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

WEBOVÁ APLIKACE PRO KONTROLU VÝSLEDKŮ AUTOMATICKÉHO ZPRACOVÁNÍ VIDEA A JEHO RUČNÍHO ANOTOVÁNÍ

WEB APPLICATION FOR INSPECTING RESULTS OF AUTOMATIC VIDEO PROCESSING AND MANUAL ANNOTATIONS

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **PETR ČERVÍČEK**
AUTHOR

VEDOUCÍ PRÁCE **doc. RNDr. PAVEL SMRŽ, Ph.D.**
SUPERVISOR

BRNO 2020

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2019/2020

Zadání bakalářské práce

Student: **Červíček Petr**
Program: Informační technologie
Název: **Webová aplikace pro kontrolu výsledků automatického zpracování videa a jeho ručního anotování**
Web Application for Inspecting Results of Automatic Video Processing and Manual Annotations
Kategorie: Web

Zadání:

1. Seznamte se s problematikou rozpoznávání objektů v obraze a sledování jejich pohybu ve videu.
2. Zpracujte přehled přístupů k uživatelské anotaci obrázků a videa a vytváření datových sad pro počítačové zpracování obrazu.
3. Navrhněte a realizujte webovou aplikaci pro anotování videí s využitím existujících nástrojů pro detekci objektů a další zpracování na straně serveru.
4. Otestujte vytvořené řešení v rámci adekvátní uživatelské studie a vyhodnoťte výsledky.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle dohody s vedoucím

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

Vedoucí práce: **Smrž Pavel, doc. RNDr., Ph.D.**
Vedoucí ústavu: Černocký Jan, doc. Dr. Ing.
Datum zadání: 1. listopadu 2019
Datum odevzdání: 14. května 2020
Datum schválení: 31. března 2020

Abstrakt

Tato bakalářská práce se zabývá implementací webové aplikace pro získávání cenných dat pro anotování. Data se získávají převážně z videí a obrázků, ale mohou to také být deepfakes obrázky a gify. Pro detekování objektů ve videu se používá knihovna YOLO – You Only Look Once. Celá aplikace je vyvíjena v Node.js, na kterém běží backend aplikace a React pro vývoj frontendu. Pro lepší pochopení vývoje webových aplikací je v textu taktéž popsán historický vývoj webových technologií.

Abstract

The thesis pursues the implementation of the web application for obtaining valuable data for anotation. Data are mainly collected from videos and images, but they can be obtained from deepfakes images and gifs as well. The YOLO – You Only Look Once is used for object detection in the videos. Whole application is developed in node.js and react. Node.js for backend and react for frontend. For a better understanding of web application, there is also historical description of the web technologies.

Klíčová slova

web, webová aplikace, single page aplikace, YOLO, anotování, video, obrázky, deepfakes, Node, React, Material-UI, detekce objektů, sledování objektů

Keywords

web, web application, single page application, YOLO, annotation, video, images, deepfakes, Node, React, Material-UI, object detection, object tracking

Citace

ČERVÍČEK, Petr. *Webová aplikace pro kontrolu výsledků automatického zpracování videa a jeho ručního anotování*. Brno, 2020. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Webová aplikace pro kontrolu výsledků automatického zpracování videa a jeho ručního anotování

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana docenta Pavla Smrže. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Petr Červíček
26. května 2020

Poděkování

Docent Pavel Smrž mi poskytl mnoho užitečných informací, a tímto bych mu rád poděkoval za jeho ochotu.

Obsah

Kapitola 1

Úvod

Tato práce se zabývá implementací webové aplikace, která získává cenná data pro anotování. V současné době je převážná většina metod pro rozpoznávání obrazu a videa založená na strojovém učení s učitelem, které vyžaduje obrovské množství anotovaných dat. S tím souvisí velké množství otevřených iniciativ pro vytváření takových dat a existují i celé velké firmy, jejichž jediným zaměřením je anotace obrazových dat pro další komerční subjekty. Existují různé platformy typu Mechanical Turk 4.5, které se zaměřují právě na takovéto úkoly.

Vlastní anotování je možné provádět v různých prostředích. Většinou se ale předpokládá velmi specializované rozhodnutí pro danou konkrétní aplikaci (je podvodně změněné video vpravo nebo vlevo), nebo anotování obrazu „od začátku“, například vyznačení oblastí ohraničujících všechny tváře v obraze. Je velmi málo nástrojů, které umožňují pracovat s částečně zpracovanými daty, kde je úkolem uživatele zkontrolovat, případně doplnit a pozměnit anotace vygenerované serverem. Stejně tak je obtížné vymyslet vhodné uživatelské prostředí pro netriviální anotaci video sekvencí, např. pro sledování pohybu (tracking) lidí a objektů ve videu. Právě tímto tématem se zabývá tato práce.

Anotování videí i obrázků se skládá z více částí. Jednou z nich je využívání knihovny YOLO – You Only Look Once (viz 4.1). Ta využívá naučenou neuronovou síť k detekování objektů ve videu. Aplikace posílá http dotaz na server s YOLO a ten vrací odpověď webové aplikaci ve tvaru jmpeg stream (video, na němž jsou zvýrazněny objekty - lidé, automobily atd.) a JSON s informacemi o objektech ve videu. U obrázků je v odpovědi vrácen tentýž obrázek i s anotací. Pokud by chtěl uživatel detekovat objekty, jež YOLO v základním režimu nezvládá, může neuronovou síť naučit rozpoznávat objekty, které požaduje. K tomu je ale zapotřebí dostatečný dataset, na kterém by bylo možné neuronovou síť učit. Dále je možné, aby uživatel anotoval video ručně. Stačí, aby se dostal do módu anotování. Jedná se o pozastavení videa a následné vytvoření kopie aktuálního snímku videa. Poté je možné tento snímek anotovat. Vykreslením polygonů do snímku se získá oblast, které se toto anotování týká. Dále uživatel zadá požadované informace ohledně zvolené oblasti – například vybere osobu a připojí k ní dodatečné informace, jako jsou například emoce.

Výsledky anotování videí (obrázků) i deepfakes fotografií jsou uloženy na server. Pokud to uživatel požaduje, existuje zde možnost lokálního uložení na uživatelův počítač. Veškeré výsledky jsou uloženy ve formátu JSON. Formát byl zvolen pro svou přehlednost a jednoduchost.

Výhodou tohoto nástroje pro získávání cenných dat pro anotování je konfigurace. Uživatel si na stránce *Konfigurace* může nastavit funkcionalitu podle svých potřeb. Například nepotřebuje-li uživatel, aby YOLO 4.1 detekovalo objekty, a zároveň nechce své výsledky posílat na server, může tyto funkcionality vypnout.

Následující kapitola shrnuje základní přístupy k rozpoznávání obrazu a detekci objektů a také krátce shrnuje, jak mohou být detekované objekty sledovány ve videu. Aby byla uživateli objasněna i problematika webových aplikací, je do textu zařazena krátká kapitola týkající se vývoje webových aplikací a také kapitola zaměřující se na použité technologie a terminologii. Obě slouží k lepšímu pochopení, proč byla webová aplikace implementována právě tímto způsobem. Kapitola konkurenční anotační nástroje uvádí jiné podobné aplikace a popisuje rozdíly mezi nimi a webovou aplikací této bakalářské práce. Podrobnější popis implementace webové aplikace se pak nachází v následující kapitole.

Nakonec bude práce shrnuta v závěru, kde bude také uvedeno, které části se nepodařilo implementovat, a zároveň také obsahuje nástin, jak by bylo možné aplikaci do budoucna zdokonalit, co je jejím přínosem a v kterých oblastech by tato aplikace mohla najít uplatnění.

Kapitola 2

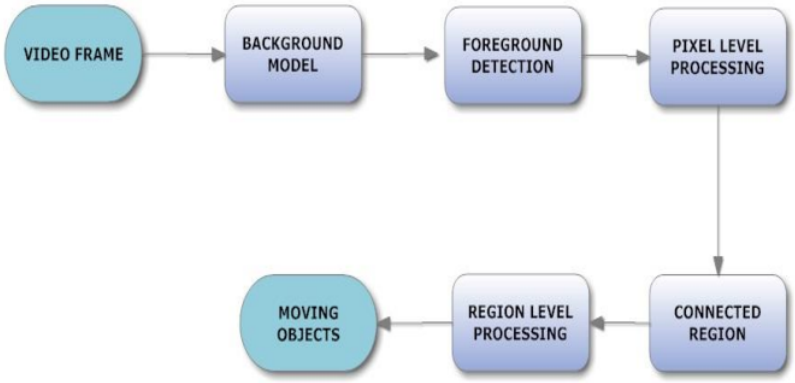
Rozpoznávání objektů v obraze a sledování jejich pohybu

Sledování videa je proces analyzování sekvencí (úseků) videa. Existují tři přístupy ke sledování videa. Jedná se o manuální, semi-manuální (částečně manuální a částečně automatizované) a plně automatizované anotování. Manuální anotování zahrnuje analýzu videa pomocí lidského faktoru. Tyto systémy jsou široce rozšířeny. Semi-automatizované systémy zahrnují některé formy počítačového zpracování obrazu, ale je zde značné zastoupení lidského zasahování. Typickým příkladem jsou systémy pro jednoduchou detekci pohybu. U plně automatizovaných systémů je sekvence videa vyhodnocena pouze strojem. Nedochází zde k lidskému zasahování a systém provádí jak nízkoúrovňové úkoly (detekce a sledování pohybu), tak i vysokoúrovňové úkoly, jako je například detekce nezvyklých událostí nebo rozpoznávání gest.

Lidské anotování dlouhotrvajícího videa je velmi nepraktické a někdy až neproveditelné. IVS (Intelligent Visual Surveillance) značí automatický proces pro vizuální monitorování, který zahrnuje analýzu a interpretaci chování objektů (detekci objektů a jejich sledování).

2.1 Detekce pohybujících se objektů

Každá metoda pro sledování objektů vyžaduje mechanismus pro detekci objektu (buď pro každý snímek, nebo když se objekt objeví poprvé). Metoda zvládá segmentaci pohybujících se objektů od stacionárních objektů na pozadí. Změny prostředí, jako je například osvětlení, ztěžují segmentaci objektů, což vytváří zásadní problém. Běžným přístupem pro detekci objektů je za použití informací z jednoho konkrétního snímku videa, ačkoliv některé metody využívají dočasné informace získané ze sekvence snímků, aby se zmenšil počet chybných detekcí.



Obrázek 2.1: Znázornění postupu pro detekci objektů [15]

.

Prvním krokem je rozpoznání objektů v popředí od objektů v pozadí. Aby bylo tohoto cíle dosaženo, lze využít kombinaci technik pro vytvoření mapy pixelů v popředí pro každý snímek. Poté se seskupí oblasti, abychom získali jednotlivé rysy objektů, jako jsou ohraničující části, oblasti objektu, obvod atd.

Detekce popředí

Téměř každý systém pro anotování videa využívá detekci objektů v popředí. Tím se napomáhá snížení výpočetního času. Prvním krokem je inicializace scény pozadí. Zde existuje mnoho metod. Související části (moduly) scény jsou izolovány a jejich propojování je minimální, aby mohl celý detekční systém pracovat flexibilně s jednotlivými moduly.

V dalším kroku dochází k detekování pixelů v popředí za využití modulů z pozadí a aktuálního snímek videa. Tato detekce na úrovni pixelů je závislá na modulech z pozadí a je využívaná pro jejich aktualizaci, aby se adaptovaly na dynamické změny. Kvůli různým defektům se může stát, že detekovaná mapa pixelů z popředí bude obsahovat nepožadované šumy (rušivé elementy). Operace pro „post-processing“ na úrovni pixelů je navržena tak, aby odstranila šum z pixelů z popředí. Jakmile jsou vyfiltrovány pixely z popředí, pomocí algoritmu jsou nalezeny oblasti, které spolu souvisejí a ty jsou následně spojeny. Taktéž jsou zároveň spočteny ohraničující oblasti objektů. V posledním kroku jsou extrahovány informace objektu, jako například celá oblast objektu, obvod, ohraničení atd.

Zpracování na úrovni pixelů

Tato část se zabývá odstraňováním rušivých elementů z detekovaného popředí snímků. Existují různé faktory, které tyto elementy mohou způsobit:

Kamerový šum Jedná se o šum způsobený součástmi fotoaparátu při snímání obrazu. Je vyvolán intenzitou pixelu, který odpovídá okraji mezi dvěma různě barevnými objekty ve scéně. Může být nastaven na barvu jednoho objektu v prvním snímku a na barvu druhého objektu ve snímku následujícím.

Šum způsobený barevnými objekty v pozadí Objekty mohou mít stejnou barvu jako má jejich pozadí. Zde je těžké detekovat pixely v popředí.

Šum způsobený odrazem Tento šum je způsobený osvětlením scény. Když zdroj světla mění svoji polohu, dochází k odrazu světla některými částmi v pozadí.

K odstranění šumů lze použít dolnopropustní filtr a morfologické operace (eroze, zvětšení) na mapu pixelů získaných z popředí. Cílem je odstranění rušivých pixelů z popředí, které neodpovídají skutečným oblastem popředí. Dále odstranění rušivých pixelů z pozadí poblíž nebo uvnitř objektových oblastí, které jsou ve skutečnosti pixely z popředí.

Dolnopropustní filtry jsou používány k rozmazání a redukci šumu. Gaussův dolnopropustní filtr se používá pro následné zpracování na úrovni pixelů. Vyhlazuje obrázek výpočtem vážených průměrů koeficientů filtru. Filtr dále upravuje pomocí konvoluce s Gaussovou funkcí vstupní signál.

Detekování souvisejících oblastí

Po detekci oblastí v popředí a použití operací pro odstranění šumů jsou vyfiltrované pixely z popředí seskupovány do souvisejících oblastí. Po nalezení jednotlivých oblastí, které odpovídají objektům, jsou vypočteny ohraničující rámečky (anglicky bounding box).

Zpracování na úrovni oblastí

I po odstranění šumů zůstávají některé malé oblasti kvůli špatné segmentaci nedotčené (TODO). Proto oblasti, které jsou menší než předem definovaný práh, jsou odstraněny z mapy pixelů z popředí. Poté lze extrahovat rysy objektů ve snímku. Těmi jsou velikost, těžiště nebo ohraničení souvisejících komponent. Tyto rysy jsou používány ke sledování objektů a jejich klasifikaci.

2.2 Sledování objektů

Sledování objektů zahrnuje spojování detekovaných objektů v popředí scény mezi snímky následujícími po sobě za pomoci jednotlivých rysů objektů, jako je pohyb, rychlost, barva, struktura. Jedná se tedy o proces sledování objektů v čase tak, že jsou objekty lokalizovány v každém snímku videa. Při sledování jsou objekty reprezentovány pomocí modelů tvaru nebo vzhledu. Model, který je vybrán pro reprezentaci tvaru, omezuje typ pohybu. Pokud je například objekt reprezentován jako bod, lze použít pouze translační model. V případě, že je pro objekt použita reprezentace geometrického tvaru jako elipsa, jsou vhodné parametrické pohybové modely, jako jsou afinní nebo projektivní transformace. Níže jsou popsané různé sledovací metody.

Bodové sledování Bodové sledování je robustní, spolehlivá a přesná metoda, kterou vymyslel Veenman a spol. Tato metoda se většinou používá pro sledování vozidel. Jsou zde vyžadovány deterministické nebo pravděpodobnostní metody. Sledovaný objekt je jako bod, který je reprezentován v detekovaných objektech ve snímcích jdoucích po sobě. Přiřazení bodů je dáno předchozím stavem objektu, který může zahrnovat polohu a pohyb objektu. Tento přístup vyžaduje pro detekování objektů v každém snímku externí mechanismus.

Kernel sledování objektů Vstupem pro Kernelovu metodu je tvar a vzhled objektu. Každý rys objektu je použit pro jeho sledování. Po vypočtení pohybu jádra objektu ve snímcích následujících po sobě lze pak tento objekt sledovat.

Silhouette sledování V tomto přístupu je extrahována z objektu jeho silueta. Jsou zde využívány informace, které jsou uloženy uvnitř objektových oblastí. Tato informace o oblasti může být například hustota vzhledu a tvarové modely.

Sledování objektů je založeno na jejich rysech a je vyžadován jejich správný výběr. To hraje nejdůležitější roli ve sledování. Z toho vyplývá, že rysy musejí být od sebe odlišné, aby mohly být objekty lehce rozlišitelné. Mohou to být například:

Barva Barva objektu je primárně ovlivněna dvěma fyzickými faktory; prvním je rozdělení světelného spektra a druhým jsou vlastnosti odrazu povrchu objektu. Pro reprezentaci je většinou použit formát RGB.

Hrany Hranice objektu obvykle generují silné změny v intenzitě obrazu. Detekce hrany jsou využívány pro identifikaci těchto změn. Důležitou vlastností hran je to, že jsou méně náchylné na změny osvícení.

Těžiště Pro každý bod je dobré zmínit, že prvním elementem těžiště je horizontální souřadnice (nebo x-souřadnice) a druhým elementem je vertikální souřadnice (nebo y-souřadnice).

Struktura Struktura objektů je používaná jak ke klasifikaci, tak i ke sledování. Tento rys je používán k identifikaci požadované oblasti či objektu. Jedná se o měření změny intenzity povrchu, které kvantifikuje vlastnosti, jako plynulost a pravidelnost. V porovnání s barvou tak struktura vyžaduje krok pro generování deskriptorů.

Rysy typu barva a struktura jsou při sledování objektů široce používané. Nevýhodou barvy je však její náchylnost na osvětlení scény [15].

Kapitola 3

Historie webových aplikací

3.1 Single Page aplikace (SPA)

Single Page aplikace je webová aplikace, která používá pouze jednu HTML stránku jako obálku pro všechny ostatní stránky. Interakce s koncovými uživateli je implementována za použití JavaScriptu, HTML a kaskádových stylů (CSS). Většina vývoje u SPA je na rozdíl od tradičního webového vývoje provedena na frontendu. Zde je většina implementace vždy na straně webového serveru (neboli backendu). Při interakci s uživatelem dochází k opětovnému načítání stránky. Single page aplikace připomínají svým chováním a implementací nativní aplikace, ale na rozdíl od nich běží v procesu prohlížeče. Nativní aplikace běží v samostatném procesu operačního systému. Pokud chceme porozumět všem výhodám single page aplikací a proč jsou v dnešní době tak populární, musíme nejdříve znát historii vývoje webových aplikací a jaké změny během ní nastaly [12].

