem byl známý obraz Mony Lisy zobrazený na obrázku 3, který byl také označen jako pornografický. Dalším problémem byl například nákladní vůz vyfocený ze dvou různých stran. Z jedné strany byl vyhodnocen jako slušný, ale fotografie nákladního vozu z druhé strany byla vyhodnocena jako pornografická. Fotografie nákladního vozu jsou porovnány na obrázku 4.

Při nahrání 22 fotografií, z nichž většina byly fotografie pornografické, se naměřily hodnoty 13,6% *false negative* a 8,3% *false positive*. Po nahrání 31 různých fotografií, přičemž žádná z nich nebyla pornografická, došlo k zablokování 6 z nich. Z toho plyne 19% *false positive*. Dalším testem bylo 42 barevných fotografií aut, autobusů a kamionů. Polovina těchto fotografií byla označena za pornografické. Pro automobily to znamená 50% *false positive* chybu. [42].

Obrázek 4: Srovnání pornografické (vlevo) a slušné verze (vpravo)

3.2.5 Detekce regionů obsahujících kůži

Vylepšenou verzí metody pro detekci kůže je rozšíření o detekci regionů, které obsahují barvu kůže. Na rozdíl od metody procentuálního zastoupení kůže tuto metodu nezajímá pouze kolik procent barvy kůže se na fotografii nachází, ale i jaký tvar a velikost mají regiony obsahující barvu kůže. Vizualizace detekovaných regionů obsahujících kůži je zobrazena na obrázku 5.



Obrázek 5: Detekované regiony s kůží

3.2.6 Nude.js

Pro detekci kůže na fotografii existuje několik open source řešení, která se touto problematikou zabývají. Jedním z rozšířených open source řešení je nude.js [51]. Nude.js je napsán ve skriptovacím jazyce JavaScript a využívá HTMLCanvas z HTML5. Výhodou tohoto řešení je, že umí detekovat i nahotu ve videích, ale to není společností Seznam.cz požadováno.

Nude.js, pro zjištění jestli je daná fotografie závadná, používá algoritmus, který byl prezentován na univerzitě De La Salle v Filipínách. Tento algoritmus dosahuje lepších výsledků než ostatní algoritmy zaměřené na problematiku odhalování erotických fotografií.

Algoritmus pracuje na tomto principu:

1. Detekuje na fotografii pixely, které mají barvu kůže.
2. Na základě těchto pixelů vytvoří regiony, které obsahují kůži.
3. Zanalyzuje nalezené regiony, zda obsahují nahotu nebo ne.
4. Klasifikuje jestli je obrázek závadný či nikoliv.

Podrobnější informace jak algoritmus pracuje jsou uvedeny v článku *Algorithm for Nudity Detection* [2].

Tento algoritmus byl schopen správně určit 94,88% fotografií z celkem 421 erotických fotografií a 635 slušných fotografií. Na této sadě fotografií měl algoritmus hodnotu *false positive* rovnu 5,04%. Výsledky byly prezentovány na 5. kongresu počítačových věd v Filipínách [2]. Výsledky jsou velmi slibné, ale zároveň velmi zavádějící, protože lze obecně utvořit takové sady fotografií, kde bude přesnost mnohém nižší. Důvodem je totiž snížená přesnost algoritmu při zpracovávání fotografií černobílých, fotografií, které mají nízkou kvalitu (například nekvalitní jpeg) nebo se na fotografiích nachází tmavší odstín kůže. Tento problém je zmíněn jednak v článku [2], ale i v článku nude.js: Nudity Detection with JavaScript [50].

3.2.7 Skin Sheriff

Jedná se o strojové učení, které je zaměřené na detekci pornografických fotografií. Popis tohoto strojového učení byl popsán ve 2. sborníku mezinárodní konference o bezpečnosti v komunikačních systémech [39].

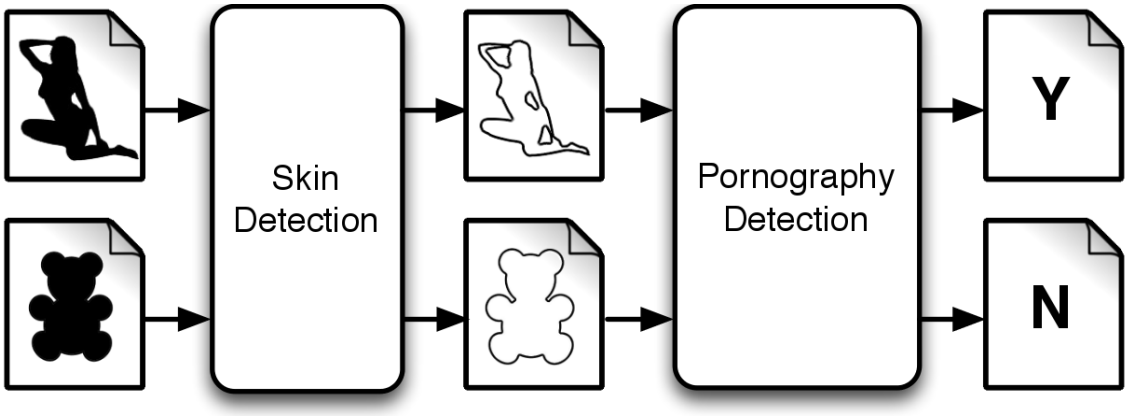
Skin Sheriff se skládá ze dvou hlavních částí. První částí je vyhledávání kůže na fotografii a výsledek tohoto prohledání je poslán na vstup druhé části, která se stará o detekci pornografie. Díky této přidané analýze dosahuje Skin Sheriff lepších výsledků než klasická detekce kůže na fotografii.

Skin Sheriff v první části pro detekci kůže využívá stejný algoritmus jako Nude.js. Výstupem tohoto algoritmu jsou regiony, které obsahují kůži. Ukázku detekovaných regionů lze vidět na obrázku 5. Následně jsou vybrány 3 největší regiony a v nich se zjistí procentuální zastoupení kůže. Tyto informace se poté použijí pro klasifikaci. Tato funkcionalita je znázorněna na obrázku 6.

Pro každý region se zjistí tyto parametry:

- Poměr kůže v daném regionu vůči minimálního ohraničujícímu obdélníku.

- Průměrné hodnoty Hue, Saturation a Value v barevném modelu HSV v daném regionu.
- Průměrná hodnota z černobílé fotografie.
- Výstřednost, orientace a eliptičnost regionu.
- Počet pixelů, které se dotýkají jiného regionu.
- Počet rohů regionů, které se dotýkají jiných regionů.
- Procentuální zastoupení kůže z celé fotografie.
- Směrodatná odchylka odstínu.
- Obvod kůže, který se v regionu nachází.
- Poměr mezi plochou kůže a jejím obvodem.
- Geometrické těžiště plochy kůže v regionu.



Obrázek 6: Skin Sheriff schéma

Pomocí těchto parametrů se poté jednotlivé regiony zpracovávají v následujících funkcích:

- Prostorová analýza kůže.
- Detekce obličeje.
- Vyřazení příliš rovných úseků kůže v regionech.
- Vyřazení dle tvaru úseku detekované kůže.
- Odstranění špatně detekovaných úseků kůže.
- Odstranění úseků kůže dotýkajících se hran fotografie.

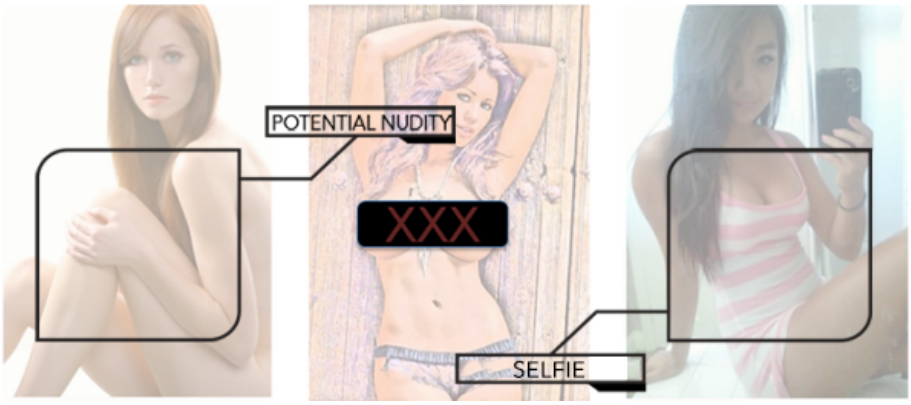
  Udávaná přesnost algoritmu je 91,9% na fotografiích z Compaq dataset [23].

3.2.8 Pokročilé detekce nahoty

Další možností jak poznat jestli je fotografie pornografického charakteru je detailně analyzovat fotografii a z ní odvodit jestli je osoba oblečená nebo ne, jestli se na fotografii nachází celá osoba nebo jen její část, vyhledávání genitálií na fotografii a případně další vyhodnocení, které pomohou pochopit v jaké poloze je osoba na fotografii a zda-li je obnažená nebo oblečená.

3.2.9 ImageVision Nudity Search

Příkladem pokročilé detekce nahoty je služba Nudity Search od společnosti ImageVision [17]. Jedná se placenou službu, která je dostupná jako API, kterému jsou zaslány fotografie a následně je navrácena odpověď, jestli daná fotografie obsahuje nahotu nebo nikoliv. Je možné si vybrat ze tří úrovní, kdy na úrovní 1 musí být osoba kompletně odhalena, aby byla fotografie označena za závadnou. Při třetí úrovni budou označeny jako závadné i fotografie s osobami ve spodním prádle [33].



Obrázek 7: Image Vision detecting exposed, sexually oriented body parts

ImageVision Nudity Search detekuje nahotu pomocí následujících analýz:

- **Detekce končetin**. Pro identifikaci nahoty ve fotografiích a videích vyhledává vlastnosti a tvary, které odpovídají lidským bytostem. Konkrétně se hledá trup a končetiny.
- **Detekce obličeje a hlavy**. Vzhledem k tomu, že nahota nemusí být vyobrazena pouze zepředu, tak je využita nejen detekce obličeje, ale i detekce hlavy.
- **Detekce textury kůže**. Na rozdíl od jiných vizuálních vyhledávacích technologií, zde se identifikuje kůže podle skutečné textury kůže a ne jen pouze podle barvy. Díky tomu je kůže přesněji identifikovaná a zároveň tato technologie funguje i na desaturovaných fotografiích.
- **Detekce zakřivení těla a zakryté části těla**. V rámci milisekund technologie identifikuje lidskou polohu. Následně určuje množství nahoty vzhledem k poloze

těla. Provádí se analýza zakřivení trupu a ploch, které jsou zakryté oblečením nebo jinými předměty.

- **Bradavky, zadek, genitálie**. V porovnání s ostatními vizuálně vyhledávacími technologiemi, které hledají charakteristické části lidského těla, dosahuje ImageVision lepších výsledků, protože používá umělou inteligenci, aby detekoval samotnou anatomii těla, která se na fotografii nachází. Detaily tohoto algoritmu nejsou veřejně k dispozici.

V rámci kapitoly bylo představeno několik přístupů k řešení problematiky klasifikace pornografických fotografií. Jak bylo zmíněno v sekci 3.2.3 algoritmus pro detekci kůže dosahuje nízkých přesností. Ani rozšíření o detekci obličeje, které sníží počet *false positive* klasifikací u fotografií, kde se nachází pouze obličej osoby nezvýší přesnost dostatečně, aby tento algoritmus mohl být použit na seznamovací službě Lidé.cz.

Implementace nazvaná Nude.js nabízí skvělé výsledky, ale pouze na fotografiích, které jsou barevné, neobsahují žádné fotografické filtry, které by upravovaly barvy a tím i odstín kůže a musí být použity fotografie s vysokým rozlišením, aby detekce správně pracovala. Z těchto důvodů není tato implementace vhodná pro seznamovací službu Lidé.cz, protože uživatelé nahrávají upravené fotografie pomocí různých filtrů nebo nahrávají fotky, které jsou pouze v odstínech šedi.

ImageVision Nudity Search je služba placená a nebylo možné zjistit aktuální přesnost klasifikace. Nevýhodou této služby je, že se platí za každou nahranou fotografii a jelikož se na Lidé.cz nahrává přibližné 7 000 fotografií denně, tak by měsíční cena za tuto službu činila 549$, což při aktuálním kurzu dolaru činí 13 400Kč měsíčně. Výhodou by bylo, že společnost Seznam.cz se nemusí nijak zabývat klasifikací fotografií, nevýhodou jsou měsíční poplatky a přidání závislosti na externě dodávané službě, která když přestane fungovat, tak přestane fungovat i nahrávaní fotografií na Lidé.cz.

Z těchto faktů vyplývá, že žádná z prozkoumaných možností není vhodná pro seznamovací službu Lidé.cz a po prozkoumání odborných článků zaměřených na klasifikaci fotografií [12] [36] [38] byla vybrána možnost využití konvolučních neuronových sítí, v nichž je tato neuronová síť schopna si sama natrénovat vzory, které v klasifikovaných fotografiích vyhledává.

3.3 Neuronová síť

Neuronová síť je složena z formálních neuronů, které jsou vzájemně propojeny. Výstup neuronu je vstupem do jednoho nebo několika dalších neuronů. Vzájemné propojení neuronů určuje topologii neuronové sítě. Neurony se dále dělí dle svého využití na neurony vstupní, výstupní a skryté (vnitřní) [49].

Neuronová síť se v průběhu svého učení vyvíjí. Mění se propojení jednotlivých neuronů, upravují se váhy a mění se stavy jednotlivých neuronů. Tyto změny můžeme rozdělit do tří kategorií. První kategorií jsou změny organizační, kdy dochází

ke změně topologie neuronové sítě. Druhá kategorie se nazývá aktivní. Zde dochází ke změnám stavů jednotlivých neuronů. Poslední kategorie se nazývá adaptivní a zde dochází ke změně konfigurace. Ve skutečné nervové soustavě toto rozdělení neexistuje, protože tyto změny probíhají současně [22].

Existují dva typy architektury. Prvním typem je cyklická architektura. Tento typ obsahuje neprázdnou množinu neuronů, která je zapojena v kruhu neboli obsahuje cyklus. Druhým typem je architektura acyklická nebo taky zvaná dopředná. Acyklická síť neobsahuje cyklus a všechny cesty vedou jedním směrem. Díky tomuto faktu lze všechny neurony rozdělit do vrstev. Kdy v každé vrstvě se nachází neurony, které spolu nejsou vzájemně propojeny a mají společnou předchozí i následující vrstvu [22].

Dle řízení neuronů se dělí neuronové sítě na dvě kategorie. První kategorie se nazývá synchronní a patří zde neuronové sítě jejíž neurony mění svůj stav nezávisle na sobě. Druhá kategorie se nazývá asynchronní a změna stavů takových neuronů je řízena centrálně [49].

Neuronová síť obsahuje počáteční konfiguraci, která se v čase mění. Na začátku se nastaví váhy všech spojení mezi neurony v síti na počáteční konfiguraci a následně postupem času dochází k adaptaci sítě. Všechny konfigurace sítě tvoří váhový prostor neuronové sítě [22].

Funkce neuronové sítě závisí na konfiguraci. Účelem adaptace je nalézt konfiguraci sítě ve váhovém prostoru, díky které bude možné realizovat předepsanou funkci [22].

3.3.1 Učení neuronové sítě

Neuronová síť se obvykle učí pomocí tréninkové množiny dat. Tato množina dat musí být dostatečně velká, aby síť mohla najít určité vzory mezi vstupními daty a vlastnosti těchto vstupních dat generalizovala tak, abychom dostali požadované výstupy.

3.3.2 Učení s učitelem

Tréninková množina dat je seskupena do množiny dvojic, kdy jsou definovány vstupy sítě a k nim je přiřazený požadovaný výstup sítě. Díky tomu síť pro každý vstup zná očekávané výstupní hodnoty a může zjistit jak je její odhad vzdálen od požadovaného výstupu. Takto je simulován učitel, který učí danou síť a proto se tento způsob adaptace nazývá učení s učitelem.

3.3.3 Učení bez učitele

V tomto případě obsahuje tréninková množina pouze vstupy sítě. V tomto případě není možnost kontroly výstupů a simuluje se tak učení bez učitele (neexistuje zpětná vazba). Neuronová síť organizuje tréninkové vzory a snaží se odhalit jejich společné vlastnosti.

3.3.4 Zpětnovazebné učení

Jedná se o učení jehož podstatou je namapovat situace na akce tak, aby se maximalizovala číselná odměna. Agentovi není řečeno, které akce má provést, jak je tomu obecně u strojového učení, ale místo toho síť musí zjistit, které akce přinášejí největší zisk a snažit se dané akce aplikovat.

Hlavními charakteristickými rysy jsou učení pomocí metody pokus omyl a opozděné odměňování. Základní myšlenkou je najít nejdůležitější aspekty skutečného problému pomocí interakce s okolím a dosáhnout požadovaného cíle. To znamená, že agent musí mít cíl nebo cíle vztahující se ke stavu okolí [4].

3.3.5 Konvoluční neuronová síť

Tato práce bude zaměřena na konvoluční neuronové sítě, které se používají i v jiných odvětvích pro detekci objektů v obrazech. Tyto konvoluční neuronové sítě se používají například pro detekci dopravních značek [43], detekci plicních uzlin v medicíně [30], rozpoznávní obličejů na fotografiích [27], rozpoznávaní psaného textu [6] a mají mnohé další příklady užití.

Z těchto odborných článků lze usuzovat, že konvoluční neuronová síť je vhodná pro trénování neuronové sítě nad obrazovými daty a proto bude modul pro vyhledávání nevhodných obrázků využívat klasifikátor založený na konvoluční neuronové síti.

Konvoluční neuronová síť je složena z jedné nebo více konvolučních vrstev, jejíchž výsledek je většinou následován akcí pod vzorkování, které převede skupinu pixelů na jednu hodnotu. Poté následuje jedna nebo více plně propojených vrstev jako je tomu u standardních vícevrstvých neuronových sítí. Architektura konvoluční neuronové sítě je navržena tak, aby využívala 2D struktury vstupního obrazu. Výhodou konvolučních neuronových sítí oproti klasickým neuronovým sítím je, že jsou snadnější na naučení a mají méně výsledných parametrů při stejném počtu neuronů ve skrytých vrstvách [47].

3.3.6 Typy vrstev v konvoluční neuronové síti

Modely neuronových sítí se skládají z vrstev, kdy každá vrstva má svůj vstup a výstup a provádí specifickou operaci. Vstupy a výstupy mohou být někdy i shodné, tudíž dojde k aktualizaci dat na vstupní vrstvě.

Convolutional

Každá konvoluční vrstva obsahuje určitý počet konvolučních matic. Tento počet matic se zapisuje v definici modelu. Konvoluční matice jsou trénovány pomocí zpětnovazebního učení. Každá z těchto matic je s konkrétními parametry, které jsou nastaveny modelem, aplikována na celý obrázek [9]. Účelem této vrstvy je získat *features*, což jsou předzpracované vstupy. Ze zadaného obrázku se neuronová síť naučí jisté vzory, kdy v první konvoluční vrstvě jsou to hrany, rohy

a linie a s každou další konvoluční vrstvou se neuronová síť naučí vyšší úroveň abstrakce těchto *features* [44].

Pooling
Pooling vrstva vypočte buď maximální nebo průměrnou hodnotu konkrétní *feature* nad konkrétním regionem obrázku a díky tomu se sníží variace obrázku. To jestli vypočte maximální nebo průměrnou hodnotu záleží na konfiguraci modelu. Díky této vrstvě je umožněno, že bude dosaženo stejných výsledků, i když se bude obrázek mírně lišit. Toto je důležitá operace pro správné fungování klasifikace a detekce objektů v obrázcích [28].

ReLU
Jedná se o vrstvu neuronů, které využívají aktivační funkce, díky které zvyšují nelinearitu modelu neuronové sítě, aniž by došlo k ovlivnění konvoluční vrstvy [31].

InnerProduct
Nazývaná také Fully Connected Layer. Jedná se o znázornění propojení všech neuronů z předchozí vrstvy v modelu s neurony, které se nacházejí v následující vrstvě modelu [32].

Dropout
Dropout vrstva slouží k vylepšení neuronové sítě, protože snižuje efekt přeučení. Hlavní myšlenkou je, aby se zabránilo vzájemnému ovlivňování neuronů ve skrytých vrstvách. Při zpracování každého obrázku z trénovacích dat je každý skrytý neuron s určitou pravděpodobností vynechán. Velikost této pravděpodobnosti se nastavuje v konkrétním modelu. Běžně se tento parametr nastavuje na hodnotu 0,5 tedy 50%. Skryté neurony takto nemohou spoléhat na dostupnost ostatních skrytých neuronů [16].

Loss
Loss vrstva představuje v modelu aplikaci Loss funkce. Tato funkce je také nazývaná jako *cost* nebo *error* funkce a vyjadřuje chybu, která vznikla na sadě dat. Většinou se užívá k určení velikosti chyby nad validačními daty. Úkolem učení je najít takový model, který dosáhne nejnižší hodnoty *loss* funkce, což znázorňuje, že tento model dosahuje na validačních datech nejmenší chyby [20].

3.4 Frameworky pro práci s neuronovými sítěmi

Seznamovací služba Lidé.cz pracuje na skriptovacím jazyce Python [13], který využívají ve verzi 2.7. Při výběru frameworku je potřeba zohlednit frameworky, které jsou určeny pro jazyk Python nebo alespoň nabízí možnost Python bindingu, aby bylo možné provádět potřebné akce v rámci Pythonu. Důležitým prvkem frameworku je jestli podporuje učení na grafické kartě. Díky podpoře grafické karty může být jednak trénování neuronové sítě rychlejší, ale co je důležitější klasifikace fotografie

na naučeném klasifikátoru bude rychlejší než na procesoru, což je potřebné pro to, aby fotografie mohla být zpracována v čase nižším než 1 sekunda. Z předchozí kapitoly taky vyplývá, že framework musí podporovat učení konvolučních neuronových sítí.

3.4.1 Cuda convnet

Cuda convnet [24] je framework napsaný v jazyce C++. Jak již název vypovídá, tak je určen výhradně pro CUDA zařízení [37]. Tudíž nepodporuje zpracování na procesorech, ale je výhradně určen pro zpracování na grafické kartě. Má vestavěný interface pro jazyk Python, takže je možné jeho využití v projektech napsaných v jazyce C++ nebo Python. Podporuje učení na konvolučních neuronových sítích, ale aktuálně není vyvíjen jak je zobrazeno v tabulce 1.

3.4.2 Decaf

Decaf je framework, který implementuje konvoluční neuronové sítě. Jeho cílem je efektivita a flexibilita. Umožňuje jednoduše zkonstruovat síť ve formě libovolného orientovaného acyklického grafu. Nepodporuje zpracování na grafické kartě, ale pouze na procesoru, což je zásadní nevýhodou tohoto frameworku [11]. Tento framework je napsán v jazyce Python, takže není potřeba žádného bindingu.

3.4.3 Caffe

Caffe je framework určený pro učení konvolučních neuronových sítí. Klade důraz na čistotu kódu, čitelnost a rychlost. Byl vytvořen Jia Yangqingem v rámci jeho disertační práce na univerzitě Berkeley a nyní je aktivně vyvíjen v Berkeley Vision and Learning Center (BVLC) a komunitních přispěvatelů. Caffe bylo vydáno pod BSD2-Clause licencí [21]. Kód Caffe frameworku ja napsaný v C++ a nabízí binding pro Python a MATLAB [34].

Sítě jsou definovány v jednoduchých konfiguračních souborech bez pevně zadaných parametrů v kódu. Přepínání mezi CPU a GPU zpracováním se děje pouze pomocí jediného příznaku. Díky tomu mohou být modely trénovány na stroji s grafickou kartou a poté použity na komoditních clusterech [48].

Rychlost je to, co dělá z Caffe ideální nástroj pro průmyslové použití. Caffe dokáže zpracovat přes 40 milionů fotografií za den na jediné grafické kartě (testováno na NVIDIA K40 a TITAN GPU). Z toho je 5ms na fotografii při učení a 2ms na fotografii při testování. Proto se Caffe řadí mezi jedny z nejrychlejších frameworků implementující konvoluční neuronové sítě, které jsou zároveň volně dostupné [48].

3.4.4 Srovnání frameworků

Tabulka 1 je převzata ze sborníku mezinárodní konference ACM o multimédiích [21]. V tabulce jsou vybrány pouze tři nejrozšířenější frameworky, které podporují

trénování konvolučních neuronových sítí a jsou zde zobrazeny pouze informace, které jsou zásadní pro výběr frameworku, který bude použit v modulu pro zpracování nevhodných obrázků.

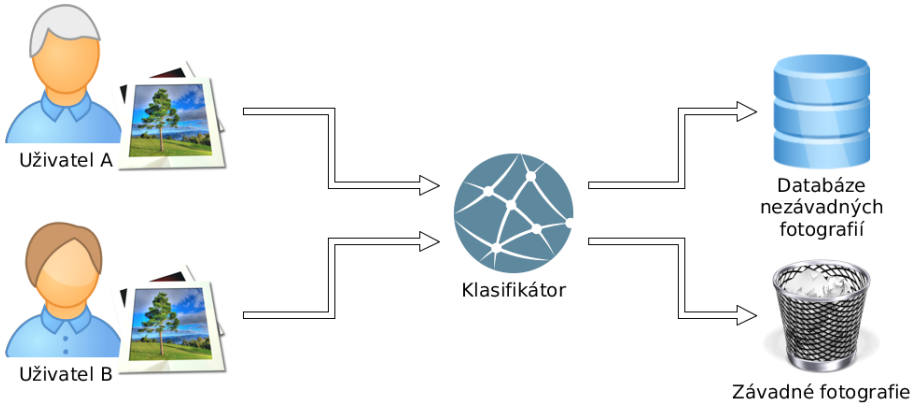
Jelikož vstupní předpoklady splňovaly dva frameworky cuda-convnet a Caffe, tak bylo po následné dohodě se společností Seznam.cz vybrán framework Caffe z toho důvodu, že obsahuje možnost učení jak na grafické kartě, tak na procesoru, ale hlavně z důvodu, že framework je aktivně vyvíjen a má aktivní komunitu, která může případně pomoct s problémy, které by mohly nastat při vývoji na frameworku nebo při jeho nasazení.

Framework	**License**	**Jazyk**	**Interface**	**CPU**	**GPU**	**Vývoj**
cuda-convnet	neznámá	C++	Python	Ne	Ano	Přerušen
Decaf	BSD	Python		Ano	Ne	Přerušen
Caffe	BSD	C++	Python, MATLAB	Ano	Ano	Distribuovaný

Tabulka 1: Srovnání frameworků

4 Analýza problému a návrh řešení

Cílem je snížit počet fotografií, které by museli administrátoři klasifikovat. Řešením je tedy použít klasifikátor, který by tuto činnost kompletně nahradil nebo alespoň zprostředkoval většinu fotografií, aby se snížil počet fotografií, které by musely být ručně klasifikovány. Použití klasifikátoru je znázorněno na obrázku 8.



Obrázek 8: Použití klasifikátoru

Klasifikátoru jsou předány fotografie od uživatelů a ten dané fotografie buď klasifikuje jako závadné, nebo nezávadné. Žádný klasifikátor nemá 100% přesnost. S tím je potřeba počítat a proto jsem při návrhu využil faktu, že administrátoři budou muset špatně klasifikované fotografie opravit. Díky tomu lze rozšiřovat databázi klasifikovaných fotografií a mít přichystáno více dat pro učení klasifikátoru, což zvyšuje pravděpodobnost, že se další nově naučený klasifikátor naučí lépe a bude dávat přesnější výsledky.

Pokud klasifikátor označí fotografii za závadnou, tak bude mít uživatel možnost přesto fotografii odeslat, ale takto odeslaná fotografie se zařadí do fronty pro schválení. Když bude fotografie označena za nezávadnou, tak se přiřadí do databáze fotografií, aby se při příští iteraci učení dostala tato fotografie do klasifikátoru a zlepšila se tak přesnost klasifikace.

