VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

ANOTÁTOR DATOVÉ SADY PRO TRÉNOVÁNÍ NEURONOVÝCH SÍTÍ

DATASET ANNOTATOR FOR NEURAL NETWORK TRAINING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MARTIN SCHNEIDER**
AUTHOR

VEDOUCÍ PRÁCE **Ing. VÍTĚZSLAV BERAN, Ph.D.**
SUPERVISOR

BRNO 2021

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2020/2021

Zadání bakalářské práce

Student: **Schneider Martin**
Program: Informační technologie
Název: **Anotátor datové sady pro trénování neuronových sítí**
Dataset Annotator for Neural Network Training
Kategorie: Uživatelská rozhraní

Zadání:

1. Prostudujte nástroje pro tvorbu moderních webových aplikací. Seznamte se s postupy trénovaní neuronových sítí a existujícími anotačními nástroji.
2. Navrhněte aplikaci, která je inicializována menší množinou anotovaných dat. Tato data jsou použita pro trénování vybraného detektoru (např. konvoluční neuronové sítě), který anotuje zbývající data. Tyto anotace jsou zkontrolovány/upraveny člověkem a použity pro další iteraci trénování.
3. Navrženou aplikaci implementujte pomocí vybrané technologie pro tvorbu moderních webových aplikací.
4. Vyhodnoťte vlastnosti výsledného systému na základě experimentů s reálnými uživateli.
5. Prezentujte klíčové vlastnosti řešení formou plakátu a krátkého videa.

Literatura:

- Brian Burke. *Gamify: how gamification motivates people to do extraordinary things.* Brookline, MA: Garthner, Inc., 2014. ISBN 9781937134853.
- Joel Marsh. *UX pro začátečníky.* Zoner Press, 2019.
- Steve Krug. *Don't make me think, revisited: a common sense approach to web usability.* San Francisco: New Riders, ISBN 978-0321965516.
- Dále dle pokynu vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Body 1, 2 a částečně bod 3.

Podrobné závazné pokyny pro vypracování práce viz https://www.fit.vut.cz/study/theses/

Vedoucí práce: **Beran Vítězslav, Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2020
Datum odevzdání: 12. května 2021
Datum schválení: 30. října 2020

Abstrakt

Cílem práce je návrh a implementace systému umožňujícího anotovat obrazová data pro učení neuronových sítí. Důležitou vlastností tohoto systému je iterativní provádění učení, kdy anotovanou množinu rozšiřuje i sama neuronová síť a uživatelé provádí korekce nad rozšiřujícími daty. Systém se skládá z informačního systému s vloženým nástrojem pro anotování komunikujícím s programem řídícím neuronové sítě. Informační systém je navržen podle architektonického modelu MVC. Frontend aplikace je postaven na knihovně React jazyka JavaScript. Design frontendu je proveden dle konvencí Material Design s využitím Material-UI frameworku. Frontend udržuje trvalé spojení s Python backendem s využitím protokolu WebSocket, jehož využívá i backend pro komunikaci s programem pro řízení neuronových sítí. NoSQL databázový systém využívá technologii MongoDB.

Abstract

The goal of this thesis is design and implementation of a system for image data annotation which will learn neural networks. An important feature of this system is the iterative approach of learning, where the annotated set is extended by the neural network itself and users make corrections of these extensions. The system consists of an information system with an embedded annotation tool. Information system communicates with neural network management program. The information system is designed according to the MVC architectural model. The frontend of the application is built on JavaScript library – React and is designed according to Material Design conventions using the Material-UI framework. The frontend maintains a permanent connection to the Python backend using the WebSocket protocol, which is also used by the backend to communicate with the neural network management program. The NoSQL database system uses MongoDB technology.

Klíčová slova

anotátor, anotační nástroj, iterativní učení, neuronová síť, grafické uživatelské rozhraní, webová aplikace, informační systém, JavaScript, React, Material Design, Python, MongoDB, WebSocket

Keywords

annotator, annotation tool, iterative learning, neural network, graphical user interface, web application, information system, JavaScript, React, Material Design, Python, MongoDB, WebSocket

Citace

SCHNEIDER, Martin. *Anotátor datové sady pro trénování neuronových sítí.* Brno, 2021. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Vítězslav Beran, Ph.D.

Anotátor datové sady pro trénování neuronových sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vítězslav Beran, Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Martin Schneider
9. května 2021

Poděkování

Tímto bych chtěl poděkovat panu Ing. Vítězslavu Beranovi, Ph.D. za veškeré konzultace, přínosné rady a trpělivý přístup. Dále děkuji všem testovacím uživatelům za jejich čas a za zpětnou vazbu k výsledné aplikaci.

Obsah

Kapitola 1

Úvod

Celosvětově se stále zvyšuje míra automatizace. Často však není reálné vytvořit program zpracovávající pro počítač složité vstupy z okolního světa jako třeba obraz a zvuk, které však člověk zpracovává přirozeně. Obecně platí, že lidský faktor je nejdražším prvkem ve službách i průmyslu. Kompromisem je využití neuronových sítí, které simulují lidský mozek. Neuronovou síť tak jako lidský mozek je však třeba naučit k námi požadovaným činnostem, kdy je třeba lidského faktoru. Pro kvalitní naučení sítě je třeba velké množství vzorových dat vytvořených člověkem.

Cílem této bakalářské práce je vytvoření systému sloužícímu pro jednoduché vytváření těchto vzorových dat – anotovaných dat, správu lidských zdrojů vytvářejících anotovaná data a komunikaci s neuronovými sítěmi za účelem předkládání anotovaných dat pro učení těchto sítí. Významnou vlastností tohoto systému je iterativní provádění učení, kdy sama neuronová síť zpětnou vazbou přispívá do množiny anotovaných dat a uživatelé pouze provádí korekce těchto anotací. Dalším pilířem systému je využití webových technologií, které umožňují efektivnější nasazení ve firemním prostředí, ale především zjednodušení práce z domova. Motivací pro uplatnění těchto vlastností je časová úspora lidského času a redukování chyb spojených s organizací lidských zdrojů a datových struktur.

Druhá kapitola pojednává o principech technologií vhodných pro řešení systému, který je hlavní náplní této bakalářské práce a zahrnuje průzkum podobných řešení a pojednává o jejich absenci. Kapitola třetí popisuje již konkrétní požadavky na systém a předkládá čtenáři návrh celého systému a jeho dílčích součástí. Čtvrtá kapitola řeší implementační rozhodnutí a konkrétní postupy při implementaci výsledného systému. Dále čtenáře informuje o tom, jak bylo výsledné řešení testováno a o možných rozšířeních, která jsou i částečnou reflexí testování a jsou uvedena pro vylepšení funkčnosti a komfortnosti používání systému. Kapitola pátá jakožto poslední kapitola je věnována zhodnocení a shrnutí celé práce.

Kapitola 2

Podklady pro návrh a realizaci systému

2.1 Nástroje pro tvorbu moderní webové aplikace

Webové aplikace mohou být jednoduchými řešeními. Pro větší projekty, které již obsahují správu uživatelů a ostatních dat je vhodné využít princip informačního systému s vlastní architekturou. Díky popularitě webových aplikací je k jejich implementaci dostupná široká škála knihoven a frameworků.

Informační systém

Informační systém se skládá z vzájemně propojených informací a procesů, které s těmito informacemi pracují. Pod pojmem procesy rozumíme funkce, které zpracovávají informace do vstupující systému a transformují je na informace vystupující ze systému. Zjednodušeně můžeme říct, že procesy jsou funkce provádějící sběr, přenos, uložení, zpracování a distribuci informací. Pod pojmem informace pak rozumíme data, která slouží zejména pro rozhodování a řízení v rozsáhlejším systému. Do celkové funkce informačního systému se také promítá nezanedbatelná položka okolí. Okolí informačního systému tvoří veškeré objekty, které změnou svých vlastností ovlivňují samotný systém, a také objekty, které naopak mění své vlastnosti v závislosti na systému. Více informací viz. [16].

Architektura MVC

Velmi populární architekturou pro webové aplikace je MVC. Vytváření aplikací s využitím architektury MVC vyžaduje vytvoření tří komponent, mezi které patří:

- Model (model), což je doménově specifická reprezentace informací, s nimiž aplikace pracuje.
- View (pohled), který převádí data reprezentovaná modelem do podoby vhodné k interaktivní prezentaci uživateli.
- Controller (řadič), který reaguje na události a zajišťuje změny v modelu. Na základě těchto změn se aktualizuje samotný pohled.

Obecně platí tento princip realizace:

1. Uživatel provede nějakou akci v uživatelském rozhraní (např. stiskne tlačítko).
2. Řadič obdrží oznámení o této akci z objektu uživatelského rozhraní.
3. Řadič přistoupí k modelu a v případě potřeby ho zaktualizuje na základě provedené uživatelské akce (např. zaktualizuje nákupní košík uživatele).
4. Model je pouze jiný název pro doménovou vrstvu. Doménová logika zpracuje změněná data (např. přepočítá celkovou cenu, daně a expediční poplatky pro položky v košíku). Některé aplikace užívají mechanismus pro perzistentní uložení dat (např. databázi). To je však otázka vztahu mezi doménovou a datovou vrstvou, která není architekturou MVC pokryta.
5. Komponenta pohled použije zaktualizovaný model pro zobrazení zaktualizovaných dat uživateli (např. vypíše obsah košíku). Komponenta pohled získává data přímo z modelu, zatímco model nepotřebuje žádné informace o komponentě View (je na ní nezávislý). Nicméně je možné použít návrhový vzor pozorovatel, umožňující modelu informovat jakoukoliv komponentu o případných změnách dat. V tom případě se komponenta view zaregistruje u modelu jako příjemce těchto informací. Je důležité podotknout, že řadič nepředává doménové objekty (model) komponentě pohledu, nicméně jí může poslat příkaz, aby svůj obsah podle modelu zaktualizovala.
6. Samotnému konečnému zobrazení výsledku uživateli ještě může u web-aplikací předcházet odpověď ze serveru na klienta, aby si ihned vyžádal obnovení stránky (client side redirect, životnost 0, takže okamžitý): Tím je zaručeno, že při obnovení stránky uživatelem (refresh, F5 v prohlížeči) nevyvolá na serveru požadovanou akci opakovaně, ale že se jedná pouze o obnovení pohledu, nyní už bez požadavku na změnu dat (modelu). Účelem je změna URL a dat http requestu, aby poslední v řadě již nebyl "server-side data-affecting"(ovlivňující model), ale pouze "read-only"(pouhé zobrazení). Celý tento client-refresh (změna URL) se děje automaticky a bez povšimnutí uživatelem.
7. Uživatelské rozhraní čeká na další akci uživatele, která celý cyklus zahájí znovu.

Podrobněji v [10].

Model – databázový systém

Databáze je organizovaný soubor dat, obvykle uložený a přístupný elektronicky z počítačového systému. Tam, kde jsou databáze složitější, se často vyvíjejí pomocí formálních návrhových a modelovacích technik. V dnešní době se využívá dvou typů databází — relačních, využívajících dotazovacího jazyka SQL a nerelačních, označovaných jako NoSQL, protože nevyužívají jazyk SQL.

SQL

- PostgreSQL je open-source relační databázový systém s nejpokročilejší funkcionalitou. Vývoj zaštiťuje PostgreSQL Global Development Group od vydání v roce 1996 (založeno na Postgres z roku 1989).

- MySQL zastává pozici nejpoužívanějšího open-source databázového systému. Díky tomu lze k němu najít velké množství literatury. Od roku 2010 jej vyvíjí Oracle Corporation, ale vznik systému je datován již k roku 1995.
- Oracle je proprietárním relačním databázovým systémem společnosti Oracle Corporation. V roce 1980 byl vytvořen jako první SQL databázový systém a je globálně nejpoužívanějším databázovým systémem.

Podrobnější srovnání těchto SQL systémů viz. [6].

NoSQL

- MongoDB od vzniku v roce 2009 je vyvíjen společností MongoDB Inc. a dosáhl nejvyšší příčky popularity v oblasti NoSQL databázových systémů a 5. pozice mezi všemi databázovými systémy. Vzhledem k jeho rozšířenosti je podporován velkým množství programovacích jazyků. Podrobnější informace viz. [4].

View – frontend

Pro zobrazení webového obsahu v dnešních prohlížečích slouží tři základní jazyky HTML, CSS a JavaScript. Základní strukturu a prezentovaná data obsahují značky jazyka HTML. Pro úpravu vzhledu jejich reprezentace se využívá stylovacího jazyka CSS. Dynamicky zasahovat do těchto předpisů lze pomocí jazyka JavaScript, jenž obecně vytváří funkcionalitu aplikace.

HTML je značkovací jazyk, který popisuje význam jednotlivých prvků a prostřednictvím odkazů je. Jeho první verze vznikla v roce 1993 a v roce 1995 byl standardizován jako RFC. Finální specifikace nejnovější verze HTML5 zaštítěna společností W3C byla vydána v roce 2014. Více informací viz. [9].

CSS jako stylovací jazyk umožňuje vytvářet styly pro jednotlivé prvky a třídy (X)HTML a XML dokumentů. Vytvořené styly udávají grafickou podobu dat obsažených v dokumentu. Vznikl v roce 1994 a záštitu W3C získal v roce 1996. Aktuálně používaná verze CSS3 je od roku 1999 ve vývoji. Podrobnější informace viz. [5, 8].

JavaScript je dynamicky typovaný interpretovaný programovací jazyk, který vytváří dynamičnost pro dokumenty v jazyce (X)HTML obohacením o interaktivní prvky. Statické typování implementuje jeho nástavba TypeScript. JavaScript můžeme rozdělit na prohlížečovou část a na jádro ECMAScript, které neobsahuje žádné metody pro manipulaci s HTML dokumentem a které lze používat i mimo webový prohlížeč. Existuje velké množství knihoven a frameworků, jejichž balíčky je možné spravovat například pomocí NPM nebo Yarn. První verze ECMAScript byla vydána v roce 1997 a jazyk je udržován organizací ECMA International i v současnosti. Více viz. [5].

Controller – backend

Implementace backend části informačního systému umožňuje velkou volnost ve výběru technologie. Některé technologie jako PHP a Node.js jsou přímo zaměřeny na tyto účely. Ostatní jazyky ale obvykle disponují knihovnami či frameworky, které též umožňují jednoduchou implementaci serverové části.

PHP je skriptovací programovací jazyk. Je určený především pro programování dynamických internetových stránek a webových aplikací například ve formátu HTML, XHTML či WML. PHP lze použít i k tvorbě konzolových a desktopových aplikací. Pro desktopové

použití existuje kompilovaná forma jazyka. Při použití PHP pro dynamické stránky jsou skripty prováděny na straně serveru – k uživateli je přenášen až výsledek jejich činnosti. Od roku 1995 je vyvíjen organizací The PHP Group. Konkrétnější informace viz. [13].