Historie vývoje webových aplikací

Celosvětová síť (WWW – World Wide Web) byla vytvořena v roce 1990, když Tim Berners-Lee implementoval HTML a protokol HTTP. V této době byly webové stránky složeny pouze z HTML souborů, které interpretoval webový prohlížeč. Jednotlivé stránky byly mezi sebou propojeny odkazy, při kliknutí na ně byli uživatelé přesměrováni na jinou webovou stránku. Na každé stránce byl jejich obsah statický a skládal se především z textových úseků a obrázků. Pro odstranění potíží se statickými stránkami byly později vyvinuty serverové skriptovací jazyky, které slouží k vytváření dynamických stránek. Těmito jazyky jsou například PHP nebo Active Server Pages (ASP). Principem těchto skriptovacích jazyků je dynamické sestavení stránek na webovém serveru a následné odeslání klientovi do prohlížeče. To umožňovalo implementaci mnohem složitějších a vyspělejších webových aplikací, které mohly také reagovat na uživatelské vstupy.

JavaScript byl představen kolem roku 1996 jako skriptovací jazyk pro klientskou stranu webových aplikací. Vývojáři mohli díky JavaScriptu implementovat novou logiku aplikace na klientské straně. Právě díky této nové logice se naskytly nové možnosti v implementaci dynamických stránek. Ačkoliv už díky JavaScriptu vytváření dynamických stránek existovalo, většina vývojářů stále vyvíjela stránky záměrně s logikou na straně webového serveru. Neviděli totiž v JavaScriptu žádný velký potenciál, neboť ten byl tehdy využíván hlavně k dynamickým změnám uživatelského rozhraní. Je dobré také zmínit, že v této době byla uživatelská přívětivost webových aplikací nízká. Dáno to bylo hlavně kvůli malé rychlosti

připojení k internetu a nutnosti znovunačtení celé stránky ze serveru po každé interakci s touto stránkou.

V roce 2000 přišla další změna. Byly představeny nové technologie, Flash a Java applety. Na webové stránce bylo umožněno mít vloženou aplikaci, která byla implementována ve Flashi nebo Javě a která poskytovala prostředí, pomocí něhož mohli uživatelé s aplikací interagovat. Tyto vložené objekty mohou být samozřejmě na webových stránkách až dodnes, ale kvůli bezpečnosti je mohou některé prohlížeče blokovat. Aby mohly být tyto nové technologie využívány, je nutná instalace aplikací třetích stran (pro spuštění aplikací ve Flashi je nutná jeho instalace, stejně tak Java a Java applety). Microsoft přišel okolo roku 2007 s technologií Silverlight. Byl to další plugin do prohlížeče, ale protože měl stejné nevýhody jako Flash a Java, tak se tato technologie neujala.

Jednou z dalších možností tvorby webových aplikací v rámci jedné webové stránky bylo vložení tzv. iframů. Výhodou této technologie bylo znovunačítání stránky pouze v rámci tohoto jednoho iframu, když uživatel se stránkou interagoval. Díky tomu se nemusela načítat znovu celá stránka. Toto řešení však představovalo další problémy s bezpečností. Jednou z nevýhod iframů bylo, že vývojáři museli udržovat jak hostitelskou verzi aplikace, tak i tu vloženou v iframu a ještě k tomu řešit komunikaci mezi těmito dvěma částmi.

JavaScript, na rozdíl od již zmíněných pluginů, je součástí prohlížeče. To je oproti výše zmíněným technologiím velkou výhodou JavaScriptu. A tou je fakt, že funguje, aniž by uživatel musel instalovat dodatečný software třetích stran. Jak již bylo nastíněno, další výhodou je, že Javascript umožňuje měnit obsah stránek, aniž by bylo nutné jakkoliv znovu načítat nebo obnovovat stránky. Hlavním důvodem, který brzdil použitelnost JavaScriptu ve své době, byl fakt, že jeho funkcionalita nebyla tak obsáhlá jako funktionalita Flashe. Navíc interpreti pro JavaScript byli v prohlížečích velmi pomalí. Tyto důvody umožnily rozšíření Flashe [19].

Ajax

Termín Ajax byl poprvé použit 18. února 2005, když Jesse James Garrett vydal článek *Ajax: A New Approach to Web Applications* [13]. Jedná se o složeninu slov Asynchronous JavaScript and XML. Nástupem technologie Ajax se začalo JavaScriptu blýskat na lepší časy. Jak již název napovídá, jedná se o technologii pro vývoj webových aplikací, která běží na straně klienta a vytváří asynchronní webovou aplikaci. Díky tomu může tato aplikace posílat a přijímat data ze serveru asynchronně (na pozadí) bez narušování chování právě využívané stránky. Ačkoliv Ajax využívá XML, v dnešní době je místo XML v praxi využíván spíše JSON [32].

Ajax není jedna technologie, ale spíše seskupení více technologií v jednu. HTML a CSS mohou být zkombinovány pro vyznačení a stylování jednotlivých informací. Poté může být webová stránka upravena pomocí JavaScriptu tak, aby dynamicky ukazovala (zároveň dovoluje uživateli interagovat) nové informace. Od roku 2017 je do JavaScriptu přidána nová funkcionalita fetch, která je využívána ke spuštění Ajaxu na webových stránkách. To dovoluje načítání obsahu stránek, aniž by bylo potřeba stránku znovu načíst.

Je dobré zmínit, že Ajax není žádná nová technologie nebo nový programovací jazyk, ale je to nový způsob využívání již existujících technologií.

HTML5

HTML5 neboli HyperText Markup Language verze 5 je balíček softwarových systémů, které definují vlastnosti a chování obsahu webových stránek. To je implementováno pomocí tak-

zvaných „značek“. Implementace HTML5 začala již v roce 2006. K prvnímu veřejně fungujícímu návrhu HTML5 specifikace došlo v roce 2008 [38]. Hlavním cílem je lepší podpora pro multimédia. Dále to může být například lehká čitelnost kódu, ale zároveň je kladen důraz na srozumitelnost pro strojový překlad bez striktních pravidel, která jsou dána XHTML. HTML je také zpětně kompatibilní se staršími verzemi – nezahrnuje pouze HTML4, ale také XHTML a DOM Level 2 HTML [39].

HTML5 obsahuje mnoho nových prvků moderního webu, které jsou specifické pro JavaScript. Umožňuje ovládat komunikaci, grafiku, multimédia a další oblasti, kde JavaScript zastává důležitou roli.

Web Hypertext Application Technology Working Group (WHATWG) začal vyvíjet nový standard v roce 2004. V této době nebylo HTML4 aktualizováno od roku 2000 [37]. W3C a WHATWG začaly okolo roku 2004 spolupracovat na novém HTML standardu založeném na HTML4.

Důležitou funkcí, jež je zahrnuta v HTML5, je využití API hardwaru zařízení, na kterém běží webová aplikace. Může to být například geolokační API. To umožňuje zjišťování polohy zařízení, které obsahuje GPS modul. Dále je možné přistupovat ke komponentům počítače – například mikrofonu nebo webové kameře. Tyto funkce pak přispívají k tomu, že se JavaScript využívá k vytváření mobilních aplikací [19].

Kapitola 4

Technologie a terminologie

4.1 YOLO: Real-Time Object Detection

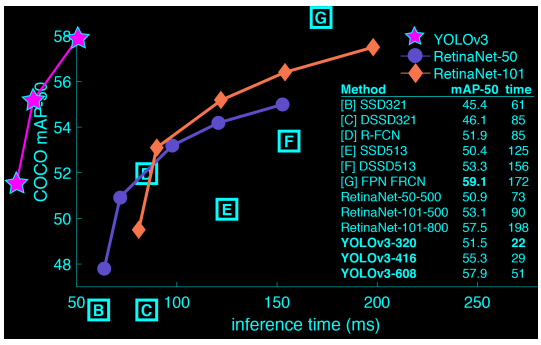
YOLO neboli You Only Look Once je systém, který je schopen detekovat objekty v reálném čase. Může tomu tak být ve videu, obrázku nebo třeba z webové kamery. Pro klasifikování je použita pouze jediná neuronová síť pro celý obrázek (snímek z videa). Tato neuronová síť rozdělí obrázek na části a předpovídá všechny pravděpodobnosti a ohraničení v každé části. Každému ohraničení je přiřazena váha podle předpovězených pravděpodobností. [26].

Tento model má oproti systémům, které jsou založené na *classifier-based* mnoho výhod. Nerozděluje obraz na části, ale vstupem modelu je celý obraz. Díky tomu jsou předpovědi tohoto modelu informovány pomocí celého kontextu obrazu. Jak bylo již zmíněno, systém využívá pouze jednu neuronovou síť na rozdíl od systémů s rekurentní neuronovou sítí (R-CNN), která by na tento proces potřebovala tisíce sítí. To činí YOLO systém více než 1000x rychlejší než *R-CNN* a 100x rychlejší než *fast R-CNN*.

Postup instalace YOLO systému je zde https://pjreddie.com/darknet/yolo/.

Porovnání s jinými detekčními systémy

YOLO je velmi rychlý a přesný. YOLO verze 3 je, co se týče takzvaného *focal loss*, systémem průměrným, ale je asi 4x rychlejší než jiné systémy. Je velmi snadné udělat kompromis mezi rychlostí a přesností pouze pomocí změny velikosti modelu. Jeho přetrénování není nutné!



Obrázek 4.1: Graf znázorňující porovnání YOLO s jinými systémy. Převzato z https://pjreddie.com/darknet/yolo/.

Darknet

Darknet je *open source framework* využívající neuronovou síť a je napsaný v jazyce C a CUDA. Podporuje zpracování příkazů jak pomocí CPU, tak také pomocí GPU. Darknet open source najdeme na stránce https://github.com/pjreddie/darknet. Slouží jako backend pro YOLO [24]. Návod pro instalaci darknetu je zde: https://pjreddie.com/darknet/install/.

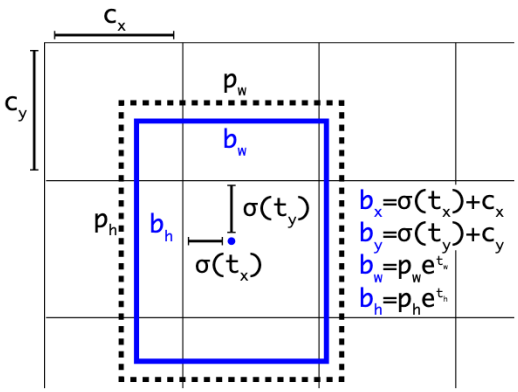
Predikování ohraničení objektů

Neuronová síť predikuje pro každé ohraničení čtyři souřadnice – t_x, t_y, t_w, t_h. Pokud je buňka posunuta od levého horního rohu obrázku o $[c_x, c_y]$ a ohraničení má šířku a výšku p_w, p_h, potom předpovědi odpovídají výpočtu:

$$b_x = \sigma(t_x) + c_x$$
$$b_y = \sigma(t_y) + c_y$$
$$b_w = p_w e^{tw}$$
$$b_h = p_h e^{th}$$

Během trénování je využíván součet čtverců chybových ztrát. Pokud skutečná hodnota predikce některé souřadnice je $\hat{t}_*$, tak gradient je roven pravdivostní hodnotě bez hodnoty predikce: $\hat{t}_* - t_*$.

YOLOv3 využívá logistickou regresi pro predikci objektového skóre každého ohraničení. Výběr je proveden tak, že ohraničení překrývá objekt více než jakékoli jiné předchozí ohraničení. To znamená, že pro objekt jsou vypočítávány různá ohraničení a každé je porovnáno s ohraničením, které je objektu přiřazeno. Nově vypočtené ohraničení nahrazuje předešlé, pokud lépe odpovídá objektu. Pokud ohraničení není optimální, ale překrývá skutečnou hodnotu objektu více než určený práh, tak je tato predikce ignorována, viz [27]. U YOLOv3 je tento práh nastaven na 0.5. Každému objektu je tedy přiřazeno maximálně jedno ohraničení [25].



Obrázek 4.2: Ukázka výpočtu ohraničení objektů. Převzato z https://pjreddie.com/media/files/papers/YOLOv3.pdf.

Predikování tříd

Pro každé ohraničení jsou predikovány možné třídy, které by mohlo obsahovat, za použití multi-label klasifikace. Není zde využívána funkce *softmax*, protože pro potřeby YOLOv3

není potřebná a na výkon nemá žádný vliv. Místo toho jsou používány nezávislé logistické klasifikátory a na predikci tříd během trénování je použita ztrátová funkce *cross-entropy* [25].

Trénování

Pokud je nutné využívat YOLO s jiným datasetem, je možné ho natrénovat. Musíme mít k dispozici všechna VOC data 4.2 z let 2007 až 2012. Poté je potřeba vygenerovat soubory obsahující značky objektů, které Darknet využívá. Jedná se o soubory s příponou .txt pro každý obrázek. Jednotlivé řádky obsahují informace ohledně objektů ve tvaru:

<třída objektu> <x> <y> <šířka> <výška>

Kde x, y, šířka a výška jsou relativní vzhledem k dimenzím obrázku.

Druhým možným způsobem je trénování pomocí COCO datasetu[1]. Zde je potřeba mít všechny COCO data a značky objektů. [26]

4.2 The PASCAL Visual Object Classes (VOC) Challenge

Pascal VOC Challenge je velmi populární dataset pro vytváření a vyhodnocování algoritmů, které slouží ke klasifikaci obrázků, detekování objektů a segmentaci. Jedná se o kritérium v oblasti rozpoznávání a detekce objektů v obraze poskytující veřejné datasety obrázků, anotací a standardních vyhodnocovacích procedur [10].

4.3 Deepfakes

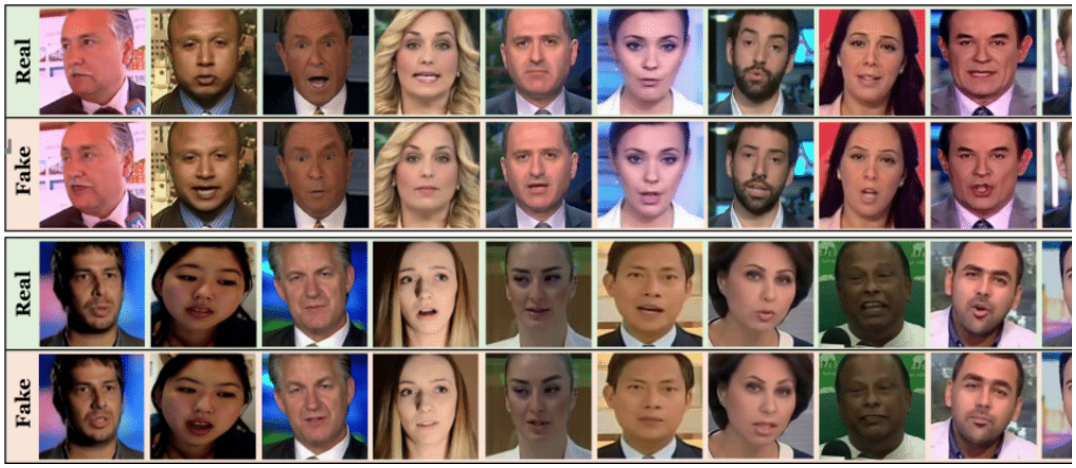
Deepfake (složenina dvou slov *Deep learning* a *Fake*) je technika výměny obličejů a můžeme ji použít u fotografií nebo u videí. Využívá se při tom umělých neuronových sítí známých jako *autoenkodér* anglicky *autoencoder*. Jedná se o enkodér, který redukuje kvalitu obrazu, a dekodér pak tento obraz znovu sestaví s již nahrazeným obličejem. Protože se tato práce nevěnuje tvorbě deepfakes, ale pouze jejich rozpoznávání, tak není potřeba se touto problematikou zabývat důkladněji.

S deepfakes se nejčastěji setkáme v pornografii, kdy jsou používány obličeje celebrit jako výsledek deepfake rekonstrukce. Tyto výměny jsou bohužel prováděny bez souhlasu celebrit. První takovou celebritou, jejíhož obličeje bylo použito, byla herečka Daisy Ridley. Dalším odvětvím, ve kterém se aplikace deepfake uplatňuje, je politika. Tato technika zde umožňuje šíření falešných zpráv známými politiky.

Abychom nezmiňovali pouze negativní účinky deepfake technologie, uveďme příklad využití ve filmovém průmyslu. Výměna tváří se zde využívá pro omlazování postav.

Na závěr jsou zde přiloženy ukázky, jak tato technika vypadá:

[1] http://cocodataset.org/#home



Obrázek 4.3: Příklady deepfakes. Převzato z https://www.biometricupdate.com/201910/harmful-application-of-deepfakes-growing-rapidly-online-new-report-warns.

Jak je vidět, mnohdy je těžké přesně určit, kdy se jedná o takzvaný „fake“ a kdy o originál.

4.4 Node.js

Node.js je asynchronní open source JavaScriptová platforma, která spouští JavaScriptový kód mimo prohlížeč (na backendu aplikace). Běží jako jeden jediný proces bez nutnosti vytváření nového vlákna pro každý požadavek. Node.js poskytuje ve své knihovně sadu asynchronních I/O primitiv, která zabraňují vzájemnému blokování. Těmito primitivy může být například přistupování do databáze, čtení a zápis ze souborového systému, komunikace na síti.

Node.js umožňuje spravovat tisíce souběžných připojení v rámci jednoho serveru bez nutnosti spravování souběžnosti vláken. Pokud by programátor musel spravovat i tato souběžné vlákna, vedlo by to k mnoha dalším *bugům* v aplikaci.

Velkou výhodou, kterou Nodej.js poskytuje, je to, že mnoho frontend vývojářů, kteří ovládají JavaScript, je nyní schopno také psát serverovou část kódu, aniž by se museli učit nový programovací jazyk.

K rozšíření Node.js pomohl balíček *npm*, který má velmi jednoduchou strukturu [20].