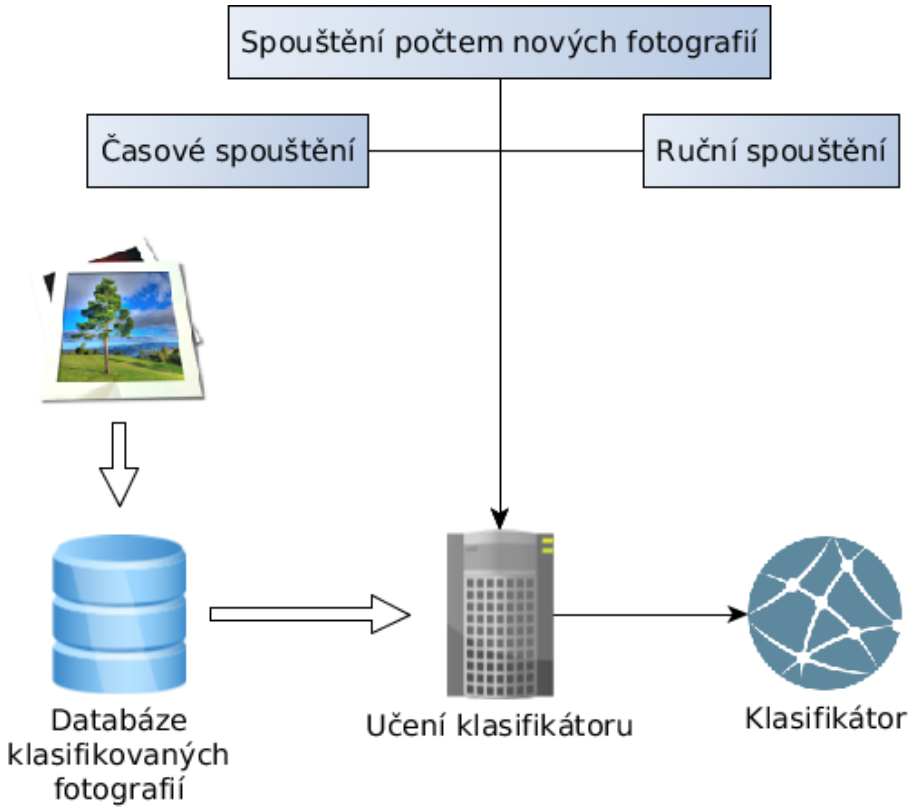
Obdobný případ nastane, pokud bude daná fotografie automaticky schválena, ale nějaký uživatel ji nahlásí jako závadnou. Fotografie bude opět zařazen do fronty pro schválení a pokud bude označena jako závadná, tak bude přidána do databáze fotografií jako závadná.

Vyvíjený modul má obstarat vše, co je potřebné, aby mohlo být spouštěno, zastaveno a nebo znovu naplánováno učení z dané databáze fotografií. Databází fotografií může být i více než jedna, ale učení bude vždy probíhat pouze nad jednou konkrétní databází. Mezi dalšími metodami modulu musí být metoda, která zajišťuje klasifikaci dané fotografie nebo skupiny fotografií. Tato metoda vrátí procentuální příslušnost do jednotlivých zkoumaných skupin. V tomto případě se jedná o skupiny s názvy *závadná* a *nezávadná*.

Klasifikátor nemusí ze začátku dosahovat požadovaných přesností z důvodu malého počtu klasifikovaných fotografií, které bude mít k dispozici. Jak bylo výše popsáno, tak databáze klasifikovaných fotografií se bude postupem času rozšiřovat, ale to znamená, že se klasifikátor musí znovu na nových fotografiích naučit. Z toho důvodu je potřeba spouštět učení klasifikátoru, jak je zobrazeno na obrázku 9.

Učení klasifikátoru je možné začít pomocí:

- Času - například po třech měsících se automaticky vytvoří nový klasifikátor.
- Počtu nových fotografií - například po 500 nových fotografiích.
- Ruční spuštění - kdy zaměstnanci Seznam.cz uznají za vhodné.

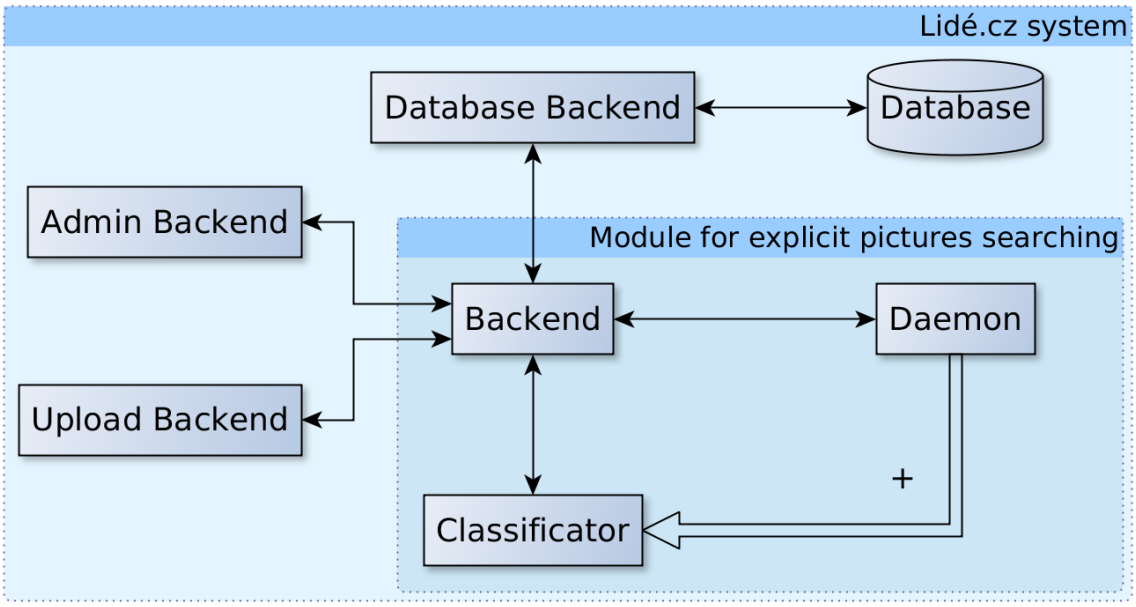


Obrázek 9: Diagram učení klasifikátoru

4.1 Komunikace modulu se systémem Lidé.cz

Vytvořený modul musí zvládat komunikaci se systémem, na kterém běží Lidé.cz. Tato komunikace je znázorněna na obrázku 10. Konkrétně bude modul komunikovat s backendem databáze, backendem, který řídí nahrávání fotografií na Lidé.cz a administrátorským backendem, kde bude umožněno spouštět a zastavovat učení

klasifikátoru. Databázový backend bude aktivně využíván navrhovaným modulem, protože bude potřeba načítat data o modelech klasifikátorů a fotografie, které mají být naučeny. Více informací o datech, které budou zpracovány *backendem* modulu se lze dočíst v kapitole 4.5. Backend určený pro upload nebude navrhovaným modulem využíván, ale naopak tento backend pro nahrávání bude komunikovat s modulem, který mu předá výsledek klasifikace. Administrátorský Backend také nebude využíván navrhovaným modulem, ale administrátorský backend bude mít možnost ovládat a sledovat učení klasifikátorů. Více informací ohledně administrátorském backendu se nachází v kapitole 4.6.



Obrázek 10: Diagram komunikace

Modul pro vyhledávání nevhodných obrázků se skládá ze tří částí, kterými jsou *Backend*, *Daemon* a *Classificator*. Jednoduché šipky v diagramu znamenají komunikaci, kde šipka udává směr komunikace. Zdvojená šipka s plusem v diagramu znázorňuje vytváření entit. Tato šipka je použita pouze v případe *Daemona*, který vytváří naučené klasifikátory.

Z diagramu je patrné, že veškerá komunikace okolního systému Lidé.cz s navrhovaným modulem probíhá pomocí entity Backend. Tato entita poté komunikuje s entitou *Daemon* nebo entitou *Classificator* v závislosti na tom, jaký požadavek byl na Backend zaslán. V případě, že na Backend bude zaslán požadavek na klasifikaci, tak Backend zajistí klasifikaci dané fotografie a pošle zpátky výsledek klasifikace. Pokud bude na *Backendu* přijat požadavek na zastavení nebo spuštění učení klasifikátoru, tak bude Backend komunikovat s *Daemonem*, který dané akce spojené s učením klasifikátorů zajistí.

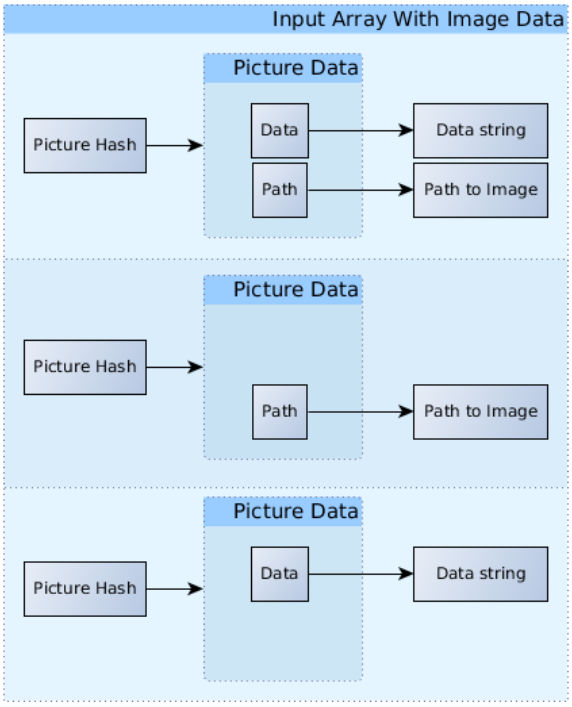
Jak bylo zmíněno v odstavci výše, tak *Daemon* běží na pozadí a jeho účelem je obstarávat požadavky ohledně učení klasifikátorů. Konkrétně se jedná o požadavky

započni učení, zastav učení a nebo pokračuj v učení od dané iterace. Dalšími požadavky, které *Daemon* obsluhuje je zjištění stavu učení klasifikátoru. *Daemon* by měl vrátit statistiky učení, které zobrazí jaký je stav klasifikátoru, aby si mohl administrátor udělat obrázek o tom, zda-li je potřeba v učení klasifikátoru pokračovat či nikoliv.

Entita *Classificator* představuje konkrétní naučený klasifikátor, který umí zpracovat příchozí fotografie. Tyto fotografie klasifikuje do kategorií nad kterými byl naučen a příslušnost fotografií do jednotlivých kategorií vrátí zpět *Backendu*, který výsledky zašle příslušnému *Backendu*, jenž jej zavolal.

4.1.1 Entita Backend

Slouží jako komunikační rozhraní mezi modulem pro vyhledávání nevhodných obrázků a systémem Lidé.cz. Zároveň obsahuje logiku ke komunikaci s klasifikátory a *Daemonem* pro učení modelů. Pokud *Daemon* potřebuje získat data z databáze, tak činí pomocí entity Backend, která mu potřebná data zprostředkuje.

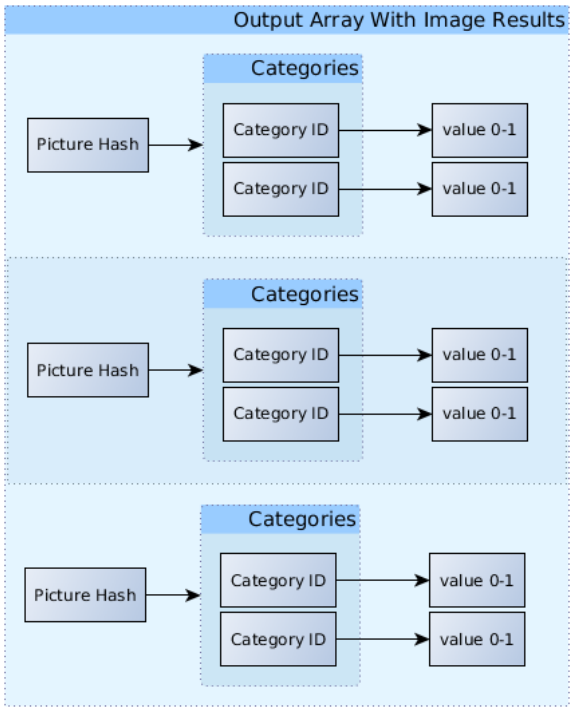


Obrázek 11: Vstupní pole dat pro klasifikaci

Pro všechny tyto účely komunikace musí Backend podporovat následující metody:

classify(model, pictures)

Metoda určena pro klasifikaci fotografií na vybraném modelu. Model bude určen



Obrázek 12: Výstupní pole s výsledky klasifikace

pomocí identifikátoru. Fotografie jsou předány jako pole, které obsahuje strukturu vyobrazenou na obrázku 11. Toto pole obsahuje záznamy, kde klíčem je hash fotografie, který je unikátní v rámci služby Lidé.cz. Pod každým z klíčů se nachází pole, které může obsahovat hodnoty pod dvěma klíči. Jedním z klíčů je path, pod takovým klíčem se očekává cesta k fotografii, který se má zpracovat. Tato cesta musí být dostupná z klasifikátoru a toto obstarává společnost Seznam.cz. Druhým klíčem může být položka data, pod kterou jsou uložena binární data fotografie, která využívá společnost Seznam.cz pro předávání obrázkových dat mezi moduly. Pro rozkódování těchto dat existuje dodaná funkce. Tudíž implementace kódování a dekódování včetně struktury dat není podstatná. Pod hashem fotografie musí existovat minimálně klíč path nebo klíč data. Mohou existovat i oba klíče současně, ale v tom případě se použije hodnota předaná pod klíčem data.

Výstupem potom je pole, které je zobrazeno na obrázku 12. Opět hlavním klíčem je hash fotografie na Lidé.cz, pomocí kterého můžeme nalézt výsledek klasifikace pro danou fotografii. Výsledek obsahuje další pole, kde klíčem jsou identifikátory kategorií, do kterých umí klasifikátor klasifikovat a hodnota daného klíče je příslušnost do dané kategorie. Hodnoty příslušnosti nabývají rozsahu $\langle 0, 1 \rangle$, kde 0 znamená, že daná fotografie do této kategorie vůbec nepatří a 1 znamená 100% příslušnost do dané kategorie.

getNextFromQueue())
Pomocí této metody se bude *Daemon* dotazovat na získání dalšího prvku z fronty pro učení. Tato akce je zobrazena na obrázku 16, kde je vidět jak si *Daemon* vybírá informace o učeném modelu, vybrané databázi fotografií a případné další informace z fronty pro učení klasifikátorů. Metoda neočekává žádné parametry a vrací pole dat s informacemi, které jsou nezbytné pro start učení daného klasifikátoru.

listQueue()
Neočekává žádné parametry a vrací pole s informacemi o všech učeních, která jsou uložena ve frontě pro učení klasifikátorů. Tato metoda je využita v administračním rozhraní, kdy bude umožňovat administrátorům výpis fronty. Více o administračním rozhraní se můžete dočíst v kapitole 4.6.

addToQueue(modelId, picturesDbId, data))
Přidá do fronty pro učení klasifikátorů nový záznam pro vybraný model a databázi fotografií. Data je pole s informacemi, které jsou potřeba znát pro započatí učení. Toto pole je koncipováno tak, že obsahuje spárované dvojice klíče a hodnoty, kde klíč udává název sloupce v databázi, do kterého se má zadaná hodnota uložit. Přidávání do fronty je využito v administračním rozhraní.

editQueue(modelId, picturesDbId, data))
Upraví zadaný záznam z fronty pro učení klasifikátorů. Identifikace je provedena pomocí identifikátoru modelu a identifikátoru databáze fotografií. Data označuje pole s informacemi, které lze editovat. Toto pole je koncipováno stejně jako v případě metody *addToQueue()* popsané výše. Využito pouze v administračním rozhraní.

deleteQueue(modelId, picturesDbId)
Smazání záznamu z fronty pro učení klasifikátorů. Záznam je identifikován pomocí identifikátoru modelu a identifikátoru databáze fotografií. Používá se v administračním rozhraní.

listModels()
Vrátí pole s hodnotami o všech modelech. Tato metoda je využita v administračním rozhraní, kdy si administrátor bude mít možnost zobrazit seznam všech modelů v systému. Slouží k získání přehledu o modelech, tudíž tato metoda by neměla vracet všechny dostupné informace o modelech.

getModel(id)
Metoda vrátí pole s detailními informacemi o modelu. Tato metoda by měla vracet více informací než metoda *listModels()* popsaná výše. Tato metoda slouží i pro získání hodnot k předvyplnění editačního formuláře v administračním rozhraní.

addModel(data)
Díky této metodě bude umožněno v administračním rozhraní přidávat nové modely do databáze modelů. Data je pole s informacemi, které jsou potřeba znát pro vytvoření modelu. Toto pole je koncipováno tak, že obsahuje spárované dvojice klíče a hodnoty, kde klíč udává název sloupce v databázi do kterého se má zadaná hodnota přiřazená ke klíči uložit.

editModel(id, data)
Upraví model, který odpovídá zadanému identifikátoru modelu. Data je pole s informacemi, které lze u modelu editovat. Toto pole je koncipováno stejně jako v případě metody *addModel()* popsané výše. Využívá se pouze v administračním rozhraní.

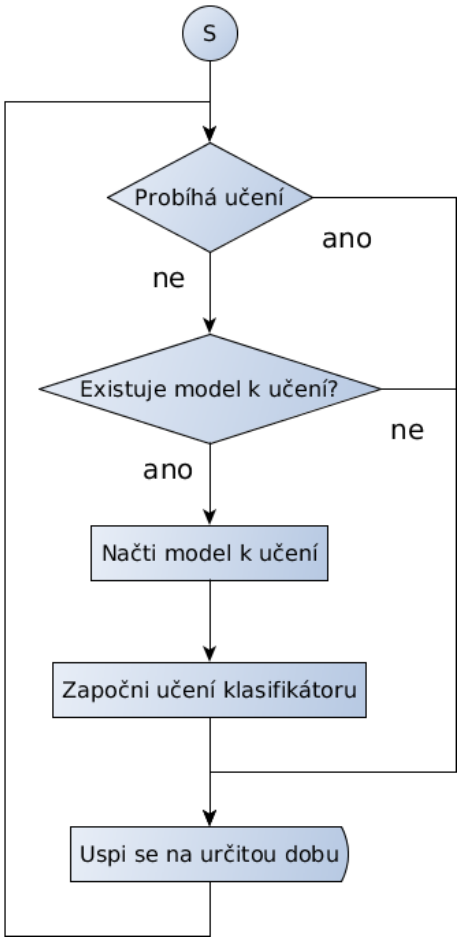
deleteModel(id)
Smazání modelu z databáze. Tato metoda je velmi nebezpečná a před jejím vyvoláním musí být uživatel upozorněn jestli si opravdu přeje model vymazat. Tato metoda vymaže nejen model a případné naplánované učení, ale také všechna naučená data, dočasně vytvořená data a klasifikátory, které byly vytvořeny v průběhu učení.

learningStatus(modelId, picturesDbId)
Pro vypsání stavu učení slouží právě tato metoda, která jako parametry očekává identifikátor modelu, o kterém má vypsat statistiku učení a databázi fotografií, nad kterou se učení provádělo. Návratovou hodnotou je pole s daty, které bude možné v administračním rozhraním vykreslit do podoby grafu nebo případné jiné vizualizace.

4.1.2 Entita Daemon

Daemon řídí všechny akce, které se dějí na pozadí. Tedy slouží jako rozhraní mezi *Backendem* a učením klasifikátoru. Mezi jeho hlavní úkoly patří zahájení a ukončení učení klasifikátoru a zjištění aktuálního stavu učení klasifikátoru, aby administrátoři měli zpětnou vazbu ohledně kvality naučení daného klasifikátoru. Všechny níže uvedené metody vyjma interní metody *processIteration()* jsou využívány v administračním rozhraní.

processIteration()
Daemon pracuje v hlavní smyčce a cyklicky se dotazuje *Backendu* jestli existuje nějaký záznam ve frontě pro učení klasifikátorů, který by mohl zpracovat. Pokud žádný takový není, tak se *Daemon* uspí na určitý časový interval a po probuzení pokračuje znovu zavoláním této metody. Vývojový diagram této metody je zobrazen na obrázku 13. Nejdříve *Daemon* zjistí jestli aktuálně na stroji neprobíhá nějaké učení. Tímto se zajistí, že na každém stroji poběží právě jedna instance učení. Pokud se učení neprovádí, tak se *Daemon* dotáže *Backendu*, jestli existuje nějaký model ve frontě pro učení klasifikátorů. Pokud existuje, tak jej načte



Obrázek 13: Cyklus Daemona

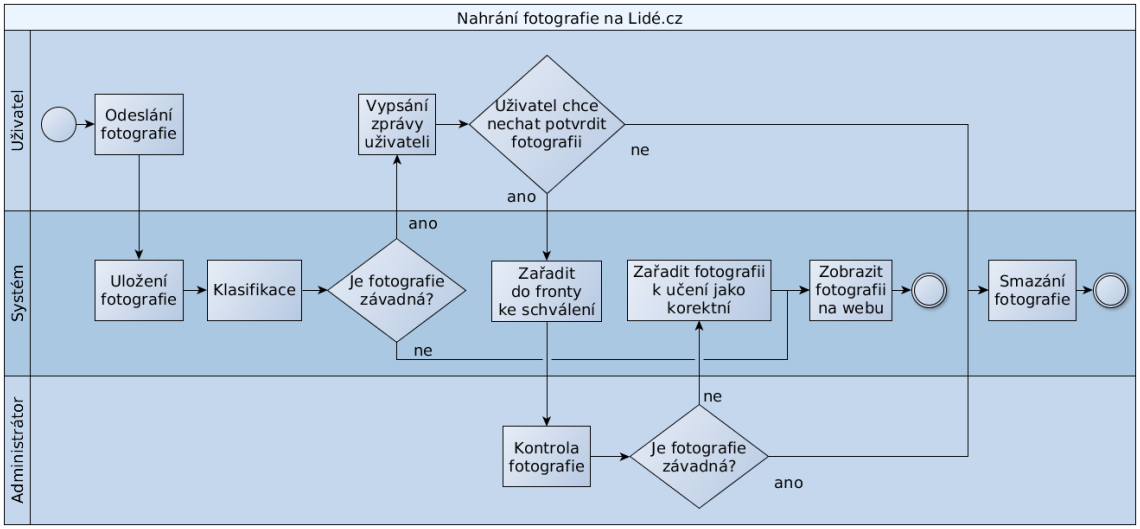
a započne učení nad daným modelem a nad danou databází fotografií. Pokud učení aktuálně probíhá, nebo pokud neexistuje model ve frontě pro učení klasifikátorů, tak cyklus *Daemona* končí a *Daemon* se na daný časový úsek uspí. Doba na kterou se *Daemon* bude uspávat po každém provedeném cyklu bude nastavitelná. Zaměstnanci společnosti Seznam.cz si ji budou moct nastavit, jak uznají za vhodné.

startLearning(data)
Metoda pro započatí učení klasifikátoru. Informace potřebné k učení jsou předány pomocí pole data, které obsahuje všechny potřebné informace pro učení klasifikátoru.

stopLearning(data)
Administrátor si může vyžádat zastavení učení modelu klasifikátoru. K tomu může využít tuto metodu, která z předaných dat zastaví právě probíhající učení klasifikátoru.

4.2 Nahrání fotografie

Nahrání fotografie je jediná akce v systému, která předá klasifikátoru právě nahranou fotografii. Důležitým aspektem je rychlost za kterou bude schopen klasifikátor fotografii zpracovat, protože požadavkem od společnosti Seznam.cz je, aby bylo možné provádět klasifikaci v reálném čase, kdy je nahrávána na server.



Obrázek 14: Diagram aktivit - nahrání fotografie

Navržený diagram aktivit pro nahrání fotografie je vidět na obrázku 14. Diagram obsahuje dva subjekty uživatele a administrátora a jeden objekt a to je systém. Systémem je myšleno celý systém na kterém běží služba Lidé.cz a klasifikátor je obsažen v tomto systému.

Diagram začíná v bodě, kdy uživatel odešle fotografii na web Lidé.cz. V tomto místě systém obdrží fotografii, kterou si uloží. Tuto fotografii předá klasifikátoru a čeká na výsledek klasifikace. Po obdržení výsledku klasifikace se systém na základě nastavení společnosti Seznam.cz rozhodne, zda-li fotografie je nebo není závadná. Tímto nastavením se myslí porovnání konkrétního procentuálního zařazení fotografie do kategorie závadná. Seznam.cz si může určit, že pokud klasifikátor vrátí, že fotografie je na 45% závadná, tak to již bude bráno jako závadná fotografie. Nastavení tohoto prahu, kdy fotografie je a není závadná je čistě v režii společnosti Seznam.cz a tímto prahem může snížit počet *false negative* klasifikovaných fotografií. Pokud není nahraná fotografie označena jako závadná, tak je zobrazena na webu a tím tento diagram aktivit pro tuto fotografii končí.

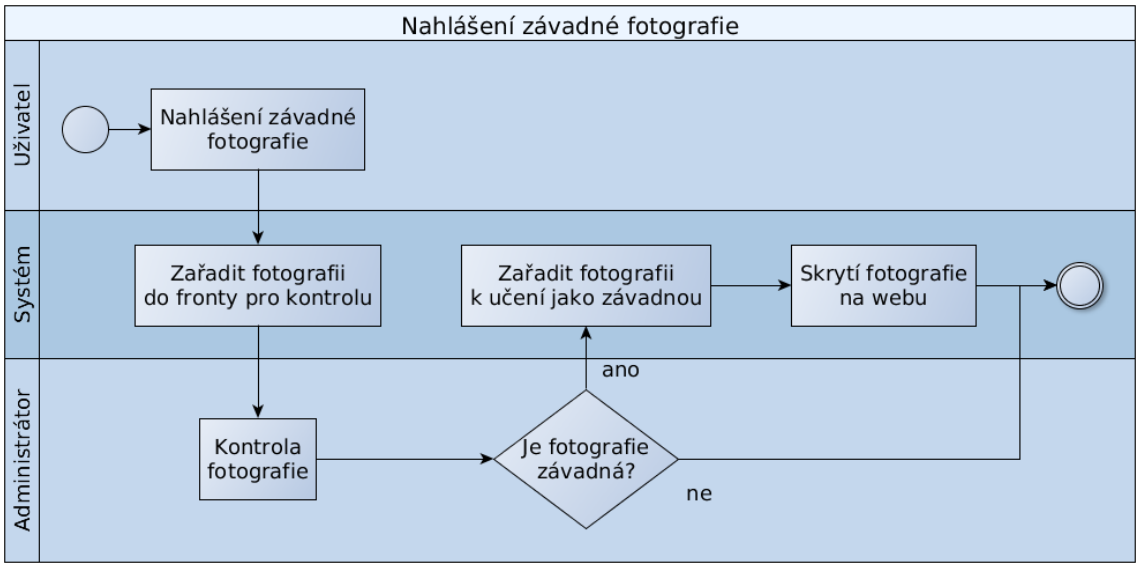
Pokud je ovšem fotografie vyhodnocena jako závadná, tak bude tato informace sdělena uživateli a uživateli je nabídnuta volba jestli chce i přesto fotografii nahrát s tím, že tato fotografie bude určena ke schválení administrátorem a že toto schválení může trvat i několik hodin. Pokud uživatel zvolí možnost, že nechce fotografii nahrát, tak může být fotografie smazána, záleží na konkrétní politice společnosti Seznam.cz jestli si chce tyto fotografie ponechat či nikoliv.

To jestli bude uživateli umožněna volba si vybrat jednu z možností nebo bude automaticky zvolena možnost nahrát fotografii do fronty pro schválení záleží na společnosti Seznam.cz a hlavně na přesnosti výsledného klasifikátoru. Pokud bude mít velký počet *true negative* výsledků, tak bude lepší zvolit možnost automatického zařazení do fronty pro schválení, aby uživatel nebyl frustrován neustále se opakujícími hláškami, že jeho fotografie jsou závadné, i když tomu tak není. Nevýhodou tohoto přístupu je vyšší počet fotografií, které musí administrátoři schválit, protože takto budou zařazeny i fotografie, které by uživatelé nepotvrdili, protože by věděli, že jsou opravdu erotického charakteru a administrátoři jim fotografii neschválí.