Node.js je softwarový systém navržený pro psaní vysoce škálovatelných internetových aplikací, především webových serverů. Programy pro Node.js jsou psané v jazyce JavaScript, hojně využívající model událostí a asynchronní I/O operace pro minimalizaci režie procesoru a maximalizaci výkonu. Vytvořil jej v roce 2009 Ryan Dahl a dále je vyvíjen především komunitou. Více viz. [11].

Python je skriptovací programovací jazyk s dynamickou kontrolu datových typů a podporuje různá programovací paradigmata, včetně objektově orientovaného, imperativního, procedurálního nebo funkcionálního. Vznikl v roce 1991 dle návrhu Guida van Rossuma a je udržován organizací Python Software Foundation. Podrobnější informace viz. [14].

Síťové komunikační protokoly

Pro kontejnerizované moduly a moduly implementované různými technologiemi je nutné použít vhodný síťový komunikační protokol.

REST je architektura rozhraní použitelná pro jednotný a snadný přístup ke zdrojům. Všechny zdroje mají vlastní identifikátor URI a REST definuje čtyři základní metody pro přístup k nim. Spojení je při každém požadavku znovu navazováno – komunikační protokol je bezstavový. REST navrhl a popsal v roce 2000 Roy Fielding (jeden ze spoluautorů protokolu http). Více o rozhraní viz. [12].

WebSocket je počítačový komunikační protokol poskytující plně duplexní (obousměrný) komunikační kanál přes jediné TCP připojení. Spojení je trvale udržováno – komunikační protokol je stavový. Protokol WebSocket byl standardizován komisí IETF jako RFC 6455 v roce 2011, a WebSocket API ve Webové IDL bylo standardizováno konsorciem W3C. Podrobnější informace viz. [15].

Konkrétní výběr implementační technologie obvykle závisí na preferencích programátora nebo například na požadavcích zaměstnavatele, který může vyžadovat konzistenci technologií svých projektů. Množina technologií je však omezena podle návrhu systému, kterým se zabývá následující kapitola 3.

2.2 Neuronové sítě

Umělá neuronová síť je kolekce propojených uzlů nazývaných umělé neurony, které se snaží modelovat biologický mozek. Každé spojení, stejně jako synapse v biologickém mozku, může přenášet signál do dalších neuronů. Umělý neuron přijímá signál, který zpracuje a vysílá signály neuronům k němu připojeným. Signál je reálné číslo a výstup každého neuronu je počítán zadanou nelineární funkcí součtu jeho vstupů. Spoje se nazývají hrany. Neurony a hrany mají obvykle váhu, která se přizpůsobuje postupujícímu učení. Neurony mohou mít prahovou hodnotu takovou, že signál je vysílán pouze v případě, že souhrnný signál překročí tuto prahovou hodnotu. Neurony se obvykle agregují do vrstev. Různé vrstvy mohou na svých vstupech provádět různé transformace. Signály putují z první vrstvy (vstupní vrstvy) do poslední vrstvy (výstupní vrstvy). Podrobnější informace viz. [7].

Architektura U-Net

U-Net je architektura neuronové sítě, která se skládá ze zadávací cesty k zachycení kontextu a symetrické rozšiřovací cesty, která umožňuje přesnou lokalizaci obrazových bodů. Této síti stačí k naučení menší počet obrazů než jiným sítím. Je také velmi rychlá. Segmentace obrazu o velikosti 512x512 pixelů trvá méně než jednu sekundu na výkonné GPU. U-Net jednoduše zřetězuje mapovací prvky kodéru s mapovacími vlastnostmi dekodéru v každé fázi, čímž vytváří strukturu podobnou žebříku. Architektura díky zkratkovým spojením umožnuje dekodéru v každé fázi učení získat relativní vlastnosti, které jsou ztraceny při sdružovací vrstvě v kodéru. Diplomová práce zabývající se sítí této architektury viz. [1].

Knihovny pro práci s neuronovými sítěmi

Pro zjednodušení vytváření struktury neuronové sítě, načítání dat a manipulaci s výpočetními jednotkami lze využít k těmto účelům existující knihovny.

TensorFlow[1]

Tato open-source knihovna může být využita na různé účely, ale hlavním zaměřením je na trénování a inference hlubokých neuronových sítí. Je využívána pro výzkum i produkci ve společnosti Google. TensorFlow stabilně podporuje Python API jazyka C a může běžet na vícero CPU i GPU s podporou CUDA. Knihovna byla vyvinuta týmem Google Brain a byla vydána pod licencí Apache License 2.0 v roce 2015.

PyTorch[2]

Je další open-source knihovna pro strojové učení založená na knihovně Torch. Používá se pro aplikace zabývající se například počítačovým viděním a zpracováním přirozeného jazyka. Podporuje CPU i GPU s technologií CUDA. Vývojem se zabývá Facebook a první vydání bylo uvolněno v roce 2016.

2.3 Trénování neuronových sítí

Neuronové sítě se učí zpracováním příkladů, z nichž každý obsahuje vstup a správný výsledek. V oblasti segmentačního zpracování obrazu jsou to obrazová data a jejich maska, mezi nimiž vznikají pravděpodobnostní vazby, které jsou uloženy v datové struktuře samotné sítě. Trénování neuronové sítě z daného příkladu se obvykle provádí určením rozdílu mezi zpracovaným výstupem sítě a cílovým výstupem, který je označen jako chyba. Síť poté upraví váhy pravděpodobnostních vazeb podle nastaveného pravidla pro učení pomocí této chybové hodnoty. Postupné úpravy způsobí, že neuronová síť bude produkovat výstup, který je stále více podobný cílovému výstupu. Po dostatečném počtu těchto úprav lze trénování ukončit na základě určitých kritérií. Pro více informací viz. [7].

[1] https://www.tensorflow.org/
[2] https://pytorch.org/

2.4 Anotace

Pro správný výsledek, vázaný ke vstupu neuronové sítě v oblasti zpracování obrazu, se využívá pojmu anotace, která může být reprezentována maskou vstupu. Pro vytváření těchto masek je třeba grafického editoru, kdy v oblasti webových aplikací není k dispozici takové množství nástrojů s tak pokročilými funkcemi jako u desktopových aplikacích.

Canvas API

Aplikační rozhraní reprezentuje element *canvas* jazyka HTML. API umožňuje kreslení pomocí jazyka JavaScript a mimo jiné i tvorbu animací, vizualizaci dat, manipulaci s obrazem a zpracování videa v reálném čase. Hlavním zaměřením aplikačního rozhraní je 2D grafika. S pomocí WebGL API lze element *canvas* využít i k hardwarově akcelerované 2D a 3D grafice. Aplikační rozhraní umožňuje velmi nízkoúrovňový přístup k plátnu, a proto existují knihovny pracující s tímto rozhraním na vyšší úrovni.

Fabric.js

Tato knihovna jazyka JavaScript umožňuje snadnější manipulaci s *canvas* elementem jazyka HTML. Fabric poskytuje chybějící objektový model pro kreslící plátno, stejně jako analyzátor SVG, vrstvu interaktivity a celou řadu dalších nástrojů. Jedná se o open-source projekt licencovaný pod MIT. Fabric byl vytvořen v roce 2010 při práci na projektu printio.ru, kde autor postrádal vyšší úroveň aplikačního rozhraní elementu canvas. V tehdejší době byl k dispozici pouze Flash, který postupně zanikl.

Konva.js

Konva je knihovnou jazyka JavaScript, která rozšiřuje 2D kontext *canvas* elementu jazyka HTML pro desktopy i mobilní zařízení. Knihovna umožňuje vytvářet animace, přechody, vnoření uzlů, vrstvení, filtrování, ukládání do mezipaměti, zpracování událostí pro stolní a mobilní aplikace a další s důrazem na vysoký výkon. První verze byla vydána jako pokračování knihovny KineticJS po ukončení jejího vývoje v roce 2015.

2.5 Existující anotační nástroje

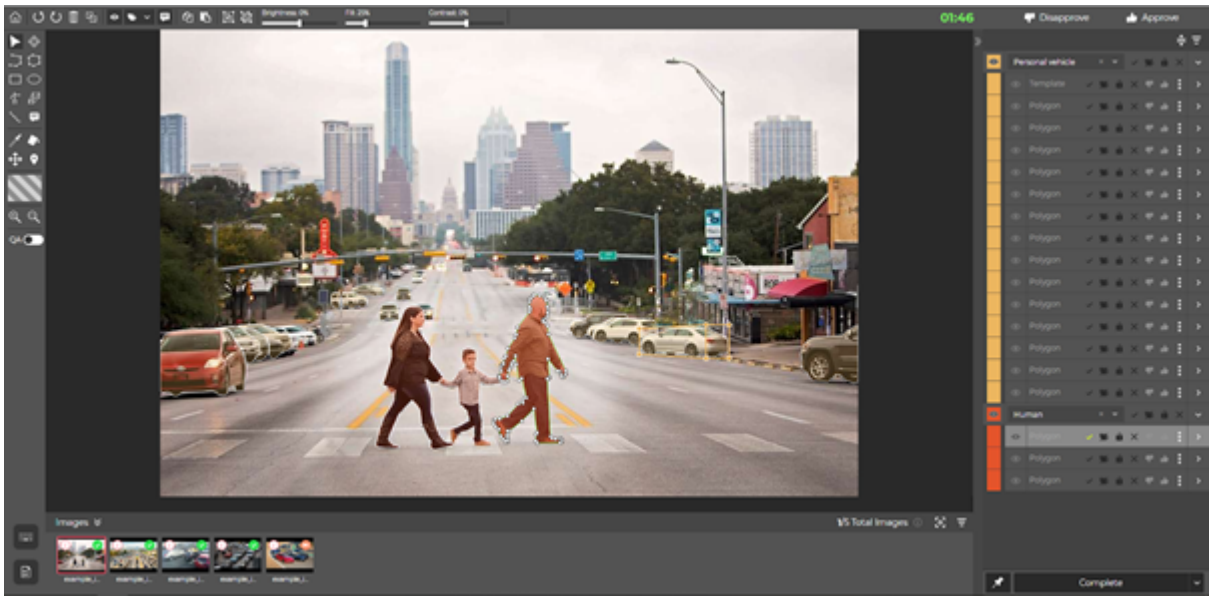
Hlavní motivací vzniku této práce je absence obdobného nástroje jak proprietárního, tak open-source. Dle průzkumu ve společnosti dodávající komplexní řešení automatizovaných výrobních linek, jejichž součástí je i detekce vadných kusů, jsou neuronové sítě pro tyto účely anotovány pomocí nástroje GIMP. Organizace lidských zdrojů probíhá přímým kontaktem. Z průzkumu dostupných řešení zde uvádím alespoň mírně podobná řešení, která však neobsahují informační systém, čímž nejsou vhodné pro nasazení na větší struktury dat, zvláště ve firemním prostředí.

SuperAnnotate[3]

Tento proprietární nástroj s moderním designem umožňuje anotování i více třídami. Neobsahuje ale informační systém pro správu uživatelů a informace o jejich práci navíc by se množstvím nástrojů mohl zdát složitým pro určité skupiny uživatelů viz. obrázek 2.1.

[3] https://superannotate.com/

SuperAnnotate je možné využít jen na neuronových sítích provozovaných na HW společnosti SuperAnnotate LLC. Pronájem NVIDIA T4 činí 0.92 USD za hodinu a u NVIDIA TESLA K80 1.58 USD za hodinu (ceny k 15.11.2020). Díky tomu je zde plná závislost na spolehlivosti, bezpečnosti a cenové politiky společnosti SuperAnnotate LLC.



Obrázek 2.1: Snímek obrazovky anotačního nástroje SuperAnnotate.

Polygon-RNN++[4]

Jedná se o další proprietární nástroj umožňující anotace i více tříd bez režimu iterací. Prostředí aplikace Polygon-RNN++ působí jednoduše, ale neintuitivně a nepropracovaně viz. obrázek 2.2. Neobsahuje také informační systém pro správu uživatelů a informace o jejich práci. Kód je k dispozici na vyžádání pro akademické účely. Komerční účely jsou řešeny individuálně.

[4] https://github.com/fidler-lab/polyrnn-pp-pytorch

Obrázek 2.2: Snímek obrazovky anotačního nástroje Polygon-RNN++.

Kapitola 3

Návrh systému

Systém je primárně určen do firemního prostředí kde anotování dat pro trénování neuronových sítí obvykle provádí více osob na více datových sadách. Informační systém umožní řízení přístupu těchto osob k datům, notifikace o dokončených anotacích správcům a jednoduché rozhraní pro učení a inferenci neuronových sítí. Obecně platí, nejen u anotování dat pro neuronové sítě, že lidský čas je cennější než strojový. Proto je systém navrhnut jako webová aplikace a umožňuje anotovat iterativním přístupem, kdy se část práce přenáší na neuronovou síť.

3.1 Iterativní trénování

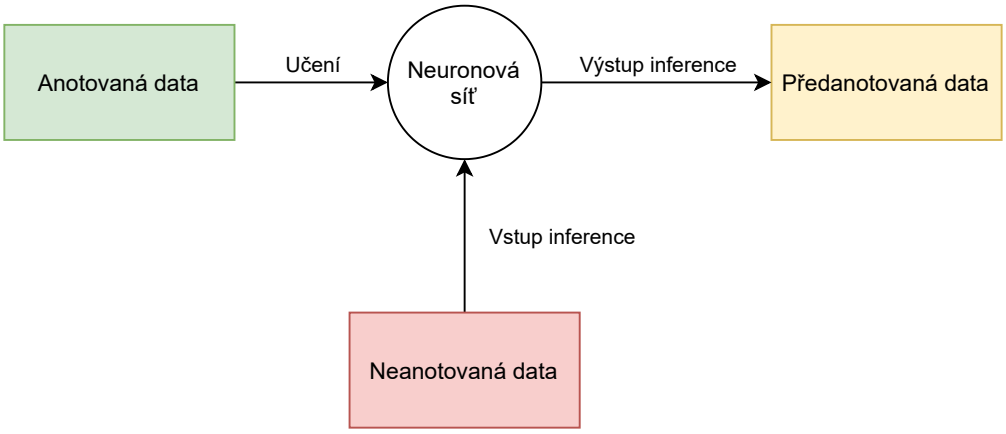
Účelem tohoto unikátního přístupu je zjednodušení práce anotujícím lidem. Neuronová síť produkuje pouze přibližné výsledky, dokud není dostatečně natrénována. Tyto výsledky však mohou být po korekci použitelné pro trénování. V případě, že je anotace špatná a je potřeba ji vytvořit znovu, může tato anotace sloužit alespoň jako vodítko, ukazující, kde přibližně se nějaký segment nachází a tím redukovat pravděpodobnost, že uživatel provádějící anotaci segment přehlédne.