NPM

Npm (Node Package Manager) je správce balíčků pro JavaScript a zároveň je taktéž výchozím správcem balíčků pro platformu Node.js. Skládá se z terminálového klienta, který interaguje se vzdálenými registry. To umožňuje uživatelům využívat a distribuovat JavaScriptové moduly, které jsou k dispozici v registrech. Balíčky v registrech mají formát *CommonJS* a zahrnují soubor s metadaty ve formátu *JSON* [21].

Npm má volně dostupnou dokumentaci ke svým balíčkům, kterou jsem v této bakalářské práci využíval.

Next.js

Next.js je framework, který umožňuje psát *React* aplikace, jež jsou renderovány na straně serveru. Výhodou této knihovny je právě to, že danou stránku je nutno nadefinovat pouze pomocí React component jednou a knihovna se již postará o správnou funkcionalitu serverového či klientského vykreslení.

Express

Express.js, zjednodušeně Express, je webový framework pro Node.js. Byl vydán jako open source software pod licencí MIT. Je navrhnut pro vytváření webových aplikací a API. Je považován za server framework standard pro Node.js.

Material-UI

Material-UI[2] je velmi populární framework pro vytváření uživatelského prostředí v Reactu. Jedná se o open source knihovnu, která patří pod licenci MIT. Vývoj začal v roce 2014, kdy šlo o snahu sjednotit React a Material Design [18]. Webová dokumentace (poskytuje i příklady použití) pro Material-UI byla v této bakalářské práci využívána pro vytvoření potřebných komponent.

Scalable Vector Graphics

Scalable Vector Graphics, častěji používáno pod zkratkou SVG, je jazyk, který popisuje dvoudimenzionální grafiky v XML. Povoluje tři typy objektů – vektorovou grafiku (skládá se z přímek a křivek), obrázky a text. Grafické objekty mohou být seskupovány, stylizovány, transformovány atd.

Vykreslování SVG může být interaktivní a dynamické. Animace je definována a spouštěna buď deklarativně, nebo pomocí skriptů.

Sofistikovanější aplikace s SVG jsou možné díky doplňkovým skriptovacím jazykům s přístupem do SVG's DOM (Document Object Model pro SVG), který poskytuje kompletní přístup ke všem elementům, atributům a vlastnostem. Bohatý soubor „event handlerů", jako je například *onmouseover* nebo *onclick*, může být přidán k jakémukoliv SVG objektu.

MIME Type pro SVG je „"image/svg"". Organizace přidá tento MIME Type jakmile bude SVG schválen jako *W3C Recommendation*. Výše zmíněná vektorová grafika obsahuje tento základní souhrn elementů: obdélníky, kruhy, elipsy, přímky, polygony a křivky [4]. V této bakalářské práci byly použity na vykreslování do videa pouze *polylines* (křivky).

4.5 Mechanical Turk

Amazon's Mechanical Turk je *online* nástroj, který sbírá data za využití skupiny osob. Umožňuje společnostem, ale i jednotlivcům, předávat jejich práci pracovníkům, kteří mohou tento úkol provést virtuálně. Může se přitom jednat o cokoliv, od validování jednoduchých dat po úkoly subjektivní, jako zadávání dotazníků [8]. Jakmile zaměstnanec (osoba, která zpracovává úkol) dokončí svoji práci, Amazon mu zaplatí podle předem stanovené smlouvy [35].

[2] https://material-ui.com/

Mohou to být i úkoly související s anotováním videí a obrázků. Jedná se o vytvoření HITs (Human Intelligence Tasks – úkoly určené pro lidi) rozdělením videa na menší sekvence. Poté se anotování těchto sekvencí rozdělí zaměstnancům [35].

Kapitola 5

Konkurenční anotační nástroje

V této kapitole budou popsány konkurenční anotační nástroje. Na úvod bych rád dodal, že žádný z prostudovaných nástrojů neobsahuje rozpoznávání deepfakes obrázků.

Video Annotation Tool from Irvine, California

Zkráceně *VATIC*[1] je online anotační nástroj pro rozpoznávání videa, který sbírá data od uživatelů pro *Amazon's Mechanical Turk*.

Vatic byl testován pouze na operačním systému Linux – Ubuntu s Apache 2.2 HTTP serverem a databází MySQL. Systém by však měl fungovat v jakémkoliv operačním systému.

Před samotným anotováním jsou z videa extrahovány jednotlivé snímky (základní rozlišení je 720x480). Ty jsou poté rozděleny na malé segmenty (pouze několik málo vteřin). Každý segment je anotován zvlášť a poté jsou tyto segmenty znovu spojeny [34].

Výhodou této bakalářské práce oproti VATIC je jeho univerzálnost a konfigurovatelnost. VATIC je navržen tak, aby sloužil pouze pro *Amazon's Mechanical Turk*. Dále jsou u mé práce zachovány dimenze videa a nedochází k jejich změně na rozlišení 720x480. Video je poté anotováno jako celek a ne po segmentech.

UltimateLabeling

UltimateLabeling[2] je víceúčelové GUI (Graphical User Interface – grafické uživatelské rozhraní) pro popisování videa, které má integrovaný SOTA detektor a tracker. Na rozdíl od mé webové aplikace je tento systém určen pouze pro videa a neumí anotovat obrázky. Aplikace je napsána v pythonu. Pro detekování objektů ve videu je zde využita knihovna YOLO 4.1 a pro *tracking* objektu SiamMask. Dále umí rozpoznat pomocí knihovny OpenPifPaf polohu těl postav. UltimateLabeling vyžaduje SSH připojení na vzdálený server s GPU. Výhodou této bakalářské práce je, žádné takové připojení nepotřebuje. [6]

Computer Vision Annotation Tool

Zkratka *CVAT*[3] označuje zdarma použitelný, online interaktivní nástroj pro anotování videí a obrázků. Tento nastroj má více uživatelských rozhraní, která vznikla na základě návrhů od uživatelů [9].

[1] https://github.com/cvondrick/vatic
[2] https://github.com/alexandre01/UltimateLabeling
[3] https://github.com/opencv/cvat

OpenLabeling

OpenLabeling[4] je open source software pro značení objektů ve videu či obrázku, který byl napsán v pythonu. Pro detekování objektů využívá více knihoven [7].

LabelImg

LabelImg[5] je grafický anotační nástroj. Na rozdíl od předchozích nástrojů slouží pouze pro anotování obrázků, nikoliv videa. Je napsán v pythonu a pro vytvoření uživatelského rozhraní využívá knihovnu *Qt*. Výstup anotace je ve formátu XML, ale taktéž využívá formát YOLO. Tento nástroj lze používat na všech operačních systémech (Windows, Linux, MacOS)[31]. LabelImg neumí na rozdíl od bakalářské práce anotovat videa.

Labelme

Labelme[6] je grafický anotační nástroj napsaný v pythonu (Qt knihovna pro uživatelské rozhraní) a využívající *polylines* (křivky). Je inspirován jiným anotačním nástrojem[7] a umožňuje klasifikování objektů [40].

Nevýhodou Labelme je nutnost jeho instalace spolu se závislostmi, což činí prvotní spuštění složité a zdlouhavé. Uživatelé, kteří potřebují jednorázovou a rychlou anotaci, zvolí jiný existující nástroj.

Visual Object Tagging Tool

Open source anotační nástroj pro obrázky a videa. Je napsán jako webová aplikace v Reactu (TypeScript) využívající redux.

Rysy nástroje:

Možnost označení objektů v obrázku či videu.

Rozšířený model pro importování dat z lokálního nebo vzdáleného úložiště.

Rozšířený model pro exportování dat obsahujících anotace z lokálního nebo vzdáleného úložiště.

VoTT[8] napomáhá ve strojovém učení – přináší nová data [36]. Problémem VoTT je chybějící funkcionalita pro sledování objektů ve videu. Anotování videa snímek po snímku je pro uživatele náročné a zdlouhavé (zde by bylo možné doporučit tuto bakalářskou práci, která umí sledovat objekty ve videu). Navíc zde nejsou žádné knihovny, které by byly schopny detekovat objekty ve videu pomocí strojového učení.

imglab

Webová aplikace pro detekování objektů v obrázku[9], které mohou být využívány pro trénování dlib a jiných objektových detektorů. Jedná se o nezávislou platformu běžící přímo v prohlížeči a vyžadující pouze minimální CPU paměť. Prozatím využívá na vykreslování

[4] https://github.com/Cartucho/OpenLabeling
[5] https://github.com/tzutalin/labelImg
[6] https://github.com/wkentaro/labelme
[7] http://labelme.csail.mit.edu/Release3.0/
[8] https://github.com/microsoft/VoTT
[9] https://github.com/NaturalIntelligence/imglab

pouze body, kružnice, obdélníky a polygony. Další tvary (elipsy, přímky, křivky) budou přidány na vyžádání uživateli [16].

Tato bakalářská práce umí oproti imglab anotovat i videa a využívá knihovnu pro detekování a sledování objektů ve videu (u obrázku pouze detekování).

Yolo_mark

Nástroj[10] pro Windows a Linux, který je určený pouze pro trénování YOLOv3 a YOLOv2 4.1. Nejedná se o webovou aplikaci [41]. Výhodou této bakalářské práce oproti Yolo_mark je její univerzálnost a konfigurovatelnost.

PixelAnnotationTool

Software[11], který umožňuje manuální a rychlé anotování obrázků. Tato metoda je pseudomanuální, protože využívá algoritmus „watershed marked“[12] pro OpenCV. Nástrojem se manuálně označí část oblasti a pak se spustí algoritmus pro označení celé oblasti. Pokud je potřeba oprava, uživatel může překreslit označení v dotyčné oblasti novým anotováním.

Závislosti:

Qt >= 5.x.

CMake >= 2.8.x.

OpenCV >= 2.4.x.

Pro Windows Visual Studio >= 2015.

Nejedná se o webovou aplikace, proto je před prvním použitím nutná instalace [5]. PixelAnnotationTool pracuje pouze s obrázky a neumí anotovat videa.

ImageTagger

ImageTagger[13] je online nástroj pro kolektivní značení objektů na obrázku. [11] Další z nástrojů, který neslouží k anotování videí.

Alturos.ImageAnnotation

Účelem tohoto projektu[14] je získání trénovacích dat pro neuronové sítě. Obrázky jsou uloženy v úložišti, jako je například Amazon S3. Obrázky se na úložiště mohou nahrát jako archiv. Pro každý archiv pak lze nastavit „tag“. Tato informace je uložena v databázi, jako je například Amazon DynamoDB.

[10] https://github.com/AlexeyAB/Yolo_mark
[11] https://github.com/abreheret/PixelAnnotationTool
[12] https://docs.opencv.org/3.1.0/d7/d1b/group___imgproc___misc.html#ga3267243e4d3f95165d55a618c65ac6e1
[13] https://github.com/bit-bots/imagetagger
[14] https://github.com/AlturosDestinations/Alturos.ImageAnnotation

Rysy nástroje:

Skupinové anotování obrázků.

Verifikace anotačních dat.

Export pro YOLO (train.txt, test.txt, obj.names) s filtry.

Žádné závislosti pro vytvoření vlastního serveru.

Pro uživatele je poněkud nepohodlné, že pomocí Alturos.ImageAnnotation nemůže sledovat objekty ve videu a musí ručně spustit skript, který rozdělí videa na snímky[2], pokud chce anotovat videa. Tento krok provádí tato bakalářská práce automaticky.

DeepLabel

DeepLabel[15] je platforma pro anotování obrázků s možností detekování objektů. Typickým příkladem užití tohoto programu je značení významných objektů. Výsledky jsou poté využity jako vstup pro aplikace využívající strojové učení. DeepLabel lze využívat na platformách Windows, Linux a MacOS [33]. Další z nástrojů, který oproti této práci neumí anotovat videa.

MedTagger

Framework pro kolektivní získávání anotací dat ze zdravotnictví. Hlavním cílem projektu bylo navrhnout a vyvinout software, který pomůže agregovat a značit obrovské množství dat z lékařských snímků. Dále pak projekt poskytuje algoritmus pro validaci již anotovaných snímků.

MedTagger[16] je stále v procesu vývoje. Proto se může stát, že některé rysy tohoto softwaru se mohou v novějších verzích změnit a nemusí být zpětně kompatibilní. Dokumentace se nachází zde[17][17].

Tento software není vůbec univerzální, proto bych uživatelům, kteří nepotřebují anotovat medicínské obrázky (rentgenové snímky atd.) doporučil využít spíše moji webovou aplikaci, která umí navíc i anotovat videa a sledovat objekty.

Labelbox

Nástroj[18] pro anotování dat a následné vytvoření aplikací s umělou inteligencí. Je zde možnost vytvoření vlastního prostředí, které bude přesně vyhovovat požadovanému úkolu. Ale existují i již předem vytvořená prostředí, která jsou všechna open source [28].

turktool

Turktool[19] je aplikace napsaná v Reactu pro anotování obrázků a využívající „bounding boxy“. Lze ji připojit na Amazon Mechanical Turk nebo na vlastní backend. Není tu podporována klasifikace objektů!

[15] https://github.com/jveitchmichaelis/deeplabel
[16] https://github.com/medtagger/MedTagger
[17] https://github.com/medtagger/MedTagger/tree/master/docs
[18] https://github.com/Labelbox/Labelbox
[19] https://github.com/jaxony/turktool

Rysy nástroje:

Využití klávesových zkratek pro smazání nebo změnu „bounding boxů“.

Nahrávání anotací na Mechanical Turk pomocí klávesnice.

Získání obrázku z URL adresy.

Tento anotační nástroj pouze pro obrázky je velmi jednoduchý a zvládá jen základní úkoly [30]. Nezvládá anotování videí a pro polygony neexistují žádné přednastavené anotační formuláře, které by ulehčovaly uživateli práci.

Pixie

Pixie[20] je aplikace pro anotování obrázků, která podporuje vykreslování „bounding boxů“, polygonů a sémantické segmentace.

Tento nástroj je napsán v Javě (java runtime environment JRE je nezbytný pro chod aplikace) a byl testován s využitím JRE 8 od firmy Oracle i OpenJDK [23].

Nejedná se tedy o online nástroj, nýbrž je nutné jeho stažení a následná instalace. Pokud by chtěl uživatel anotovat video, je nutné ho anotovat snímek po snímku. Bohužel tu neexistuje žádné sledování objektů nebo aspoň jejich detekce.

OpenLabeler

OpenLabeler[21] je open source aplikace pro anotování objektů. Dokáže generovat soubory ve formátu PASCAL VOC XML, které jsou využívány pro trénování umělé inteligence. Unikátním aspektem této aplikace je její schopnost používat pro zlepšení přesnosti a urychlení procesu anotování dedukci [22]. Další z nástrojů, který na rozdíl od této bakalářské práce není online službou. Tudíž není vhodný pro uživatele, kteří potřebují pouze jednorázovou anotaci. Navíc OpenLabeler je určen pouze pro anotování obrázků s jasným cílem.

Anno-Mage: A Semi Automatic Image Annotation Tool

Anno-Mage: A Semi Automatic Image Annotation Tool[22] je částečně automatický anotační nástroj obrázků s algoritmem RetinaNet pro napovídání. Algoritmus navrhne 80 objektových tříd z MS COCO datasetu pomocí dříve trénovaného RetinaNet modelu [3]. Tato bakalářská práci sice neumí napovídat, ale obsahuje nabídku nejčastějších objektů, což vede k podobně rychlé anotaci. Navíc je-li u mé práce zapnut režim částečně automatizovaného anotování, nedochází k našeptávání, ale rovnou k určování objektů nejen v obrázku, ale i ve videu.

makesense.ai

Motto – „For AI to be free we need not just Open Source, but also a strong Open Data movement.“. Jak motto napovídá, makesense[23] je open source webová aplikace pro značení objektů ve fotografiích. Vhodné pro „deeplearning“ projekty (rychlé a jednoduché předpracování datasetů). Aplikace je napsaná v TypeScriptu [29]. Jedná se o další z anotačních nástrojů s jasným záměrem, který není určen pro anotování videí.

[20] https://github.com/buni-rock/Pixie
[21] https://github.com/kinhong/OpenLabeler
[22] https://github.com/virajmavani/semi-auto-image-annotation-tool
[23] https://github.com/SkalskiP/make-sense

LOST - Label Objects and Save Time

Částečně automatický flexibilní webový framework pro anotování obrázků. Poskytuje mnoho anotačních rozhraní pro rychlé anotování.

LOST[24] je flexibilní od doby, kdy dovoluje, aby více odlišných anotačních rozhraní, nástrojů nebo algoritmů bylo kombinováno dohromady.

Nasazení může probíhat více způsoby. Využitím dockeru na lokální stroj nebo nasazením na webový server, na kterém poté bude anotování dostupné po celém světě [14]. Toto nasazení může být náročné, a proto bych doporučil raději využívat webovou aplikaci, jako je tato bakalářská práce, která navíc umí i anotování videí.

[24] https://github.com/l3p-cv/lost

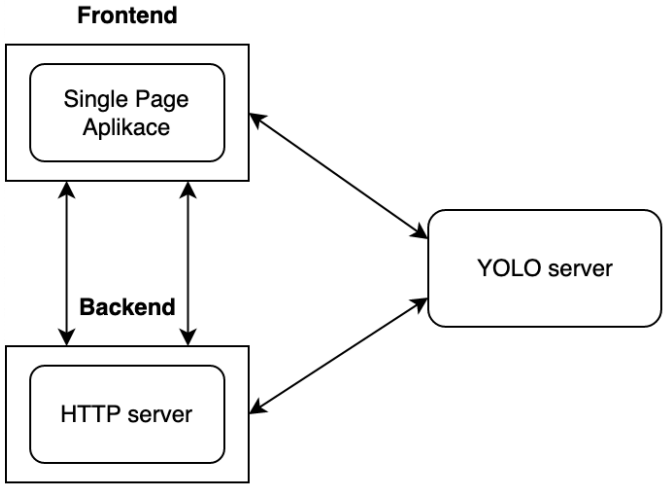
Kapitola 6

Návrh webové aplikace

Návrh webové aplikace by bylo možné rozdělit na část backendovou (HTTP server Express.js, YOLO server) a frontendovou (React, Material-UI). V této kapitole bude popsána celková struktura webové aplikace i návrhy jednotlivých komponent.