Z fronty fotografií pro schválení administrátoři vybírají fotografie jako je tomu doposud a tyto fotografie klasifikují, pokud administrátor zvolí, že fotografie je závadná, tak může být smazaná, opět záleží na politice společnosti Seznam.cz, jestli si dané fotografie ponechává nebo ne. Když administrátor označí fotografii z této fronty jako nezávadnou, tak bude fotografie zároveň přidána do databáze klasifikovaných fotografií s tím, že bude označena jako závadná a při příští iteraci učení klasifikátoru bude klasifikátor učen i na této fotografii. Tím dojde ke zlepšení klasifikátoru na závadných fotografiích, které uživatelé služby Lidé.cz nahrávají.

4.3 Nahlášení fotografie

Uživatelé služby Lidé.cz mají možnost označovat fotografie jako závadné. Tímto je umožněno upozornit administrátory na závadnou fotografii. Aktuálně je to jediný způsob jakým jsou odstraňovány a kontrolovány fotografie, které nejsou profilové. Administrátoři totiž schvalují pouze profilové fotografie a ostatní fotografie, které si uživatelé nahrávají na profil již kontrolovány nejsou.



Obrázek 15: Diagram aktivit - nahlášení fotografie

Diagram aktivit, který je zobrazen na obrázku 15 opět obsahuje dva subjekty uživatele a administrátora a jeden objekt, kterým je systém. Pod pojmem systém je myšlen celý systém na kterém běží služba Lidé.cz a klasifikátor je zamýšlen jako část tohoto systému.

Diagram začíná akcí uživatele, kterou je nahlášení závadné fotografie. Systém zařadí tuto fotografii do seznamu fotografií čekající na kontrolu administrátorem. Administrátor poté provede kontrolu fotografie a pokud ji zhodnotí jako nezávadnou, tak nahlášení fotografie zruší a žádná akce se neprovede. Když ale administrátor vyhodnotí fotografii skutečně jako závadnou, tak daná fotografie bude zařazena do databáze klasifikovaných fotografií jako závadná a při příští iteraci učení klasifikátoru bude klasifikátor naučen i na této fotografii.

4.4 Učení klasifikátoru

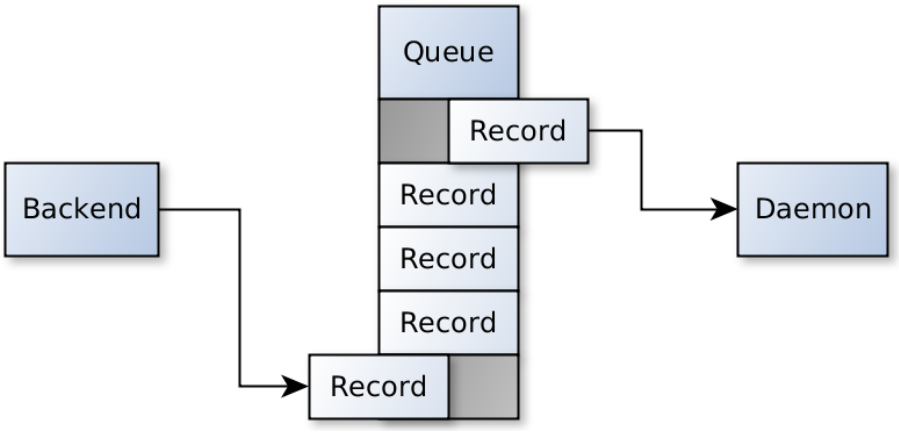
Doba pro učení klasifikátoru je mnohonásobně vyšší než doba pro klasifikaci fotografie. Na jednom stroji se bude moct učit pouze jeden klasifikátor, protože kdyby se více klasifikátorů učilo současně tak doba pro jejich učení bude vyšší než pokud budeme každý z klasifikátorů učit zvlášť. Tento jev je způsobený zvýšenou režií přístupu k datům a k výpočetním prostředkům jako je procesor nebo grafická karta. Učení více klasifikátorů na jednom stroji by bylo možné, pokud by stroj obsahoval více výpočetních prostředků, tedy například více grafických karet a existoval paralelní přístup k učícím fotografiím, ale tato možnost nebyla společností Seznam.cz požadována a proto se s ní nepočítá. Existuje ale možnost, že by existovalo více strojů, které by mohly učit klasifikátory a je tedy potřeba s touto možností při návrhu počítat.

4.4.1 Fronta pro učení klasifikátorů

Jelikož se klasifikátor může učit i několik týdnů je potřeba vytvořit frontu pro učení klasifikátoru, která bude obsahovat informace o tom, který model klasifikátoru se má učit a nad jakou databází fotografií má být učení provedeno. Schéma této fronty je zobrazeno na obrázku 16.

Backend na základě příkazu z Administrátorského *Backendu* vytvoří požadavek na naučení konkrétního modelu klasifikátoru včetně informace nad kterou databází fotografií se má model naučit. Tomuto záznamu Backend přidělí stav waiting, aby bylo jasné, že daný model klasifikátoru čeká ve frontě na naučení. *Daemon* periodicky kontroluje stav této fronty a pokud nalezne záznam, který je ve stavu waiting, tak mu nastaví stav learning, aby bylo jasné že daný požadavek na naučení je zpracován a započne učení daného modelu klasifikátoru.

Operace pro kontrolu stavu fronty a nastavení stavu na learning musí být atomická z toho důvodu, že může existovat více strojů na kterých může probíhat učení a každý z těchto strojů bude mít spuštěný svého Deamona, který bude kontrolovat stav učící fronty. Takto by mohlo dojít k situaci, kdy se setkají dva požadavky



Obrázek 16: Fronta pro učení klasifikátorů

pro získání stavu fronty, oba by získali informaci, že existuje požadavek a poté by začaly dva stroje učit stejný model klasifikátoru nad stejnou databází fotografií, což je nepřípustné. Proto je potřeba vytvořit k této frontě výlučný přístup.

Učení modelu klasifikátoru může být ukončeno ručně příkazem administrátora nebo v rámci učícího procesu, který může mít nastaven maximální počet učících iterací nebo požadovanou přesnost, které když dosáhne, tak učení ukončí. Po ukončení učení Deamon odebere požadavek na učení, který zpracoval a který se nacházel ve stavu learning.

4.5 ER Diagram

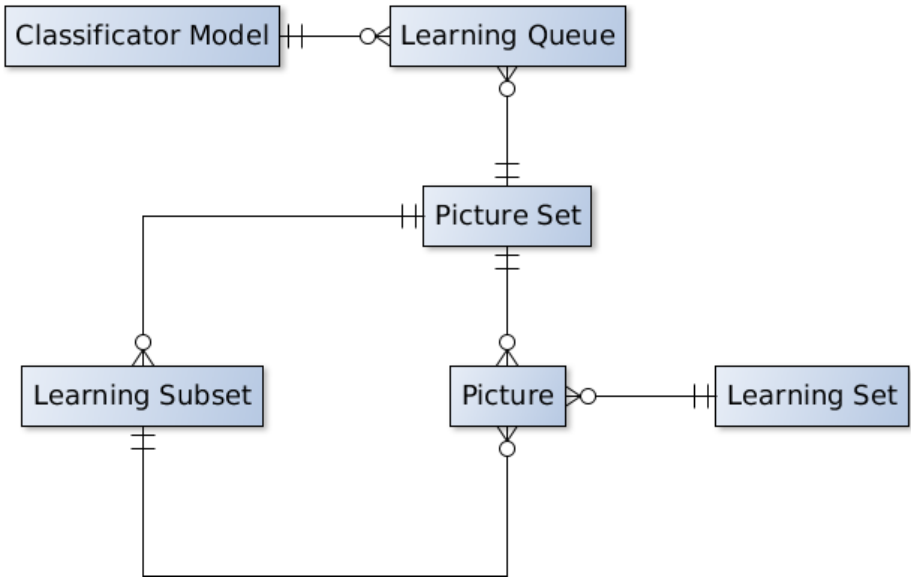
V rámci návrhu byl vytvořen ER Diagram zobrazující entity a relace mezi entitami, které jsou potřeba pro učení klasifikátorů. ER Diagram byl navržen tak, aby bylo možné vytvářet frontu pro učení klasifikátorů. Protože na jednom stroji se předpokládá, že bude učen pouze jeden klasifikátor v daném čase. Dále bylo zohledněna možnost využívat jeden model klasifikátoru nad více databázemi fotografií, které jsou zde znázorněny jako entita Picture Set. ER Diagram je zobrazen na obrázku 17.

4.5.1 Classificator Model

Tato entita znázorňuje jednotlivé modely klasifikátorů, které jsou v systému k dispozici. Tyto modely je možné kdykoliv přidávat a upravovat jejich parametry, které mají dopad na výsledek naučeného klasifikátoru.

4.5.2 Learning Queue

Entity znázorňující klasifikačního model *Classificator Model* ve frontě na učení nad danou databází fotografií Picture Set. Každý takový klasifikační model má jeden ze



Obrázek 17: ER Diagram

stavů ve kterém se nachází. Stavy mohou být dva a to stav čekám (waiting) nebo stav učím se (learning). Pokud žádný klasifikační model právě není učen, tak *Daemon* vybere jeden záznam z fronty, který je ve stavu *waiting* a započne učení s daným klasifikačním modelem a databází fotografií. Jeden klasifikační model může být ve frontě několikrát, ale pokaždé pouze s jinou databází fotografií. Není možné mít vícekrát připravené učení stejného modelu klasifikátoru a stejnou databázi fotografií a tato situace musí být ošetřena.

4.5.3 Picture Set

Tato entita představuje databází fotografií se kterou lze učit jednotlivé klasifikační modely. Tato databáze fotografií vyžaduje mít definovanou minimálně dvě kategorie do kterých bude zařazovat fotografie. Pokud tyto kategorie nebudou existovat nebo databáze fotografií nebude obsahovat žádnou fotografii, tak nebude mít klasifikátor data ze kterých by se mohl učit. Tento stav musí být ošetřen na straně klasifikátoru, aby se klasifikátor dokázal s tímto stavem správně ukončit a uvolnil korektně frontu pro jiné klasifikátory. V ideálním případě by databáze fotografií měla obsahovat minimálně dvě kategorie, aby učení klasifikátoru mělo smysl.

4.5.4 Learning Subset

Zde se nachází jednotlivé kategorie do kterých lze v databázích fotografií zařazovat fotografie. Každá databáze fotografií ze které by se měl učit klasifikátor by měla obsahovat alespoň dvě kategorie. Kategorie bude obsahovat název a unikátní hodnotu pro klasifikátor, který bude schopen kategorie dle této hodnoty rozlišit. Ze zadání diplomové práce by stačilo počítat pouze se dvěma sekcemi a to závadná a nezávadná.

Ale pro obecné použití klasifikátoru je správa sekcí vyřešená takto, aby bylo možné využívat klasifikaci i pro jiné účely než jen pro odhalení jestli daná fotografie je pornografického charakteru.

4.5.5 Learning Set

Sekce fotografií obsahují seznamy všech sekcí do kterých je možné v databázích fotografií přiřazovat fotografie. Minimálně musí obsahovat dvě entity. Entitu pro učení (learning) a entitu pro validaci (validation). Doporučená je i třetí entita pro testování (testing), díky které bude možné přiřadit databázím fotografií nějaké testovací fotografie.

4.5.6 Picture

Aby bylo možné klasifikační model naučit, tak musí databáze fotografií obsahovat minimálně sekce fotografií pro učení (learning) a pro validaci (validation) jinak klasifikátor nebude mít fotografie, které by použil pro učení. Tato skutečnost musí být pouze ošetřena na straně klasifikátoru, aby se klasifikátor dokázal s tímto stavem správně ukončit a uvolnil frontu pro jiné klasifikátory. Databáze fotografií může mít i další volitelné podsekce fotografií. Konkrétně to může být podsekce pro testování (testing), která není potřebná k učení klasifikátoru, ale je potřebná pokud chceme otestovat skutečnou přesnost finálního naučeného klasifikátoru na datech, které neměl přístupné v době učení. Entita fotografie musí obsahovat také unikátní řetězec hash, který je typu char a má délku 255 znaků. Tato hodnota slouží jako unikátní identifikátor fotografie v rámci databáze společnosti Seznam.cz.

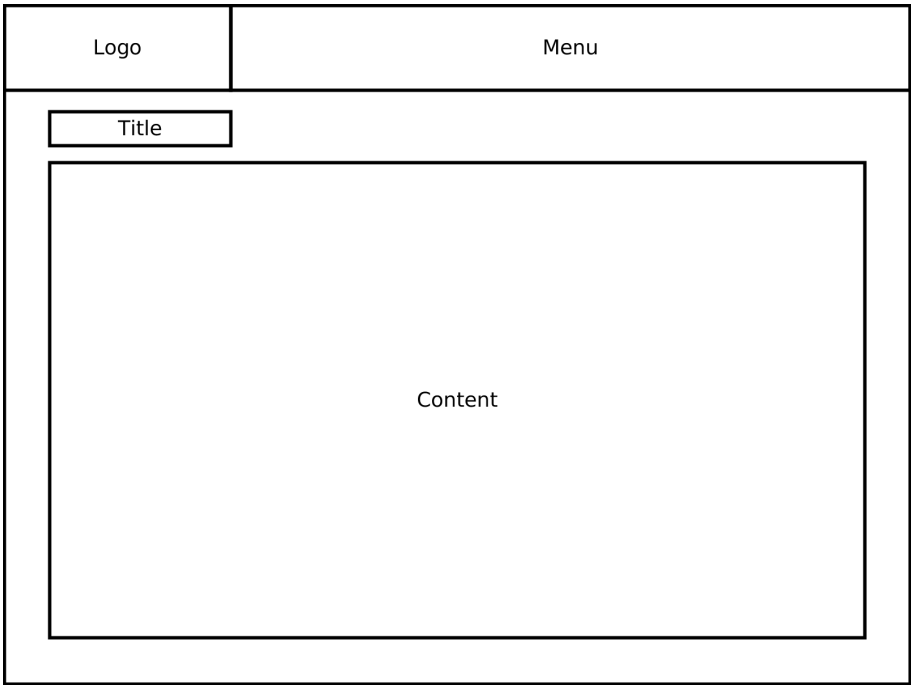
4.6 Administrační rozhraní

Administrační rozhraní zastřešuje funkcionalitu modulu pro vyhledávání nevhodných obrázků a umožňuje pohodlnější a snadnější ovládání učení klasifikačních modelů včetně jejich spravování. Je určena pouze pro zaměstnance společnosti Seznam.cz. Rozhraní bude přístupné pouze z vnitřní sítě a není vyžadována implementace přihlašování. Na tuto stránku se bude moct dostat libovolný zaměstnanec společnosti Seznam.cz, který bude znát správnou adresu administračního rozhraní.

Funkce administračního rozhraní:

- Správa databází fotografií
- Správa učení klasifikačních modelů
- Správa klasifikačních modelů
- Zobrazení stavu učení klasifikačních modelů

Navržený design administračního rozhraní má být především přehledný a snáze pochopitelný. Na obrázku 18 lze vidět, že byl zvolený běžný layout, kdy vlevo nahoře se nachází logo rozhraní a vedle něj se nachází hlavní menu. V dalších podsekcích, které jsou věnované hlavním stránkám administračního rozhraní jsou uvedeny wireframe obrázky, které jsou více podrobnější a obsahují již i návrh konkrétního obsahu stránky. Na těchto obrázcích jsou znázorněny modře akční prvky na které může uživatel klikat a červeně jsou označeny prvky, které vyvolají speciální akci. Typicky touto akcí může být uživatelský dialog pro potvrzení akce. Tmavší modrá barva udává právě vybraný prvek z hlavního menu.



Obrázek 18: Wireframe GUI administrace

Hlavní menu obsahuje tři položky:

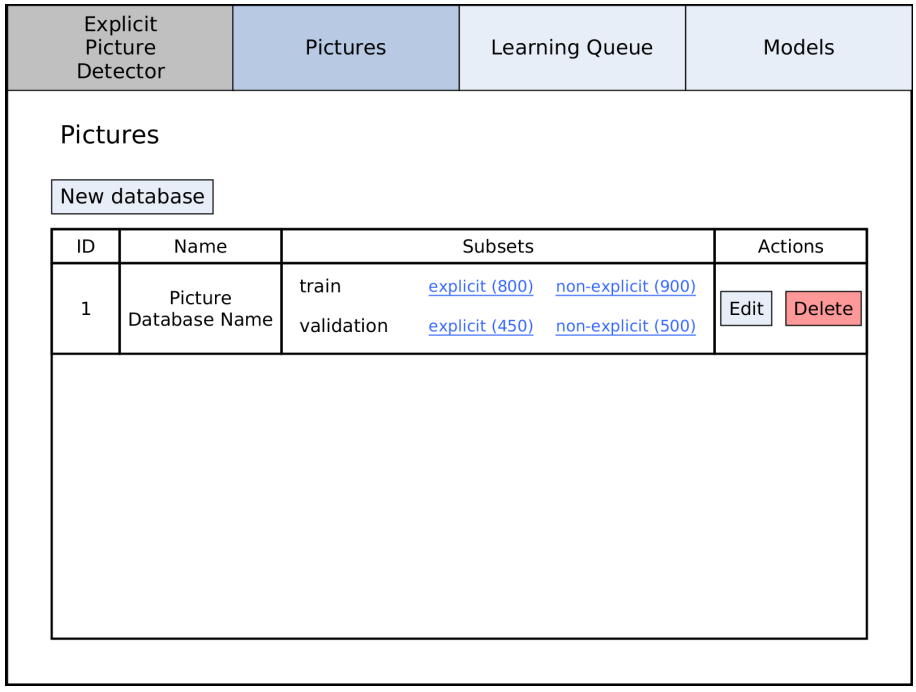
- Databáze fotografií
- Učící fronta
- Klasifikační modely

Pod hlavním menu se nachází nadpis s názvem aktuální stránky. Tento nadpis je zde proto, aby uživatel věděl na které stránce se aktuálně nachází. Hlavní obsah stránky je umístěn pod nadpisem stránky a zabírá zbytek zobrazované plochy.

4.6.1 Databáze fotografií

Na této stránce má administrátor možnost správy databáze fotografií. Každá databáze obsahuje kategorie do kterých se učí klasifikátor rozpoznávat a každá z těchto kategorií obsahuje sekci pro učení, validaci a případně další sekce určeny pro reálné testování. V těchto kategoriích jsou již zařazeny konkrétní fotografie. Wireframe návrh této stránky lze vidět na obrázku 19.

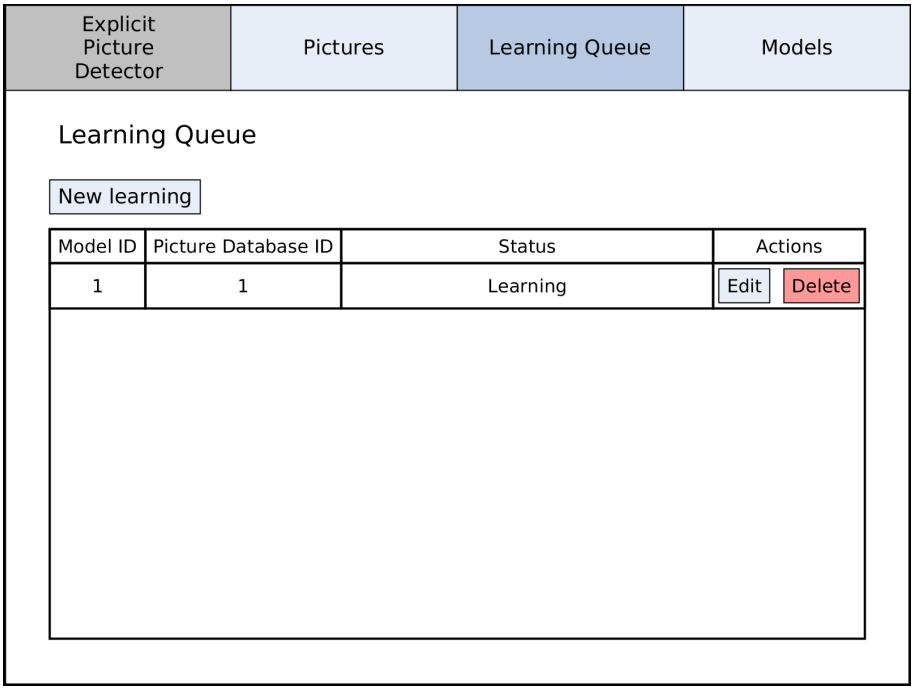
Administrátor má možnost přidávat, upravovat a mazat databáze fotografií. Při pokusu o smazání databáze fotografií musí být administrátor upozorněn dialogem, aby se předešlo nechtěným smazáním databáze fotografií. Po kliknutí na vybranou sekci se uživatel dostane do formuláře, kde může přidávat a odebírat přiřazení fotografií do této sekce. V takovém výpisu musí být minimálně zobrazeny sloupce pro identifikátor, název, sekce a sloupeček pro akce s danou databází fotografií.



Obrázek 19: Wireframe GUI administrace databází obrázků

4.6.2 Učící fronta

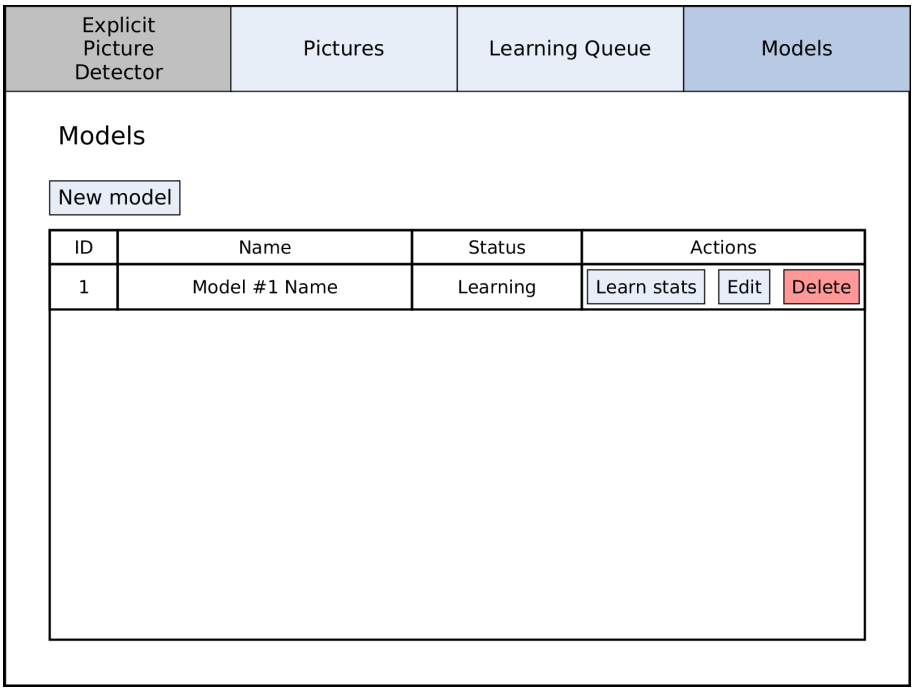
Stránka určena pro správu fronty pro učení klasifikátorů. Administrátor může přidávat, editovat a mazat záznamy v této frontě. Při pokusu o smazání musí být implementován dialog, kterým administrátor potvrdí, že chce skutečně odebrat záznam z fronty. Sloupce odpovídají wireframe návrhu z obrázku 20. Zobrazeny musí být identifikátor modelu a databáze fotografií. Dalšími položkami v tabulce je aktuální stav učení a sloupec s tlačítky pro akci spojenou s příslušným záznamem ve frontě pro učení klasifikátorů.



Obrázek 20: Wireframe GUI administrace front učení

4.6.3 Klasifikační modely

Wireframe stránky pro klasifikační modely je zobrazen na obrázku 21. Tato stránka obsahuje seznam všech dostupných klasifikačních modelů v systému. Administrátor zde může přidávat, editovat, mazat modely a zároveň je mu umožněno se dostat na stránku se statistikami učení daného modelu. Před smazáním modelu musí uživatel potvrdit mazání, aby se předešlo nechtěným smazáním modelu, protože tato akce smaže nejen model, ale i veškeré naučené klasifikátory a dočasně vytvořené soubory potřebné k učení. Minimálně musí být zobrazeny informace o identifikátoru modelu, názvu modelu, stavu modelu a sloupec pro tlačítka, které provádějí akci nad vybraným modelem.



Obrázek 21: Wireframe GUI administrace modelů

5 Řešení

Třetí částí diplomové práce je implementace navrženého řešení. Implementace proběhla na základě analýzy současného stavu. Pro klasifikování je využito konvoluční neuronové sítě, která se učí za pomocí učení s učitelem. Pro toto učení byl vybrán framework Caffe, protože nejvíce odpovídá požadavkům společnosti Seznam.cz. Více informací o implementaci frameworku naleznete v kapitole 5.1.

Implementace řešení obsahuje některé odlišnosti oproti navrhovanému řešení a především se jedná o upřesnění tříd a ER-diagramu, kdy byly doplněny specifické hodnoty potřebné pro učení klasifikátorů nad frameworkem Caffe.

5.1 Framework Caffe

Framework Caffe slouží k učení konvolučních neuronových sítí. Zvládá i více typů neuronových sítí, ale v této práci budeme využívat pouze konvoluční neuronové sítě. Tento framework je dodáván s pomocnými skripty, které jsou určeny pro usnadnění práce, obzvlášť co se učení klasifikátorů týče, protože před započetím učení je potřeba připravit data do potřebného formátu.

5.1.1 Konfigurační soubory

Caffe využívá Google Protocol Buffers [15] jako formát pro veškeré nastavení. Pod toto nastavení patří nastavení modelu pro klasifikaci, nastavení modelu pro učení a nastavení učení. Tyto soubory budou pro přehlednost uváděny pod následujícími sjednocenými názvy, které vyplývají z použití v Caffe Frameworku. Soubor s nastavením učení bude nazýván *solver konfigurace*, soubor s nastavením modelu pro klasifikaci bude nazýván *deploy konfigurace* a soubor s nastavením modelu pro učení bude nazýván *train konfigurace*. Ukázka *solver konfigurace* je zobrazena v příloze F, ukázka *deploy konfigurace* je zobrazena v příloze G a ukázka *train konfigurace* je zobrazena v příloze H.

V *solver konfiguraci* je uvedeno několik zásadních nastavení, které je potřeba správně nastavit. Prvním z nich je položka net, která obsahuje cestu k *train konfiguraci*. Pokud tato položka nebude řádně vyplněna, tak učení klasifikátoru selže. Dalším důležitou položkou je *snapshot_prefix*, která udává cestu včetně prefixu názvu naučeného klasifikátoru a uloženého stavu učení. Tato cesta musí být shodná s cestou, kterou má uloženou Backend v nastavení, aby bylo možné na straně *Backendu* zvolit pouze číslo iterace a z toho se utvořila kompletní cesta k naučenému klasifikátoru.

Ostatní parametry v *solver konfiguraci* ovlivňují učení klasifikátoru a mohou být libovolně měněny pro potřeby učení klasifikátoru. Z těchto parametrů by se měly zachovat parametry *test_interval* a *snapshot*. Parametr *test_interval* ovlivňuje jak často se bude testovat neuronová síť nad fotografiemi ze sekce validation. Tento parametr ovlivňuje přesnost výstupních grafů, respektive rozmezí mezi body v grafu učící

křivky na ose x. Parametr snapshot představuje počet iterací po kterých se má uložit naučený klasifikátor a uložit stav učení z kterého bude možné pokračovat. Tento parametr by měl být stejný jako parametr *test_interval* nebo být jeho násobkem, aby jsme měli konkrétní hodnoty v učící křivce pro konkrétní uložené iterace učení.