První iterace pro nenatrénovanou neuronovou síť začíná anotací zadaného počtu neanotovaných prvků datové sady této neuronové sítě uživateli systému. Po dosažení limitu anotací jsou neuronové síti předána anotovaná data pro účely učení. Pro vstup inference jsou síti předána další neanotovaná data téže datové sady. Výstupem inference jsou masky pro tato neanotovaná data, která se tímto stávají předanotovanými a bude třeba na nich provést korekci. Graf tohoto procesu viz. 3.1

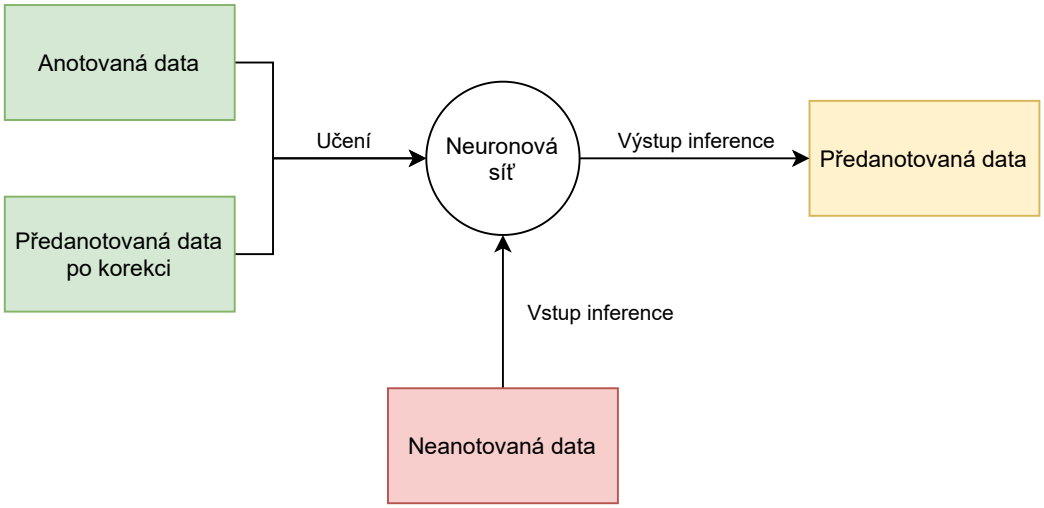
Po provedení první iterace uživatelé již neanotují neanotovaná data, ale provádí korekce předanotovaných dat čímž rozšiřují množinu anotovaných dat určených pro trénování neuronové sítě. Po dokončení korekcí na předanotovaných datech je neuronová síť opět natrénována pomocí všech dat z množiny anotovaných dat, která je nyní rozšířena o předanotovaná data po korekci viz. graf dalších iterací 3.2.

3.2 Použití systému

Tato podkapitola předkládá scénář použití systému společně s diagramem případů užití 3.3.



Obrázek 3.1: Graf průběhu první iterace.



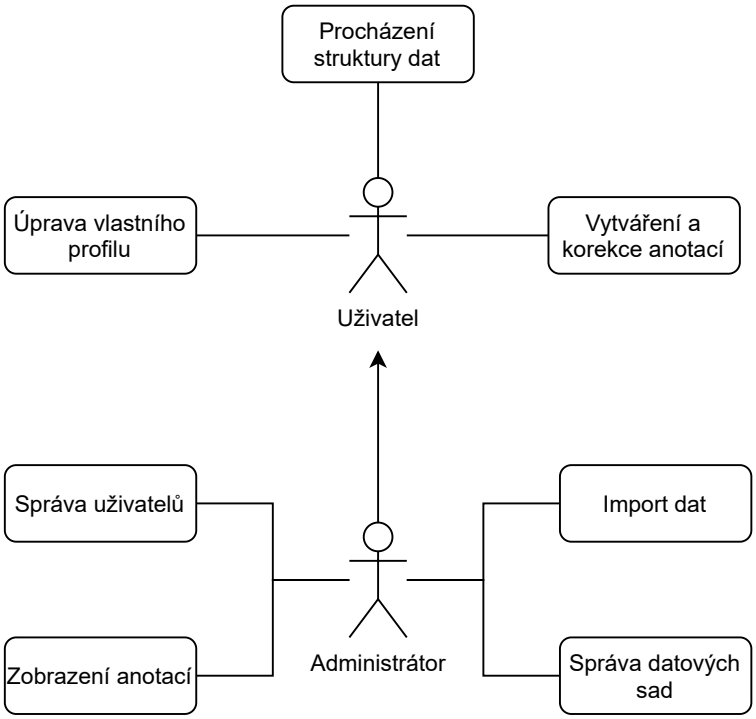
Obrázek 3.2: Graf průběhu dalších iterací.

Import dat

Pro importování dat slouží pokyn, který je zadáván společně s cestou k datovým sadám. Po dokončení importu je třeba notifikovat administrátora, který zadal pokyn k importu datových sad. Datové sady je možné libovolně zanořovat.

Vytváření anotací

Administrátor importovanou datovou sadu, případně více datových sad uvolní uživatelům k anotaci. Součástí uvolnění je limit anotací iteračního kroku systému. Uživatelé jsou schopni procházet strukturou datových sad a pro svou práci zvolit uvolněnou datovou sadu, která zároveň nedosáhla limitu anotací. Po zvolení jsou jim předkládány neanotovaná data vybrané datové sady až do dosažení jejího limitu. Ke každé dokončené anotaci je třeba uchovat informace o tom kdo ji vytvořil, kdy s anotací začal a kdy skončil, pro účely kontroly kvality anotace a zobrazení odpracovaného času.



Obrázek 3.3: Graf průběhu dalších iterací.

Učení a inference

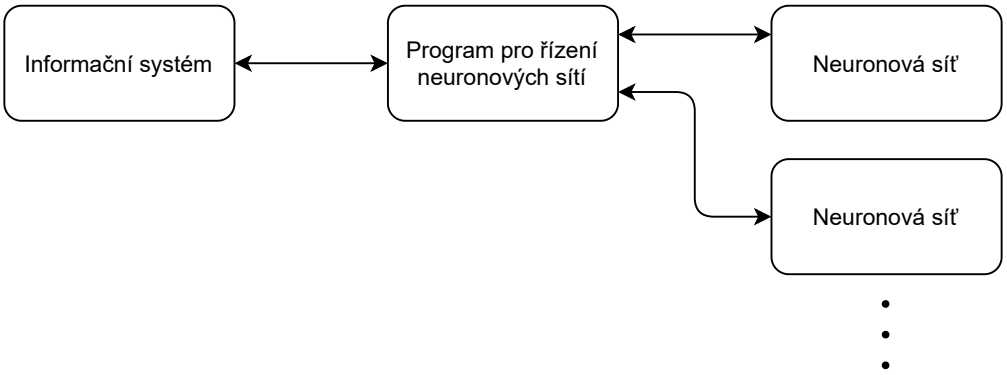
Po dosažení limitu anotací datové sady je všem administrátorům přihlášeným k této sadě odeslána notifikace. Administrátor prostřednictvím informačního systému dává pokyn pro učení a inferenci dané datové sady společně se zadaným počtem žádaných inferovaných dat. Tento požadavek je odeslán programu pro řízení neuronových sítí, který jej předá samotné neuronové síti.

Další iterace

Po dokončení práce neuronové sítě jsou administrátoři přihlášení ke zpracovávané datové sadě notifikováni. Administrátor opět uvolňuje datovou sadu pro anotace se zvoleným limitem. V této a další iteraci jsou uživatelům prioritně předkládána data vrácená inferencí neuronové sítě ke korekci.

3.3 Architektura systému

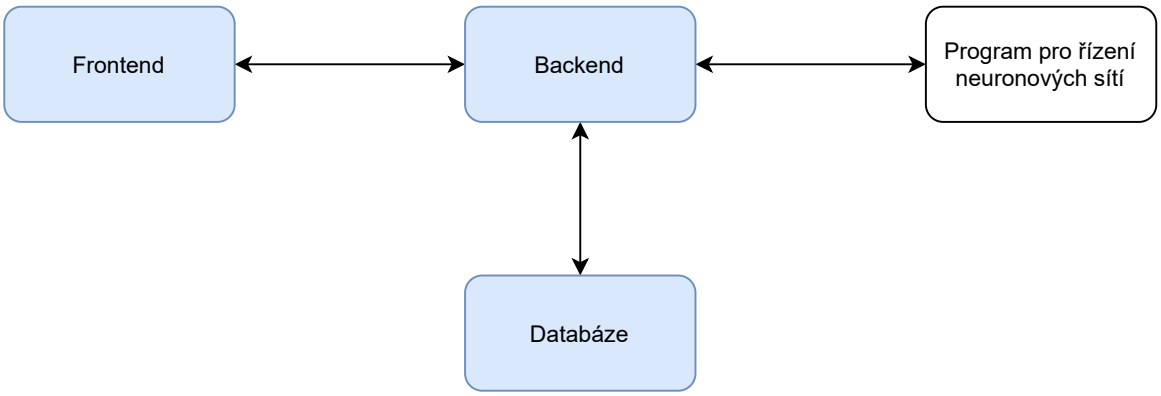
Základem návrhu řešení komplexního systému je popis jeho architektury, sloužící pro získání přehledu o jednotlivých jeho součástích a jejich vztahů. K vizualizaci slouží schéma 3.4. Systém se skládá z informačního systému, programu pro řízení neuronových sítí a samotných neuronových sítí. Informační systém obousměrně komunikuje s programem pro řízení neuronových sítí, přičemž odesílá pokyny k trénování a inferenci a přijímá informace o stavu provádění odeslaných pokynů. Program pro řízení neuronových sítí dle implementace tyto pokyny zpracovává a předává dále příslušným neuronovým sítím.



Obrázek 3.4: Schéma architektury celého systému.

Informační systém

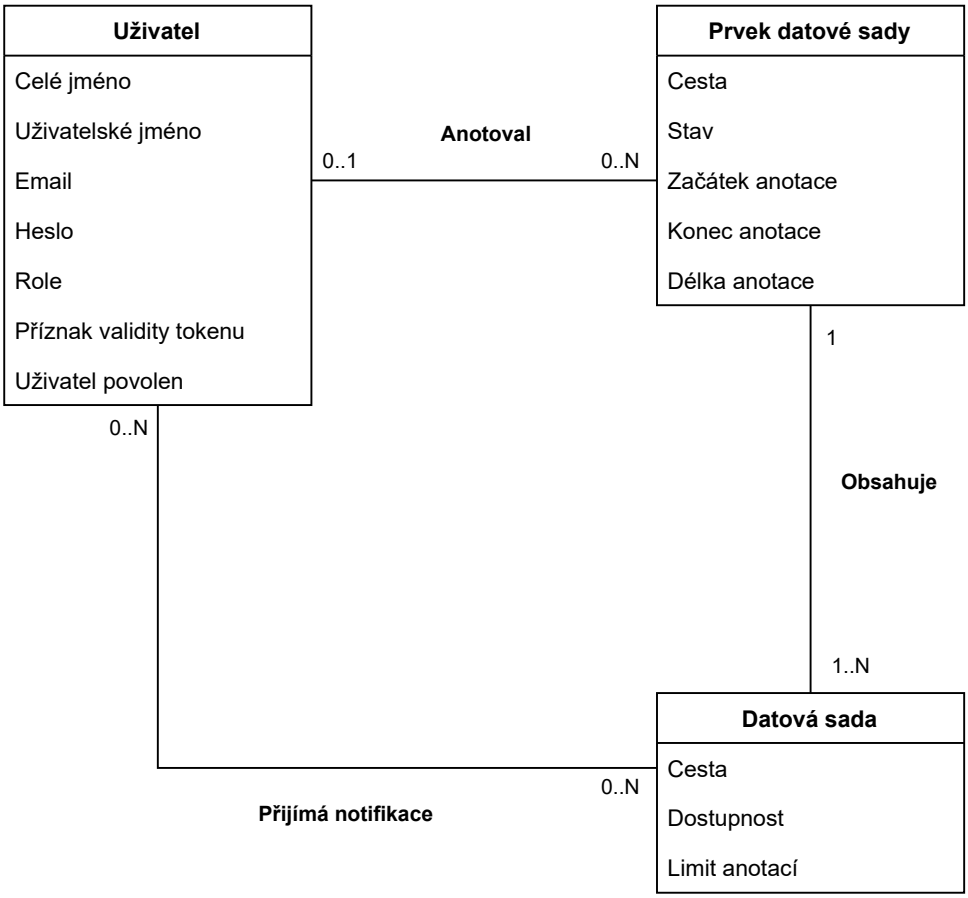
Architektura informační systému je navržena podle návrhového vzoru MVC, kdy model je reprezentován databází, view frontendem a controller backendem. Frontend informačního systému obousměrně komunikuje s backendem. Získává od něj data, která reprezentuje svým uživatelům a umožňuje spravovat datové sady a uživatele. Backend společně se získávanými záznamy z databáze řeší přístupová práva v komunikaci s frontendem a dále tvoří prostředníka pro přístup k programu pro řízení neuronových sítí. Schéma informačního systému viz. 3.5



Obrázek 3.5: Schéma architektury informační systému.

3.4 Datové struktury

Systém pracuje s uživateli, o kterých je třeba vést informace. Tyto informace jsou uloženy v databázi společně s metadaty o datových sadách a jejich prvcích. Kvůli jednoduchému přístupu k obrazovým datům prvků datových sad jsou tato data uložena v souborovém systému.



Obrázek 3.6: Schéma databáze.

Struktura databáze

Entita *uživatel* reprezentuje jednotlivé uživatele. O uživateli je evidováno jeho celé jméno, uživatelské jméno pro přihlášení, email pro zasílání notifikací, heslo, jeho role, zneplatnění vygenerovaného přístupového tokenu a příznak, zdali je uživatel povolen nebo zakázán. Uživatelé tvoří vazby s entitou *prvek datové sady* po vytvoření nebo korekci anotace tohoto prvku. Administrátoři mohou po přihlášení odběru notifikací datové sady vytvořit vazbu na entitu *datová sada*.

Datové sady jsou reprezentovány entitou *datová sada*, kdy tato entita tvoří vazby k jednotlivým prvkům entity *prvek datové sady*, vážící se k této datové sadě. O entitě *datová sada* je uchována její cesta v souborovém systému, zdali je dostupná pro anotace a aktuální limit pro anotace.

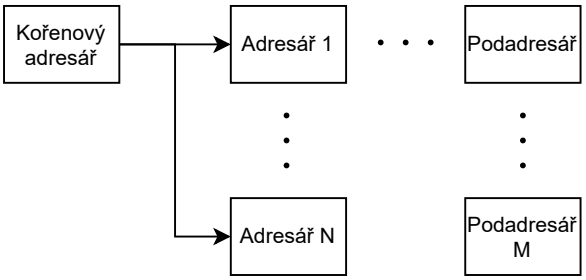
O jednotlivých prvcích datových sad jsou vedeny záznamy jako entity *prvek datové sady*, které evidují svou cestu v souborovém systému a stav, udávající, jestli anotace prvku nebyla vůbec provedena, byla provedena neuronovou sítí a je třeba ji zkontrolovat anebo je považována za dokončenou.