6.1 Architektura

Tento anotační nástroj je standardní webovou aplikací, proto je pro samotné zobrazení stránky využíván HTTP server. Frontend se tedy zobrazí pomocí HTTP serveru a veškerá data jsou načítána z backendu aplikace. Pro detekování objektů ve videu a obrázku je potom použita knihovna YOLO, která běží odděleně od webové aplikace – nachází se na YOLO serveru.



Obrázek 6.1: Diagram architektury webové aplikace.

6.2 Základní princip webové aplikace

Při vytváření konceptu webové hry jsem se inspiroval různými koncepty z jiných anotačních nástrojů. Webová aplikace obsahuje určování deepfakes. Tuto funkcionalitu neobsahuje žádná jiná anotační aplikace, kterou jsem během vymýšlení konceptu prostudoval. Deepfakes se používají stále častěji, jsou stále dokonalejší a je těžké je rozeznat, proto jsem

považoval za dobré přidat do této webové aplikace uživatelské rozpoznávání. Hlavním účelem webové aplikace však stále zůstává rozpoznávání objektů v obrázku a videu. V případě videa i sledování objektů. Anotování může být manuální a nebo částečně automatizované pomocí knihovny YOLO. Důležité je získat informace ohledně sledovaných objektů od uživatele. K tomu slouží anotační formulář, který je součástí každého vykresleného polygonu. V obraze se některé typy objektů objevují častěji než jiné (automobily, osoby, motorky atd.). Proto aplikace umožňuje při anotování objektů výběr z nejčastějších typů.

Výhodou webové aplikace je její konfigurace, která byla navržena, tak aby byl uživatel schopen si celou aplikaci nastavit podle svých potřeb. Jedná se o zapínání a vypínání všech funkcionalit webové aplikace.

Deepfakes

Fotografie a gify byly již staženy na server pro urychlení práce, aby je uživatel nemusel pokaždé nahrávat. Tato funkcionalita byla navržena tak, aby byla co nejtriviálnější a uživateli nezabírala moc času. Proto se jedná jen o klasické klikání na tlačítka, která reprezentují informaci o daném deepfake.

Anotování videa

Hlavním myšlenkou je rozdělení videa na snímky. Ty jsou pak postupně a rychle načítány, což vede k dojmu přehrávání videa. Spolu s každým snímkem jsou načítány i uložené anotace tohoto snímku. Při anotování konkrétního snímku dojde k pozastavení přehrávání videa a otevření modálního okna, které obsahuje konkrétní snímek a všechny potřebné funkcionality k určování anotace snímku.

Anotování obrázku

Návrh na anotování obrázku byl převzat z návrhu modálního okna pro anotování konkrétního snímku videa. Spolu s nahráním obrázku jsou nahrány i jeho uložené anotace. Veškerá potřebná funkcionalita k anotování se nachází pod obrázkem. Není zde žádné modální okno.

Kapitola 7

Implementace webové aplikace

Tato kapitola se zaměřuje na popis implementace webové aplikace. Budou zde popsány jednotlivé algoritmy a komponenty uplatněné při této implementaci. Taktéž zde budou popsány problémy, jež bylo nutno během vývoje vyřešit. Celá aplikace je rozdělena na stránky; ty se skládají z komponent, které je možné znovu použít.

7.1 Stránky

Jak již bylo zmíněno, celá aplikace je rozdělena na stránky. Všechny jsou složeny z komponent. Každá stránka je dostupná pomocí vlastní URL adresy ve tvaru http://{URL}/{pageName}, kde *URL* je doména a *pageName* je identifikátor stránky. Pokud pracujeme na localhost s portem 3000, potom bude tato URL adresa vypadat následovně: http://localhost:3000/video (stránka pro anotování videí). Při prvním spuštění aplikace se stránka sestaví a načte na straně serveru – takzvaný „server side rendering“. U dalších přechodů pak sestavení stránek již nenastává na straně serveru, ale na straně klienta – takzvaný „client side rendering“.

Celá webová aplikace se skládá z 5 stránek. Uživatelskou přívětivost zajišťuje úvodní stránka. Ta se po spuštění serveru vykreslí jako první. Tato stránka má funkci pouze informativní. Jsou zde informace ohledně celé aplikace a po kliknutí na modální okna může uživatel získat také informace o jednotlivých stránkách. Najdeme zde taktéž odkazy na stránky, které už dále slouží pro anotování. Jednou z těchto stránek je *deepfake* stránka, na kterou je možné se dostat přes URL */deeps*. Zde se nachází veškerá funkcionalita pro získávání informací ohledně deepfakes 4.3 fotografií a gifů. Druhou anotační stránkou je stránka s videi s adresou */video*. Ta je však mnohem složitější než stránka předcházející. Důvodem je využívání knihovny YOLO 4.1 pro detekování objektů ve videu. Taktéž samotné uživatelské anotování videí je mnohem složitější než určování deepfakes. Třetí anotační stránka popisuje práci s obrázky. Zde se rovněž může využívat knihovna YOLO. Poslední stránka je pak využívána pro konfigurační změny celé aplikace.

Do aplikace je pro usnadnění práce implementována komponenta *MainAppBar*, která obsahuje odkazy na všechny stránky. Nemusíme se tudíž dostávat na jednotlivé stránky přes úvodní stránku nebo přes vkládání URL adres.

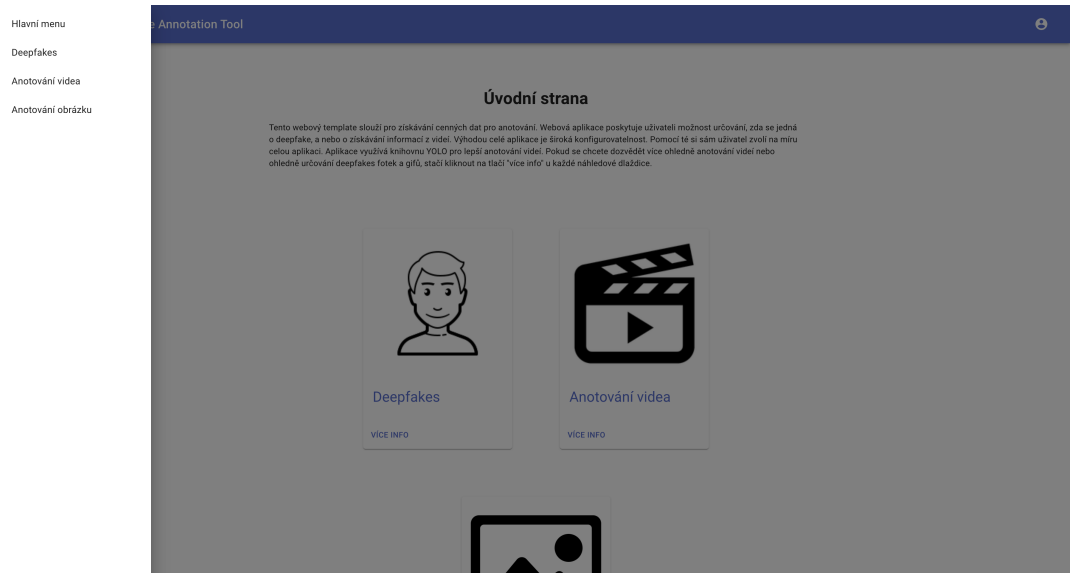
7.2 Komponenty

Tato sekce bude popisovat všechny vytvořené komponenty. Funkcionalita, která s těmito komponentami souvisí, pak bude vysvětlena v jiné sekci. Komponenty jsou uloženy v jedné

složce *components*, ve které jsou dále děleny do složek podle jejich užití. Toto dělení odpovídá každé stránce. Tudíž komponenty potřebné pro každou stránku jsou odděleny od ostatních. To napomáhá lepší orientaci. Jediná konfigurační stránka neobsahuje žádné vlastní komponenty. Dále existují dvě globální komponenty, které jsou využity na všech stránkách webové aplikace včetně konfigurační stránky. Je-li zmíněno, že do některé kategorie patří určitý počet komponent, jsou tím myšleny pouze komponenty, které uživatel vytvořil sám. Nepočítají se do toho komponenty z knihovny *Material-UI* 4.4.

Globální komponenty

První kategorií komponent jsou globální komponenty. Jak bylo již zmíněno, tyto komponenty jsou využívány na všech stránkách. Jednou z nich je *HamburgerMenu* komponenta. Ta vytváří tlačítko, které otevírá a zavírá postranní menu. V něm se nacházejí reference na všechny stránky kromě konfigurační (reference na konfigurační stránku se nachází na jiném místě, viz níže). Cílem implementace je usnadnění práce. Chce-li uživatel přejít na jinou stránku, nemusí se vždy vracet na hlavní stranu. Tato komponenta je poté importována do druhé globální komponenty, kterou je *MainAppBar*. Ta je zapotřebí k vytvoření horního panelu. Kromě *HamburgerMenu* komponenty tento panel obsahuje název celé webové aplikace a referenci na konfigurační stránku. Níže jsou ukázky obou globálních komponent.



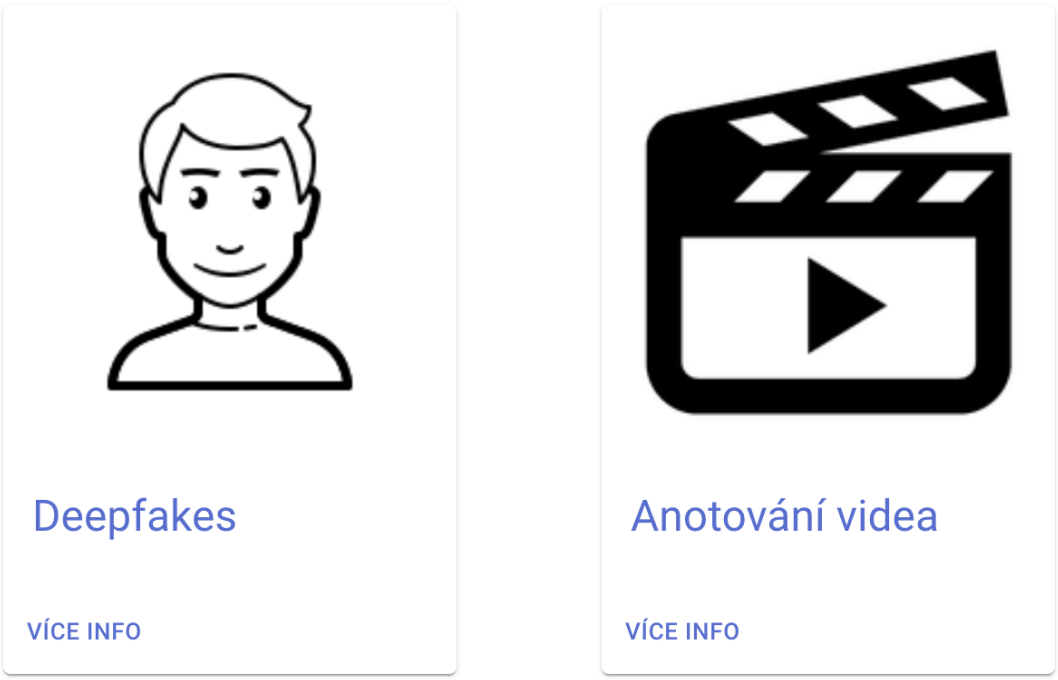
Obrázek 7.1: Ukázka postranního menu.

Obrázek 7.2: Ukázka horního panelu.

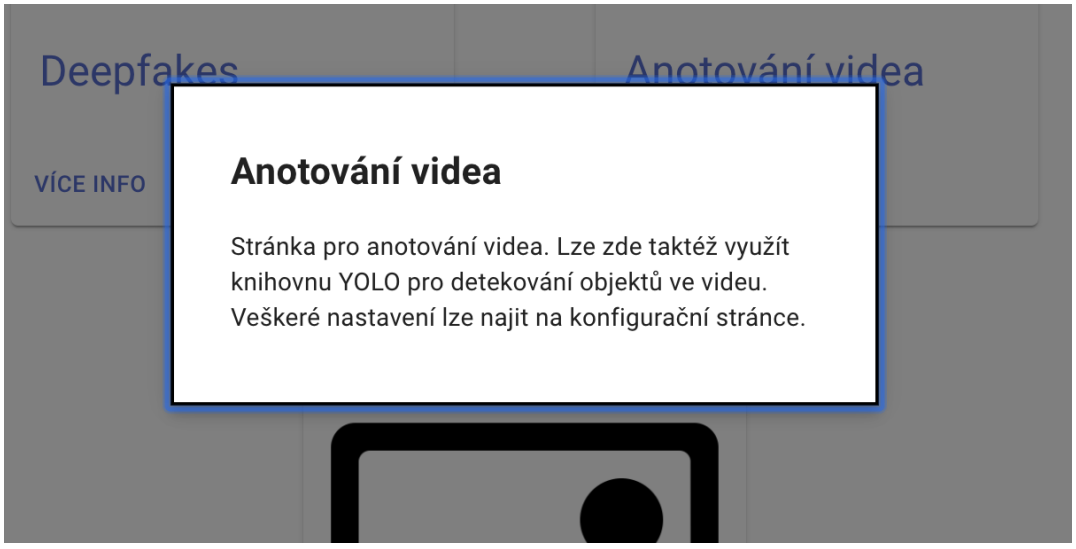
Komponenty úvodní stránky

Do této sekce patří komponenty, ze kterých se skládá úvodní strana. Tvoří ji dvě komponenty. První z nich je komponenta *IconCard* sloužící k vytvoření karet pro stránku s deepfakes, pro stránku pro anotování videa a pro stránku pro anotování obrázku. Každá

karta obsahuje obrázek, který naznačuje funkcionalitu stránky, referenci na danou stránku a druhou komponentu *ModalButton*. Tato komponenta slouží k vypsání informací ohledně stránky, na kterou tato karta má referenci. Po kliknutí na tlačítko *VÍCE INFO* se právě tato zmíněná informace zobrazí.



Obrázek 7.3: Ukázka komponent IconCard.

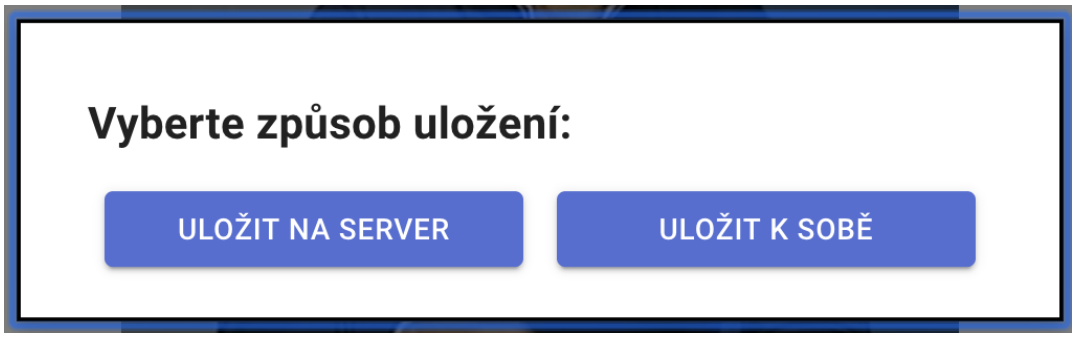


Obrázek 7.4: Ukázka komponenty ModalButton.

Komponenty stránky deepfakes

Tato stránka se skládá ze tří vytvořených komponent. Jedna je k vytvoření funkcionality pro určování , zda se jedná o deepfakes či nikoliv, a přepínání mezi nimi. Druhá pak slouží k implementaci funkcionality pro ukládání výsledků. Výsledky poté mohou být uloženy buď na server, nebo lokálně na uživatelský počítač. Poslední obsahuje funkcionalitu pro nahrávání nových deepfakes na server. Její implementace byla inspirována zde[1]

Obrázek 7.5: Ukázka komponenty pro určování deepfakes.

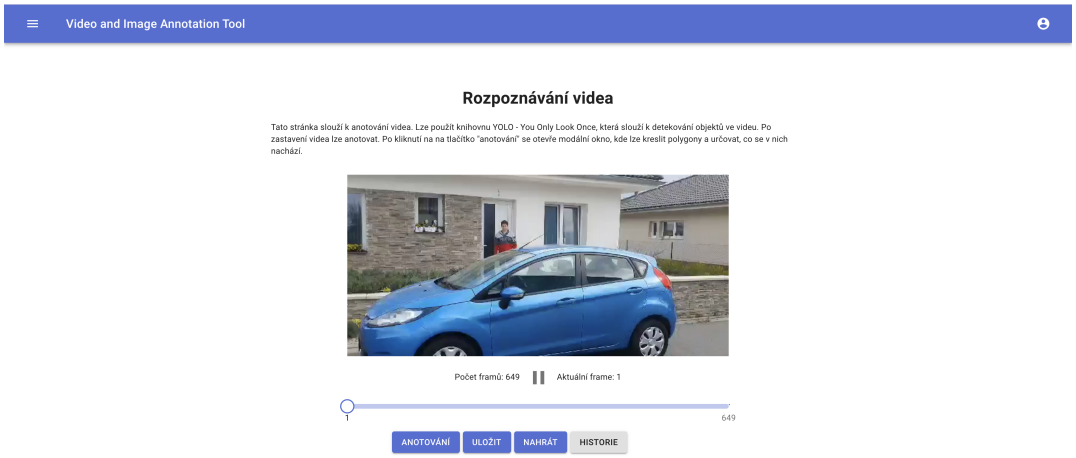


Obrázek 7.6: Ukázka komponenty pro ukládání výsledků.