Konfigurace učení obsahuje kromě jiného datové vrstvy, které jsou definované pomocí cesty k souboru s daty. Tyto soubory musí být správně nastaveny jinak učení klasifikátoru selže. Tyto hodnoty by měly být vyplněny na konkrétní cesty, které odpovídají adresářové struktuře, která je uvedena v sekci 5.7. Tyto cesty jsou zobrazeny v administrační části a administrátor musí vložit tyto konkrétní uvedené cesty, aby bylo zajištěno správné učení klasifikátoru. Kromě těchto datových zdrojů mohou být v této konfiguraci ještě volitelně cesty s parametrem *mean_file*, které udávají cestu k souboru s průměrnými hodnotami barev pro jednotlivé kanály. Tento soubor je dle Caffe dokumentace využíván pro zrychlení učícího procesu, kdy není potřeba vypočítávat mean file jednotlivých fotografií, ale je vypočítán nad databází fotografií a výsledek je uložen do souboru.

5.1.2 Vstupní data pro klasifikaci

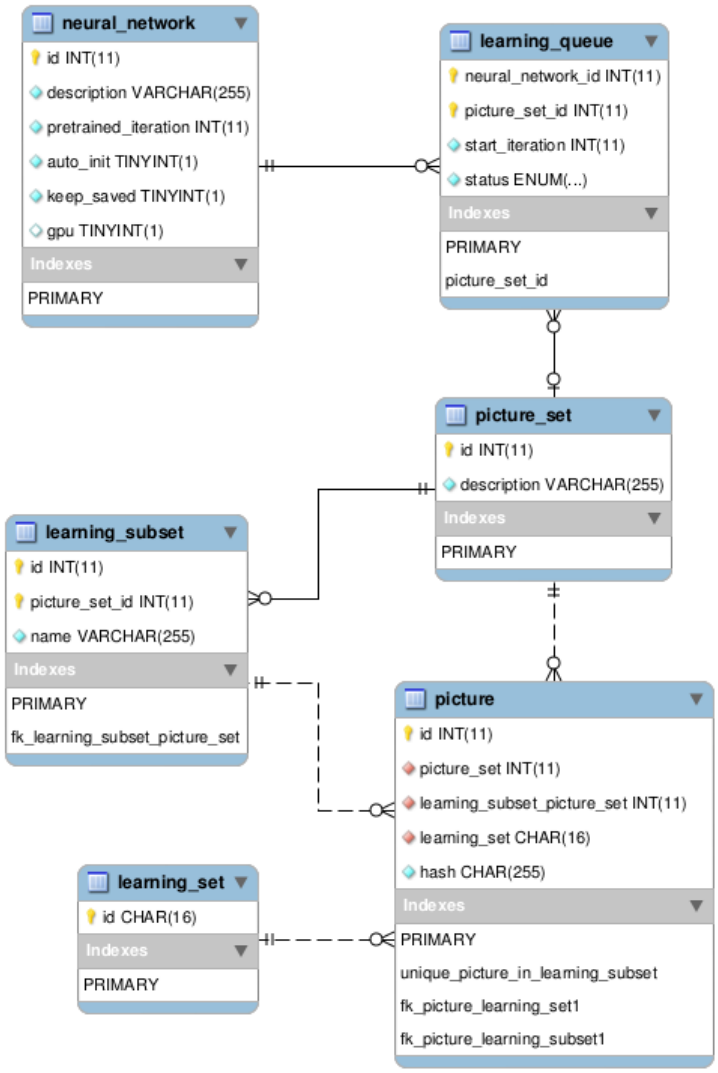
Vstupní data pro vytvoření databáze fotografií pro Caffe mají speciální formát. Formát dat je následující: Jeden záznam je vždy na jednom řádku a jako první hodnota je cesta k fotografii, která je následně oddělena mezerou za kterou následuje identifikátor kategorie do které je fotografie zaražena. Tento identifikátor je pouze číselný a začíná číslem nula. Na to je potřeba si dát pozor, protože ve výchozím stavu jsou hodnoty v databázi číslované od jedničky, což může způsobovat problémy. Po konzultaci se zaměstnanci společnosti Seznam.cz se došlo k závěru, že v databázi se nechají čísla čítat od jedničky a v kódu *Backendu* se provede zvýšení čísla kategorie o jedničku, aby odpovídaly identifikátory v databázi s identifikátory, které vrátí Backend. Dalším problémem by mohly být mezery v cestě k fotografii. Tento problém však ve společnosti Seznam.cz nenastává, takže jej není potřeba speciálně ošetřovat.

Dále je potřeba vytvořit z těchto souborů databázi z které umí framework Caffe číst fotografie. Na výběr jsou dva typy databází LevelDB [14] a LMDB [8] jelikož je na stránkách frameworku Caffe napsáno, že LMDB narozdíl od LevelDB podporuje souběžné čtení, takže z tohoto důvodu je využita LMDB databáze [18]. Tato databáze obsahuje již načtená data fotografií a v konfigurací modelů bude ve vrstvách, které jsou určeny jako vstupní cesta k takto vytvořené databázi.

5.2 Schéma databáze

Na obrázku 22 je zobrazeno databázové schéma, které obsahuje všechny potřebné tabulky pro správnou funkcionalitu modulu. Oproti původnímu návrhu došlo k upřesnění pojmenování tabulek, kdy místo obecného názvu *Classificator Model*, byla tabulka pojmenována *neural_network*. Ostatní tabulky mají stejné názvy, pouze

byly převedeny do podtržítkové notace, aby byly konzistentní v rámci ostatních databázových tabulek v systému Lidé.cz.



Obrázek 22: Databázové schéma

5.2.1 neural_network

Tabulka *neural_network* obsahuje tyto atributy:

id
: Identifikátor modelu neuronové sítě. Zároveň slouží jako primární klíč.

description
: Krátký textový popisek. V administračním rozhraní slouží k odlišení modelů. Z původního návrhu nahrazuje atribut name.

pretrained_iteration
Číslo naučené iterace klasifikátoru, který je uložen a který se má použít pro klasifikaci na daném modelu. Číslo specifikující iteraci klasifikátoru musí být pouze z existující množiny uložených stavů klasifikátoru. Tato situace musí být ošetřena v administračním rozhraní, protože nad úrovní databáze není možné takové ošetření vytvořit.

auto_init
Příznak označující, jestli se má vybraný klasifikátor (s vybranou konkrétní iterací) inicializovat při startu *Daemona*. Díky tomuto je dosaženo vyšší rychlosti klasifikace. Nevýhodou je, že klasifikátor je inicializován, i když se nikdy nevyužije.

keep_saved
Druhá možnost zrychlení klasifikace. Klasifikátor, který má tento příznak nastaven na hodnotu 1 bude při prvním požadavku na klasifikaci, který zpracovává *Daemon*, uložen a při příštím požadavku na klasifikaci bude již načten z cache inicializovaný.

gpu
Tímto příznak se dá definovat jestli klasifikace má probíhat pomocí grafické karty nebo jen čistě na procesoru.

5.2.2 learning_queue

Tabulka *learning_queue* je oproti ER diagramu rozšířena o atribut *start_iteration* díky kterému je možné pokračovat v učení od uložené iterace klasifikátoru. V tabulce je využit složený primární klíč z atributů *neural_network_id* a *picture_set_id*. Tento primární klíč zajišťuje požadované chování, kdy je možné do fronty pro učení vložit pouze jednu instanci učení pro jednu neuronovou síť a jednu databázi fotografií.
Tabulka *learning_queue* obsahuje tyto atributy:

neural_network_id
Identifikátor neuronové sítě, která má být naučena.

picture_set_id
Identifikátor databáze fotografií z které se má neuronová síť naučit.

start_iteration
Počáteční iterace učení. Pokud se má neuronová síť učit od začátku, tak se zde uvede hodnota 0. Číslo určující iteraci klasifikátoru od kterého se má v učení pokračovat musí být pouze z existující množiny uložených stavů klasifikátoru. Tato situace musí být ošetřena v administračním rozhraní, protože nad úrovní databáze není možné takové ošetření vytvořit.

status
Stav ve kterém se učení právě nachází. Atribut je typu *enum* a obsahuje dvě hodnoty. Waiting pro znázornění stavu, že záznam čeká ve frontě a Learning pro stav, kdy se neuronová síť právě učí. Neuronové sítě, které selžou nebo jsou zrušeny, jsou z této fronty odebrány.

5.2.3 picture_set

Tato tabulka uchovává databáze fotografií, které jsou v systému definované. Tabulka *picture_set* obsahuje tyto atributy:

id
Identifikátor databáze fotografií. Zároveň slouží jako primární klíč.

description
Krátký popisek označující databázi fotografií. Maximální délka popisku je 255 znaků a tento atribut je určen k rozeznání databází fotografií v administračním rozhraní.

5.2.4 picture

V této tabulce jsou přiřazeny fotografie včetně jejich rozdělení do databáze fotografií, do příslušné kategorie a sekce. Nejsou zde uloženy fotografie samotné, ale je zde pouze atribut hash, což je v systému Lidé.cz unikátní hash fotografie, který je stejný jako vygenerovaný název fotografie. Tudíž z tohoto hashe je možné vytvořit cestu k dané fotografii.
Tabulka *picture_set* obsahuje tyto atributy:

id
Primární klíč tabulky. Tento primární klíč je zde především pro zjednodušení manipulace nad daty, protože umožňuje manipulaci s položkou pomocí jedné hodnoty a nemusíme pracovat s celým složeným primárním klíčem, který by zahrnoval všechny čtyři atributy této tabulky.

picture_set
Přiřazuje fotografii k dané databázi fotografií. Každá databáze fotografií může mít určitou fotografii použitou pouze jednou. To znamená, že například nemůžeme použít stejnou fotografii pro učení a pro testování, protože by došlo ke zkreslení výsledků klasifikátoru. Stejné omezení je potřeba vytvořit v rámci kategorií. Nesmí být povoleno přiřadit fotografii do dvou kategorií jelikož by to snižovalo přesnost naučeného klasifikátoru. Tyto skutečnosti jsou ošetřeny v databází pomocí unikátního indexu *unique_picture_in_learning_subset*, který v sobě kontroluje unikátnost hodnot pomocí atributů *picture_set* a *hash*. Tento index zajistí, že uživateli nebude umožněno vložit stejnou fotografii do různých sekcí nebo do různých kategorií v rámci jedné databáze fotografií.

learning_subset_picture_set
Tímto atributem se fotografie přiřadí do kategorie do které se bude daný klasifikátor učit. Každá fotografie může být pouze v jedné kategorii v dané databázi fotografií.

learning_set
Tento atribut udává do jaké sekce fotografie spadá. Sekce musí být v systému definovány minimálně dvě a to sekce pro učení a sekce pro validaci. Tyto dvě sekce jsou využity při učení klasifikátoru a bez nich by učení neuronové sítě nemohlo probíhat. Třetí volitelná sekce je pro testování, kdy jsou fotografie v této sekci odeslány na klasifikaci a z této klasifikace se získá skutečná přesnost klasifikátoru z testovacích fotografií, které klasifikátor neznal v době učení.

hash
Unikátní hash fotografie, který musí mít předepsanou strukturu, aby byl kompatibilní v rámci celého systému Lidé.cz. Tento hash je typu varchar a má délku 255 znaků. Díky tomuto hashi je možné vygenerovat cestu k fotografii a provést klasifikaci fotografie, která bude načtena z konkrétní cesty na disku.

5.2.5 learning_subset

Tabulka definující kategorie, které máme v systému a do kterých se klasifikátory učí klasifikovat. Každá kategorie patří do předem vytvořené databázi fotografií.
Atributy tabulky *learning_subset*:

id
Unikátní klíč kategorie v rámci databáze fotografií. Tento unikátní klíč je číslován vždy od jedničky a je tomu tak pro každou databázi fotografií. Tento unikátní klíč je spojen s číslem kategorie, které se udává při učení klasifikátoru v rámci frameworku Caffe, kde vstupní data pro učení klasifikátoru mají zápis typu cesta k souboru za kterým následuje mezera a poté se nachází číslo kategorie do které fotografie patří. Tento atribut zároveň tvoří část primárního klíče, který se skládá z atributů *id* a *picture_set_id*.

picture_set_id
Přiřazení kategorie k databází fotografií. Atribut tvoří část primárního klíče, který se skládá z atributů *id* a *picture_set_id*.

name
Název kategorie, který se zobrazuje v administračním rozhraní. Maximální délka názvu je 255 znaků.

5.2.6 learning_set

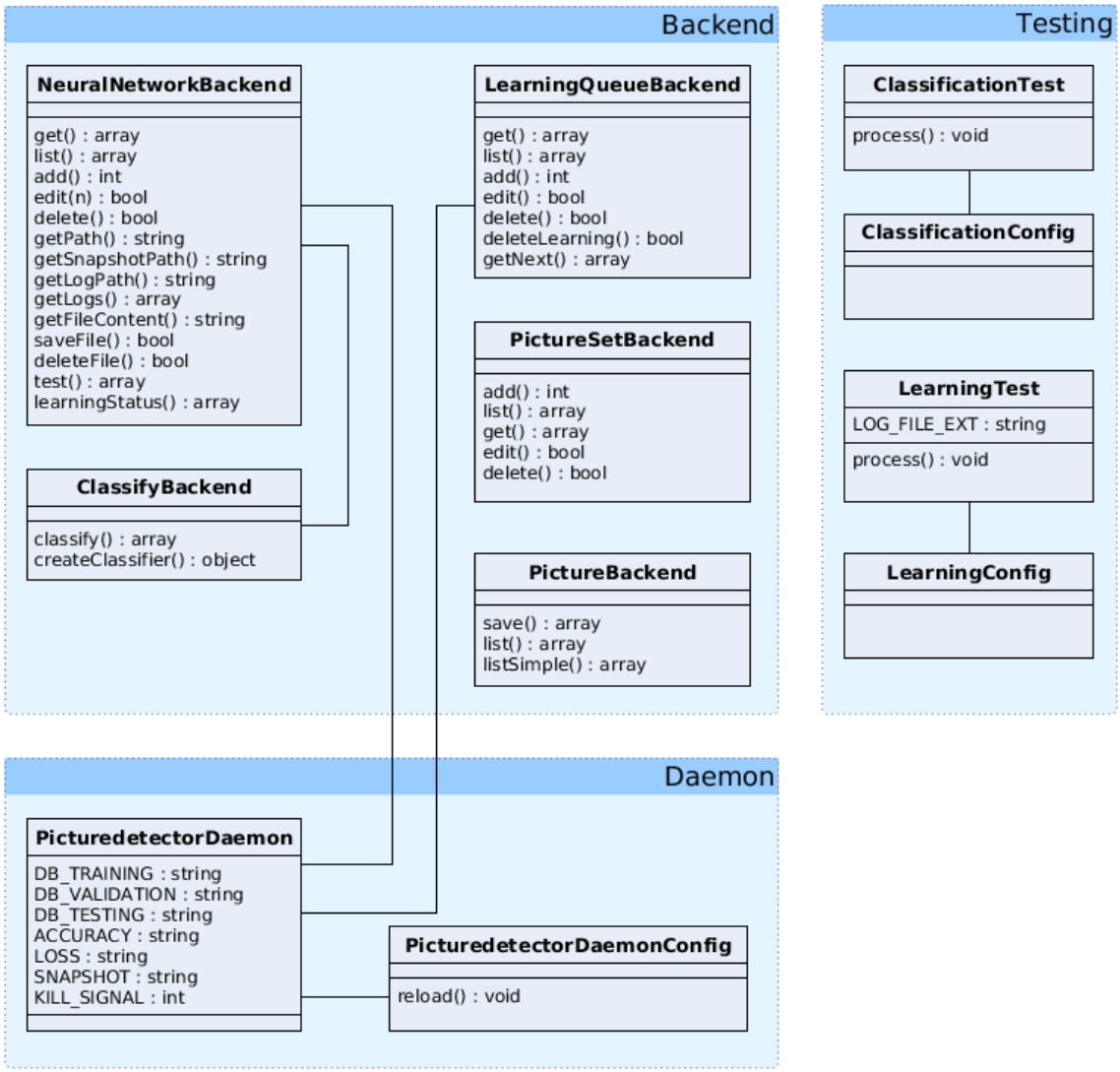
Tato tabulka představuje číselník sekcí do kterých lze fotografie přiřazovat. Tento číselník by mohl být nahrazen atributem typu *enum*, ale pro případné snadnější

rozšiřování číselníku bylo společností Seznam.cz upřednostněna možnost, že bude vytvořena speciální tabulka s číselníkem.
Atributy tabulky *learning_set*:

id

Atribut tvoří primární klíč tabulky a zároveň se jedná o textovou hodnotu, která udává název sekce. Tato hodnota může mít maximální délku 16 znaků a aktuálně se v systému nacházejí tři sekce: *learning*, *testing* a *validation*.

5.3 Diagram tříd



Obrázek 23: Diagram tříd

Na obrázku 23 je uveden diagram tříd, který byl vytvořen v rámci modulu pro detekci závadných obrázků. Tyto třídy jsou rozděleny do tří částí. První částí je Backend a zde se nacházejí třídy *Backendu*, které jsou rozděleny dle entit s kterými pracují. Pouze entity *Learning Subset* a *Learning Set* nemají vlastní třídu a jejich obsluha je zařazena do třídy *PictureSetBackend*. Je zde i speciální třída *ClassifyBackend*, která má na starosti klasifikaci a není svázaná s žádnou konkrétní entitou z ER diagramu.

Druhou část diagramu tříd tvoří *Daemon*, zde se nacházejí dvě třídy *PicturedetectorDaemon* a *PicturedetectorDaemonConfig*. Třída *PicturedetectorDaemonConfig* pouze obstarává načítání konfigurace pro třídu *PicturedetectorDaemon* a je to nutná třída, která zajišťuje správné nakonfigurování *Daemon* třídy v rámci interního systému Lidé.cz.

Třetí část tvoří *Testing*, což jsou třídy, které vznikly v rámci testování vytvořených klasifikátorů. Testovala se především rychlost klasifikace a rychlost učení, aby bylo možné zjistit jestli je reálné, aby se klasifikace fotografií prováděla reálném čase při nahrání fotografie. Opět třídy *ClassificationConfig* a *LearningConfig* slouží pro načtení konfigurace pro příslušné třídy *ClassificationTest* a *LearningTest* a jsou to nutné třídy, které umožňují správné nakonfigurování tříd v rámci interního systému na kterém běží Lidé.cz.

5.4 Backend třídy

Backend třídy slouží pro komunikaci modulu pro vyhledávání závadných obrázků s okolním systémem.

5.4.1 NeuralNetworkBackend

Jedná se o hlavní třídu *Backendu*. Zde se nacházejí metody, které pracují s modely neuronových sítí a jsou zde pomocné metody pro *Daemona*, který pomocí nich získává informace potřebné pro započetí učení klasifikátoru.

Metody třídy:

get
: Metoda pro získání detailních informací o neuronové síti. V návrhu odpovídá metodě *getModel()*.

list
: Díky této metodě lze získat seznam všech základních informací o neuronových sítích. Tato metoda odpovídá metodě listModels z návrhu.

add
: Přidání neuronové sítě do databáze dostupných neuronových sítí. Odpovídá metodě addModel z návrhu.

edit
Úprava neuronové sítě uložené v databázi dostupných neuronových sítí. Tato metoda odpovídá metodě editModel ze sekce návrh.

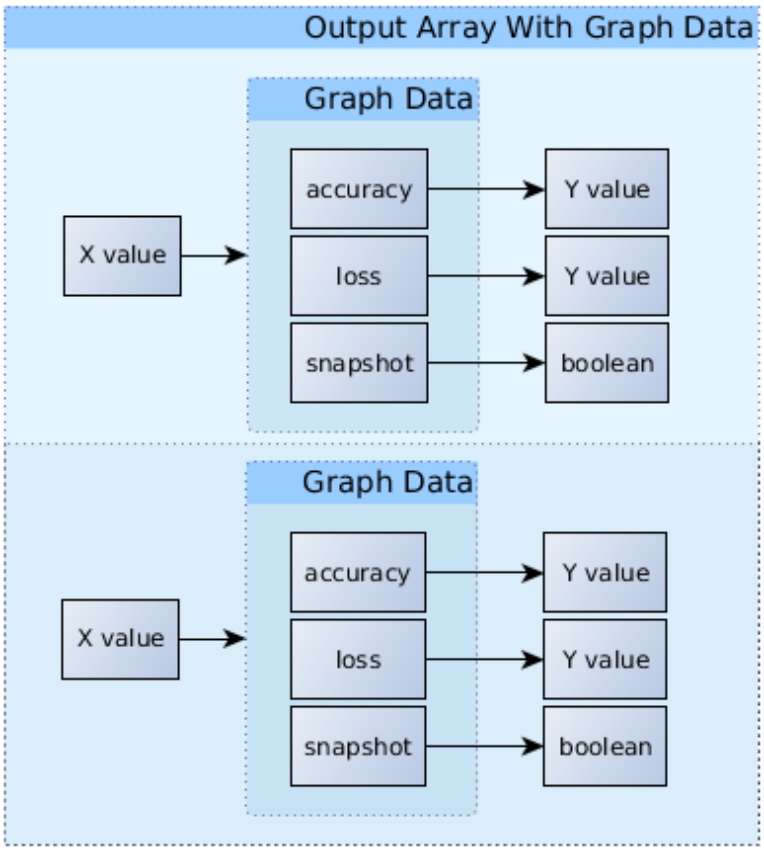
delete
Smazání vybrané neuronové sítě z databáze neuronových sítí. Metoda vznikla na základě metody deleteModel v sekci návrh.

test
Metoda, která je schopna provést testování přesnosti neuronové sítě. Očekávané parametry metody jsou identifikátor neuronové sítě a identifikátor databáze fotografií, která se má použit pro testování. Zadaná databáze fotografií musí mít nějaké přiřazené fotografie do sekce *testing*, protože z této sekce se budou fotografie načítat. Je možné model testovat i na jiné databázi fotografií než na které byl naučen. To se může hodit, pokud chceme otestovat vhodnost naučeného klasifikátoru pro jinou úlohu nebo pokud chceme mít více testovacích databází fotografií pro daný klasifikátor. Návratová hodnota metody je pole ve kterém je po řádcích uložen log obsahující výsledky testování.

learningStatus
Účelem této metody je získat statistiku učení neuronové sítě pro daný log učení. Parametry metody jsou identifikátor neuronové sítě a název logu, který se má zpracovat. Pokud log učení neexistuje, tak tato metoda vyvolá vyjímku. Díky využití metody *getLogs()* pro zobrazení seznamu logů v administračním rozhraní je zajištěno, že log skutečně existuje. Návratová hodnota metody je zobrazena na obrázku 24. Klíčem v poli je číslo iterace, které se má v grafu zobrazit na ose x. Dále v poli pod tímto klíčem jsou tři prvky, kde prvek pod klíčem *accuracy* obsahuje hodnotu y, která udává přesnost klasifikátoru nad učícími daty. Druhý prvek je pod klíčem *loss* a obsahuje hodnotu y, která udává velikost chyby nad validačními daty. Třetí prvek pod klíčem snapshot obsahuje boolean hodnotu, která nám sděluje jestli se v dané iteraci ukládal snapshot či nikoliv. Tato informace je vhodná pokud v konfiguraci solveru jsou nastaveny rozdílné hodnoty parametrů *test_interval* a *snapshot* což způsobí, že ukládání probíhá jindy než validace klasifikátoru. Jelikož se tyto hodnoty čtou z logu, tak může nastat případ, že metoda bude vracet, že se klasifikátor v dané iteraci uložil, ale přitom klasifikátor na disku neexistuje. To může nastat pokud administrátor ručně smaže daný soubor z disku. Tato informace sděluje, jestli byl stav klasifikátoru uložen, nikoliv jestli aktuálně existuje.

5.4.2 ClassifyBackend

Třída která obsluhuje klasifikaci na natrénovaném klasifikátoru. Zároveň slouží jako interface mezi *Backendem* a frameworkem Caffe, protože je to jediná třída, která importuje framework Caffe a následně volá jeho metody.



Obrázek 24: Pole se statistikami učení klasifikátoru

classify

Metoda pro klasifikaci fotografií. Odpovídá metodě *classify()* z návrhu. Jako parametry očekává identifikátor neuronové sítě a pole s fotografiemi, které musí být ve formátu jak je specifikováno na obrázku 11. Výstupem metody je poté pole, které odpovídá navrhovanému výstupu a je zobrazeno na obrázku 12.

5.4.3 LearningQueueBackend

V této třídě se nacházejí metody, které obsluhují frontu pro učení nových neuronových sítí.

Třída obsahuje následující metody:

list

Metoda vypisuje seznam informací pro všechny záznamy ve frontě pro učení nových neuronových sítí. Nevyžaduje žádný parametr a návratovou hodnotou je pole, které obsahuje informace načtené z databáze a hodnoty jsou uloženy vždy pod klíčem, který má stejný název jako sloupec v databázi. Tato metoda je shodná s metodou *listQueue()*, která byla definována v návrhu.

add

Pomocí této metody lze přidávat nové záznamy do fronty pro učení. Očekáván je jeden parametr typu pole. V tomto poli musí být uloženy hodnoty pro započatí učení a musí být uloženy pod konkrétními klíči, které mají stejné názvy jako sloupečky v příslušné databázové tabulce *learning_queue*. Metoda odpovídá metodě *addToQueue()*, která byla zjednodušena o počet předávaných parametrů, kdy navrhované parametry *modelId* a *picturesDbId* jsou předány přímo v poli s daty.

edit

Metoda sloužící k editaci záznamu z fronty pro učení. Odpovídá metodě *editQueue()* z návrhu a zachovává i stejné parametry, kdy prvním parametrem je identifikátor neuronové sítě, druhým parametrem je identifikátor databáze fotografií a třetí parametr je pole, které obsahuje data, které lze změnit. V tomto poli jsou hodnoty uloženy pod klíči, které odpovídají názvům sloupců v databázi. Konkrétně lze měnit atributy *status* a *start_iteration*.

delete

Tato metoda se používá k mazání záznamu z fronty pro učení. Její definice odpovídá metodě, která se v návrhu jmenovala *deleteQueue()*. Očekává dva parametry, kterými se identifikuje záznam v databázi. Těmito parametry jsou identifikátor neuronové sítě a identifikátor databáze fotografií.

deleteLearning

Aktuálně může v systému probíhat pouze jedna instance učení neuronoví sítě a proto byla vytvořena tato metoda, která nevyžaduje žádné parametry a je schopna odstranit právě učený záznam. Této metody je využito v případě, kdy běžící *Daemon* dostane signál SIGABRT, což znamená že má ukončit aktuálně učenou instanci klasifikátoru a případně začít učení klasifikátoru jiného, pokud ve frontě pro učení neuronových sítí je nějaký záznam ve stavu waiting.

getNext

Daemon, který neučí žádnou neuronovou síť se cyklicky s daným intervalem dotazuje *Backendu* jestli existuje ve frontě pro učení nějaká neuronová síť, kterou by mohl zpracovat. Proto existuje tato metoda, která odpovídá metodě *getNextFromQueue()*. Metoda neočekává žádný parametr. Pokud existuje neuronová síť, tak vrátí načtené informace o tomto záznamu učení ve formě pole a pokud žádná neuronová síť není ve frontě ve stavu waiting, tak vrátí hodnotu False.

5.4.4 PictureBackend

Třída určena pro práci s fotografiemi. Umožňuje jejich přiřazování do kategorií a do sekcí a získávání seznamu přiřazených fotografií.

Třída obsahuje tyto metody:

save
Tato metoda umožňuje nahrát novou fotografii do databáze fotografií. Metoda očekává čtyři parametry, kdy první tři parametry jsou identifikátory a čtvrtý parametr jsou binární data fotografie, která je nahrána z administračního rozhraní. První parametr je identifikátor databáze fotografií, druhý identifikuje kategorii do které fotografie patří a třetí identifikátor slouží pro upřesnění sekce ve které má být fotografie použita. Výstupem metody je pole, které obsahuje dvě hodnoty. Hodnota pod klíčem *id* udává identifikátor fotografie a hodnota pod klíčem *hash* udává unikátní hash v rámci systému Lidé.cz, který byl fotografii přidělen. Tento hash zároveň slouží jako název fotografie, která je na serveru uložena s příponou .jpg.

list
Metoda pro vypsání seznamu fotografií včetně jejich informací o příslušných kategorií do kterých spadají. Očekávány je jeden povinný parametr a to identifikátor databáze fotografií pro kterou se mají fotografie získat a jeden volitelný parametr typu pole, který obsahuje filtr, který je možné nastavit dle potřeby. Hodnoty ve filtru jsou nastaveny tak, že název sloupce podle kterého se má filtrovat je uložen jako klíč v poli a hodnota pod tímto klíčem udává hodnotu, dle které se má filtr provést. Filtrovat je možné dle sloupců *learning_set*, *learning_subset_id* a *learning_subset*. Návratová hodnota je typu pole, kde jsou uloženy všechny informace z databázových tabulek *picture* a *learning_subset*.