Struktura souborového systému

Pro jednoduchost přístupu k obrazovým datům jsou tato data uložena v souborovém systému backendu. Neuronovým sítím se pro učení předávají cesty v souborovém systému, což

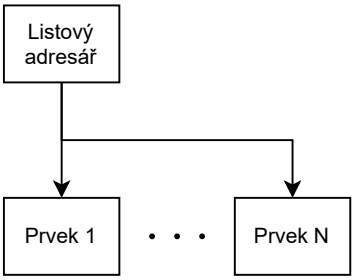
vytváří univerzální rozhraní, kdy není třeba aby neuronové sítě komunikovaly s databází. Je také možné data jednodušeji zobrazit a manipulovat s nimi.

Souborový systém zde nemá předepsanou hloubku zanoření a je možné jeho strom škálovat jak do šířky, tak do hloubky viz. 3.7.



Obrázek 3.7: Struktura adresářů.

List stromu souborového systému je považován jako adresář reprezentující data jedné datové sady. Adresář obsahuje jednotlivé prvky určené k anotaci viz. 3.8.

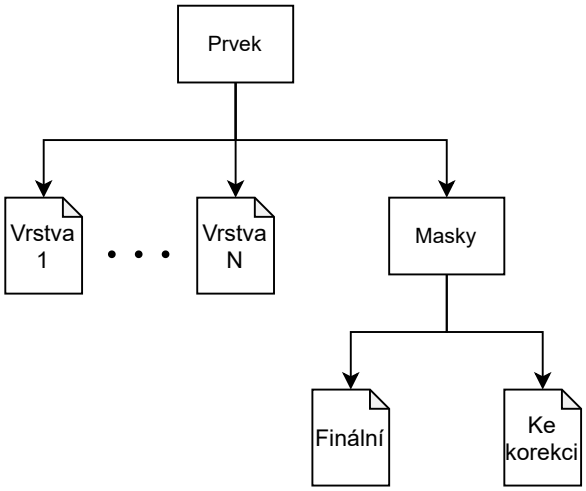


Obrázek 3.8: Struktura datové sady.

Adresář prvku obsahuje soubory jednotlivých vrstev sloužících jako podklady pro anotaci a adresář s maskami vytvořené uživatelem – finální, případně inferencí neuronové sítě – ke korekci viz. 3.9.

3.5 Anotační modul

Velmi důležitou součástí systému je anotační modul, se kterým bude pracováno po většinu času procesu anotace dat. Tento modul je určen pro práci uživatelů bez odborných znalostí. Vyžaduje proto jednoduchost a přívětivost. Návrh funkcí vychází z mého pozorování práce a konzultace s pracovníky vytvářejícími anotace v rastrovém grafickém editoru GIMP. Panel nástrojů obsahuje pro úpravu masky kreslící a mazací nástroj s nastavitelnou šířkou stopy. Velmi často byl při průzkumu pouze nakreslen obrys anotace a tento obrys byl následně vyplněn vyplňovacím nástrojem, který využívá flood-fill algoritmu, a proto je také součástí modulu. Pro rychlé vrácení nechtěného kroku slouží historie úprav, ve které se lze pohybovat vzad i vpřed. Hlavní část – plátno obsahuje pro přesné zakreslení anotace možnost přiblížení, oddálení a posuv vrstvy masky společně s vrstvou podkladovou. Výběr obrazových dat pro podkladovou vrstvu se provádí v panelu pro vrstvy, obsahující náhledy těchto dat. Anotační nástroj je součástí frontendu informačního systému.



Obrázek 3.9: Struktura prvku datové sady.

3.6 Řízení trénování a inference

Trénování a inference jsou klíčové pro uskutečnění iterativního anotování, a proto je možné tyto procesy ovládat přímo z aplikace. Pokyn pro trénování a inferenci zadává administrátor po splnění limitu datové sady v aktuální iteraci společně s žádaným počtem vrácených předanotovaných prvků datové sady. Po zpracování pokynu backendem jsou do datové sady předány cesty souborového systému v souboru formátu JSON:

```
{
    "train": {
        "masks": [
        ],
        "images": [
            [
            ]
        ]
    },
    "inference": {
        "images": [
            [
            ]
        ],
        "masks": [
        ]
    }
}
```

Objekt *train* obsahuje pole *masks* a *images*, kde jsou obsaženy cesty k maskám a pole cest k vrstvám obrazových dat. Cesty pro inferenci jsou obsaženy v objektu *inference*, kde pole *images*, obsahuje pole cest k vrstvám obrazových dat a cesty v poli *masks* pro výstup inferovaných masek.

Program pro řízení neuronových sítí

Samotné řízení neuronových sítí řeší program pro řízení neuronových sítí, které mohou mít rozdílná rozhraní. Tento program vzhledem k možné diverzitě rozhraní neuronových sítí počítá s implementací až při nasazení systému do produkce. Program přijímá od informačního systému zprávy vážící se k určité datové sadě společně s požadovaným počtem inferovaných dat pro zahájení trénování a inference. Tyto pokyny předává společně s daty z JSON souboru konkrétním neuronovým sítím. Dále je možné informačnímu systému předávat informace o průběhu aktuálně probíhajících operací.

Aplikační rozhraní

Příchozí zprávy

- Pokyn k iteraci

  Název události: iteration

  Argumenty události:

  1. dataset – cesta k datové sadě určené k trénování a inferenci
  2. userSid – zpětná vazba na uživatele o úspěšném zahájení provádění iterace

Odchozí zprávy

- Synchronizace s backendem

  Název události: sync

  Argumenty události:

  1. receivedIterations – { cesta k datové sadě: postup (0-100) }

- Oznámení začátku provádění iterace

  Název události: start

  Argumenty události:

  1. dataset – cesta k datové sadě
  2. sid – sid získané v příchozí události iteration

- Aktualizace postupu

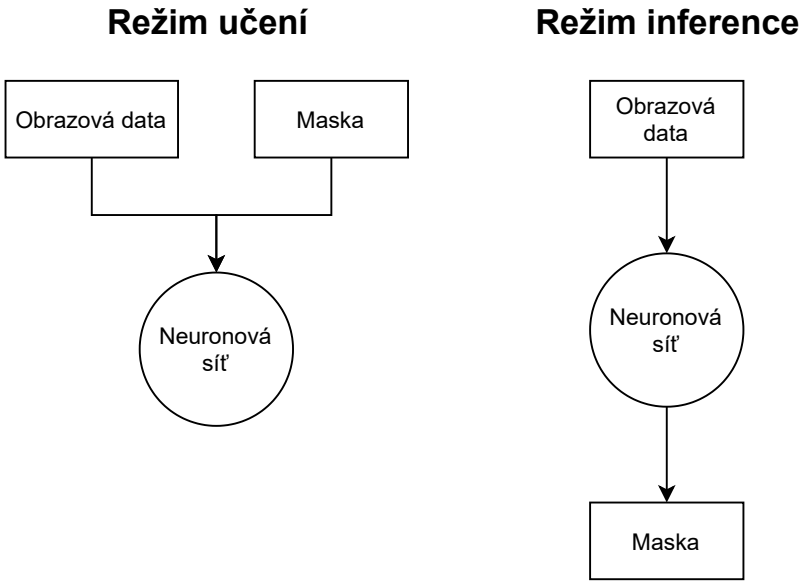
  Název události: update

  Argumenty události:

  1. dataset – cesta k datové sadě
  2. progress – postup (0-100)

3.7 Testovací neuronová síť

Pro prezentaci a vyzkoušení systému jako celku je třeba neuronové sítě. Neuronová síť je velmi jednoduchá a je počítáno s návrhem a implementací vlastního řešení při použití tohoto nástroje. Neuronová síť při učení přijímá obrazová data a k nim vážící se masky. Výsledkem učení je soubor obsahující váhy neuronů. V režimu trénování se váhy neuronů načtou do struktury neuronové sítě a jsou zpracována předložená obrazová data, ke kterým jsou generovány masky viz. 3.10.

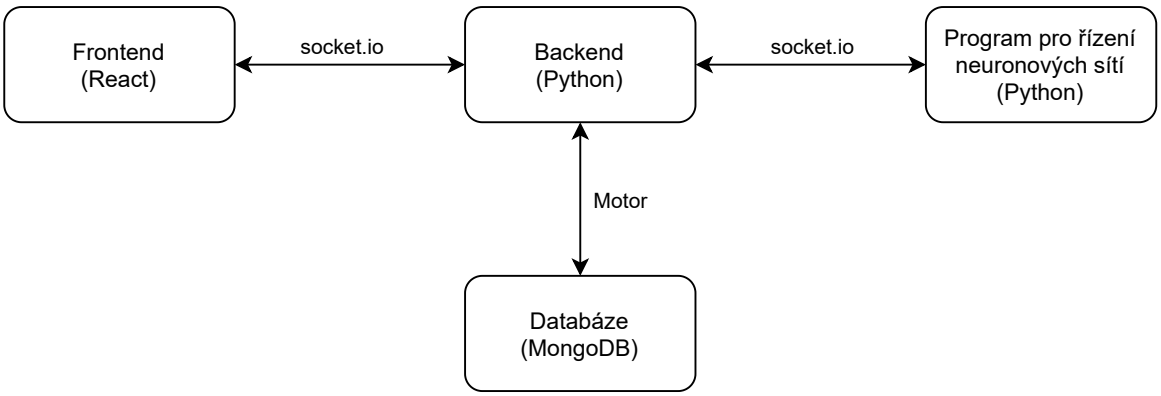


Obrázek 3.10: Vstupy a výstupy neuronové sítě.

Kapitola 4

Realizace systému

Tato kapitola se zabývá implementací systému, jeho sestavením a testováním. Frontend informačního je postaven na knihovně React jazyka JavaScript a komunikuje asynchronně s backendem pomocí knihovny socket.io založené na protokolu WebSocket. Implementační jazyk pro backend i program pro řízení neuronových sítí byl zvolen Python, protože je jazykem často využívaným frameworky pro práci s neuronovými sítěmi. Komunikace mezi oběma moduly je implementována také pomocí knihovny socket.io. Pro uložení metadat o datových sadách a datech o uživatelích slouží nerelační MongoDB databáze. Schéma technologií v návaznosti na části systému viz. 4.1



Obrázek 4.1: Schéma systému s použitými technologiemi.

4.1 Informační systém

Dle návrhu podle architektury MVC se informační systém skládá ze tří částí – frontend, backend a databáze, jejichž komunikace a technologie jsou zaznačeny do schématu viz. 4.1.

Frontend

Aplikace pro frontend využívá knihovny React. Pro pokročilé řízení stavů je použita knihovna React Redux, poskytující jeden globální store pro všechny komponenty. Design aplikace respektuje Material Design s využitím Material-UI frameworku. Navigace po celé aplikaci využívá URL s funkční historií (zpět, vpřed). Při každé akci čekající na odpověď od backendu je aplikace nastavena do načítacího režimu, kdy jsou tlačítka odesílající požadavky

zablokována a načítání je symbolizováno neurčitým lineárním prvkem postupu. Obecně po příchodu dat je načítací režim zrušen a v případě chyby je navíc chyba signalizována pomocí komponenty snackbar. Komponenty vyžadující data při svém vzniku odesílají zprávu backendu a do přijetí dat je aplikace v režimu načítání. V případě změny dat jsou uživatelům odesílány diferenciální data k pohledu, na kterém se právě nacházejí a tím vzniká okamžitá propagace změn v systému pro všechny připojené uživatele.

Připojování k backendu – komponenta Connecting

Po spuštění frontend aplikace se aplikace snaží opakovaně připojit k backendu pomocí socket.io rozhraní. Nenavázané připojení je indikováno neurčitým kruhovým prvkem postupu s textem. V případě výpadku spojení se opět vykreslí tato komponenta a aplikace se opět snaží připojit, dokud spojení není znovu navázáno.

Přihlášení uživatele – komponenta Login

Po připojení je modulem implementujícím zprávy socket.io rozhraní zkontrolována existence přihlašovacího tokenu v cookies a token je případně odeslán na validaci backendu. Pokud token není nalezen nebo je nevalidní aplikace vykreslí přihlašovací formulář, který po vyplnění odešle přihlašovací údaje backendu ke kontrole. Při úspěchu socket.io modul přijme přihlašovací token a uloží jej do cookies a do globálního store pro řízení stavů, kde ovlivňuje vykreslování aplikace dle uživatelských práv. V případě neúspěchu je příslušná chyba vypsána komponentou snackbar (neexistující uživatel, neplatné heslo nebo zakázaný učet).

Navigační panel – komponenta NavBar

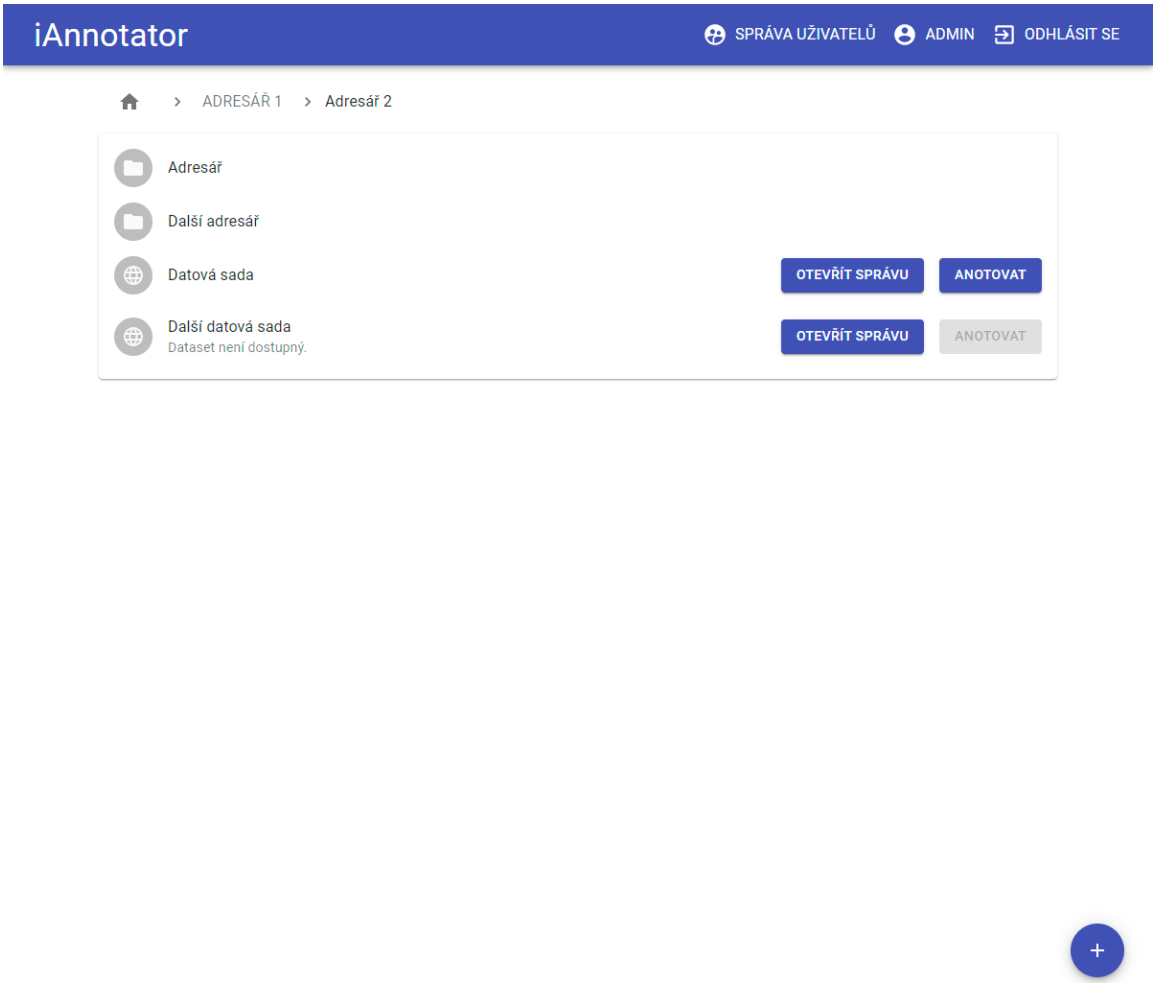
Tato komponenta reprezentující horní navigační lištu je pro autentizované uživatele vykreslována vždy. Obsahuje název projektu sloužící jako odkaz na domovskou stránku – procházení datových struktur, odkaz na správu vlastního profilu zobrazující se jako uživatelské jméno a tlačítko pro odhlášení, které maže uložený přihlašovací token z cookies a odesílá zprávu backendu. Po přijetí zprávy o úspěšném odhlášení od backendu jsou z důvodu bezpečnosti vymazána všechna přijatá data. Administrátorům se navíc oproti běžným uživatelům zobrazuje v této komponentě odkaz na správu uživatelů.