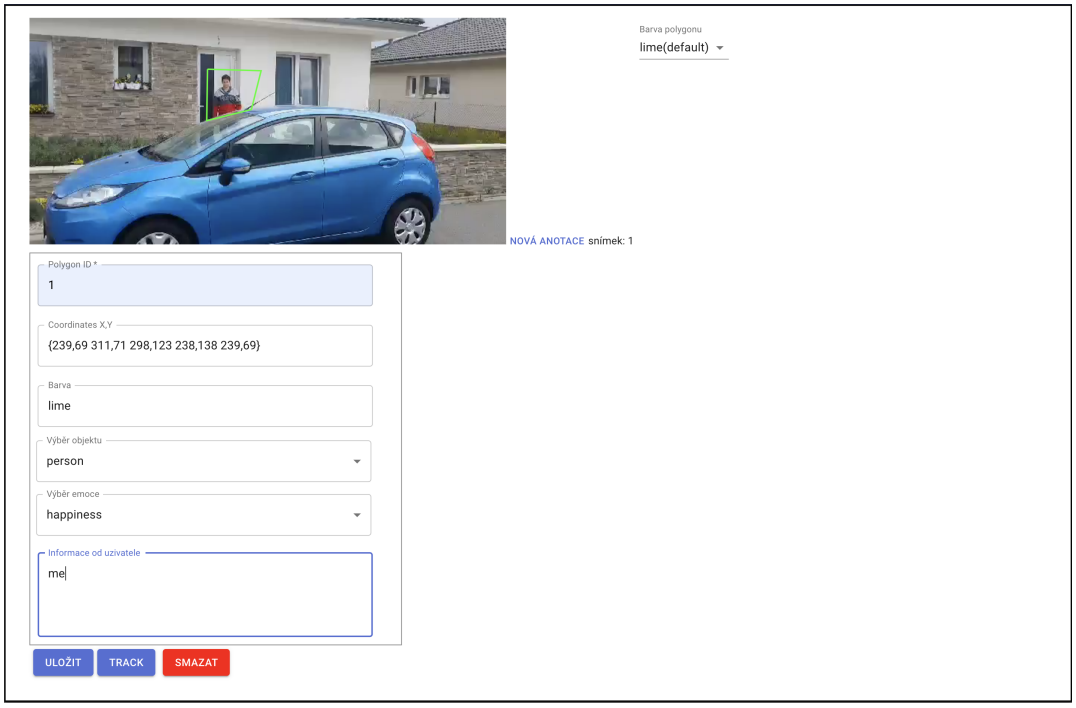
[1] https://react-dropzone.js.org/#!/Opening%20File%20Dialog%20Programmatically

Komponenty stránky pro anotování videa

Implementování stránky pro anotování videa obsahuje nejvíce funkcionality a navíc je funkcionalita poměrně složitá. Proto tato část obsahuje taktéž i nejvíce vlastně vytvořených komponent. Celkově jich je pět, přičemž dvě z nich jsou si dost podobné – *videoPlayer* a *yoloPlayer*. *VideoPlayer* je komponenta, která obsahuje veškerou funkcionalitu k nahrání lokálního videa, jeho defragmentaci na snímky, přehrávání videa a ukládání výsledků. Tato komponenta řeší pouze přehrávání videa bez využití YOLO knihovny 4.1. Právě pro řešení videa s využitím YOLO knihovny je použita druhá komponenta, *yoloPlayer*. Obsahuje stejnou funkcionalitu jako předchozí knihovna, ale jak již vyplývá z názvu, řeší tuto funkcionalitu pro anotování využívající knihovnu YOLO. Rozdíly mezi těmito „přehrávači“ budou řešeny dále, viz 7.6. Další komponentou je *VideoHistory*, která řeší nahrávání již přehraných videí z historie videí. Mimo jiné dále dokáže tato videa z historie vymazat. Funkcionalita pro uživatelské anotování je řešena v komponentě *AnnotationButton*. Je tvořena formou modálního okna, které obsahuje snímek videa, vykreslené polygony tohoto snímku, informace o videu (název, číslo snímku atd.) a poslední komponentu, *AnnotationForm*. Ta vytváří formulář pro získávání informací o oblasti, která se nachází uvnitř daného polygonu.



Obrázek 7.7: Ukázka komponenty přehrávače.



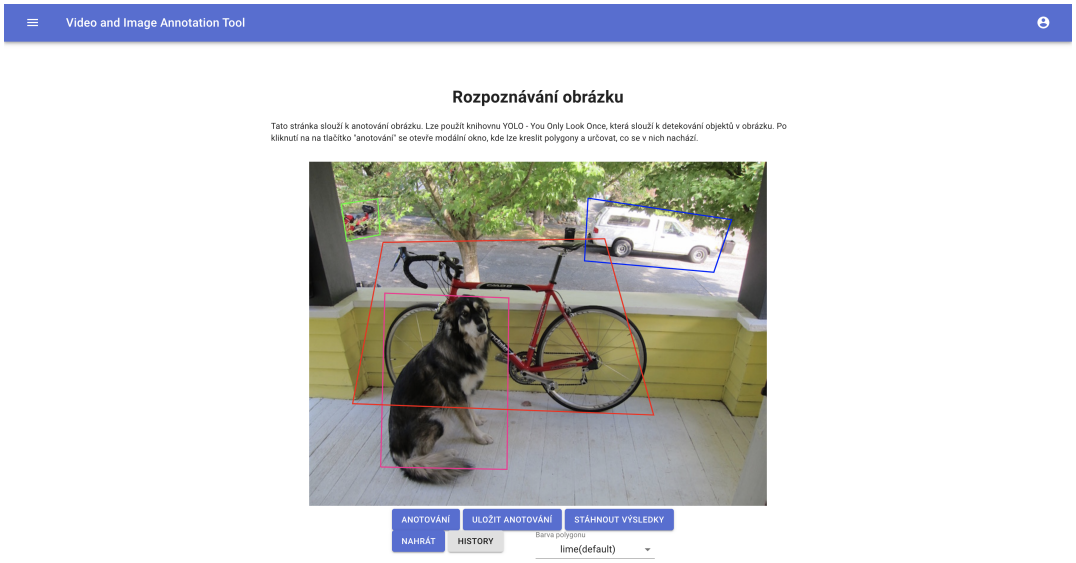
Obrázek 7.8: Ukázka komponenty pro anotování snímku videa.

Komponenty stránky pro anotování obrázku

Stránka pro anotování obrázků se skládá ze dvou vlastních komponent a jedné komponenty pro stránku s videem – *annotationForm*. O této komponentě pro videa se můžete dozvědět více informací výše, viz 7.2. Zde budou popsány pouze komponenty pro obrázky.

Main komponenta obsahuje veškerou funkcionalitu pro nahrání obrázku, anotování obrázku, ukládání a stahování výsledků. Druhou komponentou této stránky je *ImageHistory*, která řeší nahrávání a mazání obrázků z historie.

Lze zde vidět podobnost s komponentami pro stránku anotující videa.



Obrázek 7.9: Ukázka komponenty zobrazující obrázek. Obrázek psa byl převzat z githubu https://github.com/AlexeyAB/darknet/tree/master/data.

7.3 YOLO server

YOLO server je oddělený od samotné webové aplikace. Server běží a naslouchá na portu 3001. Webová anotační aplikace posílá REST požadavek na YOLO server. Tento požadavek obsahuje video nebo obrázek, ve kterém mají býti detekovány objekty pomocí knihovny YOLO.

Po obdržení je požadavek s videem uložen na server a je zavolána knihovna YOLO. Ta začne zpracovávat video a detekovat objekty v něm. Výsledky jsou téměř ihned k dispozici díky streamu. Odpověď z YOLO knihovny se objeví na dvou portech, které uživatel volí v konfigurační stránce (výchozí 8090 a 8070). První port (8090) je využíván k MJPEG streamu. To znamená, že jednotlivé zpracované snímky videa jsou zasílány na tento port v podobě streamu a na straně webové aplikace jsou zachytávány. Druhý port (8070) slouží k vysílání JSON streamu, ve kterém jsou uloženy informace o objektech detekovaných v každém snímku. Pokud by knihovna YOLO neposlala informace ke snímku a uživatel by chtěl vidět tyto informace, vypíše se chybová hláška „YOLO server (knihovna) neposlal informace k tomuto snímku!“

Pokud je požadováno detekování na videu z historie, neposílá se v požadavku video, pouze jeho jméno. Podle něho si YOLO server načte dané video ze svých uložených videí. Zbytek detekování je stejný.

Pokud je zpracováváno video a přijde nový požadavek, zastaví YOLO server všechny procesy na obou definovaných portech. Zpracuje nově příchozí požadavek a detekování objektu nového objektu započne. Akce je prováděna pomocí skriptu, který za využití terminálových příkazů zjistí všechny procesy běžící na zadaném portu a pak všechny tyto procesy ukončí. Tento skript byl převzat[2].

V případě, kdy je obdržen požadavek s obrázkem, je postup stejný jako u videa. Požadavek je taktéž uložen na YOLO server a po skončení detekce objektů pomocí knihovny YOLO je výsledek uložen do souboru *predictions.png*. Ten je následně poslán v odpovědi

[2]https://gist.github.com/malyw/0a6e1d528b32c1be6c4390b6ab234180#file-killprocessbyport-js

zpět na server webové aplikace. Proces anotování knihovny není automaticky ukončen, proto je nutné po každým anotování tento proces ukončit. K tomu je využíván terminálový příkaz, který je automaticky spouštěn. Proces detekování objektů v obrázku z historie je stejný jako u videa.

YOLO server prozatím funguje pouze v operačním systému Windows. V rámci zlepšování tohoto systém by se v budoucnu přidala i implementace YOLO serveru na ostatní operační systémy, především Linux.

7.4 Implementace konfigurace aplikace

Šablona pro konfigurační stránku je uložena na serveru v podobě JSON objektu. Před vyrenderováním stránky je JSON soubor načten a podle něj jsou vyrenderována potřebná tlačítka a přepínátka stránky. Tudíž pokud je potřeba přidat do aplikace jakoukoliv konfiguraci, je rozšíření konfigurační stránky velmi jednoduché, stačí změnit tento JSON soubor.

Veškeré změny v konfiguraci jsou promítány do stavu konfigurační stránky. Tyto změny nejsou trvalé. Ale jakmile uživatel zvolí uložení konfigurace, je momentální stav uložen do souboru na serveru a následně je uživatel přesměrován na hlavní stránku. Tento soubor je poté načítán v ostatních částech aplikace a určuje konfiguraci celé webové aplikace.

Taktéž je možné vrátit celý konfigurační stav do prvotního stavu. K tomu slouží tlačítko reset na konfigurační stránce.

7.5 Implementace deepfakes

Výše 7.2 bylo vysvětleno, které komponenty byly použity pro vytvoření „deepfake stránky“. Zde pak bude vysvětlena funkcionalita, která byla implementována pro správné chování.

Před samotným vyrenderováním stránky se načtou z backendu webové stránky všechny deepfakes fotografie a gify spolu s jejich jmény, které byly staženy z githubu[3]. Obojí se uloží do pole. Až poté dojde k vyrenderování celé deepfakes stránky. Pak se na stránce objeví první obrázek či gif podle uložení na serveru. Pod ním se nachází jméno tohoto „deepfake“ a tlačítka, která odpovídají konfiguraci z konfigurační stránky. Dvě na přecházení mezi „deepfakes“ a tři na určovaní, zda se o deepfake jedná. Deepfake – jedná se o deepfakes, není deepfake – nejedná se o deepfakes, nelze poznat – uživatel nemůže poznat, o co se jedná. Klikne-li uživatel na jedno z těchto tří tlačítek, zavolá se funkce, která vytváří JSON objekt (tento objekt si poté může uložit). Zároveň se také přepne na nový „deepfake“.

Ukládání

Je-li povoleno v konfiguraci webové aplikaci automatické ukládání výsledků, znamená to, že pokaždé, když je změněný JSON objekt (uživatel klikl na některý ze tří tlačítek), je tento výsledek aktualizován a uložen na server.

Toto automatické ukládání je v zásadě totožné, jako když si uživatel vybere *uložení výsledků na server*. V tomto případě se vezme celý JSON objekt, který je vytvořen pomocí určování „deepfakes“ fotografií či gifů a uloží se do souboru na server. V druhém případě, kdy si uživatel zvolí uložení lokálně k sobě, je vytvořen nejdříve JSON soubor obsahující právě tento JSON objekt a automaticky uložen k uživateli.

[3] https://github.com/ondyari/FaceForensics/tree/master/images

Při ukládání výsledků na server je součástí názvu uloženého souboru taktéž identifikátor za podtržítkem – results_{userID}.json. To reprezentuje uživatele. V současném stavu aplikace je tento identifikátor pevně dán v konfiguraci celé aplikace. Tato funkcionalita je zde pro budoucí plánované rozšíření webové aplikace, kdy se přidá autentizace uživatelů a napojení databáze.

7.6 Implementace anotování videa a obrázku

V této sekci bude představena implementace všech funkcionalit potřebných pro anotování videa a obrázku. Nejedná se jenom o uživatelské anotování, ale i použití knihovny YOLO, vysvětlení počítání vzdáleností mezi polygony, počítání kroků pro každý polygon atd. Informace ohledně použitých komponent je možné zjistit v sekci komponent 7.2 a 7.2.

Přehrávání videa bez použití YOLO

Začátek přehrávání videa může být vyvolán dvěma možnými způsoby. Buď uživatel nahraje z lokálního souboru video, které chce anotovat, a nebo využije historii videí a požadované video si nahraje odtud. Samozřejmě, v historii jsou pouze videa, která uživatel již někdy předtím anotoval. Totéž platí i pro nahrání obrázku před začátkem anotování.

Vybere-li si uživatel, že chce anotovat nové video, zobrazí se mu pomocí funkce „fileDialog“, která je dostupná z npm balíčků 4.4, „file dialog“. Zde jsou vyfiltrovaná pouze videa. Po zvolení videa nastává extrahování částí a vytvoření snímků tohoto videa ve formátu PNG a následné uložení na serveru. Po této extrakci je na frontendu uložen počet těchto snímků. Asynchronně s extrakcí je ještě načten z backendu aplikace soubor, ve kterém je uložena předchozí uživatelská anotace pro dané video.

Přehrávání je poté řešeno tak, že je spuštěna funkce, která inkrementuje v cyklu hodnotu reprezentující snímek videa. Při pozastavení/spuštění videa je tento cyklus pozastaven/spuštěn. Z toho lze i vyvodit, jak funguje přetáčení videa. Posuvný slider, jehož rozsah odpovídá množství snímků, se stará o funkcionalitu přetáčení videa. Posune-li uživatel tento slider, je vrácená hodnota reprezentující snímek, na který se má video posunout. Zároveň je video pozastaveno.

Veškeré informace ohledně počtu snímků a aktuálním snímku vidí uživatel hned pod videem.

Přehrávání videa s použitím YOLO

Přehrávání bez použití knihovny YOLO i s jejím užitím si je velmi podobné. I tady je možnost nahrát lokální video či pustit video z historie.

Před nahráním nového videa (i načteného z historie) je nutno nejdříve odstranit předchozí anotované video. Tím je myšleno odstranění všech snímků, které byly zaslány YOLO serverem 7.3 a soubory obsahující JSON informace o anotovaných snímcích. Poté je umožněno uživateli vybrat video, které je následně posláno na YOLO server. Ten vrací jednotlivé již anotované snímky a k nim potřebné informace v podobě JSON formátu. Obojí je uloženo na serveru.

Co se týče samotného přehrávání videa, je postup velmi podobný jako u přehrávání bez použití YOLO knihovny. Odlišné je to, že velikost slideru neodpovídá všem snímkům videa. Odpovídá pouze snímkům, které byly přijaty z YOLO serveru. Tato velikost slideru se během přehrávání videa aktualizuje, dokud nejsou přijaty veškeré snímky.

Anotační okno

Pokud si uživatel přeje začít anotovat videa, stačí kliknout na tlačítko pod videem s názvem *Anotování*. Video se automaticky pozastaví a otevře se modální okno pro anotování. V něm se nachází aktuální snímek, který má být uživatelem anotován, pořadí tohoto snímku, tlačítko pro začátek vykreslování polygonů a výběr barvy polygonů. Při začátku kreslení polygonů do snímku se zobrazí i formulář pro vyplnění údajů o konkrétním polygonu – ID, souřadnice, barva polygonu, jaký objekt se v něm nachází (automobil, osoba, pes, kolo atd.) a dodatečné informace.

Vykreslování polygonů

Implementace vykreslování polygonů je založená na principu SVG (Scalable Vector Graphics) 4.4 a pro videa i obrázky je tato funkcionalita totožná.

Po načtení požadovaného objektu (videa či obrázku) jsou extrahovány dimenze – šířka a výška. Ty pak určují velikost překrývajících se oblastí objektu a SVG oblasti. Poté je možné do SVG oblasti vykreslovat všechny typy objektů, viz 4.4. Pokud existuje soubor s uživatelskou anotací, který není prázdný, je načten jeho obsah a všechny polygony pro daný snímek videa či obrázku jsou vykresleny pomocí SVG objektu *polyline*. Dále jsou zde vykreslovány nové polygony od uživatele.