5.4.5 PictureSetBackend

Tato třída je určena pro práci nad databázemi fotografií. Umožňuje přidávat, upravovat a mazat databáze fotografií.

V této třídě se nacházejí tyto metody:

add
Metoda pro vytvoření nové prázdné databáze fotografií. Očekávány jsou dva vstupní parametry, kdy první parametr je typu pole a obsahuje hodnoty, které se mají uložit do tabulky *picture_set*. Klíčem v tomto poli je vždy název sloupce do kterého se má hodnota uložit. Aktuálně je možné nastavit pouze atribut *description*. Druhým parametrem je pole s názvy učících kategorií. Tyto kategorie musí být minimálně dvě jinak bude metoda vyvolá vyjímku. Maximální počet kategorií není nijak omezen. Výsledkem metody je vytvořený identifikátor databáze fotografií.

delete
Slouží k vymazání konkrétní databáze fotografií, která je specifikovaná pomocí identifikátoru databáze fotografií. Spolu s databází budou smazány také ka-

tegorie, které této databázi fotografií příslušely. Dojde i k odebrání přiřazení fotografií do příslušných smazaných kategorií a sekcí.

edit
Metoda sloužící pro úpravu databáze fotografií. Je očekáván jeden vstupní parametr typu pole, který obsahuje jako klíče názvy sloupců, které mají být aktualizovány a hodnoty pole jsou nové hodnoty, které se mají nastavit do databáze. Aktuálně je podporován pouze atribut *description*. Aktuálně není možnost upravovat kategorie, což po konzultaci ve společnosti Seznam.cz není ani vyžadováno a předpokládá se, že kategorie budou založeny správně již při založení databáze fotografií.

get
Získání podrobných informací o vybrané databázi fotografií, která je specifikovaná pomocí jejího identifikátoru. Návratová hodnota metody je pole, kde jsou všechny informaci z databázové tabulky *picture_set* a navíc pod klíčem *pictures_count* jsou uloženy informace o počtu fotografií v jednotlivých sekcích. Pod klíčem *learning_subsets* jsou pak uloženy názvy kategorií, které daná databáze fotografií obsahuje. Názvy kategorií mají jako klíč uloženy identifikátor kategorie.

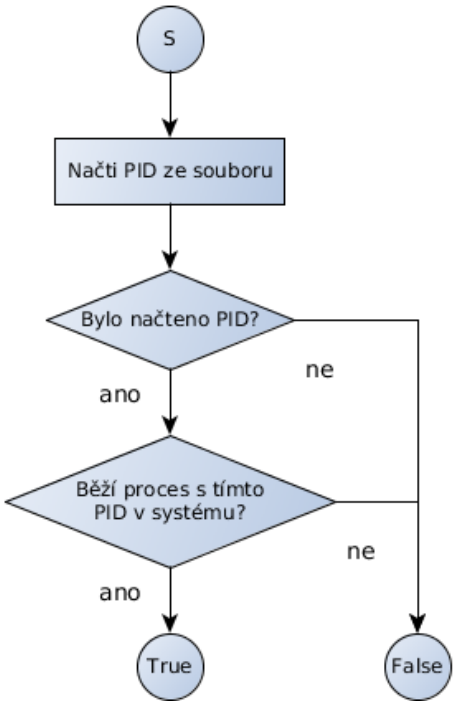
list
Vypíše seznam databází fotografií, které jsou v systému dostupné. Neočekává žádné vstupní parametry a výstup je shodný s výstupem metody *get*.

5.5 Daemon třídy

Daemon je spuštěn na pozadí a plní požadavky, které na něj přicházejí. *Daemon* třídy jsou zaměřené pro učení a testování neuronových sítí a jsou to akce, které se vyznačují dlouhou dobou zpracování a proto není možné očekávat okamžitou odpověď, protože učení neuronové sítě probíhá v řádech dnů až týdnů a testování neuronové sítě může pohybovat v řádu sekund až minut. Obě tyto časové rozmezí závisí na složitosti neuronové sítě a počtu fotografií nad kterými se testování provádí.

5.5.1 PicturedetectorDaemon

Tato třída běží na pozadí a zajišťuje učení neuronových sítí nad frameworkem Caffe. Třída je úzce spojena se systémem na kterém běží Lidé.cz a proto obsahuje řadu metod, které nebyly v původním návrhu a jsou potřebné pro správné pracování na daném systému. Pro zjišťování jestli probíhá nějaké učení je využito PID souboru, který je uložen na lokaci, která je specifikována v konfiguraci *Daemona*. Metody, které začínají dvěma podtržítky jsou interní metody *Daemona*, které jsou volány pouze z hlavní smyčky ve které Daemon pracuje.



Obrázek 25: Kontrola jestli probíhá učení

Třída má tyto metody:

__learningInProgress
: Interní metoda *Daemona*, která zjišťuje jestli právě probíhá nějaké učení. Metoda nevyžaduje žádné parametry a vrací hodnotu typu boolean konkrétně hodnotu True pokud nějaké učení právě probíhá. Metoda funguje tak, že pokud existuje soubor s uloženým PID, tak jej načte a následně zkontroluje jestli v systému běží proces se zadaným PID. Pokud neběží, tak vypíše do logu chybu, že daný proces neběžel a vrátí False, čímž dá najevo, že žádné aktuální učení neprobíhá. Diagram zobrazující jak funguje testování právě probíhajícího učení je zobrazen na obrázku 25.

__stopLearningProcess
: Zastavení učícího procesu. Tato interní metoda je zavolána pokud Daemon dostane systémový signál SIGABRT. Po přijetí tohoto signálu dojde ke kontrole jestli nějaké učení běží a pokud ano, tak bude následně ukončeno. V dalším cyklu *Daemona* pak může započíst učení nové. Odpovídá metodě *stopLearning* ze sekce návrhu.

_startCaffeLearning
: Pomocná metoda, která zastřešuje započatí učení nad Frameworkem Caffe. V této metodě se volají veškeré pomocné skripty, které jsou potřeba, aby mohlo započít učení. Diagram znázorňující akce, které v této metodě probíhají je

zobrazen na obrázku 26. Metoda očekává tři parametry kterými jsou identifikátor neuronové sítě, identifikátor databáze fotografií a volitelný parametr, který udává číslo iterace od kterého se má učení spustit. Tento parametr se využívá, pokud již klasifikátor naučený byl a chceme v učení pokračovat. Pokud se tento parametr nevyplní, tak se jako výchozí hodnota nastaví hodnota 0, tudíž bude klasifikátor učen od začátku.

_processIteration
Hlavní cyklus *Daemona*. Zde probíhá všechna logika *Daemona*, který zde kontroluje jestli aktuálně běží učení a pokud ne, tak jestli existuje neuronová síť, která čeká na učení. Odpovídá metodě *processIteration* ze sekce návrh. Logika této metody je zobrazena na obrázku 13.

_createClassifyMeanFile
Jelikož framework Caffe vyžaduje jiný formát mean file souboru při učení a při klasifikaci, tak tato metody vytváří mean file soubor v numpy [3] formátu, který se načítá při vytváření klasifikátoru a zajišťuje rychlejší klasifikaci. Vstupními parametry metody je cesta k trénovacímu mean file souboru a cesta, kde se má uložit výsledný mean file v numpy formátu.

5.5.2 PicturedetectorDaemonConfig

Tato třída je zde kvůli funkcionalitě se systémem na kterém běží Lidé.cz a obstarává načítání konfigurace.

5.6 Testing třídy

Tyto třídy vznikly pro testování rychlosti frameworku Caffe a zjišťování jestli je možné provádět klasifikaci v reálném čase. Zároveň se testovala rychlost učení, aby bylo možné říct přibližné časové rozmezí pro naučení jednoho klasifikátoru.

5.6.1 ClassificationTest

Třída zaměřená na testování rychlosti klasifikace klasifikátoru.

V třídě jsou následující metody:

process
Provede časový test na všech neuronových sítích, které se nacházejí v databázi. Test rychlosti klasifikace se provede pro 1, 2, 5, 10 a 20 fotografií. Fotografie nejsou načteny z databáze, ale ze složky, která je definovaná v konfiguraci. Je potřebné dodržet, aby složka obsahovala dostatečný počet fotografií, tedy v tomto případně minimálně 20 fotografií.

5.6.2 LearningTest

Účelem této třídy je měřit rychlost učení jednotlivých modelů neuronových sítí. Nenačítá modely z databáze, nýbrž načítá modely ze zadaná složky, kterou má nastavenou v konfiguraci.

Třída obsahuje následující metody:

process
Metoda pro provedení časového měření na modelech neuronových sítí, které nalezne v pevně dané složce nastavené v konfiguraci. Provádí se dva testy. První test je spuštění Caffe skriptu, který je určen na časový test učení. Druhý test je klasické spuštění učení jako to dělá Daemon. Počet iterací učení, které se mají provést je uveden v konfiguraci.

5.7 Adresářová struktura klasifikátoru

Pro zjednodušení operací s učením neuronových sítí byla vytvořena jednotná adresářová struktura, která umožňuje přehledné uložení vygenerovaných dat.

Každá neuronová síť má svojí složku s identifikátorem a v této složce jsou následující podsložky:

- logs
- snapshots
- temp

V kořenové složce neuronové sítě jsou uloženy *protocol buffer* soubory s konfiguracemi a vygenerovaný mean file v *numpy* formátu určený pro klasifikaci.

Složka *logs* obsahuje všechny logy učení, které byly nad danou neuronovou sítí naučeny. Tyto logovací soubory jsou využívány při zobrazování statistik učení v administračním rozhraní.

Složka *snapshots* obsahuje jednotlivé iterace naučených klasifikátorů a uložených stavů klasifikátoru pro pozdější možnost pokračování v učení. V administračním rozhraní lze nastavit pokračování v učení pouze z těch klasifikátorů, které jsou uloženy v této složce. Obdobně lze zvolit pro klasifikování pouze tu iteraci klasifikátoru, která existuje v této složce.

Složka *temp* obsahuje dočasné soubory, které jsou potřebné k naučení neuronové sítě. Po úspěšném naučení neuronové sítě mohou být soubory v této složce smazány.

5.8 Učení klasifikátoru

Než započne samotné učení klasifikátoru, tak je potřeba pro framework Caffe připravit několik věcí. Některé z nich nejsou povinné (například vytvoření mean file), ale jsou doporučené, protože díky ním může učení probíhat rychleji. Diagram

znázorňující akce, které se provádí před začátkem učení je zobrazen na obrázku 26. Při učení klasifikátorů se využívá jak povinných, tak doporučených kroků, aby bylo zajištěno nejrychlejší možné učení klasifikátorů.

Na začátku se přečte z *Backendu* cesta k souboru se *solver konfigurací*. Po získání tohoto souboru se načte *solver konfigurace*, která obsahuje nastavené cesty k souboru s *train konfigurací* a cestu, kde se mají ukládat naučené klasifikátory. Následně jsou vytvořeny soubory s fotografiemi pro framework Caffe. Tyto soubory jsou uloženy do dočasné složky dané neuronové sítě. Poté se vytvoří nový soubor, který bude obsahovat záznam učení klasifikátoru. Tento záznam učení je uložen do složky pro záznamy učení neuronové sítě. Tento soubor se záznamem učení bude poté možno číst a vytvářet z něj grafy učení neuronové sítě.

Jelikož má framework Caffe problém při vytváření LMDB databází fotografií pokud již cílová složka existuje, tak se provede vymazání LMDB databází s fotografiemi v dočasné složce neuronové sítě, aby se předešlo případným problémům při znovu učení neuronové sítě. Následně se pomocí skriptu dodaného spolu s frameworkem Caffe vytvoří LMDB databáze fotografií z kterých umí framework Caffe načítat data. Tyto databáze jsou vytvořeny dvě. Jedna z databází je pro učící fotografie a druhá databáze je pro validační fotografie.

Když je vytvořena učící LMDB databáze, tak je možné z této databáze vytvořit mean file soubor, který obsahuje průměrné hodnoty fotografií. Vytvoření tohoto souboru je volitelné, ale přispívá k rychlejšímu učení neuronové sítě, takže je doporučené jej vytvořit. V průběhu tohoto procesu se shromažďují parametry pro učící skript, který je dodáván spolu s Caffe frameworkem.

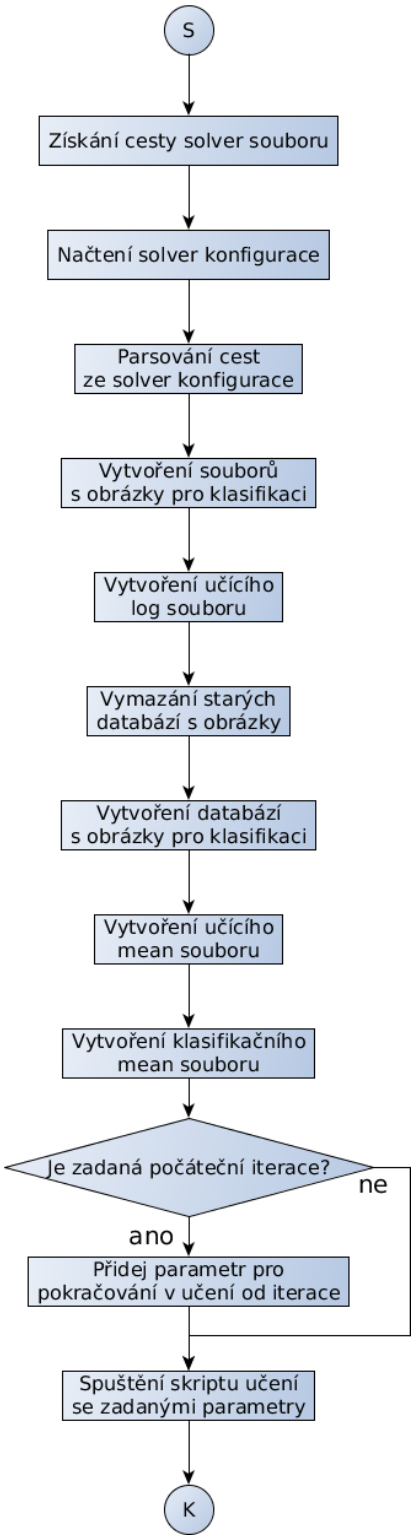
Pro učení klasifikátoru jsou předány tyto parametry:

- `train` - udává typ akce, kterou chceme s Caffe frameworkem provádět. V tomto případě je to trénování klasifikátoru.
- `-solver` - cesta k souboru se *solver konfigurací*
- `-snapshot` - volitelný parametr, který očekává cestu k souboru s uloženým stavem učení klasifikátoru. Tento parametr doplníme pouze v případě jestli bylo zvoleno pokračování učení od určité iterace.

Následně je s těmito parametry spuštěn proces jehož PID si uložíme a předáme jej *Daemonovi*, který si jej uloží do souboru s aktuálně spuštěným učením.

5.9 Modely

V rámci trénování klasifikátoru bylo vyzkoušeno několik modelů neuronových sítí, které se liší svojí topologií a byly vyzkoušeny také různé variace těchto modelů, kdy byly pouze pozměněné parametry modelu, které by mohly ovlivnit trénování klasifikátoru. Celkem takto vzniklo v systému devět konfigurací modelů z nichž



Obrázek 26: Postup učení klasifikátoru

některé se podařilo naučit a jiné modely se ani po opakovaných pokusech naučit nepodařilo.

V této sekci se budu zmiňovat o takzvaných *features*, což jsou předzpracované data z fotografií, které v modelech vstupují do neuronové sítě. Tyto data jsou předzpracovány pomocí konvolučních a pool [19] vrstev v modelu a díky tomuto předzpracování dosahují modely vyšších přesností než když se učí přímo nad surovými vstupními daty [40]. *Features* představují reálný vstup do neuronové sítě, která je definovaná konfiguraci modelů.

5.9.1 AlexNet

Model vznikl na základě odborného článku [26] a je určen pro zpracování datasetu LSVRC 2010 [41]. Schéma modelu je zobrazeno v příloze B.

Červeně jsou označeny jednotlivé konvoluční vrstvy, zeleně jsou pak zobrazeny ReLU vrstvy, které znázorňují Rectified Linear Unit [31] neboli vrstva s aktivační funkcí a zeleně jsou pak také znázorněny *Dropout* vrstvy [46], které snižují efekt přeučení neuronové sítě.

Celkem obsahuje model osm vrstev, které upravují vstupní fotografii na naučené *features*, které se pak stanou vstupem klasické Feed-Forward Neural Network [5], která se skládá z tří vrstev, kde propojení těchto vrstev neuronové sítě je znázorněno pomocí fialových *InnerProduct* vrstev, které provádějí skalární součin vektorů [19].

5.9.2 CaffeNet

Referenční CaffeNet model, který vychází z AlexNet modelu a obsahuje některé úpravy navíc. Má stejný počet vrstev, ale liší se v uspořádání činností, které se provádějí mezi jednotlivými konvolučními vrstvami. Je určen pro zpracování datasetu ILSVRC 2012 [41]. V příloze A je zobrazen model této neuronové sítě.

V prvním případě je vidět, že než se výsledek první konvoluční vrstvy uloží do oranžové POOL vrstvy [19], tak dojde k normalizaci hodnot pomocí Local Response Normalization [19]. Takto je to i u druhé konvoluční vrstvy. Následující vrstvy modelu AlexNet jsou již totožné s referenčním modelem CaffeNet.

Jelikož se tento model vyvinul z modelu AlexNet [26], tak se očekává, že bude dosahovat lepších přesností než model AlexNet.

5.9.3 CaffeNet MF

Na základě úspěchů modelu CaffeNet jsem se rozhodl upravit parametry modelu a získat tak lepší přesnosti na pornografických fotografiích. Celý název tohoto modelu je CaffeNet More Features a jak lze z názvu odvodit, byly upraveny parametry tak, aby se model mohl naučit více *featur*. Tento model vznikl z úvahy, že databáze fotografií obsahuje velké množství závadných fotografií, které jsou velmi rozdílné a 96 *featur*, které jsou výchozí hodnotou v první konvoluční vrstvě modelu Caffe nemusí

stačit a proto jejich počet byl zvýšen o jednu třetinu tedy celkem na 128 *featur*. Definice modelu je stejná jako v případě CaffeNet jen se liší hodnoty parametrů.

5.9.4 CaffeNet LF

Tento model také vychází z modelu CaffeNet, ale zde jsem aplikoval jinou myšlenku a to, že na fotografiích hledáme jen malý počet vzorů, které rozhodují jestli daná fotografie je pornografická. Především se jedná o pohlavní orgány. Snížením počtu *featur*, které projdou první konvolucí by mohlo přispět k tomu, že se neuronová síť naučí vyhledávat skutečně tyto vzory a nebude ovlivněna jinými vzory, které mohou mít fotografie společné. Například nějaké pozadí fotografie. Definice modelu je stejná jako v případě CaffeNet jsou pozměněny hodnoty parametrů.

5.9.5 SeznamNet

Model SeznamNet je mnou vytvořený model, který myšlenkově vycházel z modelu CaffeNet. Myšlenka tohoto modelu vycházela z předpokladu, že model CaffeNet byl vytvořen pro dataset ILSVRC 2012 [41], který obsahuje celkem 1 000 různých kategorií fotografií a to co se na daných fotografiích hledá je velmi rozdílné.

V modelu SeznamNet bylo použito méně konvolučních vrstev, které provádějí předzpracování fotografie a byly upraveny parametry včetně počtu neuronů v neuronové síti, který byl snížen z 4 096 na 256 a model má otestovat jestli je možné pracovat s jednodušším modelem neuronové sítě, který má klasifikovat fotografie pouze do 2 kategorií. Definice modelu je zobrazena v příloze C.

5.9.6 SeznamNet LD

Modely byly průběžně testovány a bylo zjištěno, že model SeznamNet není možné z daných fotografií naučit. Po prozkoumání modelu jsem zjistil, že byl sice snížen počet neuronů v neuronové síti, ale nebyl změněn parametr funkce *Dropout* [19], který zajišťuje aby v neuronové síti nedošlo k přeučení. Tento parametr je pro daný počet neuronů vysoký a může způsobit, že to co se neuronová síť naučí bude zničeno efektem *Dropout* funkce. Definice modelu je stejná jako v případě SeznamNet pouze se liší v parametrech. Model je zobrazen v příloze D.

5.9.7 SAN

Tento model vychází z modelu AlexNet, který byl zjednodušen pomocí odebrání konvolučních vrstev a bylo testováno, jestli si model poradí s trénovacími daty. Definice modelu je zobrazena v příloze D. Úpravy, které byly provedené byly velmi podobné s úpravami v modelu SeznamNet. Byly odstraněny konvoluční vrstvy a došlo ke zmenšení počtu neuronů v neuronové síti.

5.9.8 Cifar10

Model Cifar10 je určen pro dataset CIFAR-10 [25], který obsahuje fotografie 32x32 pixelů a rozlišuje tyto fotografie do deseti tříd. Jelikož tento model nelze v základu využít, protože fotografie 32x32 pixelů jsou příliš malé pro hledání pornografických rysů ve fotografii. Proto byl výchozí model upraven tak, aby mohl přijímat fotografie o velikosti 256x256 pixelů, jako ostatní modely. Definice modelu je zobrazena v příloze E.

5.9.9 SeznamCifar10

Po natrénování modelu Cifar10 jsem se snažil vylepšit výslednou přesnost klasifikace. Jelikož u modelu Caffenet MF došlo k zlepšení výsledků pokud bylo klasifikátoru nastaven větší počet *featur*, které z konvoluční vrstvy vystupují, tak jsem stejné úpravy provedl i u modelu SeznamCifar10, kde byl počet *featur* zdvojnásoben z 32 na 64. Definice modelu je stejná s Cifar10 a je zobrazena v příloze E.

6 Testování

Jednou z podmínek společnosti Seznam.cz bylo, aby bylo možné fotografii klasifikovat v reálném čase. Proto byl proveden test doby klasifikace fotografií. Nutno podotknout, že dosažené časy jsou závislé na konkrétním hardware na kterém byly spuštěny. V mém případě se jedná o sestavu s procesorem i5-3570K a grafickou kartou GeForce GTX 660. Jedná se o grafickou kartu, která se nachází v běžných domácnostech a dosahuje výkonu 1 981 GFLOPS. Pro průmyslové nasazení se používají grafické karty s mnohonásobně vyšším výkonem, takže lze předpokládat, že výsledné časy klasifikací budou ještě nižší. Časy uvedené v tabulce 2 jsou průměrné časy z padesáti měření, které byly prováděny na modelu CaffeNet MF.

Počet fotografií	Doba klasifikace [s]
1	0,071
2	0,136
5	0,363
10	0,718
20	1,355

Tabulka 2: Časy pro klasifikaci fotografie

Celkově bylo k dispozici 164 397 fotografií. Z toho 95 166 jsou závadné fotografie a zbylých 69 231 jsou nezávadné fotografie. Tyto fotografie byly rozděleny do tří sekcí jak je znázorněno v tabulce 3.

	nezávadné	závadné
učení	52 600	71 767
validace	13 808	20 579
testování	2 823	2 820

Tabulka 3: Rozdělení fotografií

Každý z odzkoušených modelů byl naučen a testován na stejných fotografiích s totožným rozdělením fotografií. Fotografie byly zpracovávány barevně, tedy se všemi třemi kanály a byly sjednocené na velikosti 256x256 pixelů. Byly prováděny pokusy s fotografiemi, které měly rozměry 128x128 pixelů na modelech CaffeNet a AlexNet, ale výsledné klasifikátory dosahovaly nižších přesností. Proto všechny klasifikátory byly již testovány pouze na velikosti fotografií 256x256 pixelů. Modely byly testovány na fotografiích, které byly v testovací sekci a celkem se jednalo o 5 643 fotografií.

Nejlepší přesnost naučených klasifikátorů, která bude uvedena u jednotlivých modelů je vypočtena jako součet *true positive* a *true negative* klasifikací podělená celkový, počtem fotografií v testovací sekci z použité databáze fotografií.

Grafy, které jsou obsaženy v této kapitole obsahují dvě křivky. První z nich je *accuracy* křivka, která bude v legendě zobrazena jako *Acc* a znázorňuje přesnost

naučeného klasifikátoru na trénovacích datech. Jedná se o procentuální vyjádření přesnosti v rozsahu hodnot $\langle 0, 1\rangle$. Druhou křivkou je takzvaná *loss* křivka, která udává chybu klasifikátoru na validačních datech. Rozsah této křivky není stanoven a může se pohybovat v kladných reálných hodnotách. Jelikož nejvyšší chyba, která byla na modelech naměřena byla 0,7, tak v rámci úspory místa a zvýšení přehlednosti budou tyto křivky vždy vloženy do jednoho grafu, aby bylo umožněno pozorování závislostí křivek *accuracy* a *loss*.

Loss křivky, které jsou grafech zobrazeny nemají typický průběh, protože modely neuronových sítí obsahují vrstvu DROP, která zajišťuje, že určité vazby mezi neurony budou zrušeny čímž se zabraňuje přeučení neuronové sítě. Díky tomu není u *loss* křivek pozorovatelný vzestup. Zvýšení *loss* chyby na validačních datech lze pozorovat až při větším počtu iterací. Například na obrázku 28 již lze pozorovat, že od 100 000 iterace nastává přeučení modelu.

Accuracy i *loss* křivky se mohou jevit jako velmi nekonzistentní, protože se neustále zvyšují nebo snižují. To může být obecně způsobeno velkou učící konstantou, ale v tomto případě je to způsobeno tím, jak framework Caffe provádí učení a validaci. V *train konfiguraci* se nastavuje, kolik fotografií se má načíst z LMDB databáze pro jednu iteraci učení a jednu iteraci validace. Pro představu tyto hodnoty jsou v modelech nastaveny na hodnotu 64 pro učení a 50 pro validaci. Větší hodnoty nešlo nastavit z důvodu omezené grafické paměti, kterou mám k dispozici. Při jednotlivých iteracích se z LMDB databáze postupně čte zadaný počet fotografií a ten se zpracuje. Může se tedy stát, že se v konkrétní testované iteraci zpracovává větší množství fotografií, které klasifikátor umí správně klasifikovat a jindy se může stát přesný opak, že se načtou fotografie, které klasifikátor ještě správně neklasifikuje a tímto je způsobeno ono zakřivení křivek v grafu. Testování klasifikátoru je nastaveno tak, že se testuje každých 1 000 naučených iterací.