Prohlížeč datové struktury – komponenta ProjectSelect

Tato komponenta je zároveň domovskou stránkou aplikace. Po obdržení dat zobrazuje datovou strukturu podobným způsobem jako běžné prohlížeče souborového systému. Zobrazeny jsou tři typy záznamů – povolená datová sada, nepovolená datová sada a adresář umožňující zanoření po kliknutí. Mezi nepovolené sady se řadí i sady s dosaženým limitem nebo ty na kterých právě probíhá učení nebo inference. Datové sady obsahují odkaz na anotační nástroj implementovaný komponentou Annotate. V případě, že je datová sada nedostupná je tlačítko zakázané. Pro administrátory je u datových sad zobrazeno tlačítko odkazující na komponentu DatasetManage, která umožňuje datové sady spravovat. Komponenta ProjectSelect obsahuje drobečkovou navigaci pro snadný pohyb v datové struktuře.

K importu dat slouží plovoucí tlačítko, které je reprezentováno komponentou ImportData. Tato komponenta zobrazuje dialog umožňující zadat kořen dat v souborovém systému

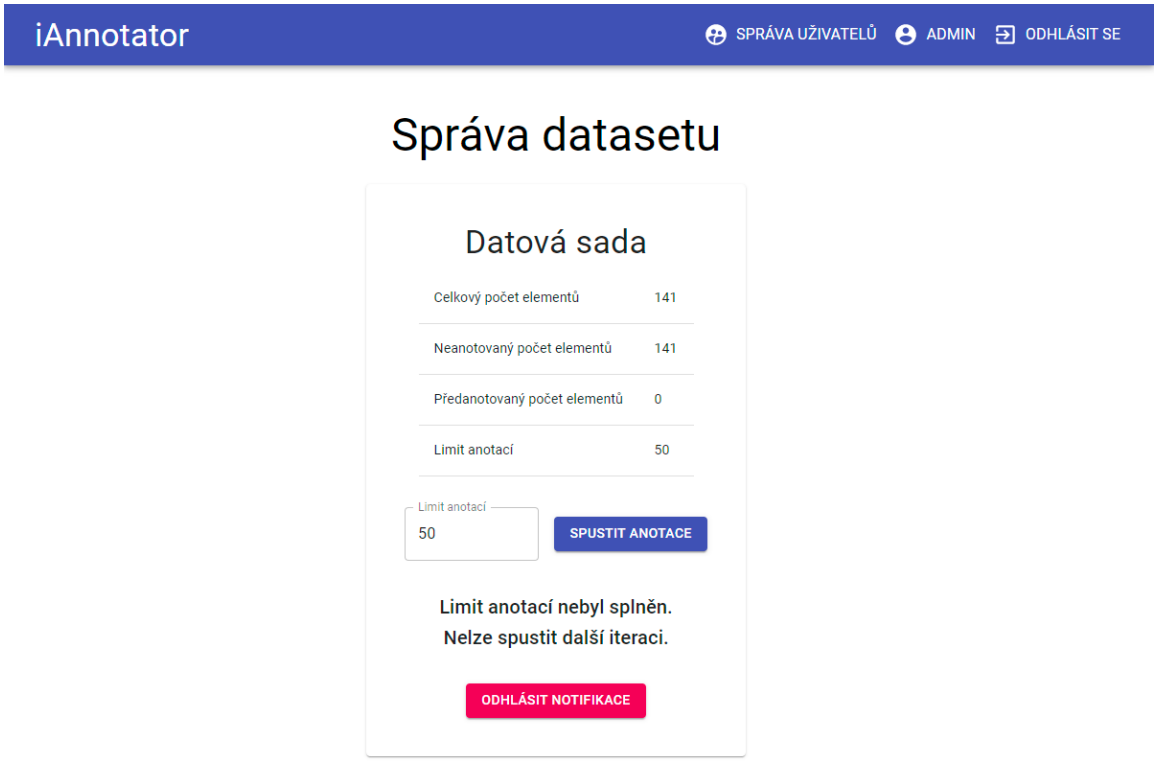
určených pro import. Po potvrzení cesty odesílá backendu zprávu obsahující zadanou cestu a aktuální cestu zobrazenou komponentou ProjectSelect. 4.2



Obrázek 4.2: Snímek obrazovky frontend komponenty ProjectSelect.

Správa datové sady – komponenta DatasetManage

Správa datové sady obsahuje informace o celkovém počtu prvků v datové sadě, počtu prvků podle jejich stavu a aktuálním limitu anotací datové sady. Komponenta umožňuje uvolnit datovou sadu k anotování se zadaným limitem v číselném vstupu, jehož hodnota se validuje zde a následně na backendu, přičemž nesmí být záporné a nesmí přesáhnout hodnotu součtu neanotovaných a předanotovaných prvků. Pokud je limit nulový, je možné odeslat příkaz pro spuštění trénování a inference se žádaným počtem vrácených anotací ke korekci, který nesmí být záporný ani větší než počet neanotovaných prvků. Při nenulovém limitu je vstup s tlačítkem nahrazen textem upozorňujícím na nesplněný limit. Pro zasílání informací o splnění limitu anotací a dokončení iterace je možné se přihlásit k notifikacím, případně se od nich odhlásit. 4.3



Obrázek 4.3: Snímek obrazovky frontend komponenty DatasetManage.

Anotační modul – komponenta Annotate

Anotační nástroj se skládá z komponent ToolBar, Canvas a Views, jejichž stavy mezi nimi zpřístupňuje komponenta Annotate. Nástrojový panel implementovaný komponentou ToolBar umožňuje pohyb v historii úprav, výběr kreslícího a mazacího nástroje s možností úpravy šířky tahu nebo plnícího nástroje – kyblíku a vymazání celé masky. V případě opuštění anotačního nástroje je odeslána zpráva backendu pro uvolnění prvku datové sady jiným uživatelům. Tlačítko pro zrušení anotace je odkazem na zobrazení struktury datových sad a slouží ke zvýšení uživatelské přívětivosti. Po dokončení anotace tlačítkem „dokončit anotaci“ se obrazová data masky odesílají společně s cestou k prvku datové sady backendu.

Kreslící plátno je obsaženo v komponentě Canvas společně s vrstvou obrazových dat a masky. Plátno se chová responzivně vzhledem k velikosti okna prohlížeče. Pomocí scrollování je možné přibližovat a oddalovat vrstvy podle daných omezení: 0.1x až 200x. Plátno umožňuje pravým tlačítkem myši přesouvat obrazovou vrstvu zároveň s maskou, přičemž je využita funkce pro omezení pozicování mimo okraje. Pokud vrstvy zabírají celý prostor plátna a více je posun omezen polovinou rozměrů plátna, v opačném případě polovinou rozměrů vrstev. Posuv a navázání funkce pro omezení řeší externí knihovna Konva. Pro plnící nástroj je implementován záplavový algoritmus.

Výběr podkladové vrstvy pro masku lze provést v komponentě Views, která zobrazuje náhledy s názvy vrstev aktuálně anotovaného prvku datové sady s barevnou indikací právě aktivní vrstvy. Změna vrstvy nijak neovlivňuje masku, pozici vrstev, ani jejich úroveň přiblížení. 4.4

Správa uživatelů – komponenta Users

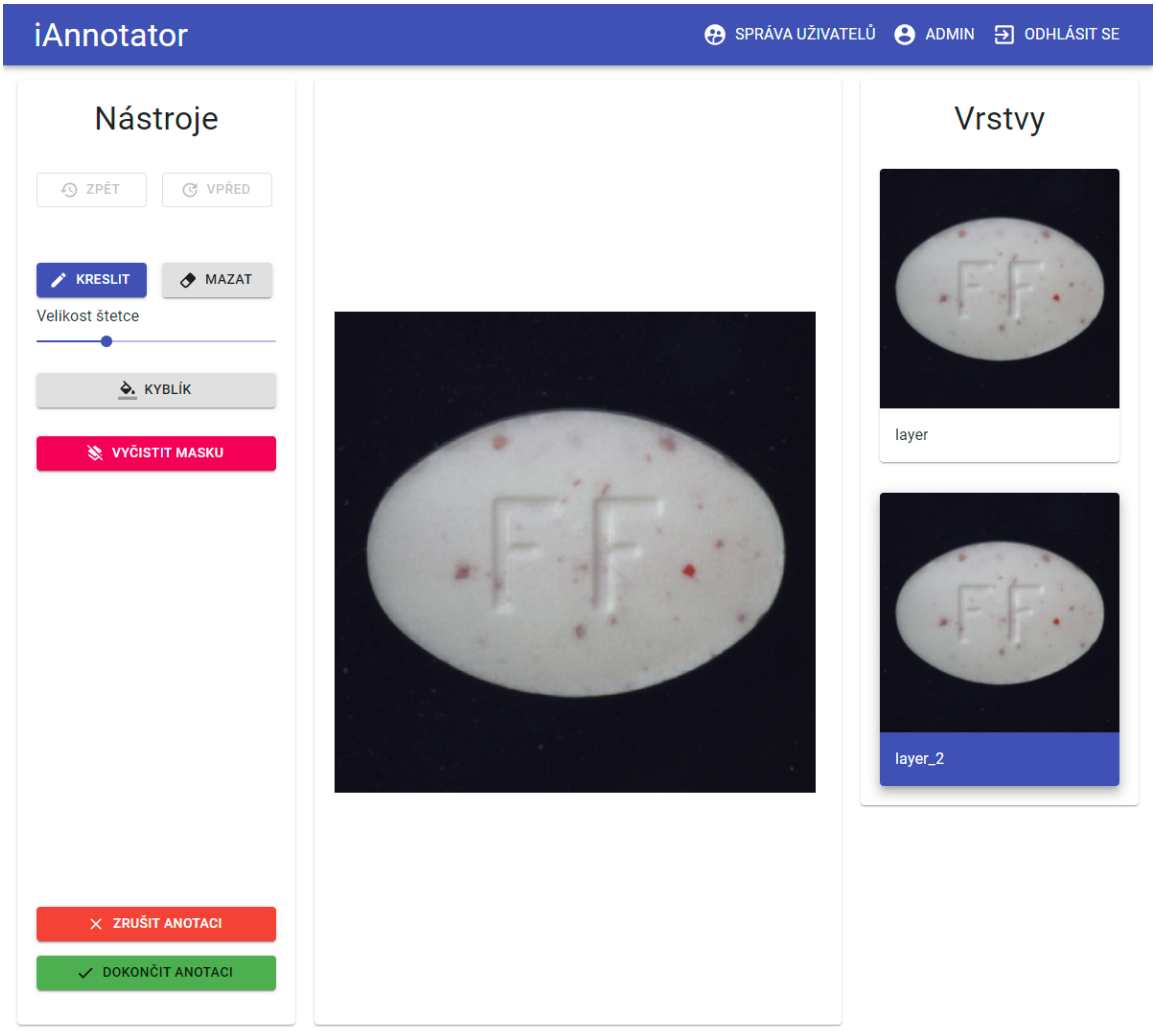
Správu uživatelů implementuje komponenta Users. Zobrazuje seznam rozbalitelných položek reprezentujících jednotlivé uživatele. Položky obsahují uživatelské jméno, celé jméno, roli uživatele a odkaz na zobrazení historie anotací daného uživatele implementovanou komponentou AnnotationHistory. Po rozbalení položky je možné editovat celé jméno, uživatelské jméno, email, heslo a roli uživatele. Z rozbalené nabídky lze uživatelský účet zakázat nebo povolit. Pro vytváření uživatelů slouží dialog komponenty CreateUser vyvolaný plovoucím tlačítkem. 4.5