Jakmile chce uživatel vykreslovat nový polygon, stačí kliknout na tlačítko „anotování“ pod obrázkem nebo „nová anotace“ v modálním okně pro anotování videa. Začátek anotování lze poznat také pomocí změny kurzoru. Po najetí kurzoru přes oblast obrázku/snímku (SVG oblasti) se změní z „default“ na „crosshair“. Poté jsou po kliknutí zachyceny souřadnice X,Y daného bodu v obrázku a jsou uloženy do pole. Také se provede kontrola, zda daný bod není zároveň koncový, tzn. má stejné souřadnice jako již existující bod. Pokud kontrola rozpozná konec polygonu, je vykreslování ukončeno a polygon je uložen v poli. Kliknout na dva stejné body je však pro uživatele náročné a nepřívětivé. Pro usnadnění proto existuje kolem bodu okruh, takže stačí, když uživatel zasáhne tento okruh. To je pak vyhodnoceno jako kliknutí na bod. Tato vzdálenost činí 5 pixelů. Nejedná-li se o koncový bod, jsou tyto souřadnice uloženy a je vykreslena přímka mezi posledním bodem a bodem předchozím. Tímto způsobem se po bodech vykresluje celý polygon, až do okamžiku, kdy je zjištěn koncový bod. Výsledné souřadnice pak mají následující tvar:

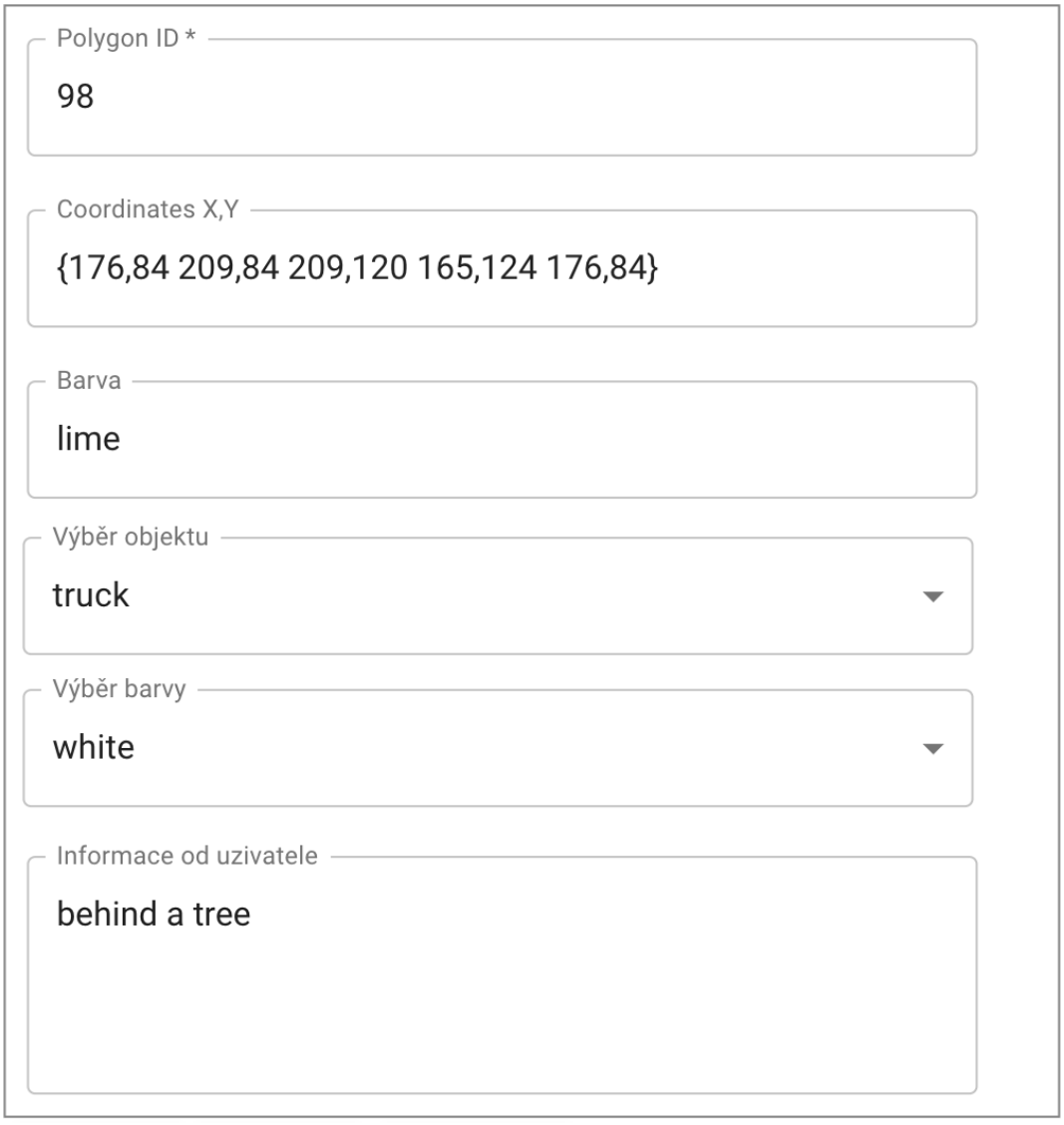
"x1,y1 x2,y2 x3,y3 x4,y4 x1,y1"

Tento finální polygon pak bude mít tvar čtverce. Zároveň se začátkem vykreslování polygonu se uživateli objeví i formulář pro zachycení informací ohledně daného polygonu. Více informací v sekci o formuláři 7.6.

Anotační formulář

Anotační formulář se skládá ze dvou komponent – textbox a select. Slouží k získání informací o polygonu od uživatele. Jsou zde i dvě předem doplněné informace, které jsou jen pro čtení a uživatel je nemůže měnit. Jsou to informace týkající se souřadnic a barvy polygonů. Poté se v anotačním formuláři nachází „Polygon ID“ textbox pro určení identifikátoru polygonu. Ten se pak dále používá u mazání polygonů 7.6 nebo „trackování“ objektů. Podle ID se zjistí začáteční a koncový polygon, více informací ohledně tohoto viz 7.6. Dále se tu nachází select „Výběr objektu“. Jedná se o seznam nejběžnějších objektů, které se nejčastěji vyskytují v obraze – například auto, osoba, kolo, atd. Tento select je implementován, aby uživatel nemusel opakovaně a zdlouhavě vypisovat často se opakující objekty, ale jenom

je jednoduše vybral ze seznamu. Vybere-li uživatel ze seznamu objekt typu „osoba“ nebo „automobil“/„nákladní automobil“, objeví se ve formuláři druhý select. Ten slouží jako doplňující informace k předchozímu selectu. U osoby to je výběr základních emocí člověka, u osobního či nákladního automobilu potom informace ohledně jeho barvy. Poslední částí, ze které se skládá anotační formulář, je textbox „Informace od uživatele“. Ten slouží k získání dodatečných informací o polygonu. Například pokud polygon obsahuje jiný objekt, než který lze vybrat ze seznamu nebo kdyby uživatel rád dodal nějaké informace k vybranému objektu.



Obrázek 7.10: Ukázka anotačního formuláře.

Mazání polygonů

Pokud chce uživatel smazat vytvořený polygon, musí nejdříve vybrat z pole ten správný. To lze poznat tak, že uživatel vybere anotační formulář obsahující správný ID polygonu (možné poznat i podle souřadnic). Po vybrání správného polygonu stačí kliknout na tla-

čítko „smazat“. Proces mazání je pak následující: nahraje se pole všech polygonů, které se nacházejí na obrázku nebo snímku videa. Pak je v tomto poli vyhledán polygon podle jeho identifikátoru a následně smazán. Dále musí dojít ještě k posunutí všech polygonů v poli, které se nacházejí za smazaným polygonem o jeden index dopředu. Poté se tato změna musí přenést do všech částí aplikace, jako jsou například data pro anotační formuláře.

Ukládání polygonů

Jakmile uživatel dokončí anotování daného snímku nebo obrázku, musí své anotace uložit, jinak budou ztraceny. Postupně se po jednom projdou všechny polygony. Pro každý z nich se sestaví JSON objekt v požadovaném formátu:

```
{
    id,
    coordinates,
    color,
    type,
    emotion,
    colorCar,
    information
}
```

Tyto objekty jsou pak následně uloženu do pole „polygons“. Do této chvíle je tento postup totožný s anotováním obrázku i videa. Dále se mírně liší.

U obrázku je pole polygonů posláno z frontendu aplikace na její backend, kde je implementována funkcionalita ukládání do souboru. Spolu s tímto polem je poslán i název obrázku. Výsledek anotování je pak uložen do souboru {název}.json, pokud není použita knihovna YOLO 4.1. Je-li tato knihovna použita, tak jméno souboru je {název}_yolo.json. Oba typy souborů jsou pak k nalezení ve složce *json/answers/image/userAnnotation/{uid}*.

V případě anotování videa jsou přidány k poli polygonů ještě informace ohledně snímku, který byl anotován. Jedná se o uložení čísla snímku (kolikátý snímek to je) a jeho indexu v poli snímků. Z těchto informací je pak sestaven další JSON objekt ve formátu:

```
{
    frameId,
    index,
    polygons[],
}
```

Tento objekt je pak poslán na backend webové aplikace, kde ukládání je stejné jako u obrázku s tím rozdílem, že výsledek je uložen ve složce *json/answers/video/userAnnotation/{uid}*.

Stažení výsledků

Jakmile je uživatel hotov s anotováním, může si stáhnout výsledky k sobě. Z backendu webové aplikace se načte požadovaný soubor podle toho, zda se jedná o anotování obrázku nebo videa a zda je využita knihovna YOLO 4.1. Tento soubor je poté z backendu poslán na frontend aplikace, kde je pak pomocí funkce *FileSaver.saveAs()* uložen soubor s výsledky k uživateli. Jedná se o soubor ve formátu JSON. Tento formát je vybrán z toho důvodu, že práce s ním je rychlá a snadná, uživatel tedy může snadno převést výsledky do tvaru, který

požaduje. To dělá anotační nástroj univerzální pro různé další použití. Funkce pro ukládání je dostupná z npm balíčků[4] 4.4.

Trackování (sledování) objektů

Předešlé rysy aplikace byly společné jako pro anotování obrázku i videa. Trackování objektů je však funkcionalita využitá pouze u anotování videa. Jedná se o zachycení jednoho objektu v obraze a jeho následné sledování až do určité doby. Tu určí sám uživatel, ale doba může být určena i podle výsledků z knihovny, která umí vyhodnocovat toto trackování a přiřazuje objektům jednotlivé identifikátory.

Chce-li uživatel trackovat nějaký objekt ve videu, musí vykreslit polygon značící začátek trackování a polygon značící jeho konec. Identifikátory obou polygonů musí být identické. Důvodem je, že jakmile uživatel vykreslí polygon a chce začít trackování, je vyhledán druhý polygon se stejným identifikátorem. Není-li druhý polygon nalezen, vypíše se chybová hláška „Druhý polygon nenalezen“. V případě nalezení druhého polygonu začne proces výpočtu trackování. Nejdříve je vypočtena vzdálenost mezi těmito polygony. Tím je myšlen počet všech snímků mezi polygony. Poté jsou extrahovány všechny souřadnice obou polygonů a vypočteny kroky. Jsou použity pro vypočtení souřadnic následujícího polygonu, který se nachází na následujícím snímku. Zde můžou nastat tři situace. Počet souřadnic obou polygonů je stejný, začínající polygon má více souřadnic než koncový polygon a nebo koncový polygon má více souřadnic než polygon začínající.

V případě, kdy délka souřadnic je stejná, je proces výpočtu polygonů následující: polygon na následujícím snímku je vypočten tak, že ke každé souřadnici X,Y je připočten daný vypočtený krok (pro souřadnici X a Y může být krok odlišný). Protože se kroky přičítají k souřadnicím podle toho, jak jsou uloženy v poli, tedy: X2[n] = X1[n]+stepX[n], Y2[n] = Y1[n]+stepY[n]], kde X1 je souřadnice X polygonu s indexem n, stepX je krok pro tuto danou souřadnici na indexu n a X2 je nově vypočtená souřadnice na stejném indexu v poli souřadnic nového polygonu, který se bude nacházet na následujícím snímku. Identicky to je se souřadnicí Y. Z toho vyplývá, že je důležité pořadí souřadnic, podle kterých se polygony vykreslují!

Máme například dva čtverce (se stejným identifikátorem) na odlišných snímcích videa, kde první = {x1,y1 x2,y2 x3,y3 x4,y4 x1,y1} a druhý = {x1,y1 x2,y2 x3,y3 x4,y4 x1,y1} 7.6 a chceme provést tracking. Berme v úvahu, že souřadnice prvního čtverce x1,y1 je bod A, x2,y2 je bod B, x3,y3 je bod C a x4,y4 je bod D. Aby se polygony (čtverce) na snímcích mezi začátkem a koncem vykreslovaly správně, tak souřadnice druhého čtverce musí být takové, že x1,y1 odpovídá bodu A, x2,y2 odpovídá bodu B, x3,y3 odpovídá bodu C a x4,y4 odpovídá bodu D. Není-li toto pravidlo dodrženo, nebudou mít nově vykreslené polygony tvar čtverce. V rámci dalšího zdokonalování webové aplikace by byla implementace vykreslování kružnic a obdélníku. To by alespoň částečně odstranilo tento problém.

Jedná-li se o případ, kdy se délky souřadnic polygonů liší, je nutné nejdříve kratší souřadnici (ta, která obsahuje méně dvojic X,Y) rozšířit tak, aby velikosti byly stejné. Poté je výpočet následujících polygonů stejný jako v případě, kdy délky souřadnic jsou již od začátku totožné.

Po výpočtu všech polygonů je ještě provedena kontrola, zda na snímcích existuje již nějaká anotace (polygon) nebo ne. Neexistuje-li, je pro daný snímek vytvořený objekt reprezentující anotaci snímku. Obsahuje číslo snímku, index snímku a pole s polygony (v

[4] https://www.npmjs.com/package/file-saver

tomto případě pouze s jedním), viz 7.6. V opačném případě je vypočtený polygon uložen na konec pole obsahující všechny polygony pro daný snímek.

Konec trackování obsahuje uložení anotací do souboru. Tato funkcionalita je již podrobněji popsána v 7.6. Jedná se pouze o poslání JSON objektu na backend webové aplikace, kde je poté uložen do souboru {název}.json nebo {název}_yolo.json podle toho, zda byla využita knihovna YOLO. Pokud je trackování přes mnoho snímků, může proces ukládání výsledků trvat delší dobu.

Pokud by nastala situace, kdy by se z nějakého důvodu neuložila celá anotace (nebyla by přes všechny snímky), stačilo by udělat znovu tutéž anotaci na konečný snímek. Algoritmus by sám našel konec předešlého trackování a pokračoval odtud. Tímto způsobem je i implementováno trackování objektů, které nemají lineární trajektorii. Stačí pouze jejich trajektorii rozdělit právě na lineární úseky a postupně je trackovat jen za využití polygonů stejných identifikátorů. Algoritmus najde konec jedné části a napojí trackování na ní.

Kapitola 8

Testování a vyhodnocení

V této kapitole jsou popsány výsledky testování. Nejdříve bylo prováděno funkční testování, protože se jedná o webovou aplikaci, tak bylo . To vedlo k odstranění co největšího počtu chyb. Jakmile byla aplikace stabilní, bylo nutné provést i uživatelskou přívětivost aplikace. K tomu jsem připravil standardní dotazník. Všichni uživatelé dostali možnost o samotě testovat webový anotační nástroj. Před začátkem ještě obdrželi dotazník s otázkami ohledně funkcionality webové aplikace. Po uplynutí doby, kterou uznal každý uživatel za dostačující, odpověděli na všechny otázky. Ty bylo poté vyhodnoceny a byl sepsán závěr vyhodnocující funkcionalitu celé webové aplikace. Kvůli aktuální situaci v České republice nebyl počet testerů vysoký.

8.1 Otázky a jejich popis

1. **Jak dobře aplikace funguje?** Tato otázka zjišťuje, jaký pocit mají uživatelé z kvality celé webové aplikace. Je zde na výběr ze čtyř možných odpovědí. Viz graf 8.2.

   (a) **velmi dobře** Aplikace funguje bez chyb. Je na takové úrovni, že by se ihned dala používat jako produkční aplikace.

   (b) **aplikace funguje s menšími vadami** Veškerá funkcionalita aplikace funguje. Pouze občas se objeví menší vada. Ale přesto aplikace slouží správně, k čemu je určena.

   (c) **aplikace obsahuje vady, ale dá se používat** Aplikace obsahuje více vad, ale přesto ji lze používat. Bylo by potřeba nedostatky opravit.

   (d) **aplikace obsahuje mnoho chyb** Aplikace obsahuje velmi mnoho chyb. Je jich tolik, že se aplikace nedá vůbec používat.

2. **Je aplikace intuitivní (user friendly)?** Otázka zjišťující intuitivnost aplikace. Tedy jestli uživatel sám dokáže webovou aplikaci používat nebo potřebuje poradit, jak aplikaci používat. Viz graf 8.2.

   (a) **velmi intuitivní** Uživatel aplikaci ovládá bez jakékoliv pomoci či nápovědy.

   (b) **občas jsem nevěděl, co dělat** Uživatel dokáže aplikaci ovládat. Občas ale váhá, jak aplikaci používat, a potřebuje poradit.

   (c) **musel jsem dlouho hledat, co dělat** Uživatel se musel více ptát, jak aplikaci používat, než aby mohl samostatně pracovat.

(d) **aplikace není vůbec intuitivní** Bez dokumentace nelze webovou aplikaci vůbec používat. Uživateli se vůbec nelíbí vzhled aplikace. Je možné, že by ji kvůli tomu vůbec nepoužíval.

3. **Je konfigurace aplikace dostatečná?** Tato otázka zjišťuje, zda konfigurace aplikace obsahuje dostatečný výběr funkcionalit nebo zda by bylo dobré nějakou přidat. Viz graf 8.2.

   (a) **ano, není co dodat** Uživatel nemá potřebu mít širší konfiguraci. Ta, která existuje, je naprosto dostačující.

   (b) **ano, ale nějaké věci by se dali přidat** Konfigurace, která je implementovaná, je dostačující, ale uživatel by měl rád ještě další možnosti konfigurace.

   (c) **není, ačkoliv nějaké užitečné věci tam jsou** Uživatel považuje existující konfiguraci za nedostatečnou. Najde se tu pouze pár možností, které uživateli připadají užitečné.

   (d) **není vůbec** Uživatel neshledává na konfiguraci webové aplikace nic užitečného. Celou by ji předělal nebo odstranil.

4. **Jak dobře funguje anotování bez použití YOLO?** Tato otázka se snaží zjistit, jak funguje anotování (videa i obrázku) bez využití knihovny YOLO – tedy pouze samotné uživatelské anotování. Viz graf 8.2.

   (a) **velmi dobře** Anotování funguje velmi dobře. Uživatel nemá co vytknout.

   (b) **anotování funguje s menšími vadami** Anotování funguje dobře. Pouze občas se objeví menší vada. Přesto však anotování slouží správně, k čemu má.

   (c) **anotování obsahuje vady, ale dá se používat** Anotování obsahuje více vad, ale přesto ho lze používat. Bylo by potřeba nedostatky opravit.

   (d) **anotování se nedá vůbec používat** Anotování obsahuje velmi mnoho chyb. Je jich tolik, že se anotování nedá vůbec používat.

5. **Jak dobře funguje anotování s použitím YOLO?** Další otázka se snaží zjistit, jak funguje anotování (videa i obrázku) s využitím knihovny YOLO – tedy uživatelské anotování a zároveň detekování objektů pomocí YOLO. Viz graf 8.2.

   (a) **velmi dobře** Anotování funguje velmi dobře. Uživatel nemá co vytknout.