6.1 CaffeNet

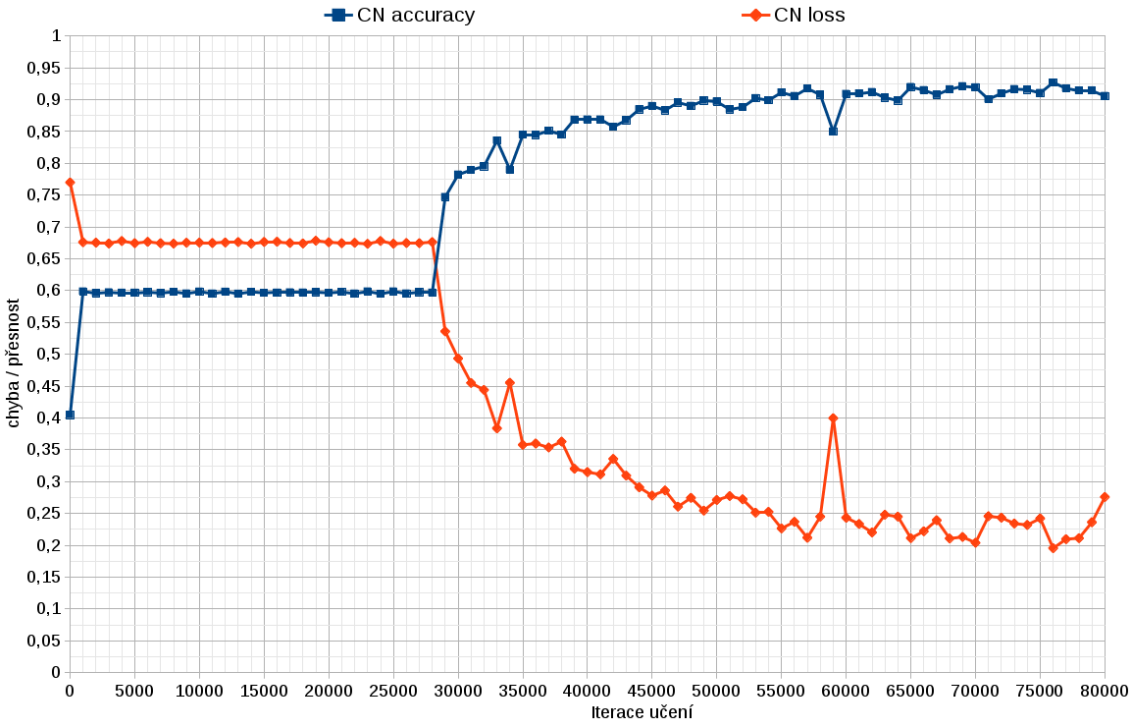
Testování na modelu CaffeNet přineslo zajímavé výsledky. Na obrázku 27 je zobrazena učící křivka tohoto modelu. Zajímavá je počáteční fáze učení, kdy modelu trvalo 28 000 iterací než si nastavil vnitřní váhy tak, aby docházelo ke zlepšení výsledků učení. Právě do této 28 000 iterace model přiřazoval všechny fotografie do kategorie závadné. Předpokládal jsem, že se jedná o nějakou interní chybu frameworku Caffe, protože tento model byl trénován již dříve, ale tento jev nebyl pozorován, ale z výsledků ostatních modelů, které byly založeny na modelu CaffeNet, vycházely podobné výsledky jen iterace, kdy se učení začalo projevovat se měnila. U ostatních modelů se tento jev neprojevil, takže nelze vyvodit, že by se jednalo o chybu frameworku Caffe.

Na tomto modelu bylo také prováděno učení klasifikátoru v rámci prvotního testování použitelnosti frameworku Caffe. Učení klasifikátoru probíhalo až do 330 000 iterace a pozoruhodné je, že i když se jedná o stejný model, tak nebyl v prvotním učení zpozorován problém se začátkem učení. Mezi prvotním a finálním učením

uběhlo několik měsíců a v této době byl aktualizován framework Caffe a byla aktualizována syntaxe zápisu konfigurace modelu. Předpokládal jsem, že některá z těchto změn mohla ovlivnit výsledek učení, ale jelikož výsledky ostatních modelů dokazují, že se klasifikátor korektně učí již od prvních iterací, tak tato teorie nebyla prokázána.

Z naučených 80 000 iterací dosáhla nejlepší výsledků iterace 76 000, která dosáhla přesnosti 90,86% na testovaných datech. Matice záměn pro danou iteraci je zobrazena v tabulce 4. Celkem bylo správně klasifikováno 5 127 fotografií a špatně klasifikováno bylo 516 fotografií. Z tabulky zobrazující matici záměn lze vyčíst, že je splněna podmínka, kdy počet *false positive* vyhodnocení je větší než počet *false negative* vyhodnocení fotografií.

Průměrná hodnota přiřazení do správné kategorie je 0,9457 a průměrná hodnota přiřazení do špatné kategorie je 0,7829. Z tohoto lze vyvodit, že lze snížit počet *false positive* a *false negative* klasifikovaných fotografií, pokud se nastaví práh například na hodnotu 0,85. Tak všechny klasifikace, které budou mít hodnotu přiřazení do kategorie nižší než 0,85, tak budou automaticky odeslány do fronty pro potvrzení administrátorem.



Obrázek 27: Učící křivka modelu CaffeNet

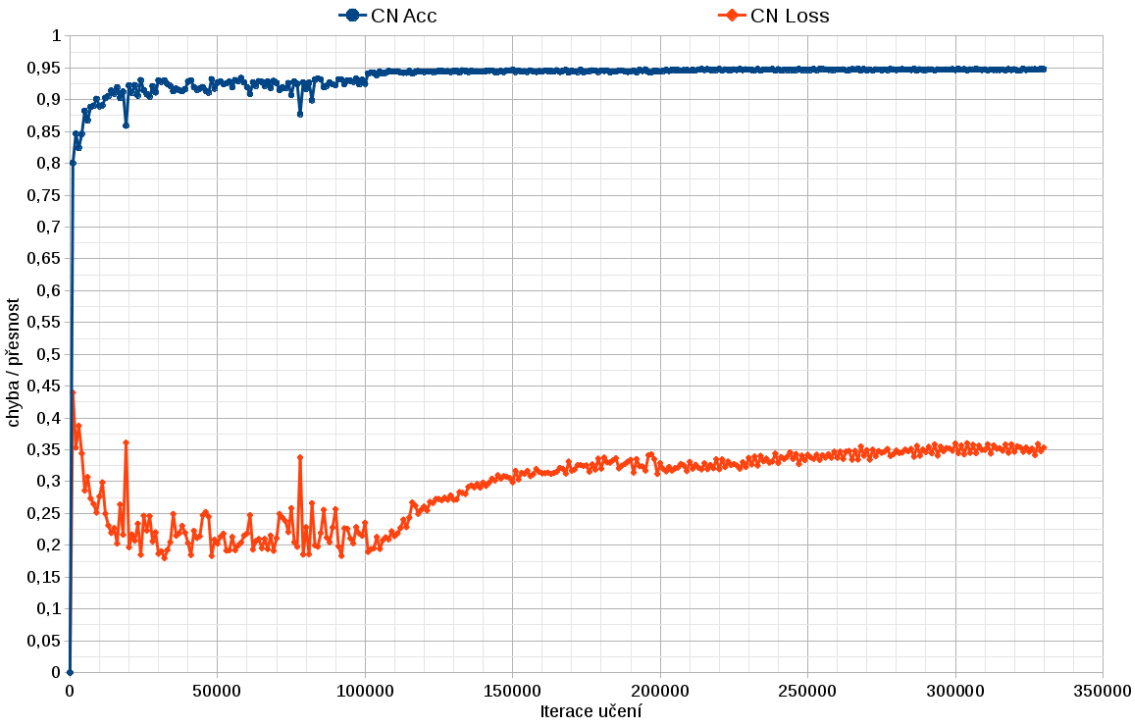
Na obrázku 28 je zobrazen graf prvotního učení modelu až do 330 000 iterace, kde lze pozorovat, že od 100 000 iterace již dochází k přeučení modelu, protože se zvyšuje chyba na validačních datech. Ale i přes výrazné přeučení na validačních datech vyšla jako nejlepší iterace tohoto klasifikátoru iterace 300 000, kdy klasifikátor dosáhl přesnosti 92,98%. Správně klasifikoval 5 247 fotografií a špatně klasi-

	nezávadné	závadné
nezávadné	2 490	333
závadné	183	2 637

Tabulka 4: Nejlepší matice záměn modelu CaffeNet

fikoval 396 fotografií. Matice záměn pro tento klasifikátor je zobrazena v tabulce 5. Je vidět, že tento klasifikátor splňuje podmínku a počet závadných fotografií, které byly označeny za nezávadné je nižší než v opačném případě. Tento klasifikátor ovšem nelze použít, protože je přeučený a neobsahuje potřebnou abstrakci pro správné klasifikování fotografií.

Průměrná hodnota příslušnosti správně klasifikovaných fotografií je 0,9942 a průměrná hodnota příslušnosti špatně klasifikovaných fotografií je 0,9308. Přesnost klasifikátoru je sice vyšší, ale v tomto případě již nelze rozumně využít specifického prahu pro snížení počtu *false postive* a *false negative* klasifikací, protože rozdíly mezi průměrnými hodnotami jsou velmi malé.



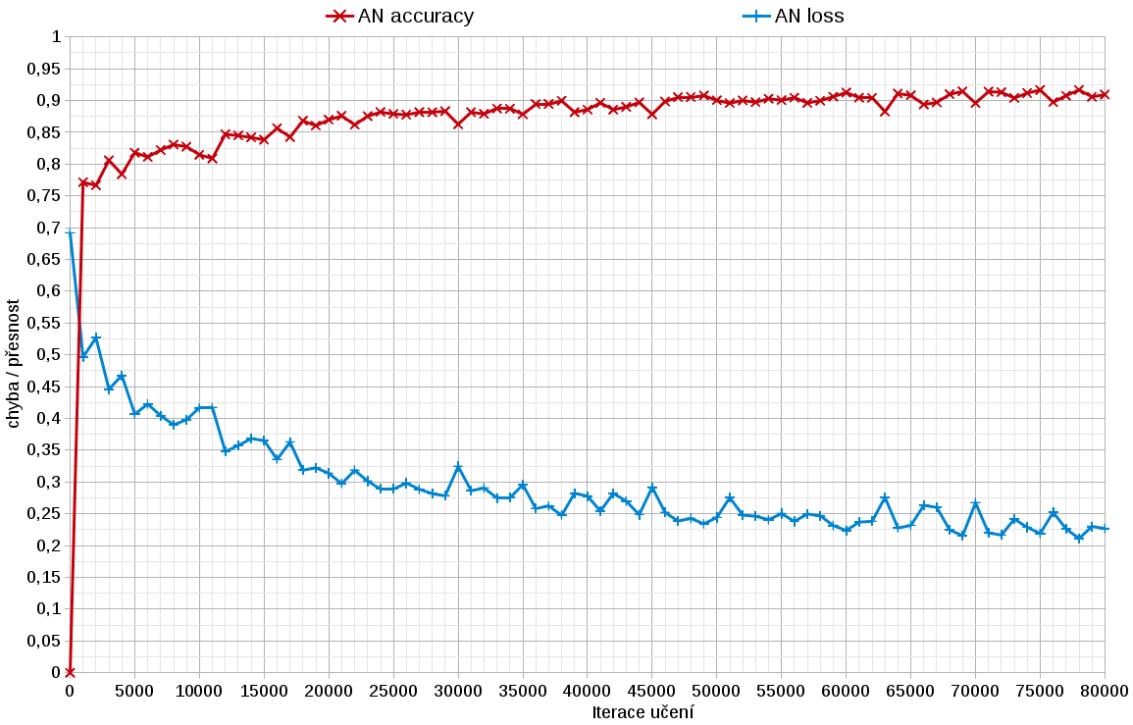
Obrázek 28: Učící křivka modelu CaffeNet při prvotním učení

	nezávadné	závadné
nezávadné	2 597	226
závadné	170	2 650

Tabulka 5: Nejlepší matice záměn modelu CaffeNet při původním učení

6.2 AlexNet

Model AlexNet má již klasičtější tvar křivky pro učení. Tento model na testovaném rozsahu 80 000 iterací dosahoval podobných výsledků jako model CaffeNet. Na obrázku 29 je zobrazena učící křivka modelu. Z této křivky se jeví jako nejlépe naučená iterace číslo 78 000, protože má nejmenší chybu na validačních datech, ale v rámci testovacích dat vyšla jako nejlepší naučená iterace klasifikátoru iterace 72 000. Matice záměn pro tuto iteraci je zobrazena v tabulce 6. Přesnost iterace 72 000 na testovacích datech činí 89,56%. Bylo správně klasifikováno 5 054 fotografií a špatně klasifikováno bylo 589 fotografií. Z matice záměn lze také vyčíst, že tato naučená iterace nesplňuje podmínku společnosti Seznam.cz a více závadných fotografií je označen jako nezávadné. To znamená, že se zvyšuje pravděpodobnost zobrazení závadných fotek na službě Lidé.cz, protože budou klasifikovány jako nezávadné. Možné zlepšení klasifikátoru může nastat pokud budeme pokračovat v učení.



Obrázek 29: Učící křivka modelu AlexNet

Rozdíl v průměrných hodnotách přiřazení do kategorií je podobný jako u modelu CaffeNet jen jsou dané hodnoty nižší. Konkrétně to je 0,9245 při zařazení do správné kategorie a 0,7527 při zařazení do špatné kategorie. Optimální práh, který by snížil počet špatně klasifikovaných fotografií, by se v tomto případě měl nastavit na hodnotu 0,83. Fotografie, které nesplňují tuto hodnotu přiřazení budou zařazeny do fronty pro potvrzení administrátorem.

	nezávadné	závadné
nezávadné	2 552	271
závadné	318	2 502

Tabulka 6: Nejlepší matice záměn modelu AlexNet

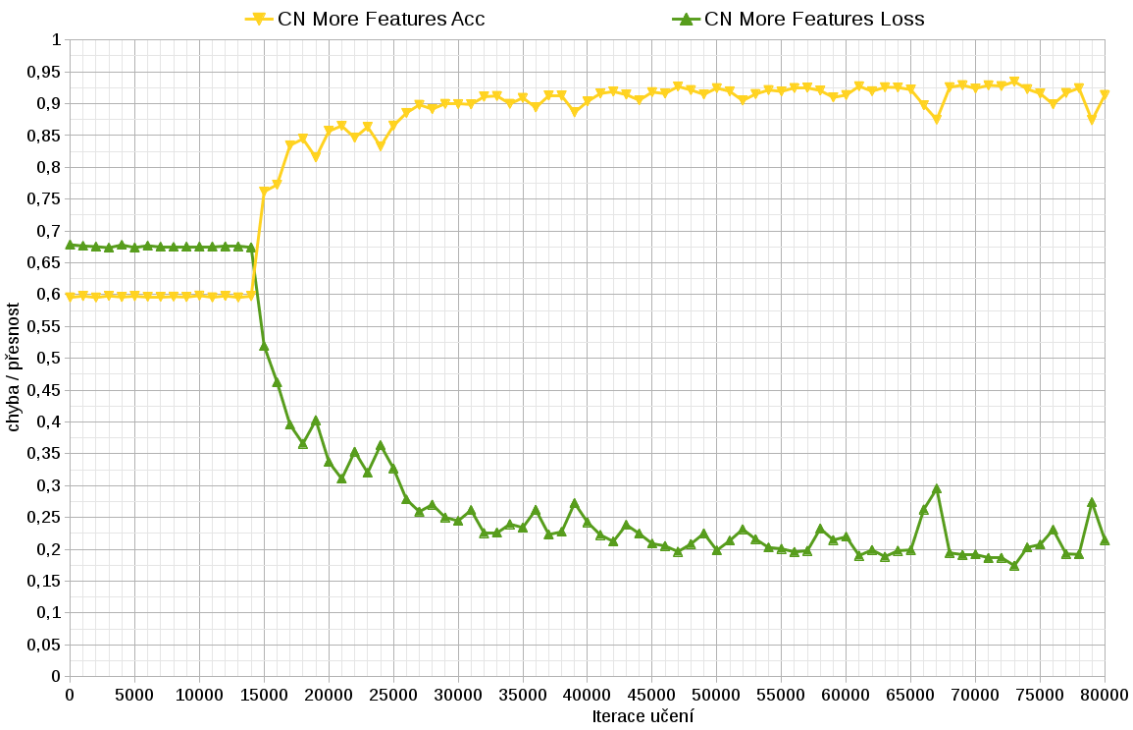
6.3 CaffeNet MF

Model CaffeNet More Features vykazoval podobné vlastnosti jako jeho výchozí model CaffeNet, ale jak si lze všimnout z učící křivky, která je zobrazena na obrázku 30, tak se snížil počet iterací, kdy se model nic nenaučil a stále pouze klasifikoval všechny fotografie jako závadné. Nyní již od 15 000 iterace došlo k výraznému zlepšení klasifikátoru, kdy z přesnosti 60% se dostal na přesnost 76%. Poté docházelo k dalšímu zlepšování klasifikátoru. Jako nejlepší iterace klasifikátoru se z grafu zdá být iterace číslo 73 000, která dosahuje nejnižší validační chyby a zároveň nejvyšší přesnosti na trénovacích datech.

Po provedení testu na testovacích fotografiích se skutečně iterace 73 000 projevila jako nejlepší a dosáhla přesnosti 91,56%, což je o 0,7% lepší výsledek než jakého dosáhl nejlepší naučený klasifikátor modelu CaffeNet. Správně bylo klasifikováno 5 167 fotografií a špatně bylo klasifikováno 476 fotografií. Matice záměn naučeného klasifikátoru z 73 000 iterace je zobrazena v tabulce 7. V tabulce je vidět, že sice se zlepšila přesnost klasifikátoru, ale klasifikátor nyní více závadných fotografií považuje za nezávadné, což nesplňuje podmínku společnosti Seznam.cz, která požadovala, aby *false negative* byl nižší než *false positive*. Toto se může zlepšit při pokračování v učení daného modelu.

Průměrná hodnota přiřazení do správné kategorie činí 0,9431 a průměrná hodnota přiřazení do chybné kategorie činí 0,7652. Opět je zde možnost využití určité hodnoty prahu, kdy by fotky s nedostatečným přiřazením do kategorie byly odeslány do fronty pro potvrzení administrátorem. Doporučená hodnota prahu je 0,89. Hodnota prahu může být vyšší než u modelu CaffeNet a to díky většímu rozdílu průměrných hodnot mezi správným a špatným zařazením.

Jelikož se tento model prokázal v prvních 80 000 iteracích jako nejpřesnější, tak jsem se rozhodl pokračovat v učení tohoto modelu od 70 000 iterace a snížil jsem velikost parametru *lr_rate* na polovinu, abych dosáhl lepších výsledků klasifikátoru. Učící křivka je zobrazena na obrázku 31. Opět je z křivky patrné, že od 100 000



Obrázek 30: Učící křivka modelu CaffeNet MF pro 80 000 iterací

	nezávadné	závadné
nezávadné	2 598	225
závadné	251	2 569

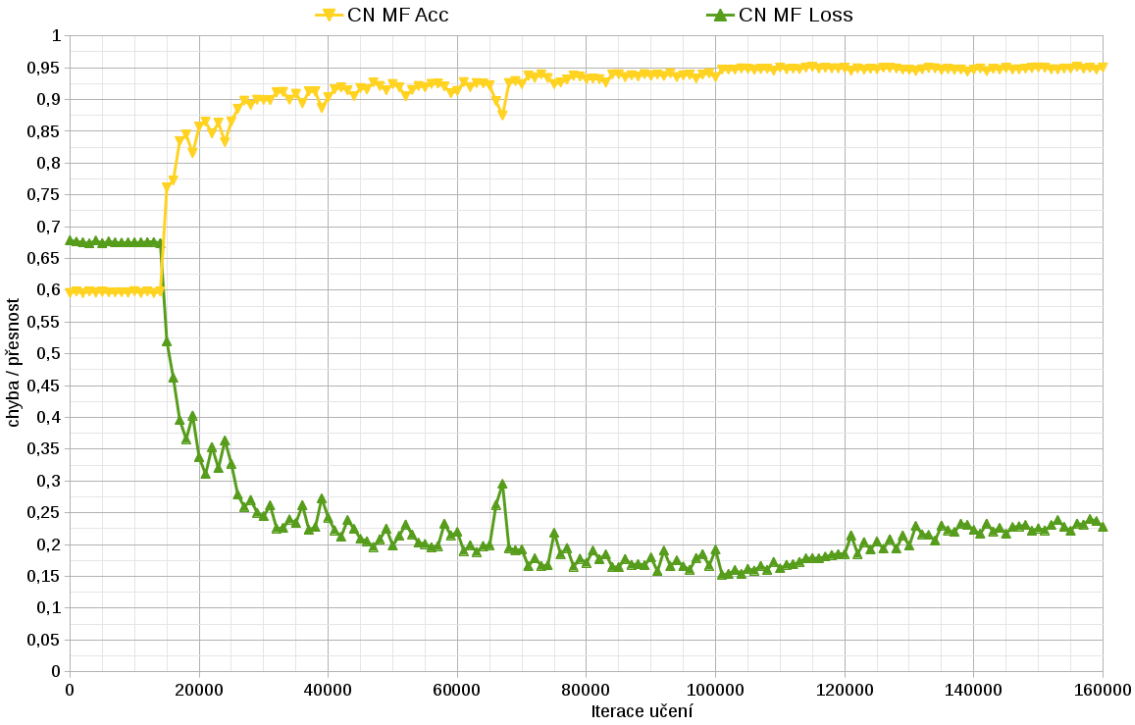
Tabulka 7: Nejlepší matice záměn modelu CaffeNet MF pro 80 000 iterací

iterace dochází k přeučení modelu, které ale není tak výrazné jako v případě modelu CaffeNet, protože doučení tohoto modelu probíhalo s nižším parametrem *lr_rate*.

Nejlepšího výsledku na testovacích datech dosahovala 153 000 iterace klasifikátoru, kde přesnost dosahovala 94,22%. Špatně klasifikováno bylo 326 fotografií a průměrná hodnoty přiřazení do správné kategorie vyšla 0,991 a průměrná hodnota přiřazení do špatné kategorie byla rovna 0,9127. Při této iteraci, ale již byl klasifikátor přeučený a proto nakonec jako nejlepší iterace klasifikátoru byla zvolena iterace 107 000. V této iteraci klasifikátor dosáhl přesnosti 93,64%, kdy špatně klasifikoval 359 z celkového počtu 5 643 fotografií. Matice záměn této iterace je zobrazena v tabulce 8.

Průměrná hodnota přiřazení do správné kategorie vyšla 0,9815 a průměrná hodnota přiřazení do špatné kategorie vychází 0,8475. Díky tomuto rozdílu, lze případně zavést práh s konkrétní hodnotou 0,92, který sníží počet *false negative* a *false positive* klasifikací. Z tabulky 8 vidíme, že klasifikátor splňuje podmínky kladené společností Seznam.cz a jelikož ze všech testovaných klasifikátorů dosáhl nejvyšší přesnosti, tak

je navržen jako klasifikátor vhodný pro využití v modulu pro zpracování nevhodných obrázků.



Obrázek 31: Učící křivka modelu CaffeNet MF pro 160 000 iterací

	nezávadné	závadné
nezávadné	2 614	209
závadné	150	2 670

Tabulka 8: Nejlepší matice záměn modelu CaffeNet MF pro 160 000 iterací

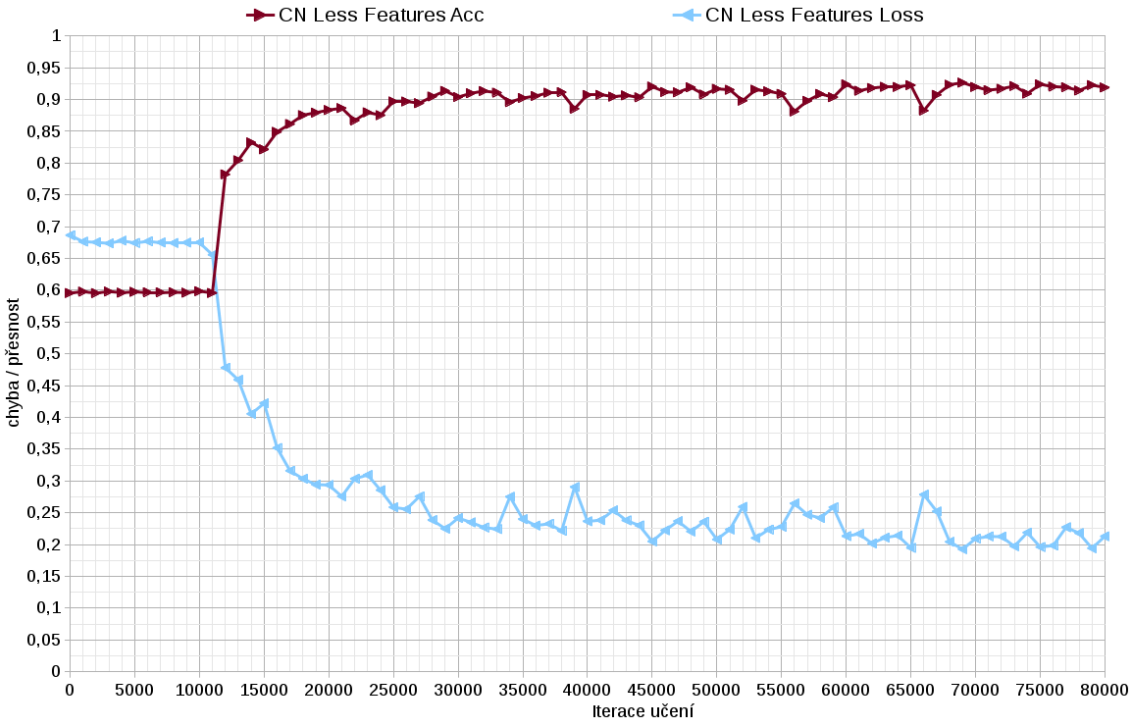
6.4 CaffeNet LF

Upravený model Caffenet, kterému byl snížen počet *featur* vykazoval podobné výsledky učení jako výchozí verze modelu Caffenet. Učící křivka modelu je zobrazena na obrázku 32. Zajímavostí je, že počet iterací, kdy se model nic nenaučil byl opět snížený a tentokrát první pokrok v učení byl zaznamenán v 12 000 iteraci, kdy se změnila hodnota přesnosti na trénovacích datech z 60% na 78%. V dalších iteracích se klasifikátor dále zlepšoval.

Byl proveden test nad testovacími fotografiemi a jako nejlepší iterace klasifikátoru vyšla iterace číslo 71 000. Tato iterace dosahovala přesnosti 90,93%. Tato hodnota je o 0,07% lepší než nejlepší klasifikátor modelu CaffeNet, ale zároveň je tato

hodnota o 0,63% horší než varianta modelu CaffeNet, který využívá více *featur* ke klasifikaci fotografií. Hypotéza tohoto modelu se tedy nepotvrdila a bylo zjištěno, že pro lepší klasifikaci není možné snižovat počet *featur*, ale naopak je potřeba *featury* přidat. Počet správně klasifikovaných fotografií je 5 131 a počet špatně klasifikovaných fotografií je 512.

Průměrná hodnota přiřazení do správné kategorie je 0,9378 a průměrná hodnota přiřazení do nesprávné kategorie činí 0,7759. V porovnání s původním CaffeNet modelem došlo pouze ke zvýšení hodnot o 0,01, ale rozsah hodnot je skoro stejný. Pokud ale porovnáme průměrné hodnoty s modelem CaffeNet MF, tak je zde vidět, že došlo k snížení rozdílu průměrných hodnot. Pokud by měl být použit práh pro tento model, tak jeho doporučená hodnota je 0,85.