Historie anotací uživatele – komponenta AnnotationHistory

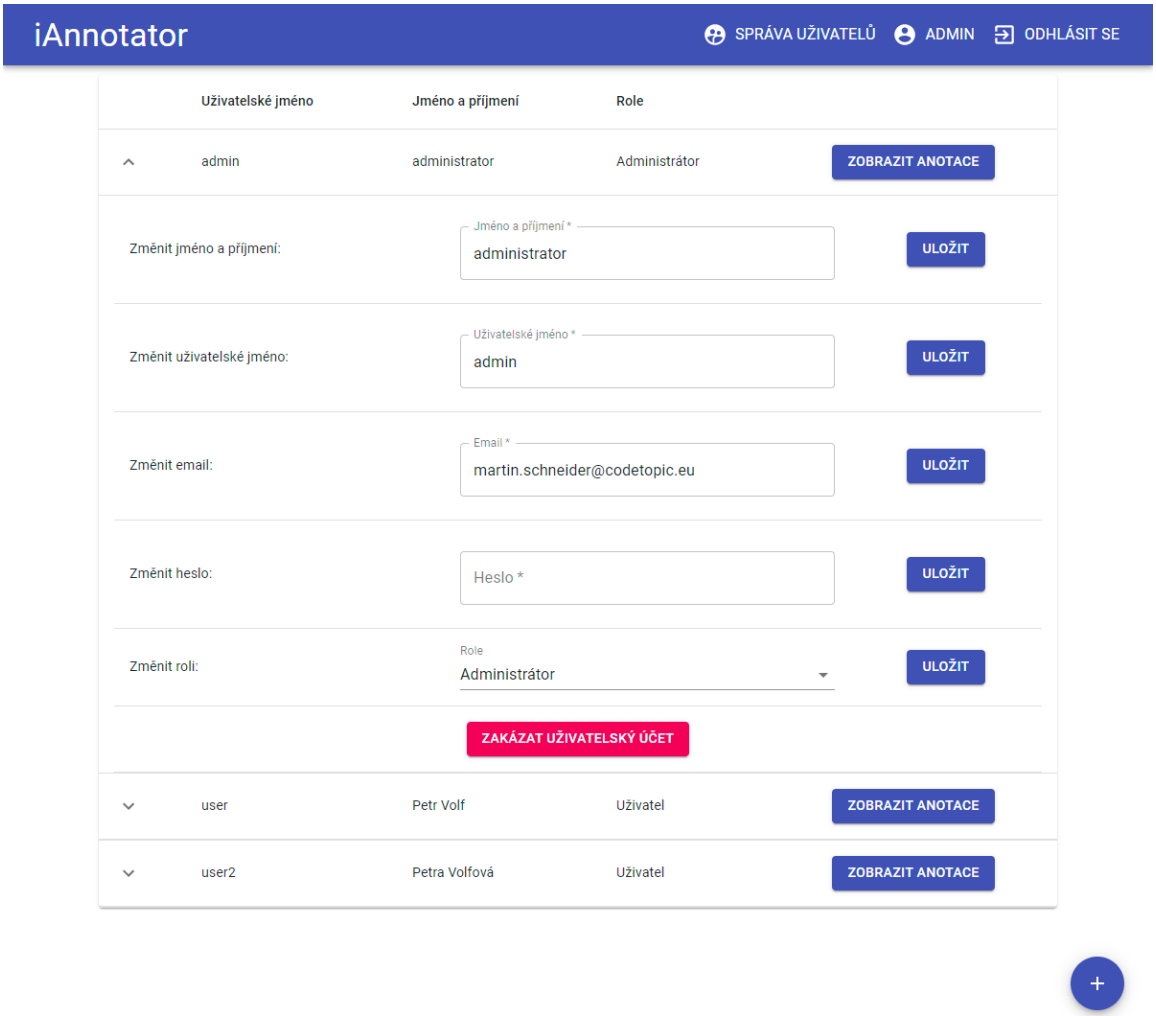
Komponenta zobrazuje historii anotací určitého uživatele. Pomocí dvou vstupů pro datum, lze filtrovat anotace v zadaném intervalu z něhož se vypočte celkový započtený čas. Vstupy po požadavku na filtrování podléhají validaci, zdali se nepřekrývají. Pro náhled anotace slouží tlačítko „zobrazit“ vyvolávající dialog komponenty AnnotationViewer. 4.6

Prohlížeč anotace na podkladových vrstvách – komponenta AnnotationViewer

Náhledy uživatelských anotací se zobrazují v dialogu komponenty AnnotationViewer vyvolaném komponentou AnnotationHistory. Dialog zobrazuje podkladovou vrstvu s maskou. Pokud existuje více podkladových vrstev, lze je přepínat šipkami po stranách nebo navigací pod vrstvami. K implementaci přepínání vrstev je využita knihovna react-material-ui-carousel. 4.7

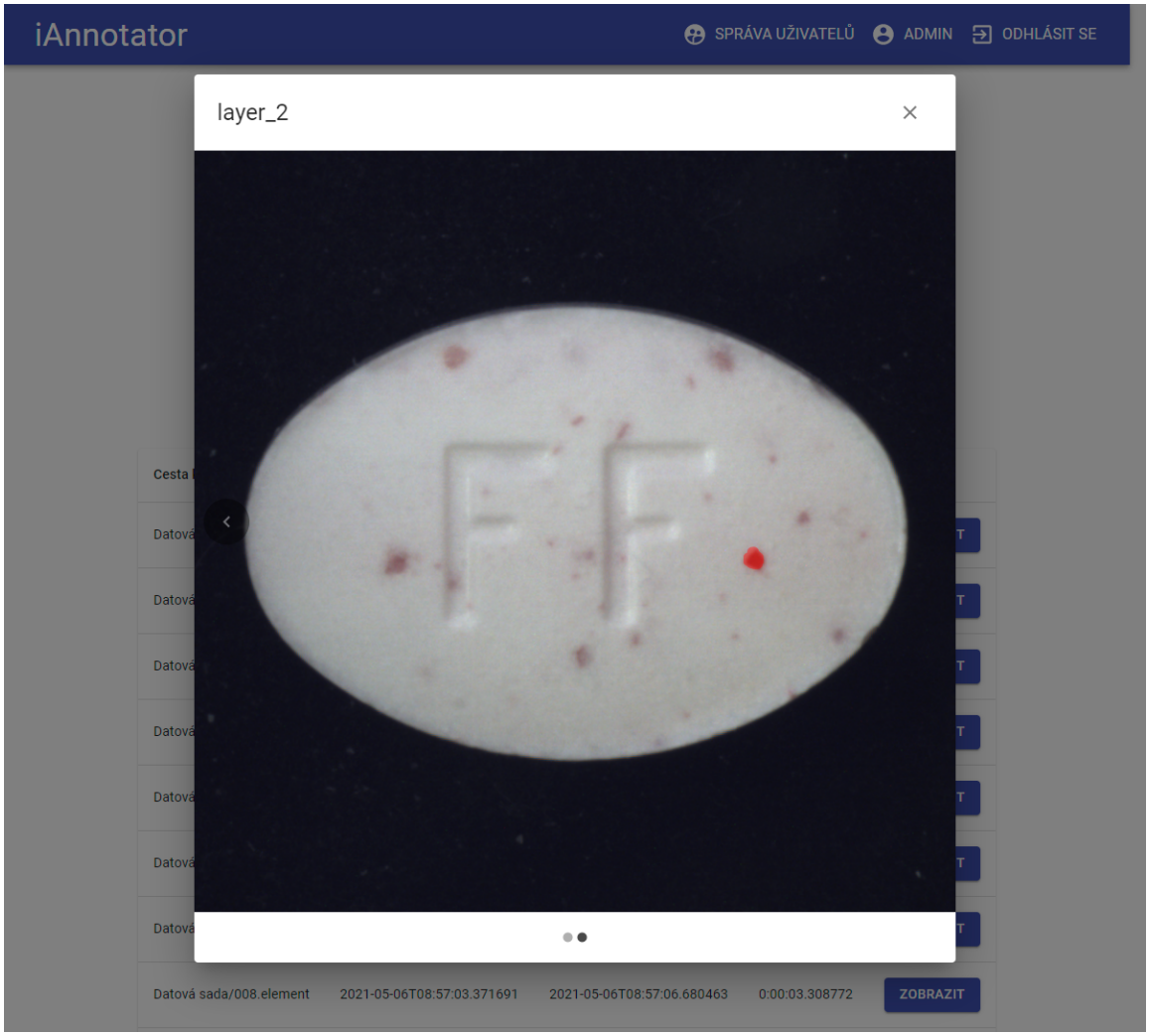


Obrázek 4.4: Snímek obrazovky frontend komponenty Annotate.



Obrázek 4.5: Snímek obrazovky frontend komponenty Users.

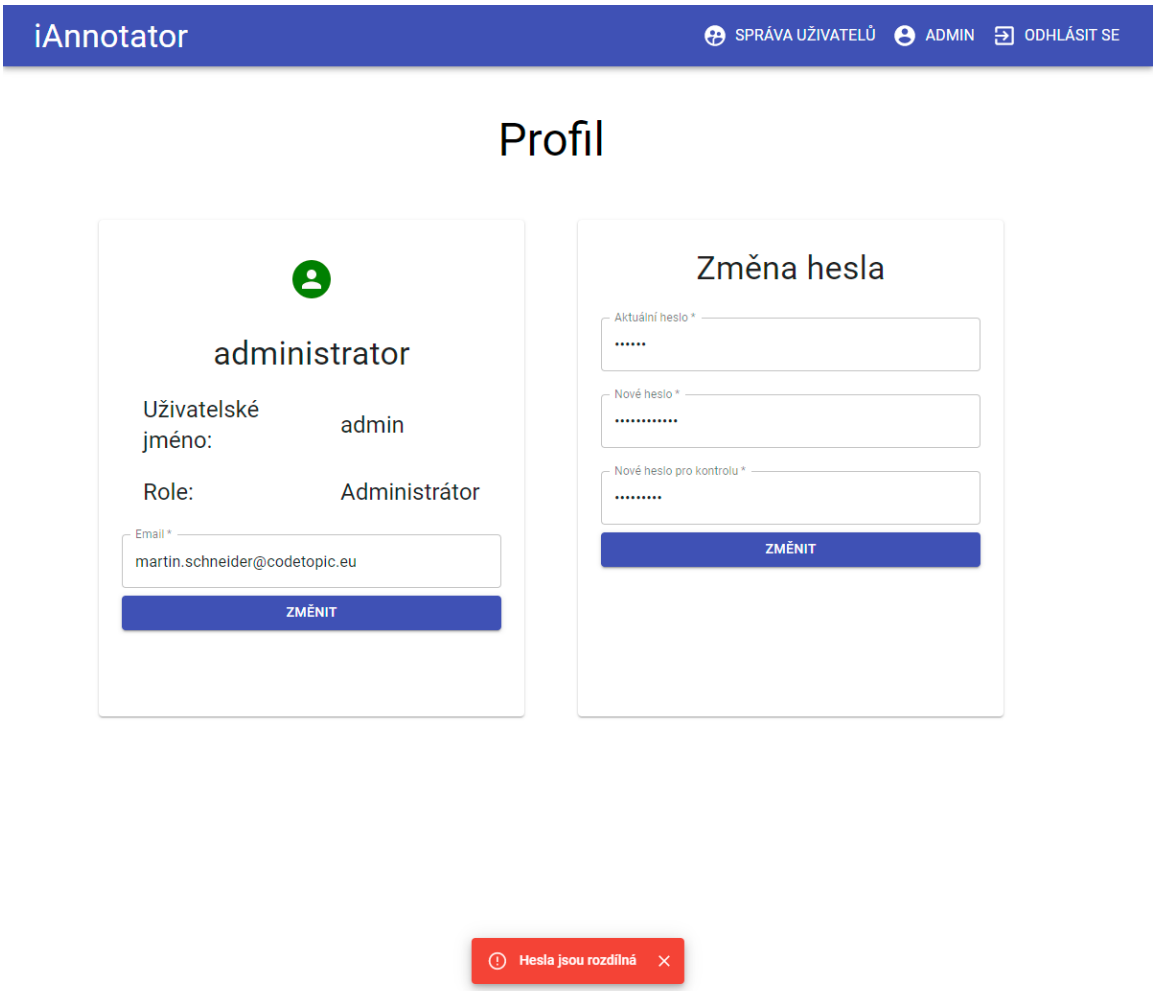
Obrázek 4.6: Snímek obrazovky frontend komponenty AnnotationHistory.



Obrázek 4.7: Snímek obrazovky frontend komponenty AnnotationViewer.

Zobrazení a editace profilu – komponenta Profile

Pro správu vlastního profilu slouží komponenta Profile, kde je možné zobrazit si informace o svém uživatelském účtu a měnit svůj email a heslo. Nové heslo je třeba zadat dvakrát stejné. 4.8



Obrázek 4.8: Snímek obrazovky frontend komponenty Profile.

Backend

Backend systému je psán v jazyce Python, který je zvolen pro zachování konzistence s oblastí neuronových sítí, kdy významné frameworky jako TensorFlow a PyTorch mají největší podporu právě pro Python. Všechny příchozí požadavky od frontendu a odchozí požadavky na data databáze a programu pro řízení neuronových sítí jsou obsluhovány asynchronně knihovnou socket.io s podporou knihovny aiohttp. Pro komunikaci s databází je využita knihovna Motor, umožňující asynchronní komunikací s MongoDB databází. Pro převod obrázků mezi uloženým formátem ve stupních šedi a pro zobrazení na frontendu jsou využity funkce knihovny Pillow.

Modul pro import dat

Tento modul slouží k importu dat na odděleném vlákně procesu. Prochází zadanou strukturu dat a kopíruje data do zadaného místa. Zápis metadat do databáze provádí po každém prvku datové sady až po jeho úspěšném překopírování pro zachování integrity dat. Do záznamu o datové sadě zapisuje jako odběratele notifikací uživatele, který požadavek na import zadal. Po dokončení importu dat modul volá funkci modulu notifications, pro odeslání notifikace o dokončení importu dat.

Struktura importovaných dat

Adresáře lze zanořovat a za datovou sadu je označen adresář obsahující adresáře s příponou *.element*, které reprezentují jednotlivé prvky datové sady. Prvky datových sad obsahují svoje vrstvy jako obrázkové soubory.

Modul pro notifikace

Pro notifikace je určen modul notifications. Implementace odesílání emailů, případně jiného způsobu není v tomto řešení obsažena a modul notifikace pouze vypisuje na standardní výstup.

Komunikace s programem pro řízení neuronových sítí

Backend v intervalech zadaných proměnnou controllerReconnect opakuje připojení k socket.io rozhraní programu pro řízení neuronových sítí, dokud není připojen. Programu předává pokyn pro zahájení trénování a inference a získává od něj informace o průběhu těchto operací, který backend ukládá v proměnné iterations. S pokynem předávaným programu pro řízení neuronových sítí je vygenerován soubor *.instructions.json* dle návrhu struktury 3.6 do složky předávané datové sady.

Komunikace s frontendem

Po připojení klienta pomocí rozhraní socket.io se údaje o spojení ukládají do datové struktury typu dictionary, kde klíčem je identifikační číslo spojení.

Dále je očekávána autorizace buď prostřednictvím tokenu nebo přihlašovacích údajů. Z hesla zadaného uživatelem je vygenerován hash algoritmem bcrypt. V případě úspěšné autorizace je vytvořen objekt pro uživatele, který je vložen do údajů o spojení. Bez tohoto objektu a příslušných práv v něm není možné vykonávat další požadavky. Uživateli je zaslán token, buď vrácený jím odeslaný validní nebo nově vygenerovaný v případě přihlášení

přihlašovacími údaji. Token je ve formátu JWT — JSON Web Tokens a manipulace s ním je řešena prostřednictvím knihovny PyJWT. Z bezpečnostních důvodů je tokenům nastavována expirace, kterou je možné v programu změnou proměnné tokenExpiration změnit. Token lze zneplatnit zápisem do databázového záznamu uživatele.

Při klientově žádosti o data konkrétního pohledu je do objektu uživatele v údajích o spojení nastavena místnost odpovídající skupině uživatelů požadujících data z tohoto pohledu a příslušná data jsou odeslána. Jakákoliv změna těchto dat způsobí odeslání diferenciálních dat všem uživatelům skupiny, které se změna týká.

Každý požadavek prochází kontrolou oprávnění a je ověřována validita vstupu. Při selhání kontrol je odeslána chybová hláška. Při manipulaci s databází v případě vrácené chyby signalizované výjimkou dekorátor obalující každý požadavek výjimku odchytí a odesílá frontendu chybu.