   (b) **anotování funguje s menšími vadami** Anotování funguje dobře. Pouze občas se objeví menší vada. Ale přesto anotování slouží správně, k čemu má.

   (c) **anotování obsahuje vady, ale dá se používat** Anotování obsahuje více vad, ale přesto ho lze používat. Bylo by potřeba nedostatky opravit.

   (d) **anotování se nedá vůbec používat** Anotování obsahuje velmi mnoho chyb. Je jich tolik, že se anotování nedá vůbec používat.

6. **Obsahuje aplikace bugy?** Předposlední otázka zjišťuje množství chyb vyskytujících se v aplikaci. Je velmi úzce spojena s předchozími otázkami. Viz graf 8.2.

   (a) **žádné** Aplikace neobsahuje žádné chyby. Toto je velmi vzácný případ, protože i mnoho produkčních webových aplikací obsahuje chyby, a proto jsou pro ně poté dělány „hotfixy“.

(b) **občas se nějaký bug objeví** Může se stát, že se občas objeví nějaká chyba, ale pořád je jich málo.

(c) **obsahuje bugy, ale přesto se dá v pořádku používat** Častější výskyt chyb. Pořád jich ale není příliš a webová aplikace se dá stále používat.

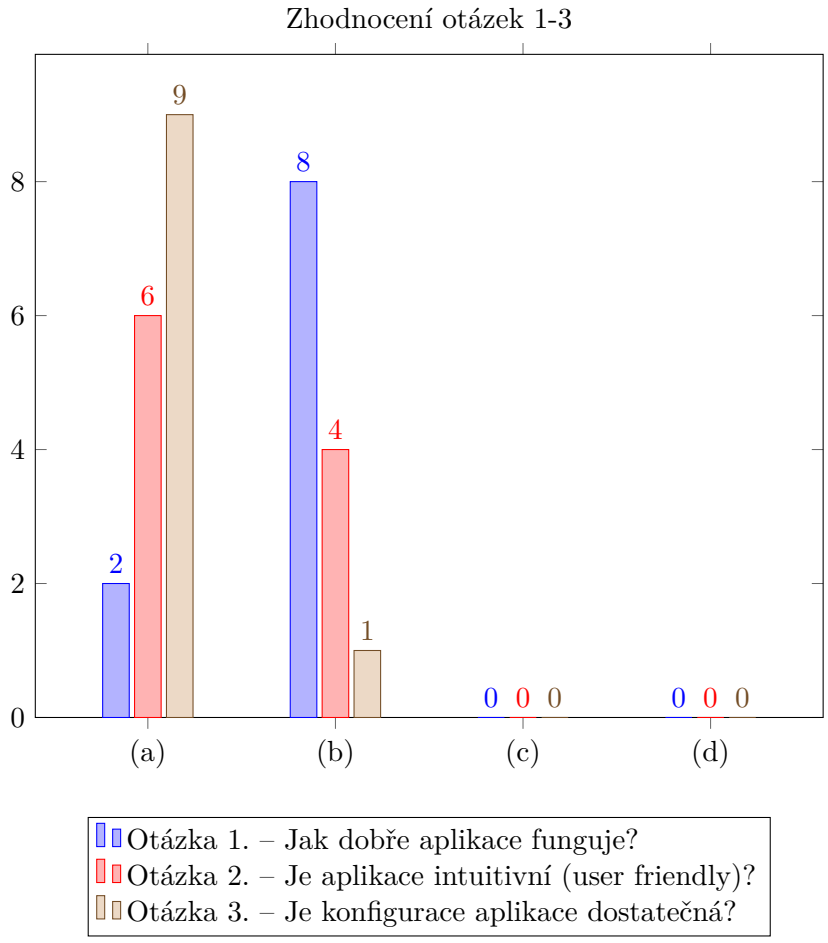
(d) **obsahuje mnoho chyb** Webová aplikace obsahuje mnoho chyb, téměř ji nelze používat. Ale přesto se najdou funkcionality, ke kterým ji lze použít.

(e) **obsahuje tolik bugů, že se nedá vůbec používat** Webová aplikace obsahuje tolik chyb, že se nedá používat.

7. **Slovy popište své shrnutí ohledně aplikace.** Slovní ohodnocení celé webové aplikace. Zde může uživatel uvést své poznatky k aplikaci a rozvést odpovědi k předešlým otázkám. Viz 8.2.

8.2 Vyhodnocení otázek

Zde budou graficky i slovně popsány výsledky z testování od uživatelů. Jedná se především o zjištění kvality a možnosti uplatnění webového anotačního nástroje.



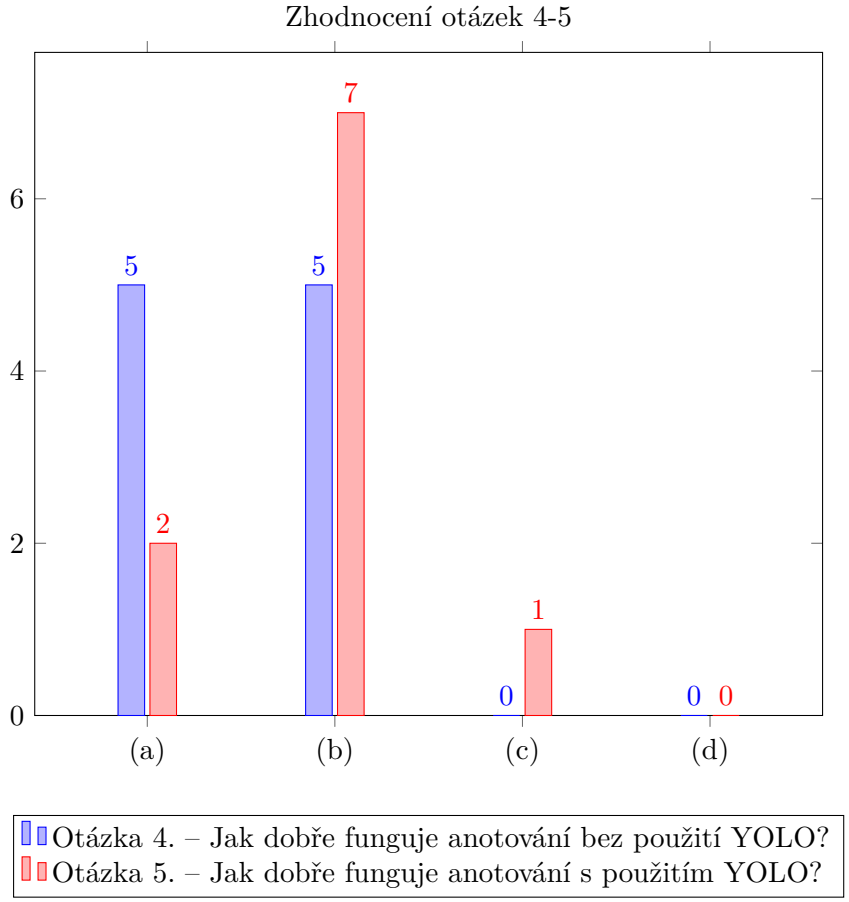
Graf ukazující výsledky odpovědí na otázky jedna až tři. Tyto tři otázky spolu souvisí tím, že podle nich se dá zjistit, zda je aplikace užitečná a dá se používat nebo zda je potřeba ji celou přepracovat. Podle výsledků z testování lze říci, že anotační nástroj je podle testerů

na takové úrovni, že ho lze bez problému používat. V aplikace je pouze několik drobností, které by bylo dobré opravit či přidat.

Podle odpovědí na první otázku můžeme říci, že aplikace funguje tak, jak má, lze podle ní anotovat videa i obrázky. Ale i přesto se najdou maličkosti, které by bylo dobré opravit. Ale zajisté jich není tolik a nejsou tak zásadní, aby nebylo možné aplikaci používat.

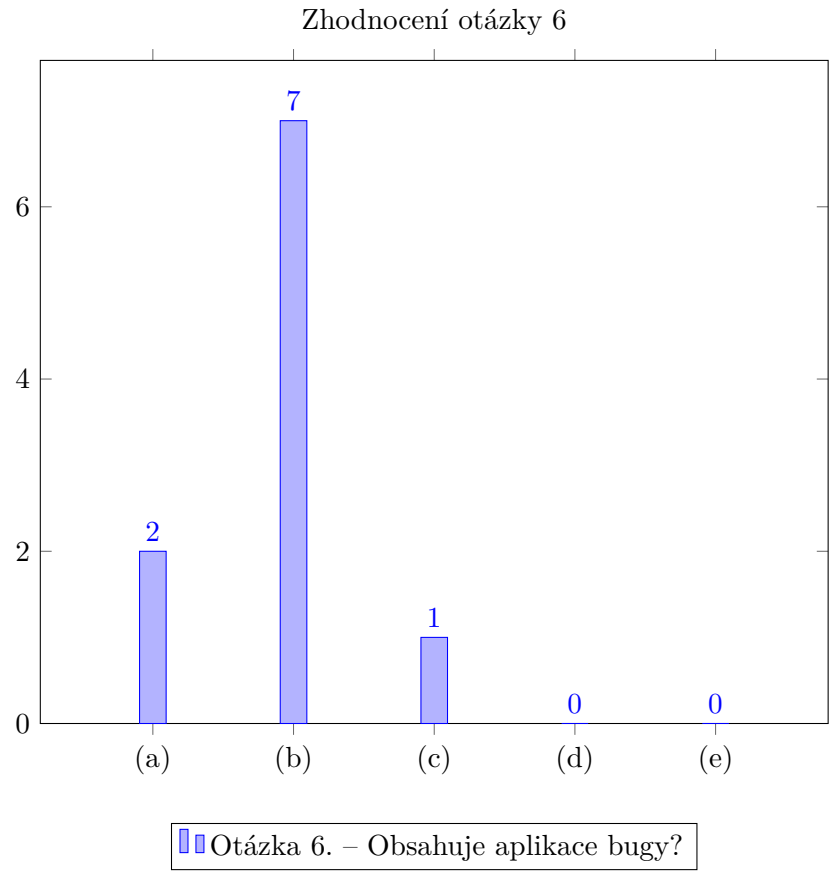
Druhá otázka je zaměřená spíše na uživatelské rozhraní než na funkcionalitu aplikace. Podle ní lze říci, že aplikace je navržená dobře a testeři ji mohli používat bez větší pomoci.

Třetí otázka je zaměřená na konfiguraci webové aplikace. Téměř všichni testeři byli spokojeni s dosavadní konfigurací a neměli potřebu přidávat k ní další možnosti.



Graf ukazující výsledek odpovědí na otázky čtyři a pět. Jedná se o dva velmi podobné případy – liší se pouze v tom, zda je použita knihovna YOLO, nebo nikoliv. Na tomto grafu lze tedy vidět rozdíl mezi tím, jak dobře webová aplikace funguje s knihovnou YOLO a bez ní. Na první pohled je vidět, že aplikace lépe funguje bez knihovny YOLO, ale rozdíl není značný.

V případě, kdy nebyla použita knihovna YOLO, uvedla polovina testerů, že aplikace funguje výborně, a druhá polovina uvedla, že aplikace funguje velmi dobře, ale občas se najde nějaká vada. V případě, kdy je použita knihovna YOLO, si většina testerů myslí, že aplikace funguje velmi dobře, ale občas se najde nějaká vada.



Graf ukazující výsledek odpovědí na otázku, zda aplikace obsahuje „bugy“. Je velmi úzce spjatá s předchozími otázkami. Z grafu je zřejmé, že anotační nástroj neobsahuje příliš mnoho chyb. Podle většiny testerů aplikace obsahuje občas nějakou chybu, ale rozhodně je to v přijatelné míře a chyby neovlivňují funkcionalitu aplikace. Ta se dá používat právě k těm účelům, ke kterým byla navržena a implementována. Našlo se i několik uživatelů, kteří si myslí, že aplikace funguje bez chyb, ale i jeden, který si myslí, že „bugů“ je více, ale pořád v míře, kdy lze aplikaci v pořádku používat.

Vyhodnocení poslední (sedmé) otázky

Aplikace se uživatelům líbila a považují ji za užitečnou, ale i přesto se najde několik faktů, které by aplikaci vytkli. Jak je vidět na předešlých grafech, obsahuje aplikace menší množství aspektů, které uživatelé posoudili jako „bugy“ ve webové aplikaci. Proto by ve fázi dalšího zdokonalování byly postupně všechny chyby opraveny. Ale jak bylo již zmíněno, mnoho i produkčních webových aplikací obsahuje „bugy“, které jsou opravovány v *hotfixech* (s dalším vývojem se objevují nové a nové chyby).

Jinak byli uživatelé s webovým anotačním nástrojem spokojeni. Co se týče konfigurace anotačního nástroje, uvedli někteří testeři, že tato funkcionalita je spíše „nice to have“ (hezké ji mít), ale není nutná. Kdyby zde vůbec nebyla, tak by to na celkové hodnocení aplikace nemělo vliv.

Kapitola 9

Závěr

Výstupem této bakalářské práce je nástroj pro získávání cenných dat pro anotování. Tato data jsou pro další snadnou a rychlou manipulaci s nimi ve formátu JSON. Práce taktéž zahrnuje studium použitých technologií.Je zde popsán i vývoj webových aplikací, aby čtenář lépe porozuměl výhodám dnešních technologií a single page aplikacím. Tudíž cílem není pouze samotná implementace webové aplikace, ale taktéž studium důležitých technologií a nalézání vhodných formátů odpovědí od uživatele.

Webová aplikace poskytuje uživateli možnost určování, zda se jedná o deepfake 4.3 a nebo o získávání informací z videí a obrázků. Výhodou celé aplikace je široká konfigurovatelnost. Pomocí té si sám uživatel zvolí na míru celou aplikaci. Pokud uživatel chce pro detekování objektů využívat knihovny YOLO 4.1, musí mít na paměti, že jsou využívány již předem natrénované neuronové sítě. Ty jsou schopny detekovat například osoby, osobní i nákladní automobily a další. V rámci dalšího zlepšování webové aplikace by bylo dobré přidat trénovací funkcionalitu. Pomocí ní by mohl uživatel nahrát na YOLO server 7.3 svá testovací data a poté by byla neuronová síť na těchto datech natrénovaná. To by vedlo ke zkvalitnění výsledků samotného anotování. Také by bylo vhodné přidat další možnosti vykreslování polygonů, jako je například vykreslování obdélníků, kružnic a dalších tvarů. To by vedlo i ke zkvalitnění trackování objektů. Nyní se používají pouze křivky.

K dalšímu zlepšení webové aplikace by jistě přispělo napojení databáze pro ukládání. V tomto případě je vše ukládáno na server. Pokud by tuto webovou aplikaci využívalo mnoho uživatelů, bylo by napojení databáze nezbytné. Pro tento účel by asi nejlépe posloužila NoSQL (Not only SQL) databáze. Výhodou je, že tato databáze nemusí obsahovat pouze relační modely. Je možné zde ukládat například data, jako JSON nebo XML. Po připojení databáze by následovalo vytvoření přihlašovaní, aby uživatelé viděli pouze svoje anotace.

Zájemcům o vývoj webových aplikací bych doporučil přečíst si mou bakalářskou práci. Může být důležitá pro ty, kteří chtějí získat teoretické znalosti v oblasti webových technologií, nebo pro ty, kdo chtějí vyvíjet vlastní podobný systém. Může se jednat například i o webovou hru.

Na závěr bych rád dodal, že si myslím, že vývoj webových aplikací má velký potenciál a pokračování na této bakalářské práci má slibnou budoucnost.

Literatura

[1] *Yolo-v4 and Yolo-v3/v2 for Windows and Linux*. 2020. Dostupné z: https://github.com/AlexeyAB/darknet/tree/master/.

[2] *Alturos.ImageAnnotation*. 2020. Dostupné z: https://github.com/AlturosDestinations/Alturos.ImageAnnotation.

[3] *Anno-Mage: A Semi Automatic Image Annotation Tool*. 2020. Dostupné z: https://github.com/virajmavani/semi-auto-image-annotation-tool.

[4] Bowler, J. et al. *Scalable Vector Graphics (SVG)1.0 Specification*. December 1999. Dostupné z: https://www.w3.org/TR/1999/WD-SVG-19991203.pdf.

[5] Bréhéret, A. *Pixel Annotation Tool*. 2017. Dostupné z: https://github.com/abreheret/PixelAnnotationTool.

[6] Carlier, A. *UltimateLabeling*. 2019. Dostupné z: https://github.com/alexandre01/UltimateLabeling.

[7] Cartucho, J., Ventura, R. a Veloso, M. Robust Object Recognition Through Symbiotic Deep Learning In Mobile Robots. In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2018, s. 2336–2341. ISBN 9781538680940.

[8] Company, A. *Amazon Mechanical Turk*. 2018. [Online; navštíveno 2. 4. 2020]. Dostupné z: https://www.mturk.com/.

[9] *Computer Vision Annotation Tool (CVAT)*. 2020. Dostupné z: https://github.com/opencv/cvat.

[10] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J. a Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*. červen 2010, sv. 88, č. 2, s. 303–338.

[11] Fiedler, N., Bestmann, M. a Hendrich, N. ImageTagger: An Open Source Online Platform for Collaborative Image Labeling. In: Springer. *RoboCup 2018: Robot World Cup XXII*. 2018. ISBN 9783030275440.

[12] Fink, G. a Flatow, I. Introducing Single Page Applications. In: *Pro Single Page Application Development*. Apress, 2014, s. 3–13. DOI: 10.1007/978-1-4302-6674-7_1. ISBN 9781430266730. Dostupné z: https://doi.org/10.1007/978-1-4302-6674-7_1.

[13] Garrett, J. J. *Ajax: A New Approach to Web Applications*. 2005. [Online; navštíveno 8. 1. 2020]. Dostupné z: https://pdfs.semanticscholar.org/c440/ae765ff19ddd3deda24a92ac39cef9570f1e.pdf?_ga=2.94336985.559245674.1578501504-238268044.1578501504.

[14] Jäger, J., Reus, G., Denzler, J., Wolff, V. a Fricke Neuderth, K. LOST: A flexible framework for semi-automatic image annotation. *ArXiv preprint arXiv:1910.07486*. 2019.