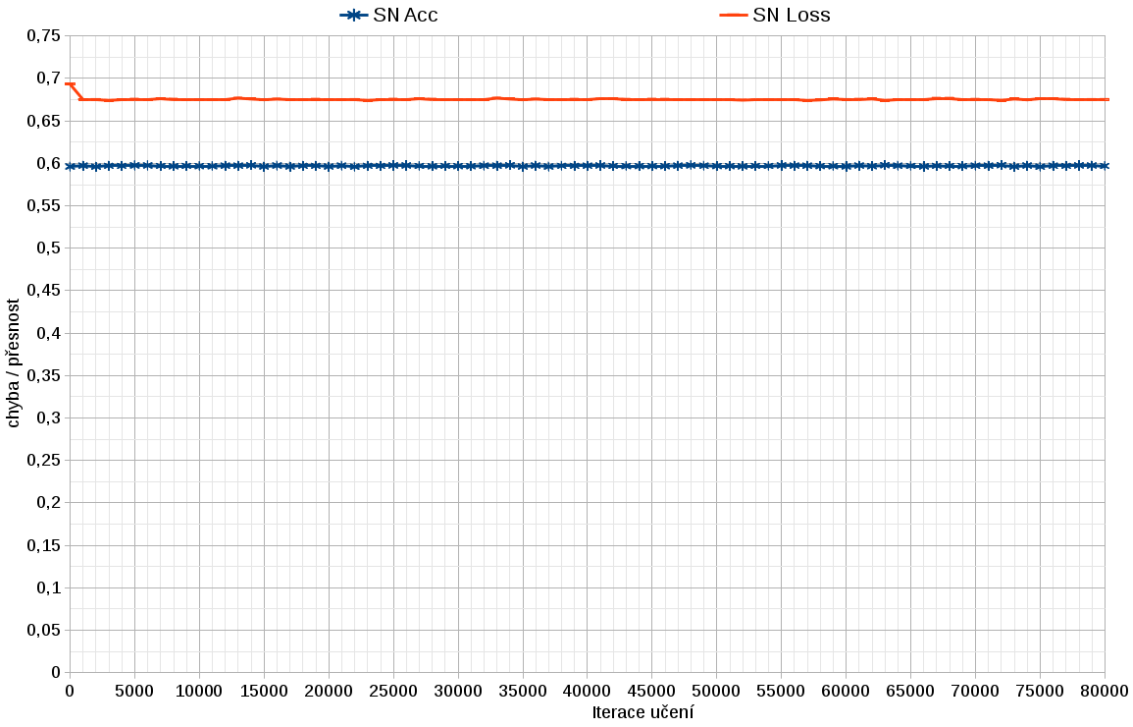
Obrázek 32: Učící křivka modelu CaffeNet LF

	nezávadné	závadné
nezávadné	2 596	227
závadné	285	2 535

Tabulka 9: Nejlepší matice záměn modelu Caffenet LF

6.5 SeznamNet

Tento model se ani po 80 000 iteracích nepodařilo naučit. Není přesně jasné, proč nedošlo k vůbec žádnému učení a model stále všechny fotografie označoval jako závadné. Zjevně odebrání konvolučních vrstev vede k nedostatečnému učení neuronové sítě. Křivka učení je zobrazena na obrázku 33 a matice záměn, která je shodná pro všechny iterace je zobrazena v tabulce 10. Tento mode nebude nadále testován a nemůže být použit jako klasifikátor.



Obrázek 33: Učící křivka modelu SeznamNet

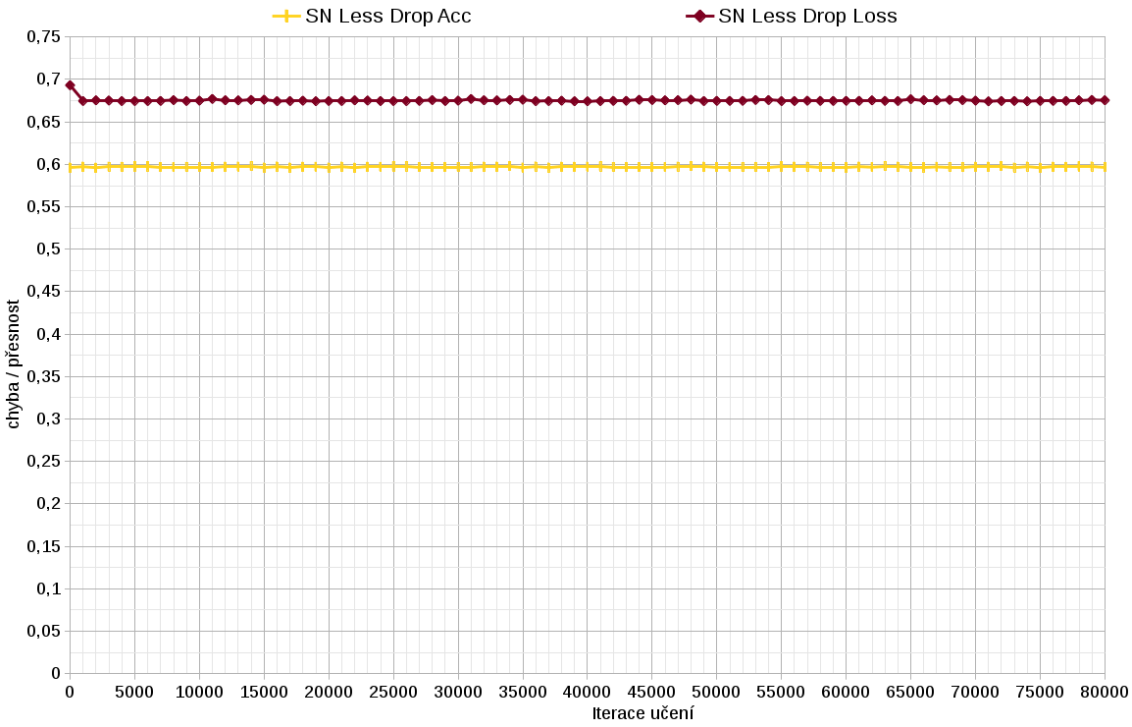
	nezávadné	závadné
nezávadné	0	2 823
závadné	0	2 820

Tabulka 10: Matice záměn modelu SeznamNet

6.6 SeznamNet LD

Ani úprava parametru pro odstraňování naučených neuronů nepomohla a tento model se ani po 80 000 iteracích nepodařilo naučit. Hypotéza, že když byl snížen počet konvolučních vrstev a byly sníženy velikosti neuronů ve vrstvách, tak bude potřeba snížit i *drop rate* parametr, který by bránil k učení neuronové sítě, se jeví jako

chybná. Tento model nebude nadále testován a nemůže být použit jako klasifikátor. Učící křivka modelu je zobrazena na obrázku 34 a matice záměn, která je shodná pro všechny učené iterace je znázorněna v tabulce 11.



Obrázek 34: Učící křivka modelu SeznamNet LD

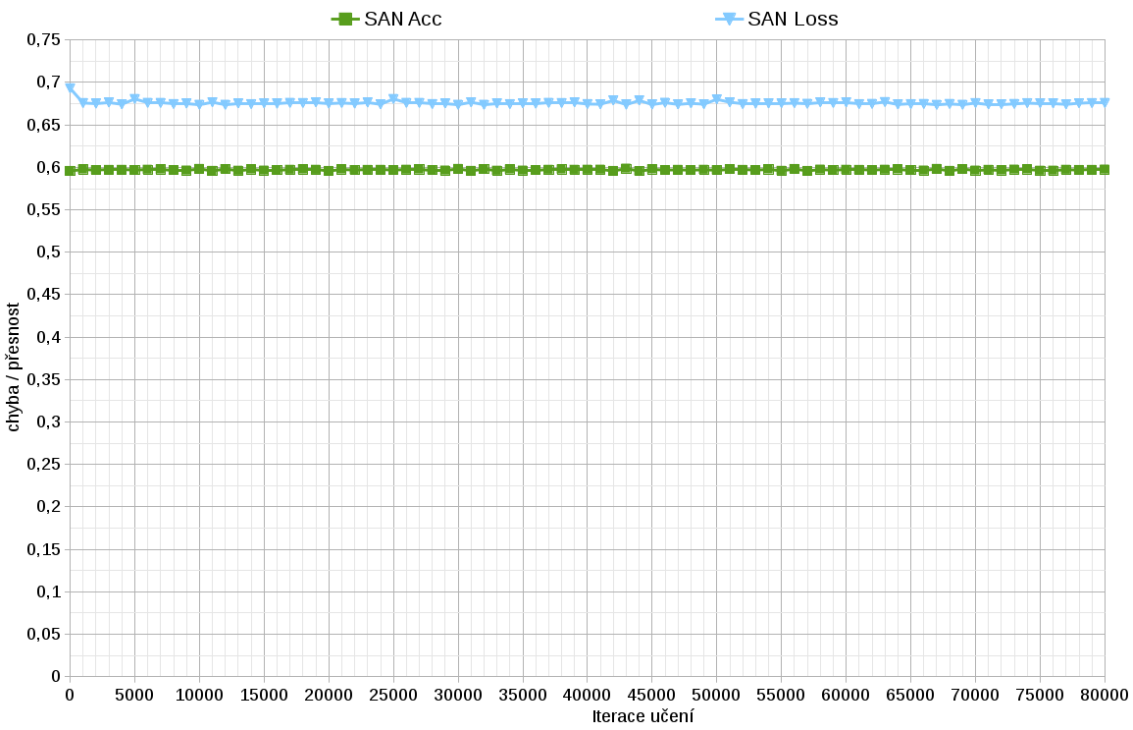
	nezávadné	závadné
nezávadné	0	2 823
závadné	0	2 820

Tabulka 11: Matice záměn modelu SeznamNet LD

6.7 SAN

Další vytvořený model, který vycházel z modelu AlexNet se také nepodařilo natrénovat. Projevoval se zde stejný problém jako u modelu SeznamNet a SeznamNet LD. Všechny fotografie byly označeny automaticky za závadné a nedošlo k žádnému naučení neuronové sítě.

Učící křivka modelu je zobrazena na obrázku 35 a matice záměn, která je shodná pro všechny naučené iterace klasifikátoru je zobrazena v tabulce 35. Tento model nemůže být použit pro klasifikaci na systému Lidé.cz.



Obrázek 35: Učící křivka modelu SAN

	nezávadné	závadné
nezávadné	0	2 823
závadné	0	2 820

Tabulka 12: Matice záměn modelu SAN

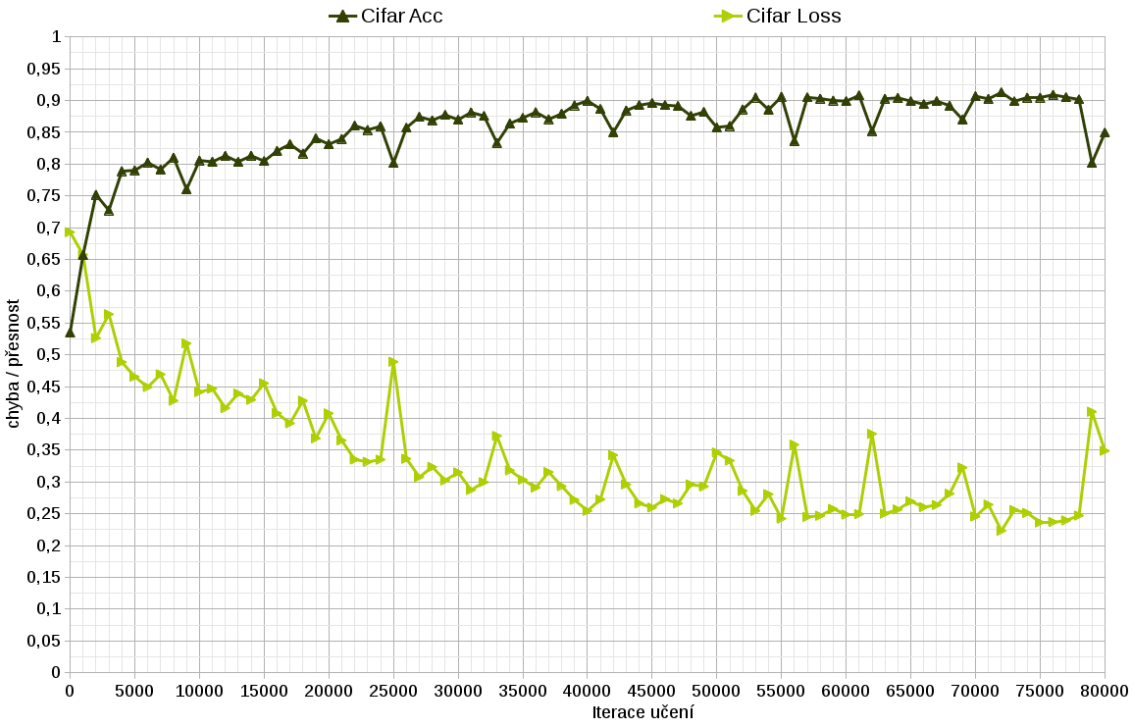
6.8 Cifar10

Již od prvních iterací bylo vidět, že model Cifar10 se úspěšně na databází fotografií učí. Učící křivka je zobrazena na obrázku 36. Již z hodnot v učící křivce lze vyvodit, že model nebude dosahovat lepších výsledků než model CaffeNet MF. Je vidět, že křivka přesnosti na učících datech se zastavila na hranici 90% a dále již neroste. Z grafu se jeví jako nejlepší iterace klasifikátoru iterace číslo 72 000, kde klasifikátor dosahuje nejnižší chyby na validačních fotografiích.

Po provedení testu na testovacích fotografiích, ale vyšla jako nejlepší iterace číslo 63 000, která dosáhla přesnosti 88,61%. Kdy správně klasifikovala 5 000 fotografií a špatně klasifikovala 643 fotografií. Tento výsledek je horší než v případě modelu CaffeNet a obou jeho verzích, které se podařilo natrénovat. Matice záměn pro nejlepší iteraci modelu Cifar10 je zobrazena v tabulce 13.

Rozsah mezi průměrnými hodnotami pro špatnou a dobrou klasifikaci zůstává podobný jako v případě modelu CaffeNet, pouze se aktuální hodnoty snížily. Průměrná hodnota pro klasifikaci do správné kategorie je 0,893 a průměrná hod-

nota pro klasifikaci špatné kategorie je 0,714. Pokud bychom chtěli opět snížit počet *false positive* a *false negative* klasifikací, tak bychom mohli použít práh s hodnotou 0,8, kdy by bylo zajištěno, že fotografie s nižší hodnotu přiřazení do kategorie budou zaslány do fronty pro potvrzení administrátorem. Výsledky tohoto modelu jsou horší v porovnání s modely CaffeNet a proto použití tohoto modelu nedoporučuji.



Obrázek 36: Učící křivka modelu Cifar10

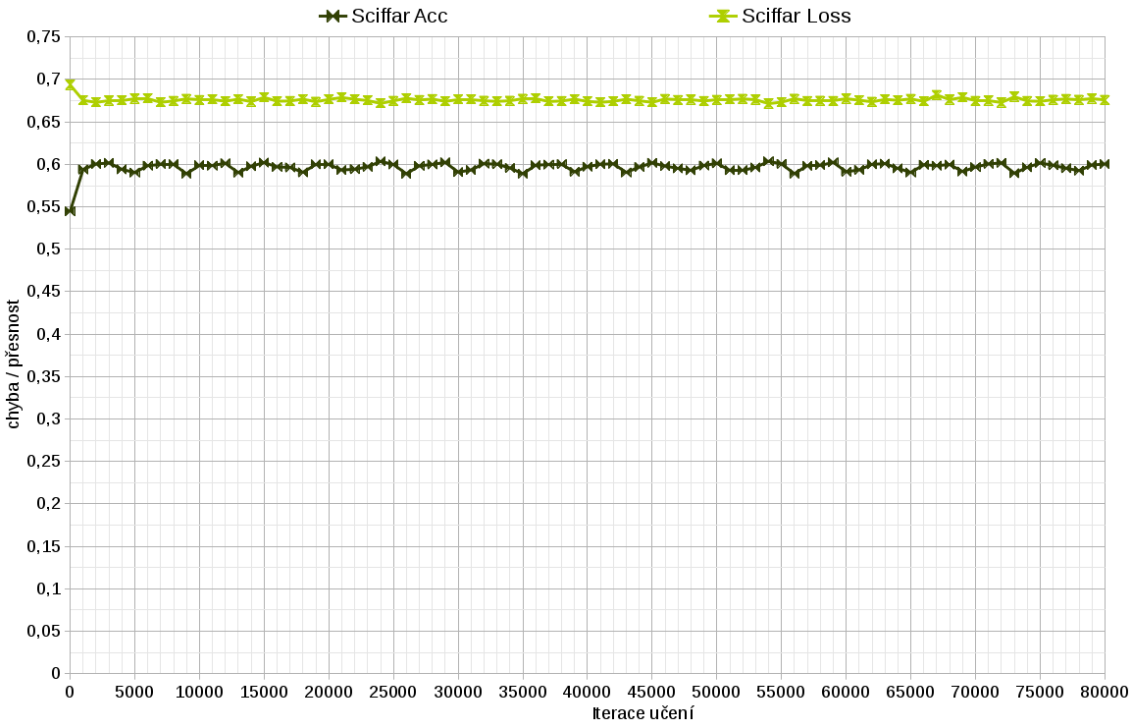
	nezávadné	závadné
nezávadné	2 521	302
závadné	341	2 479

Tabulka 13: Nejlepší matice záměn modelu Cifar10

6.9 SeznamCifar10

Poslední z modelů upravených speciálně pro potřeby služby Lidé.cz. Tento model se také nepodařilo naučit a stále jen klasifikoval veškeré vstupní fotografie jako závadné. Z učící křivky, která je zobrazena na obrázku 37 je vidět jistá snaha neuronové sítě o započatí učení, ale v následující iteraci, kdy přijdou jiné validační fotografie dojde zase k zvýšení chyby na průměrnou hodnotu. Je velmi zajímavé, že jedinou změnou mezi modely SeznamCifar10 a Cifar10 bylo zvětšení počtu *featur* na dvojnásobek.

Bylo očekáváno zlepšení výsledku učení nebo přinejhorším drobné zhoršení učení, ale výsledek byl takový, že model se nenaučil vůbec nic. Matice záměn, která je shodná pro všechny iterace klasifikátoru je zobrazena v tabulce 14.



Obrázek 37: Učící křivka modelu SeznamCifar10

	nezávadné	závadné
nezávadné	0	2 823
závadné	0	2 820

Tabulka 14: Matice záměn modelu SeznamCifar10

6.10 Shrnutí testování

Model CaffeNet MF, který vznikl z referenčního modelu CaffeNet a byl náležitě upraven, aby byl schopen se naučit více *featur*, dosáhl nejlepších výsledků, kdy dosáhl přesnosti na testovaném datasetu 93,64% při 107 000 iteraci učení, která zároveň nebyla ovlivněna přeučením klasifikátoru. Průměrný čas, pro klasifikaci fotografie činí 0,071 sekundy a splňuje všechny podmínky zadané společností Seznam.cz, které jsou popsány v sekci 3.1.4. Učení 160 000 iterací tohoto modelu trvalo na grafické kartě s výkonem 1 981 GFLOPS 44 hodin. Konkrétní konfigurace modelu jsou uvedeny v přílohách F, G a H.

7 Závěr

V rámci diplomové práce byly analyzovány možnosti klasifikace fotografií. Byl srovnán současný stav klasifikace ve společnosti Seznam.cz s možnostmi klasifikace, které se využívají ve světě. Dle studií se ukázaly jako nejvhodnější variantou konvoluční neuronové sítě, které mají při trénování možnost automatického vytvoření *featur*, které jsou následně používány ke klasifikaci.

Byla navržena komunikace modulu pro vyhledávání nevhodných obrázků s okolním systémem, na kterém pracuje služba Lidé.cz. Součástí návrhu řešení bylo vytvoření návrhu nahrávání a nahlášení fotografií. V tomto návrhu bylo počítáno s chybnými výsledky klasifikátoru a byla navržena možnost opravování klasifikátoru s možností jeho následného doučování, které zajistí, že výsledný klasifikátor bude mít s postupem času vyšší přesnost a bude produkovat méně chybných klasifikací. Dále byl vytvořen ER Diagram obsahující potřebné databázové tabulky, které se musí přidat do systému Lidé.cz, aby bylo možné provádět klasifikaci fotografií.

Dle provedených analýz současného stavu klasifikace fotografií byl vybrán jako nejvhodnější kandidát framework Caffe, který splňuje požadavky společnosti Seznam.cz a zároveň podporuje trénování konvolučních neuronových sítí. Navržené řešení bylo úspěšně implementováno nad frameworkem Caffe a byly provedené testy na devíti modelech neuronové sítě, kdy se dle požadavků společnosti Seznam.cz hledal model, který bude mít přesnost klasifikace větší než 90%, bude schopen provést klasifikaci fotografie s časem nižším než 1 sekunda a bude splňovat požadavek, že počet *false negative* klasifikací bude nižší než počet *false positive* klasifikací.

Testování modelů neuronových sítí proběhlo na sadě 5 643 fotografií, což činí přibližně 80% denního provozu na Lidé.cz. V rámci testování byl nalezen model, který splňoval všechny požadavky společnosti Seznam.cz. Jedná se o model s názvem CaffeNet MF, který dosáhl přesnosti 93,64%, kdy čas pro klasifikaci jedné fotografie činí průměrně 0,071 sekundy a zároveň počet pornografických fotografií, které jsou označeny za nezávadné, je nižší než v opačném případě.

Tento model je možné použít pro klasifikaci fotografií na seznamovací službě Lidé.cz. Pro otestování implementovaného modulu v ostrém provozu, bude modul použit pouze v části systému, která zajišťuje zobrazování právě nahraných fotografií na službu Lidé.cz. Klasifikátor v takovém případě bude pouze rozhodovat o tom, které fotografie budou uživatelům zobrazeny a které nikoliv. Pokud bude klasifikátor dosahovat vysoké spolehlivosti nasadí se i na jiné části systému Lidé.cz.

Literatura

[1] Amazon.com: Amazon Mechanical Turk. 2005-2015.
Dostupné na https://requester.mturk.com/

[2] Ap-Apid, R.: An algorithm for nudity detection. In *5th Philippine Computing Science Congress*, 2005, ISSN 1908-1146, s. 201–205.
Dostupné na http://onebit.us/x/i/814381733331796005.pdf

[3] Ascher, D.; Dubois, P. F.; Hinsen, K.; aj.: Numerical python. 2001.

[4] Barto, A. G.: *Reinforcement learning: An introduction*. MIT press, 1998, ISBN 978-0-2621-9398-6.
Dostupné na http://150.185.222.161/~dfinol/NeuroCienciaCognitiva/ReinforcemenLearmning%20BOOK%20-%20F119696Bd01.pdf

[5] Bebis, G.; Georgiopoulos, M.: Feed-forward neural networks. *Potentials, IEEE*, ročník 13, č. 4, Oct 1994: s. 27–31, ISSN 0278-6648, doi:10.1109/45.329294.

[6] Bengio, Y.; LeCun, Y.; Henderson, D.: Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden Markov models. *Advances in Neural Information Processing Systems*, 1994: s. 937–937.
Dostupné na http://yann.lecun.com/exdb/publis/pdf/bengio-lecun-henderson-94.pdf

[7] Bradski, G.; Kaehler, A.: *Learning OpenCV: Computer vision with the OpenCV library*. "O'Reilly Media, Inc.", 2008, ISBN 1-4493-1465-1.

[8] Chu, H.: The Lightning Memory-Mapped Database. *Retrieved October*, ročník 12, 2013: str. 2013.

[9] Cireşan, D. C.; Meier, U.; Masci, J.; aj.: High-performance neural networks for visual object classification. *CoRR*, ročník abs/1102.0183, 2011.
Dostupné na http://arxiv.org/abs/1102.0183

[10] Clearswift: MIMEsweeper for SMTP. © 2015.
Dostupné na http://www.mimesweeper.com/products/mimesweeper-for-smtp

[11] Donahue, J.; Jia, Y.; Vinyals, O.; aj.: DeCAF: A Deep Convolutional Activation Feature for Generic Visual Recognition. *arXiv preprint arXiv:1310.1531*, ročník abs/1310.1531, 2013.
Dostupné na http://arxiv.org/abs/1310.1531

[12] Egmont-Petersen, M.; de Ridder, D.; Handels, H.: Image processing with neural networks—a review. *Pattern Recognition*, ročník 35, č. 10, Říjen 2002: s. 2279–2301, ISSN 00313203, doi:10.1016/S0031-3203(01)00178-9.

[13] Foundation, P. S.: Our Documentation. © 2001-2015.
Dostupné na `https://www.python.org/doc/`

[14] Ghemawat, S.; Dean, J.: Leveldb. 2014.
Dostupné na `https://github.com/google/leveldb`

[15] Google Inc.: Protocol Buffers - Google Developers.
Dostupné na `https://developers.google.com/protocol-buffers`

[16] Hinton, G. E.; Srivastava, N.; Krizhevsky, A.; aj.: Improving neural networks by preventing co-adaptation of feature detectors. *CoRR*, ročník abs/1207.0580, 2012.
Dostupné na `http://arxiv.org/abs/1207.0580`

[17] ImageVision: ImageVision: Automated Nudity Detection Service. 2013.
Dostupné na `http://www.imagevision.com/NuditySearch`

[18] Jia, Y.; Shelhamer, E.: Data. 2014.
Dostupné na `http://caffe.berkeleyvision.org/tutorial/data.html`

[19] Jia, Y.; Shelhamer, E.: Layer Catalogue. 2014.
Dostupné na `http://caffe.berkeleyvision.org/tutorial/layers.html`

[20] Jia, Y.; Shelhamer, E.: Loss. 2014.
Dostupné na `http://caffe.berkeleyvision.org/tutorial/loss.html`

[21] Jia, Y.; Shelhamer, E.; Donahue, J.; aj.: Caffe: Convolutional Architecture for Fast Feature Embedding. In *Proceedings of the ACM International Conference on Multimedia*, MM '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-3063-3, s. 675–678, doi:10.1145/2647868.2654889.

[22] Šíma Jiří; Roman, N.: *Teoretické otázky neuronových sítí*. Matfyzpress, 1996, ISBN 80-85863-18-9.
Dostupné na `http://www2.cs.cas.cz/~sima/kniha.pdf`

[23] Jones, M.; Rehg, J.: Compaq skin database. 1998.

[24] Krizhevskey, A.: Cuda-convnet. 2012.
Dostupné na `https://code.google.com/p/cuda-convnet`

[25] Krizhevsky, A.; Hinton, G.: Learning multiple layers of features from tiny images. *Computer Science Department, University of Toronto, Tech. Rep*, ročník 1, č. 4, 2009: str. 7.

[26] Krizhevsky, A.; Sutskever, I.; Hinton, G. E.: ImageNet Classification with Deep Convolutional Neural Networks. In *Advances in Neural Information Processing Systems 25*, editace F. Pereira; C. Burges; L. Bottou; K. Weinberger, Curran Associates, Inc., 2012, ISBN 9781627480031, s. 1097–1105.

[27] Lawrence, S.; Giles, C.; Tsoi, A. C.; aj.: Face recognition: a convolutional neural-network approach. *Neural Networks, IEEE Transactions on*, ročník 8, č. 1, Jan 1997: s. 98–113, ISSN 1045-9227, doi:10.1109/72.554195.

[28] Lee, H.; Grosse, R.; Ranganath, R.; aj.: Convolutional Deep Belief Networks for Scalable Unsupervised Learning of Hierarchical Representations. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, New York, NY, USA: ACM, 2009, ISBN 978-1-60558-516-1, s. 609–616, doi: 10.1145/1553374.1553453.
Dostupné na `http://doi.acm.org/10.1145/1553374.1553453`

[29] Liu, B.: *Web data mining: exploring hyperlinks, contents, and usage data*. Springer Science & Business Media, 2007, ISBN 978-3-642-19459-7.

[30] Lo, S.-C. B.; Chan, H.-P.; Lin, J.-S.; aj.: Artificial Convolution Neural Network for Medical Image Pattern Recognition. *Neural Netw.*, ročník 8, č. 7-8, Prosinec 1995: s. 1201–1214, ISSN 0893-6080, doi:10.1016/0893-6080(95)00061-5.

[31] Maas, A. L.; Hannun, A. Y.; Ng, A. Y.: Rectifier nonlinearities improve neural network acoustic models. In *Proc. ICML*, ročník 30, 2013.

[32] Macukow, B.; Arsenault, H.: Neural network model using a normalized inner product as a measure of similarity. In *Neural Networks, 1988., IEEE International Conference on*, July 1988, s. 225–230 vol.1, doi:10.1109/ICNN.1988.23851.

[33] Mashape Inc.: Nudity Recognition. 2012.
Dostupné na `https://www.mashape.com/imagevision/nudity-recognition-nudity-filter-for-images/overview`

[34] MATLAB: *version 8.5 (R2015a)*. Natick, Massachusetts: The MathWorks Inc., 2015.

[35] Mitra, S.; Hayashi, Y.: Neuro-fuzzy rule generation: survey in soft computing framework. *Neural Networks, IEEE Transactions on*, ročník 11, č. 3, 2000: s. 748–768, ISSN 1045-9227, doi:10.1109/72.846746.
Dostupné na `ftp://161.24.19.221/ele/jrsantos/Leitura/NeuroFuzzy_RuleGeneration.pdf`

[36] Nebauer, C.: Evaluation of convolutional neural networks for visual recognition. *Neural Networks, IEEE Transactions on*, ročník 9, č. 4, Jul 1998: s. 685–696, ISSN 1045-9227, doi:10.1109/72.701181.