V případě manipulace s uživatelem prostřednictvím správy uživatelů je tento uživatel odhlášen s požadavkem na vymazání všech jeho lokálních dat a jeho token je v databázi zneplatněn. Server kontroluje v objektu uživatele, zdali byl uživatel připojen do nějaké místnosti a případně jej odpojuje. Objekt uživatele je nakonec vymazán. Stejného postupu využívá i odhlášení vyvolané samotným uživatelem.

Databáze

Jako databázový systém slouží kontejner z oficiálního mongoDB obrazu repozitářové služby Docker Hub[1].

[1] https://hub.docker.com/

Struktura databáze

Konkrétní struktura databáze vychází z návrhu databáze viz. 3.4

```
users = [
    {
        _id: identifikátor (ObjectID),
        username: uživatelské jméno (string),
        fullname: celé jméno (string),
        email: emailová adresa (string),
        password: hash hesla (string),
        role: role uživatele (string),
        tokenInvalid: platnost přihlašovacího tokenu (bool),
        disabled: dostupnost uživatelského účtu (bool)
    }
]
datasets = [
    {
        _id: identifikátor (ObjectID),
        path: cesta k datové sadě (string),
        available: dostupnost datové sady (bool),
        limit: limit anotací (unsigned integer),
        elements: [
            cesty k prvkům datové sady (string)
        ],
        subscribed: [
            id uživatelů přihlášených k notifikacím (ObjectID)
        ]
    }
]

elements = [
    {
        _id: identifikátor (ObjectID),
        user: id uživatele, který provádí nebo provedl anotaci (ObjectID),
        path: cesta k prvku (string),
        state: stav prvku - neanotovaný, předanotovaný, anotovaný (string),
        from: začátek anotace (string),
        to: konec anotace (string),
        time: délka trvání anotace (string)
    }
]
```

4.2 Program pro řízení neuronových sítí

Pro možnost přímého spouštění neuronové sítě implementované pomocí TensorFlow je program implementován jazykem Python, který je plně podporován tímto frameworkem. Program komunikuje s backendem pomocí knihovny socket.io s využitím knihovny aiohttp.

Podrobnosti komunikace programu s backendem viz. 4.1. Navázaná testovací neuronová síť je importována jako modul a jsou přímo volány její funkce.

4.3 Testovací neuronová síť

Neuronová síť využívá frameworku TensorFlow pro Python. Pro demonstrační účely je implementována pouze jednoduchá síť architektury U-Net. Optimalizace neuronové sítě probíhala na datových sadách publikací [3, 2], dostupných volně ke stažení na webových stránkách společnosti MVTec[2]. Datové sady jsou však malé a testovací neuronová síť nedopracovaná, proto dosahované výsledky nejsou přesné a obsahují šum.

Struktura neuronové sítě

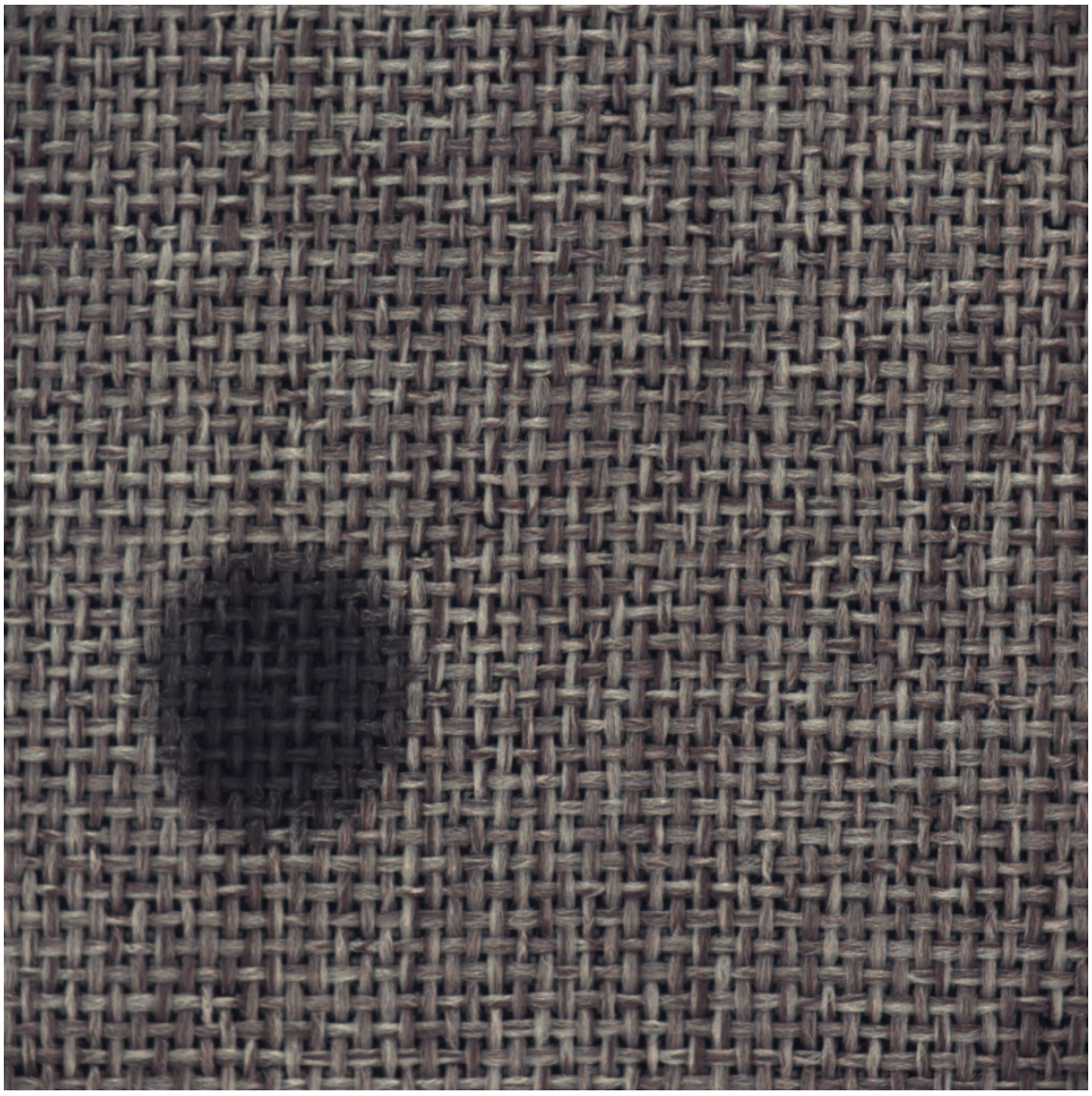
Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 600, 600, 3)]	0
conv2d (Conv2D)	(None, 600, 600, 8)	224
conv2d_1 (Conv2D)	(None, 600, 600, 8)	584
max_pooling2d (MaxPooling2D)	(None, 300, 300, 8)	0
conv2d_2 (Conv2D)	(None, 300, 300, 16)	1168
conv2d_3 (Conv2D)	(None, 300, 300, 16)	2320
max_pooling2d_1 (MaxPooling2	(None, 150, 150, 16)	0
conv2d_4 (Conv2D)	(None, 150, 150, 32)	4640
conv2d_5 (Conv2D)	(None, 150, 150, 32)	9248
max_pooling2d_2 (MaxPooling2	(None, 75, 75, 32)	0
conv2d_6 (Conv2D)	(None, 75, 75, 64)	18496
conv2d_7 (Conv2D)	(None, 75, 75, 64)	36928
conv2d_8 (Conv2D)	(None, 75, 75, 32)	18464
conv2d_9 (Conv2D)	(None, 75, 75, 32)	9248
conv2d_transpose (Conv2DTran	(None, 150, 150, 32)	9248

[2] https://www.mvtec.com/company/research/datasets/mvtec-ad

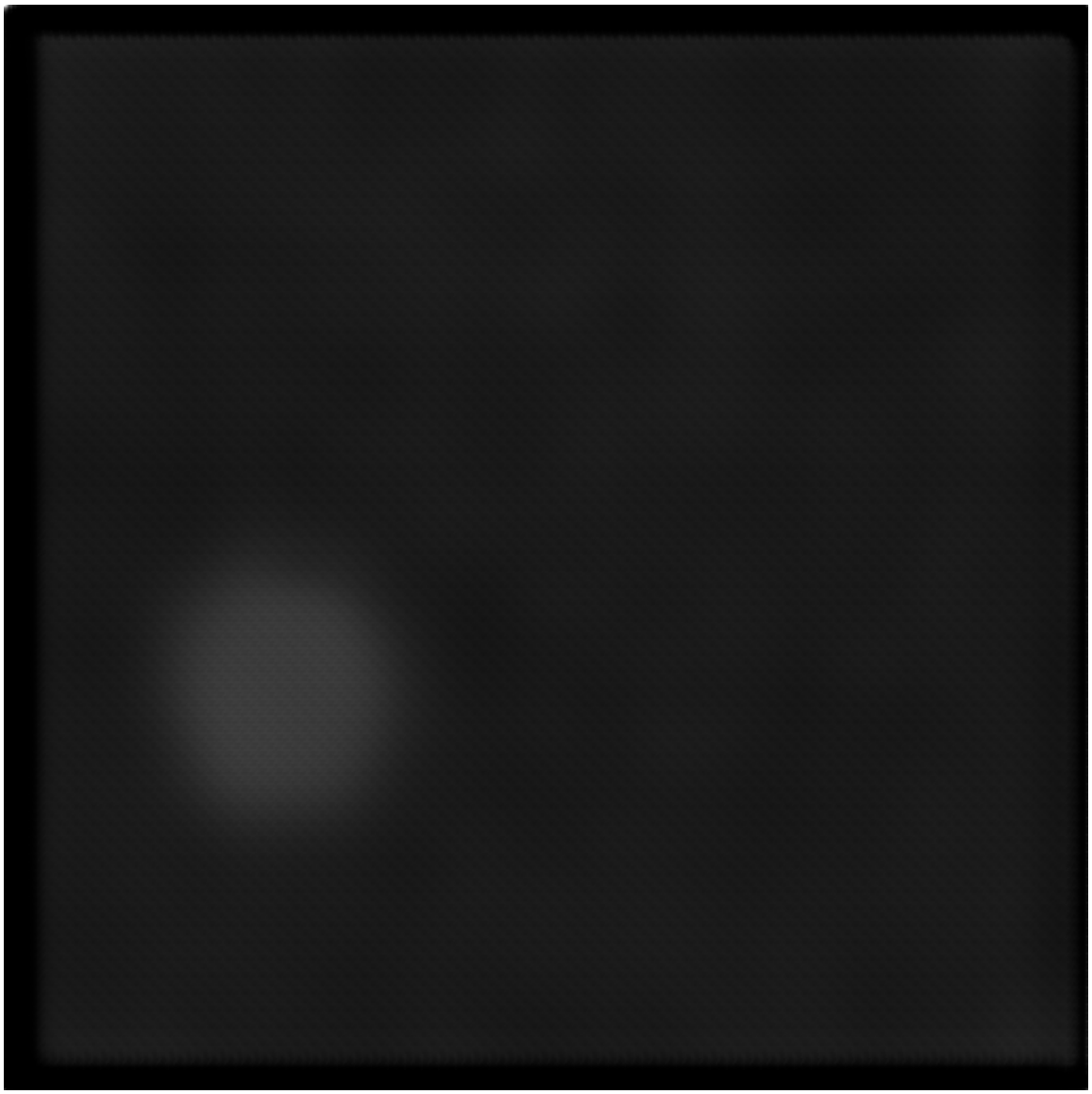
```
conv2d_10 (Conv2D)           (None, 150, 150, 16)      4624
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 150, 150, 16)      2320
_________________________________________________________________
conv2d_transpose_1 (Conv2DTr (None, 300, 300, 16)      2320
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 300, 300, 8)       1160
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 300, 300, 8)       584
_________________________________________________________________
conv2d_transpose_2 (Conv2DTr (None, 600, 600, 8)       584
_________________________________________________________________
conv2d_14 (Conv2D)           (None, 600, 600, 8)       584
_________________________________________________________________
conv2d_15 (Conv2D)           (None, 600, 600, 8)       584
_________________________________________________________________
conv2d_16 (Conv2D)           (None, 600, 600, 1)       73
=================================================================
Total params: 123,401
Trainable params: 123,401
Non-trainable params: 0
```

Výsledky neuronové sítě

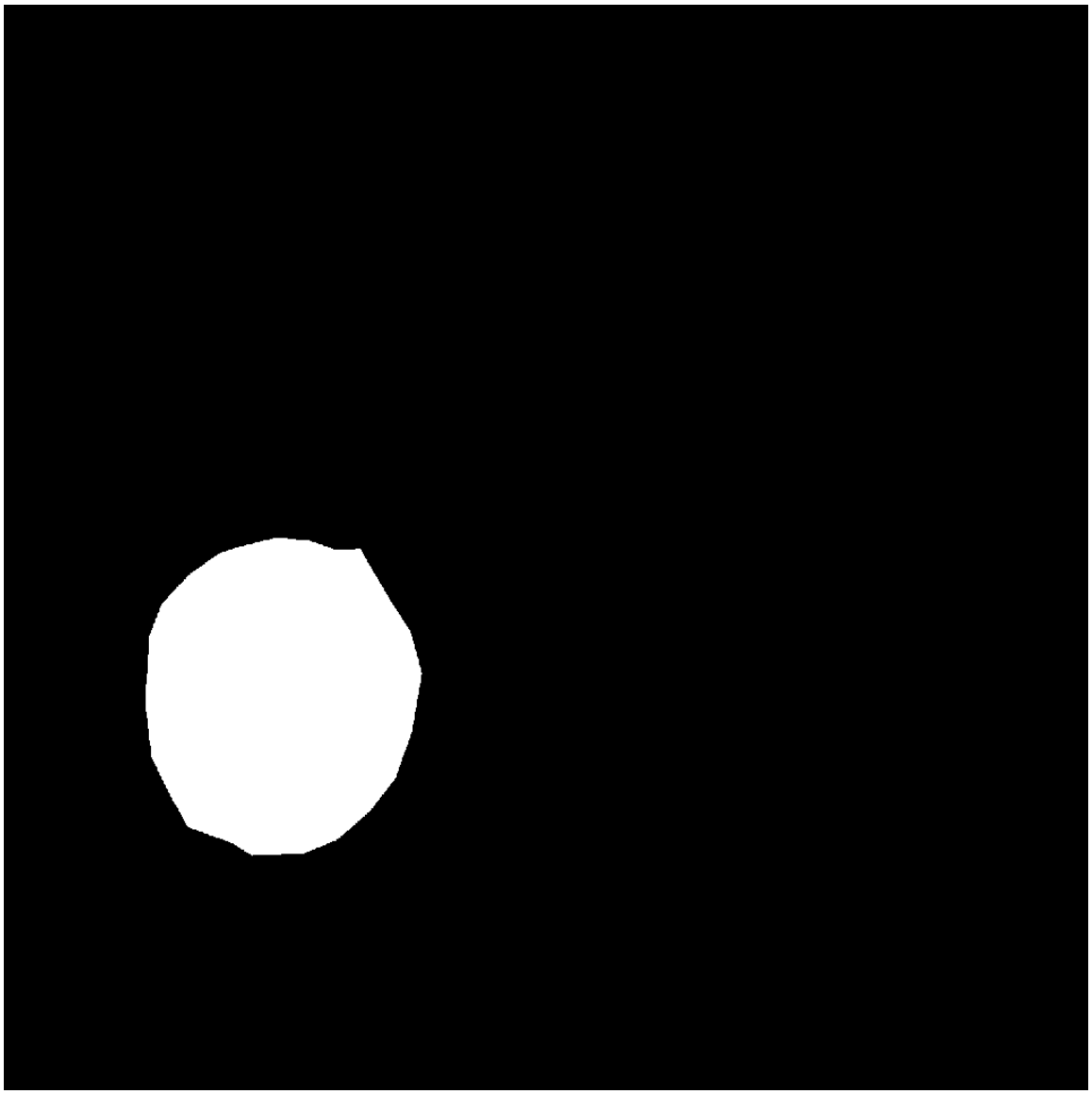
Díky jednoduché implementaci a nutnosti škálovat vstupní data kvůli nedostatku paměti je výstup neuronové sítě přibližný a obsahuje velkou míru šumu. Pro snížení nároků na paměť lze například využít takzvaný *patching*, kdy obrazová data jsou rozkládána na menší části. Vstupními daty pro neuronovou síť byla datová sada *carpet*, ze kterého byl využit i prvek viz. 4.9 pro demonstraci inference viz. 4.10. Správný výstup z této datové sady viz. 4.11.