[15] Joshi, K. A. a Thakore, D. G. A Survey on Moving Object Detection and Tracking in Video Surveillance System. *International Journal of Soft Computing and Engineering*. 2. vyd. Blue Eyes Intelligence Engineering & Sciences Publ. 2012, č. 3, s. 44–47. ISSN 2231-2307.

[16] *Imglab*. 2019. Dostupné z: https://github.com/NaturalIntelligence/imglab.

[17] *MedTagger*. 2019. Dostupné z: https://github.com/medtagger/MedTagger.

[18] MIT. *Materia-UI*. 2020. [Online; navštíveno 13. 1. 2020]. Dostupné z: https://material-ui.com/company/about/.

[19] Moravec, I. M. *Prohlížečová hra s umělou inteligencí*. 2019. Diploma thesis. Vysoké učení technické v Brně.

[20] Node.js. *Introduction to Node.js*. 2020. [Online; navštíveno 9. 1. 2020]. Dostupné z: https://nodejs.dev/.

[21] Ojamaa, A. a Düüna, K. Assessing the security of Node.js platform. In: IEEE. *2012 International Conference for Internet Technology and Secured Transactions*. 2012, s. 348–355. ISBN 9781908320087.

[22] *OpenLabeler*. 2020. Dostupné z: https://github.com/kinhong/OpenLabeler.

[23] *Pixie*. 2019. Dostupné z: https://github.com/buni-rock/Pixie.

[24] Redmon, J. *Darknet: Open Source Neural Networks in C*. 2013–2016. Dostupné z: http://pjreddie.com/darknet/.

[25] Redmon, J. a Farhadi, A. *YOLOv3: An Incremental Improvement*. 2012.

[26] Redmon, J. a Farhadi, A. *YOLOv3: An Incremental Improvement*. 2018.

[27] Ren, S., He, K., Girshick, R. a Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2015.

[28] Rieger, B. *Labelbox*. 2019. Dostupné z: https://github.com/Labelbox/Labelbox.

[29] Skalski, P. *Make Sense*. 2019. Dostupné z: https://github.com/SkalskiP/make-sense/.

[30] *Turktool*. 2018. Dostupné z: https://github.com/jaxony/turktool.

[31] Tzutalin. *LabelImg*. 2015. Dostupné z: https://github.com/tzutalin/labelImg.

[32] Ullman, C. a Dykes, L. *Beginning Ajax*. 1. vyd. Wiley Publishing, 2007. ISBN 9780470106754.

[33] Veitch Michaelis, J. *DeepLabel*. 2017. Dostupné z: https://github.com/jveitchmichaelis/deeplabel.

[34] Vondrick, C., Patterson, D. a Ramanan, D. *VATIC - Video Annotation Tool from Irvine, California*. 2012. Dostupné z: https://github.com/cvondrick/vatic.

[35] Vondrick, C., Ramanan, D. a Patterson, D. Efficiently Scaling Up Video Annotation with Crowdsourced Marketplaces. In: Daniilidis, K., Maragos, P. a Paragios, N., ed. *Computer Vision – ECCV 2010*. Springer Berlin Heidelberg, 2010, s. 610–623. DOI: 10.1007/978-3-642-15561-1_44. ISBN 9783642155611. Dostupné z: https://doi.org/10.1007/978-3-642-15561-1_44.

[36] *VoTT (Visual Object Tagging Tool)*. 2020. Dostupné z: https://github.com/microsoft/VoTT.

[37] W3C. *HTML 4 Errata*. 1999–2005. [Online; navštíveno 9. 1. 2020]. Dostupné z: https://www.w3.org/MarkUp/html4-updates/errata.

[38] W3C. *A vocabulary and associated APIs for HTML and XHTML*. 2008. [Online; navštíveno 9. 1. 2020]. Dostupné z: https://www.w3.org/TR/2008/WD-html5-20080122/.

[39] W3C. *HTML5 Differences from HTML4*. 2014. [Online; navštíveno 9. 1. 2020]. Dostupné z: https://www.w3.org/TR/2014/NOTE-html5-diff-20141209/.

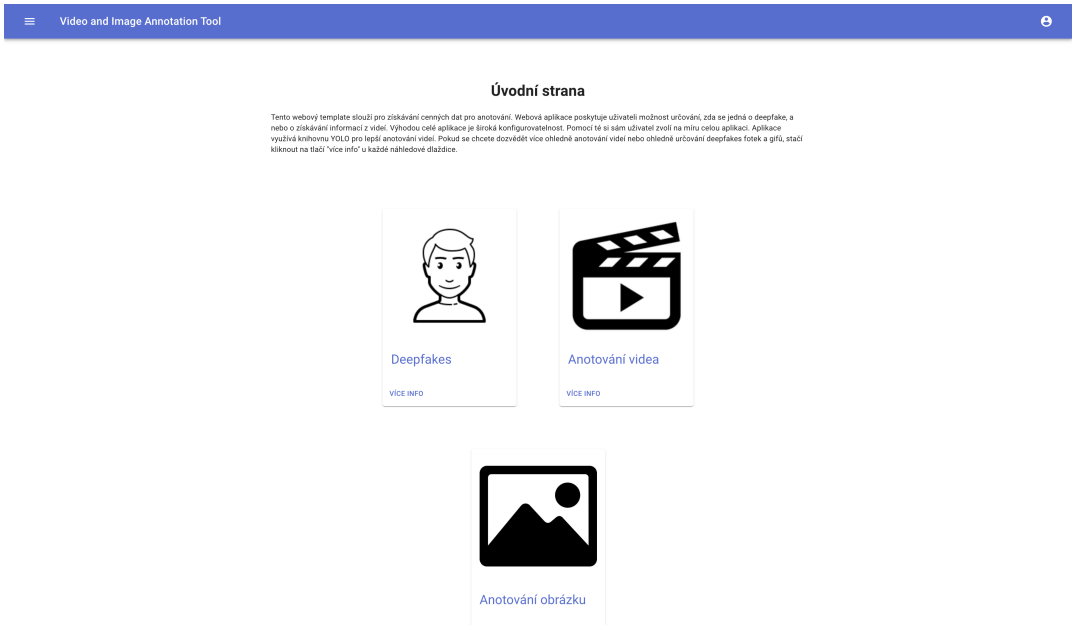
[40] Wada, K. *Labelme: Image Polygonal Annotation with Python*. 2016. Dostupné z: https://github.com/wkentaro/labelme.

[41] *Yolo_mark*. 2019. Dostupné z: https://github.com/AlexeyAB/Yolo_mark.

Příloha A

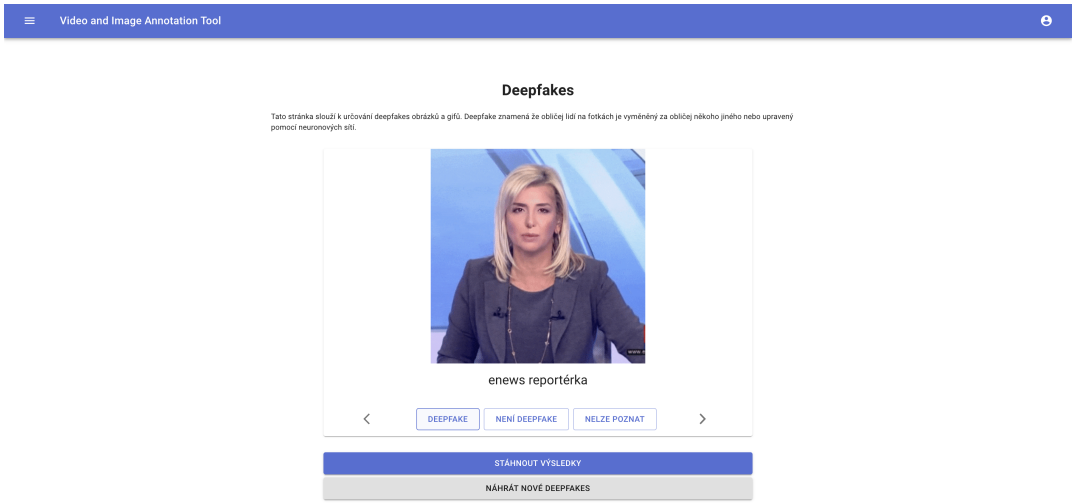
Ukázky stránek

Úvodní stránka



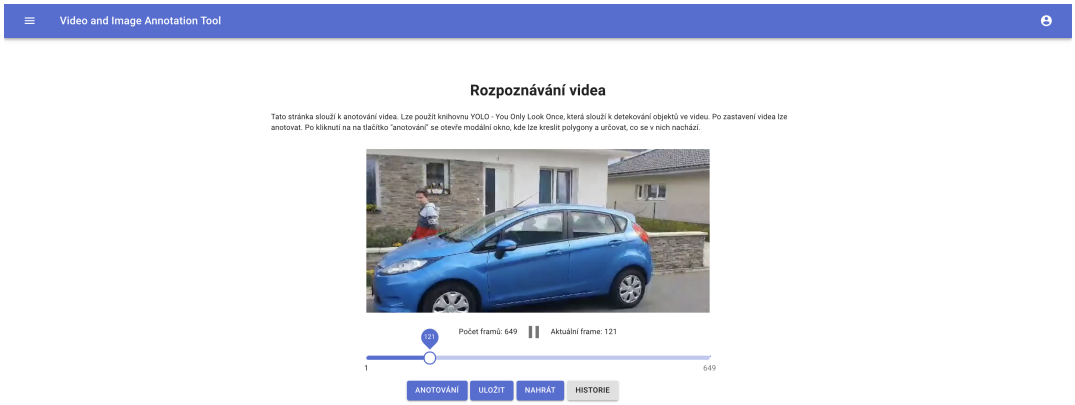
Obrázek A.1: Ukázka úvodní stránky.

Stránka s deepfakes



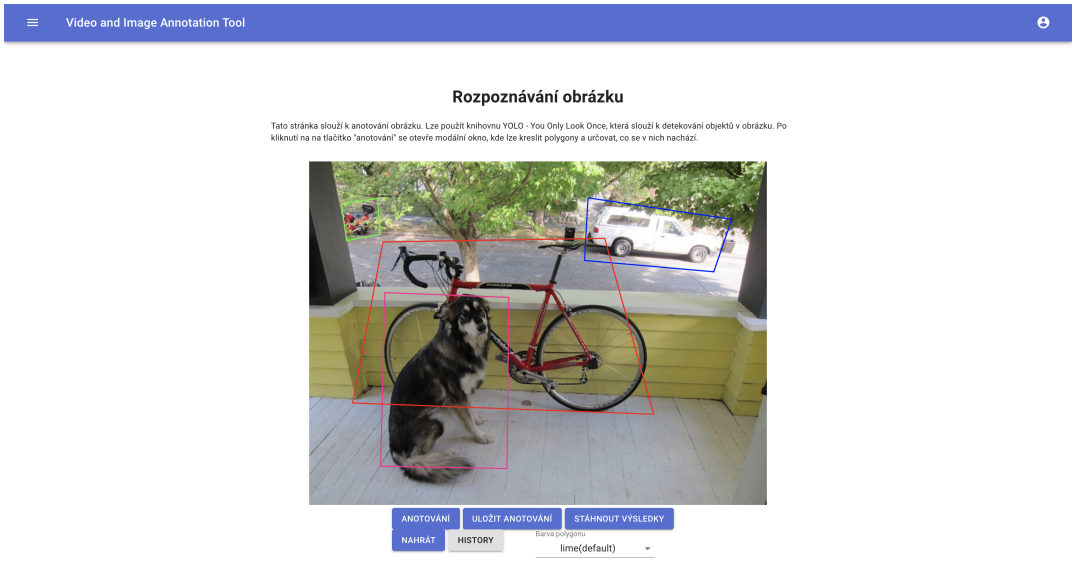
Obrázek A.2: Ukázka deepfakes stránky.

Stránka s videi



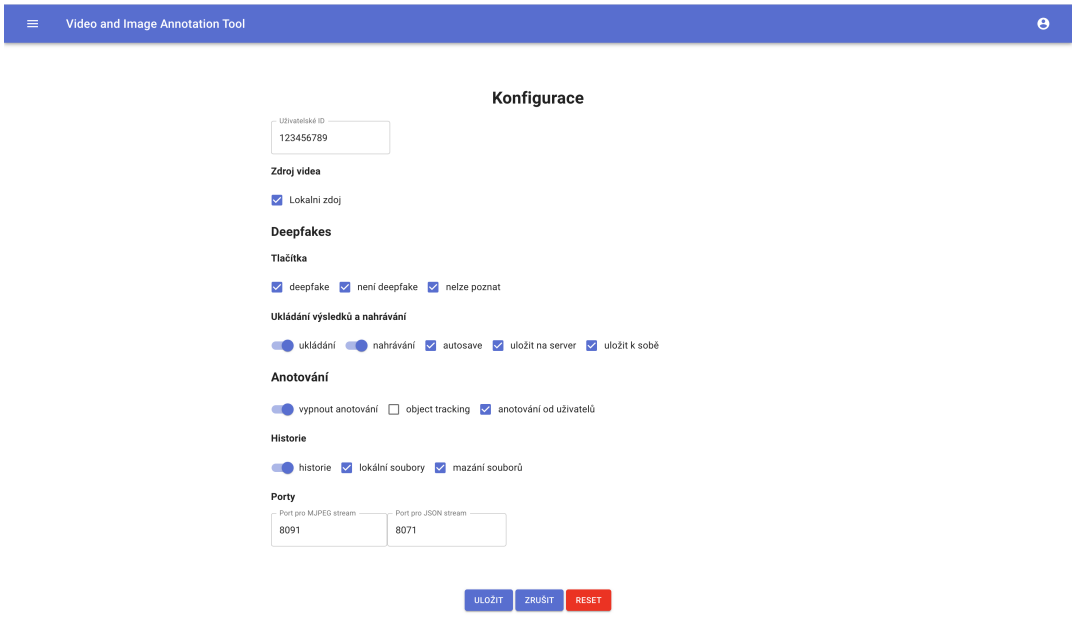
Obrázek A.3: Ukázka stránky pro anotování videí.

Stránka s obrázky



Obrázek A.4: Ukázka stránky pro anotování obrazků. Fotka byla převzata z githubu knihovny YOLO – https://github.com/AlexeyAB/darknet/tree/master/data.

Konfigurační stránka



Obrázek A.5: Ukázka konfigurační stránky.

Příloha B

Obsah přiloženého paměťového média

Paměťové médium obsahuje tyto soubory a adresáře:

- Adresář *source* obsahuje tři podadresáře, které obsahují zdrojové soubory:
  - adresář *annotation_tool* obsahuje zdrojové soubory pro webovou aplikaci
  - adresář *yolo_server* obsahuje zdrojové soubory pro YOLO server
  - adresář *test* obsahuje testovací video a obrázek
- Adresář *Latex* obsahuje kódy k vytvoření technické zprávy
- Adresář *text* obsahuje textové soubory bakalářské práce – dokumentace, technická zpráva
- Soubor *plakat.pdf* obsahuje vytvořený plakát

Příloha C

Manuál

Webová aplikace je napsaná v Node.js (backend) a React (frontend). Dále využívá knihovnu YOLO (You Only Look Once), pro kterou je nutné doinstalovat některé závislosti:

- Windows – YOLO knihovna potřebuje operační systém Windows
- CMake >= 3.8
- CUDA 10.0
- OpenCV >= 2.4
- cuDNN >= 7.0
- GPU s CC >= 3.0
- Node.js >= 12

Podrobnější návod k instalaci lze najít na githubu[1]. Po nainstalování potřebných závislostí lze aplikaci spustit následovně.

Instalace npm balíčků

Instalaci lze provést dvěma způsoby:

- npm install
- npm i

Spuštění po nainstalování balíčků

- npm run dev

Oba tyto kroky je potřeba udělat v adresáři s webovou aplikací i v adresáři s YOLO serverem.

[1] https://github.com/AlexeyAB/darknet

Příloha D

Prohlášení

K bakalářské práci byly přiloženy dva testovací soubory. Natočené domácí video a obrázek. Na videu se nachází osobní automobil, který byl pro tyto účely zapůjčený. Majitelka vozu mi dala souhlas se zveřejněním videa pro účely této bakalářské práce. Obrázek byl převzat z internetu[1] [1]. Jedná se o volně dostupnou stránku. Na obrázku se nacházejí čtyři různé objekty – pes, kolo, nákladní automobil a skútr.

[1] https://github.com/AlexeyAB/darknet/tree/master/data

Příloha E

Plakát

Webová aplikace pro kontrolu výsledků automatického zpracování videa a jeho ručního anotování

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Autor: Petr Červíček
Vedoucí: doc. RNDr. Pavel Smrž, Ph.D.
Bakalářská práce, 2020

Jedná-li se o anotování obrázků, je načten obrázek z lokálního zdroje nebo z historie spolu s jeho uloženou anotací. Poté je umožněno uživateli kreslit do obrázku polygony a určovat, co se na nich nachází.

První částí této bakalářské práce je anotování videí a obrázků bez využití knihovny pro detekování objektů. Pokud jde o případ anotování videa, je toto videa rozděleno na snímky, které mohou být dále anotovány pomocí vykreslování polygonů. Je zde také podporováno sledování objektů v obraze.

U anotování obrázků s využitím knihovny YOLO je postup následující. Obrázek je poslán na YOLO server, kde dochází k detekování objektů a v odpovědi serveru je zaslán obrázek již s detekcemi objektů. Poté může uživatel provést uživatelské anotování pomocí vykreslování polygonů.

V druhé části této práce je přidána k uživatelskému anotování knihovna, která je schopná detekovat objekty ve videu pomocí strojového učení. Jedná se o knihovnu YOLO. U videa detekuje v každém snímku objekty a výsledeky posílá v MJPEG a JSON streamu. Uživatelské anotování je stejné jako v případě bez využití knihovny pro detekování objektů.

Třetí možností je určování deepfakes. Uživateli jsou zde poskytnuty fotografie či gify a on sám poté určuje, zda se jedná o takzvaný deepfake nebo jestli osoba na fotografii je skutečná. Celá aplikace poté byla testována a předložena uživatelům, kteří poté provedli vyhodnocení. Testeři se shodli, že aplikace funguje správně a její vidí smysl v jejím používání.