[37] NVIDIA Corporation: About CUDA. 2015.
Dostupné na `https://developer.nvidia.com/about-cuda`

[38] Park, S. B.; Lee, J. W.; Kim, S. K.: Content-based Image Classification Using a Neural Network. *Pattern Recogn. Lett.*, ročník 25, č. 3, Únor 2004: s. 287–300, ISSN 0167-8655, doi:10.1016/j.patrec.2003.10.015.

[39] Platzer, C.; Stuetz, M.; Lindorfer, M.: Skin Sheriff: A Machine Learning Solution for Detecting Explicit Images. In *Proceedings of the 2Nd International Workshop on Security and Forensics in Communication Systems*, SFCS '14, New York, NY, USA: ACM, 2014, ISBN 978-1-4503-2802-9, s. 45–56, doi: 10.1145/2598918.2598920.
Dostupné na `http://doi.acm.org/10.1145/2598918.2598920`

[40] Ranzato, M. A.; lan Boureau, Y.; Cun, Y. L.: Sparse Feature Learning for Deep Belief Networks. In *Advances in Neural Information Processing Systems 20*, editace J. Platt; D. Koller; Y. Singer; S. Roweis, Curran Associates, Inc., 2008, s. 1185–1192.

[41] Russakovsky, O.; Deng, J.; Su, H.; aj.: ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)*, 2012, doi:10.1007/s11263-015-0816-y.

[42] Rutter, D.: Review: PORNsweeper. 2000.
Dostupné na `http://www.dansdata.com/pornsweeper.htm`

[43] Sermanet, P.; LeCun, Y.: Traffic sign recognition with multi-scale Convolutional Networks. In *Neural Networks (IJCNN), The 2011 International Joint Conference on*, July 2011, ISSN 2161-4393, s. 2809–2813, doi:10.1109/IJCNN.2011.6033589.

[44] Setiono, R.; Liu, H.: Neural-network feature selector. *Neural Networks, IEEE Transactions on*, ročník 8, č. 3, May 1997: s. 654–662, ISSN 1045-9227, doi: 10.1109/72.572104.

[45] Seznam.cz, a.s.: Lidé.cz. ©1996–2015.
Dostupné na `http://onas.seznam.cz/cz/lide-cz.html`

[46] Srivastava, N.; Hinton, G.; Krizhevsky, A.; aj.: Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *J. Mach. Learn. Res.*, ročník 15, č. 1, Leden 2014: s. 1929–1958, ISSN 1532-4435.
Dostupné na `http://jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf`

[47] Standford University: Unsupervised Feature Learning and Deep Learning Tutorial. 2013.
Dostupné na `http://ufldl.stanford.edu/tutorial/supervised/ConvolutionalNeuralNetwork`

[48] Vision, B.; Center, L.: Caffe — Deep Learning Framework. 2014.
Dostupné na `http://caffe.berkeleyvision.org/`

[49] Volná, E.: *Evoluční algoritmy a neuronové sítě*. Ostrava: Ostravská univerzita v Ostravě, první vydání, 2012.

Dostupné na http://www1.osu.cz/~volna/Evolucni_algoritmy_a_neuronove_site.pdf

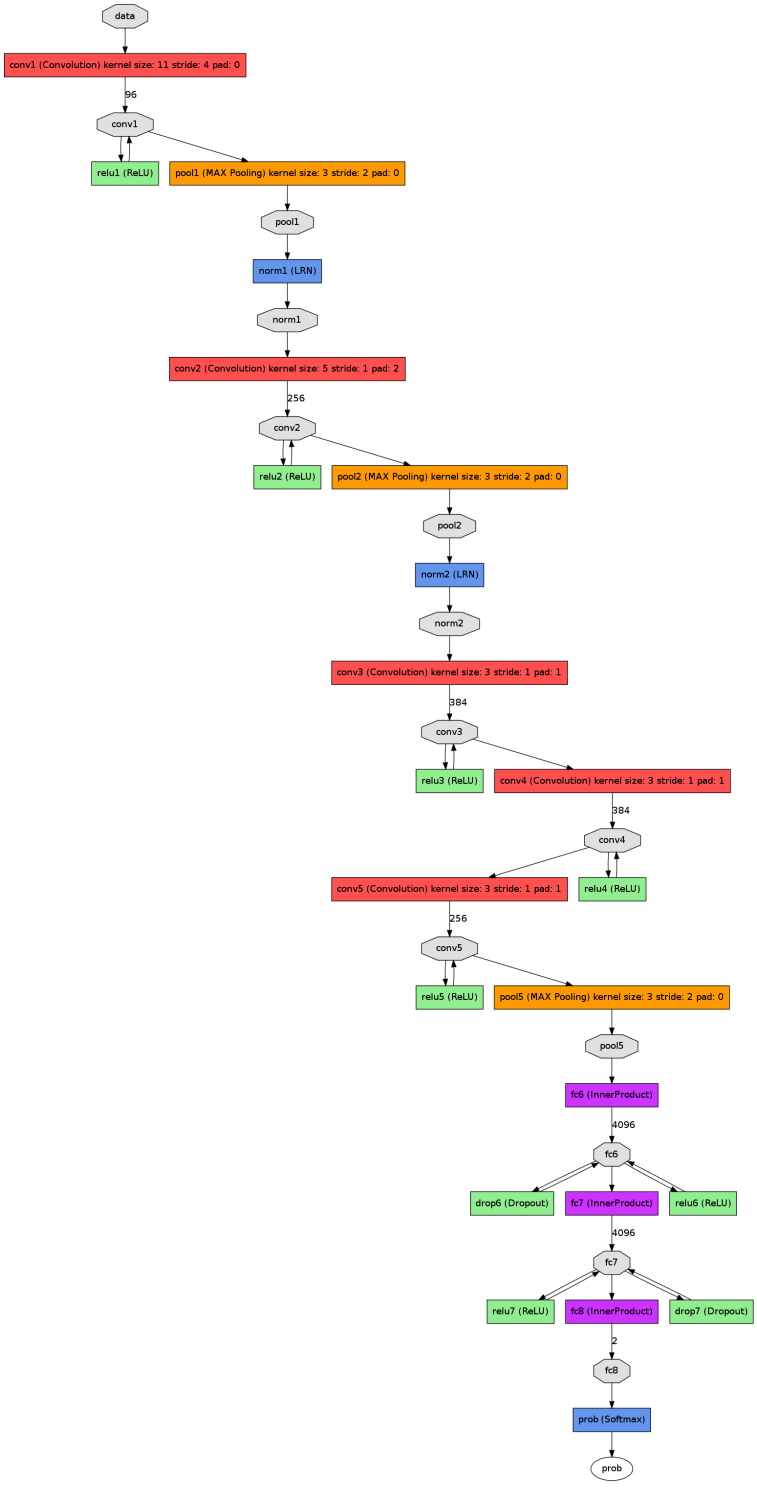
[50] Walsh, D.: Nude.js: Nudity Detection with JavaScript. 2011.
Dostupné na http://davidwalsh.name/nudejs

[51] Wied, P.: Nudity detection with JavaScript and HTMLCanvas. 2010.
Dostupné na http://www.patrick-wied.at/static/nudejs/

[52] YangSky: A Brief Survey of Porn-Detection/Porn-Removal Software. 2005.
Dostupné na http://www.yangsky.com/products/porndetect/htm/surveypornremoval.htm

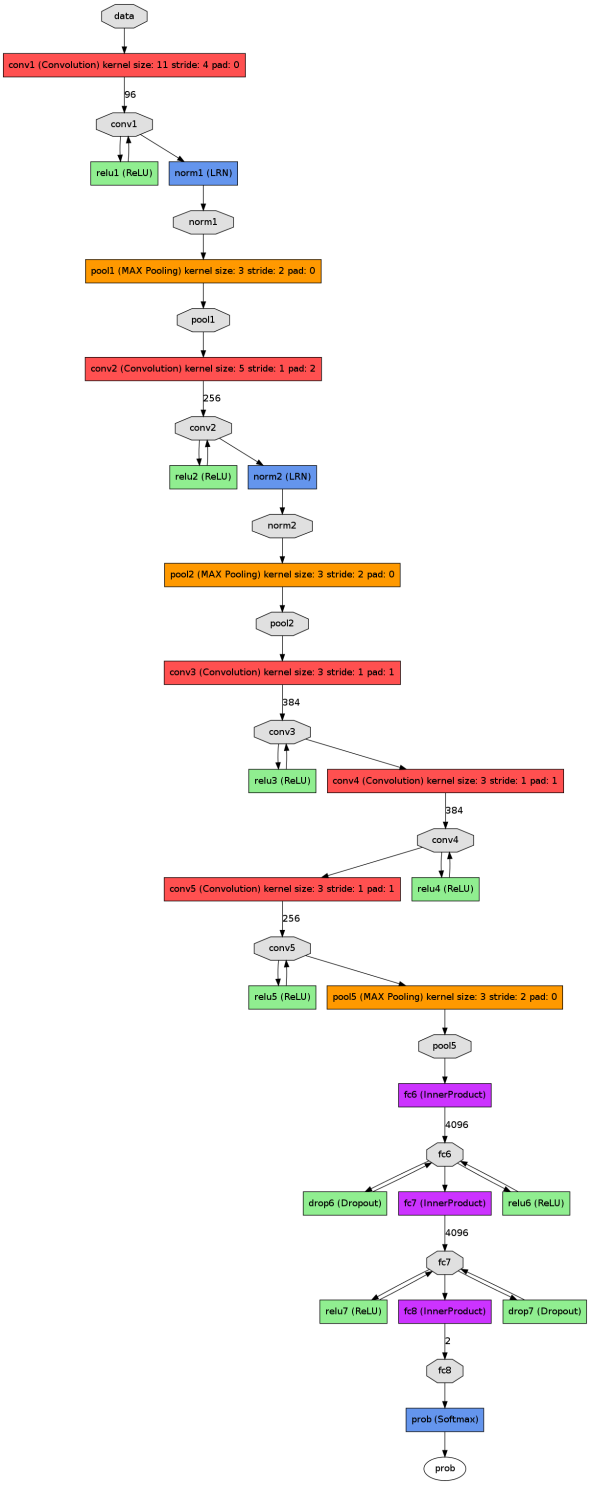
Přílohy

A Model CaffeNet



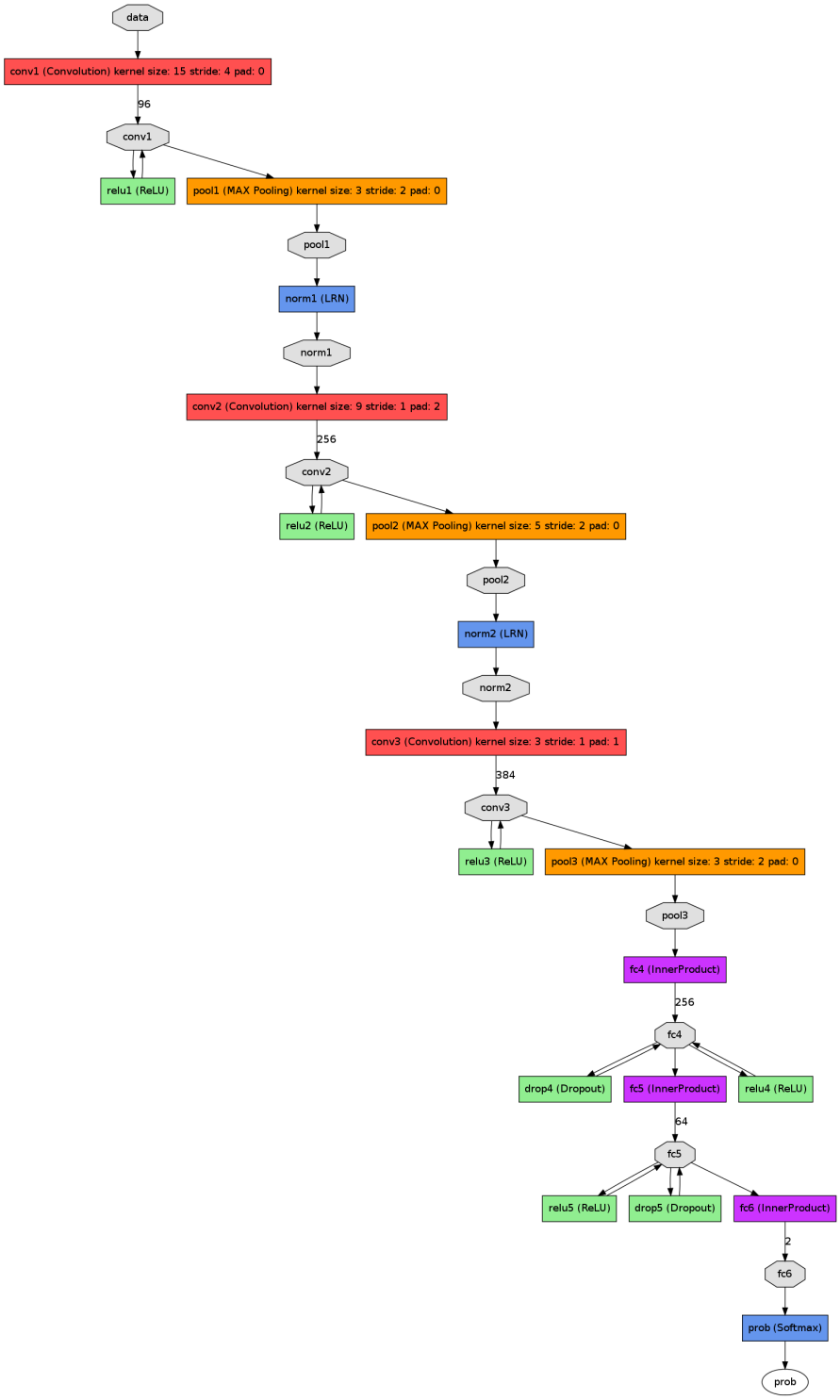
Obrázek 38: Model CaffeNet

B Model AlexNet



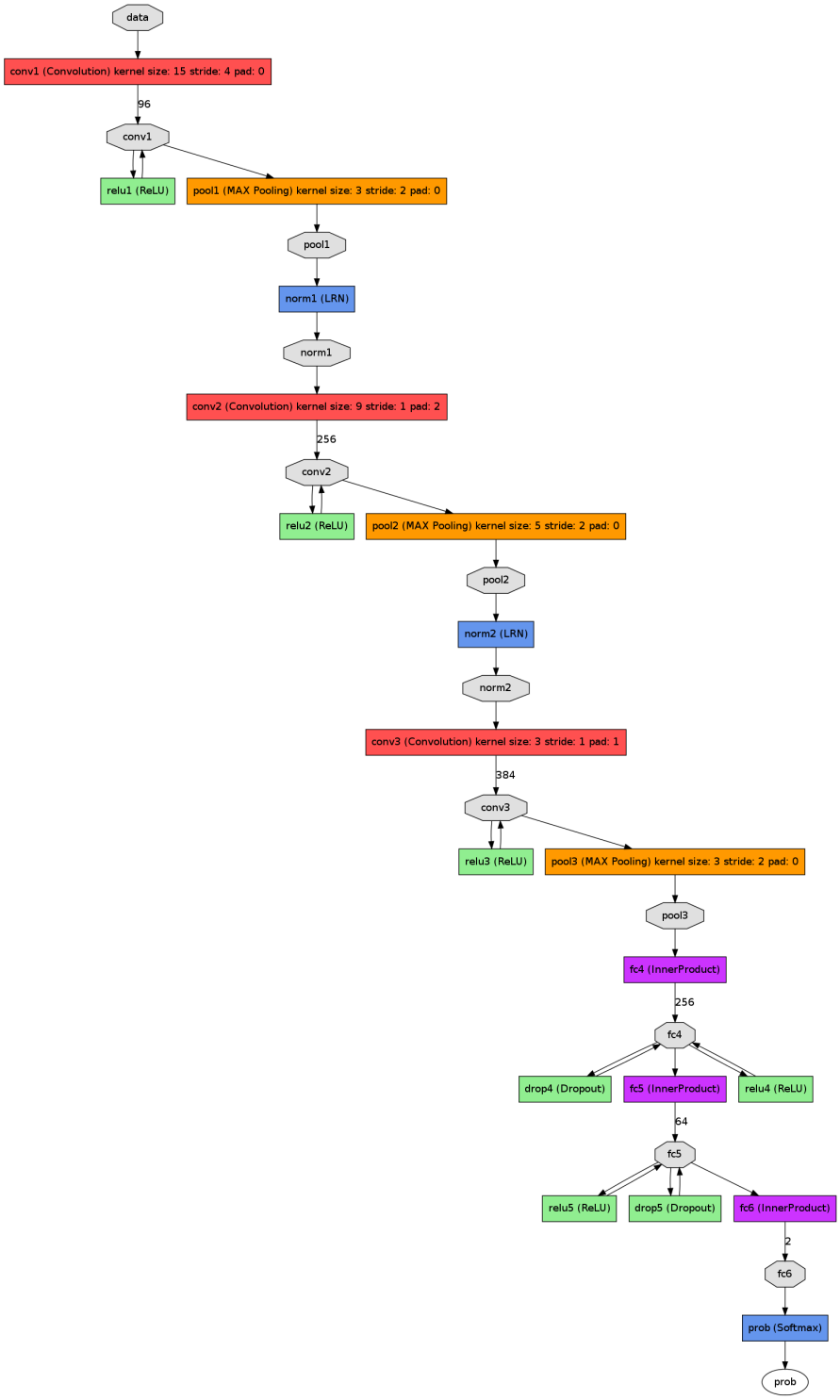
Obrázek 39: Model AlexNet

C Model SeznamNet



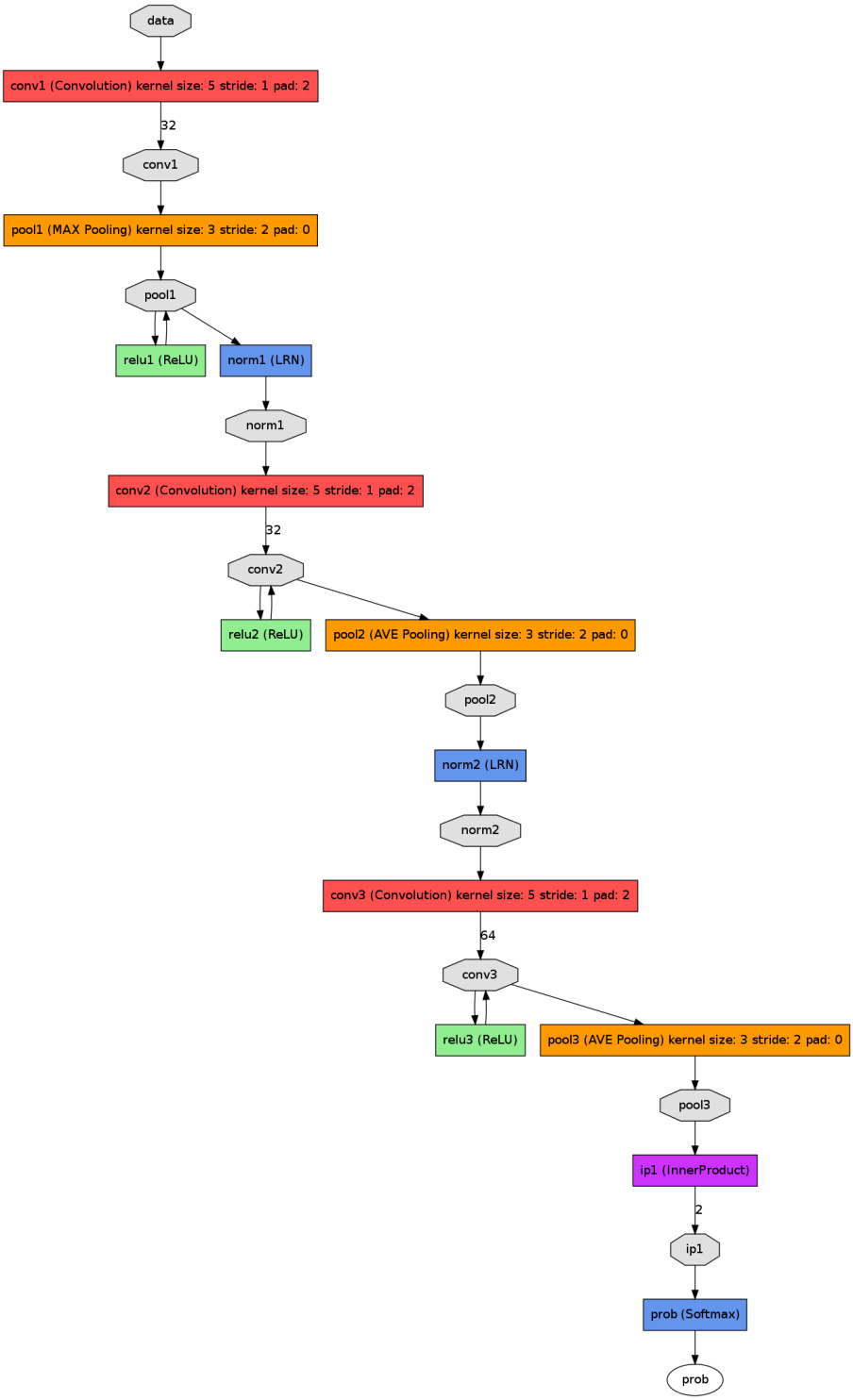
Obrázek 40: Model SeznamNet

D Model SAN



Obrázek 41: Model SAN

E Model Cifar10



Obrázek 42: Model Cifar10

F Ukázka solver konfigurace

Zde je uveden obsah *solver konfigurace* modelu CaffeNet MF.

```
net: "/www/picturedetector/backend/data/neural-networks/3/trainmodel.prototxt"
test_iter: 1000
test_interval: 1000
base_lr: 0.01
lr_policy: "step"
gamma: 0.1
stepsize: 100000
display: 100
max_iter: 80000
momentum: 0.9
weight_decay: 0.0005
snapshot: 1000
snapshot_prefix: "/www/picturedetector/backend/data/neural-networks/3/snapshots/snapshot"
solver_mode: GPU
```

G Ukázka deploy konfigurace

Zde je uveden obsah *deploy konfigurace* modelu CaffeNet MF.

```
name: "CaffeNet More Features"
input: "data"
input_dim: 10
input_dim: 3
input_dim: 227
input_dim: 227
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  convolution_param {
    num_output: 128
    kernel_size: 11
    stride: 4
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm1"
  type: "LRN"
  bottom: "pool1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "norm1"
  top: "conv2"
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
    group: 2
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
```

```
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm2"
  type: "LRN"
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm2"
  top: "conv3"
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4"
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
    group: 2
  }
}
layer {
  name: "relu4"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "conv4"
  top: "conv5"
  convolution_param {
```

```
    num_output: 256
    pad: 1
    kernel_size: 3
    group: 2
  }
}
layer {
  name: "relu5"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "fc6"
  inner_product_param {
    num_output: 4096
  }
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "fc6"
  top: "fc6"
}
layer {
  name: "drop6"
  type: "Dropout"
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc7"
  type: "InnerProduct"
  bottom: "fc6"
  top: "fc7"
  inner_product_param {
    num_output: 4096
  }
}
layer {
  name: "relu7"
  type: "ReLU"
  bottom: "fc7"
  top: "fc7"
}
layer {
  name: "drop7"
  type: "Dropout"
```

```
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  inner_product_param {
    num_output: 2
  }
}
layer {
  name: "prob"
  type: "Softmax"
  bottom: "fc8"
  top: "prob"
}
```

H Ukázka train konfigurace

Zde je uveden obsah *train konfigurace* modelu CaffeNet MF.

```
name: "CaffeNet More Features"
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TRAIN
  }
  transform_param {
    mirror: true
    crop_size: 227
    mean_file: "/www/picturedetector/backend/data/neural-networks/1/temp/mean_file_train.binaryproto"
  }
  data_param {
    source: "/www/picturedetector/backend/data/neural-networks/1/temp/imagenet_train_db"
    batch_size: 64
    backend: LMDB
  }
}
layer {
  name: "data"
  type: "Data"
  top: "data"
  top: "label"
  include {
    phase: TEST
  }
  transform_param {
    mirror: false
    crop_size: 227
    mean_file: "/www/picturedetector/backend/data/neural-networks/1/temp/mean_file_val.binaryproto"
  }
  data_param {
    source: "/www/picturedetector/backend/data/neural-networks/1/temp/imagenet_val_db"
    batch_size: 50
    backend: LMDB
  }
}
layer {
  name: "conv1"
  type: "Convolution"
  bottom: "data"
  top: "conv1"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 128
    kernel_size: 11
    stride: 4
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
```

```
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu1"
  type: "ReLU"
  bottom: "conv1"
  top: "conv1"
}
layer {
  name: "pool1"
  type: "Pooling"
  bottom: "conv1"
  top: "pool1"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm1"
  type: "LRN"
  bottom: "pool1"
  top: "norm1"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv2"
  type: "Convolution"
  bottom: "norm1"
  top: "conv2"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 256
    pad: 2
    kernel_size: 5
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu2"
  type: "ReLU"
  bottom: "conv2"
```

```
  top: "conv2"
}
layer {
  name: "pool2"
  type: "Pooling"
  bottom: "conv2"
  top: "pool2"
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "norm2"
  type: "LRN"
  bottom: "pool2"
  top: "norm2"
  lrn_param {
    local_size: 5
    alpha: 0.0001
    beta: 0.75
  }
}
layer {
  name: "conv3"
  type: "Convolution"
  bottom: "norm2"
  top: "conv3"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "relu3"
  type: "ReLU"
  bottom: "conv3"
  top: "conv3"
}
layer {
  name: "conv4"
  type: "Convolution"
  bottom: "conv3"
  top: "conv4"
  param {
    lr_mult: 1
    decay_mult: 1
```

```
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 384
    pad: 1
    kernel_size: 3
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu4"
  type: "ReLU"
  bottom: "conv4"
  top: "conv4"
}
layer {
  name: "conv5"
  type: "Convolution"
  bottom: "conv4"
  top: "conv5"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  convolution_param {
    num_output: 256
    pad: 1
    kernel_size: 3
    group: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu5"
  type: "ReLU"
  bottom: "conv5"
  top: "conv5"
}
layer {
  name: "pool5"
  type: "Pooling"
  bottom: "conv5"
  top: "pool5"
```

```
  pooling_param {
    pool: MAX
    kernel_size: 3
    stride: 2
  }
}
layer {
  name: "fc6"
  type: "InnerProduct"
  bottom: "pool5"
  top: "fc6"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu6"
  type: "ReLU"
  bottom: "fc6"
  top: "fc6"
}
layer {
  name: "drop6"
  type: "Dropout"
  bottom: "fc6"
  top: "fc6"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc7"
  type: "InnerProduct"
  bottom: "fc6"
  top: "fc7"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 4096
    weight_filler {
      type: "gaussian"
      std: 0.005
    }
```

```
    bias_filler {
      type: "constant"
      value: 1
    }
  }
}
layer {
  name: "relu7"
  type: "ReLU"
  bottom: "fc7"
  top: "fc7"
}
layer {
  name: "drop7"
  type: "Dropout"
  bottom: "fc7"
  top: "fc7"
  dropout_param {
    dropout_ratio: 0.5
  }
}
layer {
  name: "fc8"
  type: "InnerProduct"
  bottom: "fc7"
  top: "fc8"
  param {
    lr_mult: 1
    decay_mult: 1
  }
  param {
    lr_mult: 2
    decay_mult: 0
  }
  inner_product_param {
    num_output: 2
    weight_filler {
      type: "gaussian"
      std: 0.01
    }
    bias_filler {
      type: "constant"
      value: 0
    }
  }
}
layer {
  name: "accuracy"
  type: "Accuracy"
  bottom: "fc8"
  bottom: "label"
  top: "accuracy"
  include {
    phase: TEST
  }
}
layer {
  name: "loss"
  type: "SoftmaxWithLoss"
  bottom: "fc8"
  bottom: "label"
  top: "loss"
}
```