Obrázek 4.9: Vstup testovací neuronové sítě.



Obrázek 4.10: Výstup testovací neuronové sítě.



Obrázek 4.11: Předpokládaný výstup testovací neuronové sítě.

4.4 Sestavení a spuštění

Pro jednoduchost produkčního nasazení jsou frontend i databáze kontejnerizovány technologií Docker. Obraz mongo pro MongoDB databázi je veřejně dostupným obrazem na síti Docker Hub. Frontend je sestaven s pomocí obrazu node obsahujícího knihovnu Node.js a následně je zpřístupněn technologií Nginx v obrazu nginx, s konfigurací přizpůsobenou pro React aplikaci. Backend a program pro řízení neuronových sítí nejsou kontejnerizovány z důvodu zachování multiplatformního přístupu do souborového systému.

Pro sestavení a spuštění frontend kontejneru společně s databázovým kontejnerem lze z kořenové složky projektu volat příkaz:

```
docker-compose up
```

Backend spouští Python skript *server.py* ve složce */backend*. Prvním argumentem programu je umístění adresáře obsahující datovou strukturu a druhý volitelný argument *init* vytváří v databázi uživatele *admin* s heslem *admin*.

Program pro řízení neuronových je obsahem Python skriptu *controller.py* ve složce */controller*. Jediným argumentem programu je umístění adresáře obsahující datovou strukturu.

Testovací neuronová síť je modulem importovaným v programu pro řízení neuronových sítí, který jej spouští jako další vlákno.

Pro úspěšné spuštění Python programů je nutné nainstalovat balíčky v souboru *requirements.txt* , který každý ze skriptů obsahuje ve svém adresáři.

Neuronová síť byla testována interpretem Python 3.8.8 s frameworkem TensorFlow[3] 2.4.1 s grafickou kartou NVIDIA GTX 1660 Ti 6GB s knihovnami:

- CUDA Toolkit 11.0 Update1
- cuDNN v8.0.4 (September 28th, 2020), for CUDA 11.0

a NVIDIA RTX 3070 8GB s knihovnami:

- CUDA Toolkit 11.1.1
- cuDNN v8.0.4 (September 28th, 2020), for CUDA 11.1

4.5 Testování a vyhodnocení

Testování správnosti implementace

Systém byl průběžně testován dle postupu implementace – každá funkce byla manuálně otestována. Pro testování frontendu informačního systému byly využity nástroje Redux DevTools, React Developer Tools a vývojářská konzole prohlížečů Google Chrome a Mozilla Firefox. Backend, program pro řízení neuronových sítí a testovací neuronová síť byli testovány vestavěným ladícím nástrojem v editoru Visual Studio Code a výpisy na standardní výstup. Závěrem systém několikrát prošel celkovým manuálním testováním a testováním vybraných uživatelů.

[3]Návod na spuštění TensorFlow s výpočty na GPU: https://www.tensorflow.org/install/gpu

Testování na uživatelích

Uživatelům byl vysvětlen princip systému a byly jim předány přihlašovací údaje k administrátorskému účtu. Uživatelé měli za úkol importovat datovou sadu, která jim byla ukázána v souborovém systému, povolit právě importovanou datovou sadu pro tři anotace a vytvořit uživatelský účet. Z nově vytvořeného účtu bylo požadováno vytvořit tři anotace a tím splnit limit datové sady a změnit email tohoto uživatelského účtu. Po opětovném přihlášení na administrátorský účet uživatelé spustili další iteraci a zobrazili si historii a souhrn své práce na jimi vytvořeném účtu.

Uživatel Jiří (21 let)

Jiří je studentem třetího ročníku bakalářského studia informačních technologií na Fakultě informačních technologií Vysokého učení technického v Brně. Pracuje na částečný úvazek jako systémový administrátor a vývojář softwaru.

Při testování byla vytčena absence možnosti procházení souborového systému serveru při importu dat – pro svou složitost implementace a malý význam je možnost zařazena do možných rozšíření. Dále bylo zjištěno, že po odhlášení zůstává v URL poslední zobrazený pohled, který může být pro jiného uživatele nedostupný – řešeno opravou. Uživatel při vytváření anotací zmínil možnost nastavení výchozí hodnoty velikosti štětce po každé anotaci – požadavek neřešen. Po splnění limitu anotací datové sady bylo vyskakovací okno, oznamující nedostupnost datové sady jako chybu, vnímáno negativně – změna obsahu vyskakovacího okna. Při zobrazení historie práce uživatel využíval informací data a času v záznamech, které jsou však ve formátu ISO a pro uživatele byly velmi špatně čitelné – převedeno do formátu čitelného pro člověka.

Uživatel Erik (22 let)

Erik studoval obor informační technologie na Střední průmyslové škole v Jedovnicích. V dnešní době pracuje jako servisní technik elektronických zařízení.

Erik prošel celým procesem testování intuitivně. Z pozorování jeho práce jsem zjistil, že není jednoduché odhadnout tloušťku štětce, protože není vizuálně interpretována před začátkem tahu – zobrazení obrysu kurzoru přidáno jako možné rozšíření. Při anotaci červeného podkladu byla vidět menší nejistota – změna barvy masky přidána jako možné rozšíření. Na závěru testování uživatel sdělil, že aplikace je pro něj přívětivá a neměl problém s ní pracovat. Jako menší možné komplikace pro méně zkušené uživatele uvedl pozici plovoucího tlačítka pro import datových sad a vytvoření uživatele a absenci explicitního popisu tlačítka pro návrat do prohlížeče datové struktury – pro svůj malý význam uvedeno pouze zde.

4.6 Možná rozšíření

Aktuální implementace systému obsahuje nejdůležitější funkce a demonstruje základní koncept práce. Díky modularitě je možné tento systém jednoduše rozšířit dle žádoucích vlastností. Nejsou implementovány všechny možné funkce, protože by mohly pouze zvýšit komplexnost aplikace a tím snížit získanou úsporu času.

Uživatelská práva k jednotlivým datovým sadám

Při větším počtu datových sad a uživatelů může být žádoucí odstínit některé uživatele od určitých datových sad. Možné by bylo i vytvářet skupiny uživatelů, aby nemusel být každý uživatel přiřazován jednotlivě.

Skrývání/mazání datových sad

Po delší době provozu může být problém s nedostatkem místa, případně s nepřehledností struktury dat. Aktuálně je možné mazat datové sady pouze přímým zásahem do databáze a souborového systému.

Možnost referencovat data při importu

Pro úsporu času a místa při importu dat by mohlo být výhodné je pouze referencovat, případně vyjmout.

Import již anotovaných dat

Přidání podpory případu potřeby rozšířit trénovací množinu anotovanými daty pomocí jiných nástrojů.

Potvrzovací dialog pro vymazání masky a zakázání uživatele

Je třeba počítat s chybou lidského faktoru a pro některé vážnější operace je lepší se ujistit, zdali uživatel provedl akci záměrně nebo nikoliv.

Zobrazení obrysu kurzoru

Pro uživatele při výběru tloušťky štětce může být pomůckou vizualizace štětce na plátně pomocí jeho obrysu.

Změna barvy masky

Červená barva, která je použita pro zaznačování masky může být při anotaci červeného podkladu nevhodná. Pro tyto účely by bylo vhodné moci přepínat alespoň na další barvu nebo si moci barvu libovolně zvolit.

Více parametrů pro pokyn k učení a inferenci

Neuronové sítě lze velice rozsáhle konfigurovat, a proto by mohlo být žádoucí častěji měněné parametry moci zadat přímo z prostředí aplikace.

Zobrazení informací o proběhlém učení a inferenci

Při učení neuronových sítí vzniká velké množství informací, které by mohlo být žádoucí zobrazit rovnou v prostředí aplikace, a nikoliv formou výpisů v terminálovém okně.

Metrika počítání času

Systém zaznamenává začátek a dokončení anotace uživatele a jako rozdíl uvažuje dobu strávenou anotací. Pro firemní nasazení by mohlo být výhodné měřit čas u klienta s pozastavením při nečinnosti a alespoň následným ověřením na backendu, zdali je interval kratší nebo stejný jako rozdíl času začátku a dokončení anotace.

Uložení rozpracované anotace

Pro náročnější anotace by mohlo být výhodné moci rozpracovanou práci uložit. Problém by však mohl nastat při dlouhodobé neaktivitě uživatele, kdy by musel být řešeno vypršení rezervace prvku datové sady po určitý čas nebo ruční odebrání rezervace administrátorem.

Tmavý režim

V dnešní době je velmi populární funkce přepnutí vzhledu aplikace do tmavého režimu, který na určitých typech displejů šetří energii a v hlavně v tmavém prostředí je uživateli obecně vnímán lépe.

Kapitola 5

Závěr

Cílem této bakalářské práce bylo navrhnout a implementovat systém sloužící k tvorbě anotací obrazových dat pro trénování neuronových sítí se správou lidských zdrojů a dat pro učení. V rámci teoretické části jsem zpracoval informace o tvorbě moderních webových aplikacích, principech činnosti neuronových sítí a provedl průzkum existujících řešení. Z průzkumu jsem vyhodnotil, že žádný nástroj neřeší řízení lidských zdrojů a neumožňuje iterativní přístup k anotování.

Ze zjištěných informací jsem provedl návrh systému a s přihlédnutím k vlastním preferencím jsem zvolil konkrétní implementační technologie. Návrh byl při implementaci několikrát upraven, a to jak z důvodu nápadů na vylepšení, tak kvůli problémům s integritou dat nebo zabezpečením systému. Největší změna spočívala v úpravě datové struktury, kdy místo pevně dané struktury je možné datové sady libovolně zanořovat. Tato změna vytvořila náročné návrhové otázky a složitější implementaci, ale také volnost pro subjekty používající tento systém.

Funkcionalita systému byla testována průběžně po částech a následně několikrát jako celek samostatně i s pomocí uživatelů. V rámci celé práce jsem zapisoval možná rozšíření tohoto systému a do výsledného řešení zahrnul pouze ta nejdůležitější. Ostatní rozšíření, která by mohla být užitečná, jsou zaznamenána v této technické zprávě a mohou sloužit jako inspirace těm, co by chtěli tuto práci použít jak pro produkci nebo jako základ pro další práci.

Vytvořený systém je po navázání vlastní neuronové sítě a případném dokončení implementace notifikačního modulu připraven k nasazení. Poznatky z testování byly buď zaznamenány do možných rozšíření nebo rovnou řešeny. Celkově byl systém uživateli hodnocen jako jednoduchý, přívětivý a moderně zpracovaný.

Literatura

[1] Mitrenga, M. *Konvoluční neuronová síť pro segmentaci obrazu*. Brno, CZ, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií. Dostupné z: https://www.vutbr.cz/studenti/zav-prace/detail/112899?zp_id=112899.

[2] Paul Bergmann, D. S. C. S. MVTec AD – A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[3] Paul Bergmann, M. F. D. S. C. S. The MVTec Anomaly Detection Dataset: A Comprehensive Real-World Dataset for Unsupervised Anomaly Detection. *International Journal of Computer Vision*. 2021.

[4] IT solid. *MongoDB System Properties* [online]. 2021 [cit. 2021-28-04]. Dostupné z: https://db-engines.com/en/system/MongoDB.

[5] IT solid. *MongoDB System Properties* [online]. 2021 [cit. 2021-28-04]. Dostupné z: https://db-engines.com/en/system/MongoDB.

[6] IT solid. *System Properties Comparison MySQL vs. Oracle vs. PostgreSQL* [online]. 2021 [cit. 2021-28-04]. Dostupné z: https://db-engines.com/en/system/MySQL%3BOracle%3BPostgreSQL.

[7] Wikipedia contributors. *Artificial neural network — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 28-April-2021]. Dostupné z: https://en.wikipedia.org/w/index.php?title=Artificial_neural_network&oldid=1017042691.

[8] Wikipedia contributors. *CSS — Wikipedia, The Free Encyclopedia*. 2021. [Online; accessed 28-April-2021]. Dostupné z: https://en.wikipedia.org/w/index.php?title=CSS&oldid=1019611009.

[9] Wikipedie. *Hypertext Markup Language — Wikipedie: Otevřená encyklopedie*. 2020. [Online; navštíveno 28. 04. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Hypertext_Markup_Language&oldid=19168348.

[10] Wikipedie. *Model-view-controller — Wikipedie: Otevřená encyklopedie*. 2020. [Online; navštíveno 28. 04. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Model-view-controller&oldid=19143373.

[11] Wikipedie. *Node.js — Wikipedie: Otevřená encyklopedie*. 2020. [Online; navštíveno 1. 05. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Node.js&oldid=19126999.

[12] Wikipedie. *Representational State Transfer — Wikipedie: Otevřená encyklopedie*. 2020. [Online; navštíveno 1. 05. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Representational_State_Transfer&oldid=18851475.

[13] Wikipedie. *PHP — Wikipedie: Otevřená encyklopedie*. 2021. [Online; navštíveno 1. 05. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=PHP&oldid=19316920.

[14] Wikipedie. *Python — Wikipedie: Otevřená encyklopedie*. 2021. [Online; navštíveno 1. 05. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=Python&oldid=19574714.

[15] Wikipedie. *WebSocket — Wikipedie: Otevřená encyklopedie*. 2021. [Online; navštíveno 1. 05. 2021]. Dostupné z: https://cs.wikipedia.org/w/index.php?title=WebSocket&oldid=19673709.

[16] Šmíd, V. *Pojem informačního systému* [online]. [cit. 2021-28-04]. Dostupné z: https://www.fi.muni.cz/~smid/mis-infsys.htm.