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

POČÍTAČOVÉ VIDĚNÍ PRO SLEDOVÁNÍ 3D TISKU

COMPUTER VISION FOR MONITORING OF 3D PRINTING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **MIKULÁŠ HEINZ**
AUTHOR

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

BRNO 2022

Vysoké učení technické v Brně
Fakulta informačních technologií

Ústav počítačové grafiky a multimédií (UPGM) Akademický rok 2021/2022

Zadání bakalářské práce

Student: **Heinz Mikuláš**
Program: Informační technologie
Název: **Počítačové vidění pro sledování 3D tisku**
Computer Vision for Monitoring of 3D Printing
Kategorie: Umělá inteligence

Zadání:

1. Seznamte se se způsoby rozpoznávání stavu činnosti strojů na základě vizuální informace.
2. Zpracujte přehled hlavních problémů, které nastávají při tisku na běžných 3D tiskárnách.
3. Na základě získaných poznatků navrhněte a implementujte systém, který dokáže rozpoznat nejčastější případy chyb při 3D tisku.
4. Zpřístupněte vytvořený systém pomocí webového rozhraní tak, aby jej bylo možné použít např. v podmínkách naší fakulty.
5. Vytvořte stručný plakát prezentující práci, její cíle a výsledky.

Literatura:

- dle doporučení vedoucího

Pro udělení zápočtu za první semestr je požadováno:

- funkční prototyp řešení

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 2021
Datum odevzdání: 11. května 2022
Datum schválení: 1. listopadu 2021

Abstrakt

Tato práce se zabývá automatickou detekcí chyb, které nastávají v průběhu časově náročného 3D tisku. K tomu využívá počítačového vidění a umělou inteligenci. Hlavním výsledkem je systém, který pomocí Raspberry Pi a připojené kamery zaznamenává pravidelně průběh tisku a snímky zasílá na počítač uživatele k detekci. Na tomto počítači je snímek analyzován modelem konvoluční neuronové sítě a informace o nalezené chybě je zaslána uživateli pomocí SMTP protokolu. Součástí řešení je také datová sada s 385 snímky chyb při 3D tisku rozdělených podle typu.

Abstract

This thesis deals with the automatic detection of errors that can occur during time-consuming 3D printing. It uses computer vision and artificial intelligence to achieve this. The main result is a system that uses Raspberry Pi and a connected camera to periodically record the printing process and sends the images to the user's computer for detection. On this computer, the image is analysed by a convolutional neural network model and information about found error is sent to the user via a SMTP protocol. The solution also includes a dataset with 385 images of 3D printing errors sorted by type.

Klíčová slova

strojové učení, počítačové vidění, 3D tisk, zpracování obrazu, umělá inteligence, konvoluční neuronová síť, aditivní výroba

Keywords

machine learning, computer vision, 3D printing, image processing, artificial intelligence, convolution neural network, additive manufacturing

Citace

HEINZ, Mikuláš. *Počítačové vidění pro sledování 3D tisku.* Brno, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. RNDr. Pavel Smrž, Ph.D.

Počítačové vidění pro sledování 3D tisku

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana doc. RNDr. Pavla Smrže Ph.D. Uvedl jsem všechny literární prameny, publikace a další zdroje, ze kterých jsem čerpal.

.....
Mikuláš Heinz
9. května 2022

Poděkování

Chtěl bych poděkovat vedoucímu mé bakalářské práce doc. RNDr. Pavlu Smržovi za odborné vedení a cenné rady při psaní práce. Dále bych chtěl poděkovat Zdeňkovi Juříčkovi za rady a předané zkušenosti v oblasti 3D tisku.

Obsah

1 Úvod **2**

2 Pozadí tématiky 3D tisku **4**
2.1 Procesy aditivní výroby 5
2.2 Přehled chyb v průběhu tisku 8
2.3 Současná řešení detekce chyb 13

3 Využití počítačové vidění a neuronové sítě pro detekci chyb **15**
3.1 Reprezentace obrazu v počítači 15
3.2 Umělé neuronové sítě 15
3.3 Konvoluční neuronové sítě 16
3.4 Trénování CNN 17
3.5 Detekce objektů 19

4 Návrh **21**
4.1 Návrh systému 22
4.2 Detekce chyb 23
4.3 Uživatelské rozhraní 25
4.4 Datová sada 27

5 Implementace **29**
5.1 Propojení s tiskárnou a aplikací Octoprint 29
5.2 Detekce sledováním změn v obraze 31
5.3 Detekce chyby neuronovou sítí 32
5.4 Uživatelské rozhraní 33
5.5 Použité technologie 35

6 Vyhodnocení **37**
6.1 Testování uživateli 38
6.2 Možnosti rozšíření 39

7 Závěr **41**

Literatura **42**

Kapitola 1

Úvod

3D tisk vznikl v 80. letech 20. století a za tu dobu se stal značně populárním způsobem pro výrobu prototypů nebo objektů na míru. Tradičně byl užíván v celém spektru průmyslových odvětví, ale v posledních přibližně deseti letech zaznamenal 3D tisk výrazný nárůst v popularitě také mezi běžnými amatérskými uživateli. Umožňuje výrobu objektů s komplexní geometrií za výrazně nižší cenu a čas, než bylo dosud běžné nebo možné. Pořád ale jde o časově náročný proces a v mnohých případech se může jednat až o desítky hodin nebo jednotky dní soustavného tisku. Délka tisku se odvíjí především od velikosti objektu, druhu tiskárny a uživatelem nastavených parametrů, jako hustota výplně nebo výška vrstvy. Není výjimečné, že v průběhu tisku nastane chyba, jejímž následkem je krom nepoužitelného tisknutého objektu také velká časová ztráta. Pokud uživatel tiskne přes noc nebo se u tiskárny po celou dobu tisku nepohybuje, může se stát, že se vrátí ke zcela nepoužitelnému výtisku. I pokud si uživatel chyby brzy všimne, často je tisknutý objekt nenávratně zničen a uživatel tak musí započít, třeba několikadenní tisk, od začátku. Většina chyb, které mohou nastat při 3D tisku, je detekovatelná lidským okem. Protože je nepraktické ale kontrolovat tisk po celou jeho dlouhou dobu člověkem, je na místě použít počítačové vidění a chyby detekovat automaticky. Uživatel tak může být upozorněn na přítomnost chyby, tisk včas přerušit a ušetřit tak čas i materiál. U určitých chyb je také díky včasnému odhalení a zásahu uživatele možné v tisku pokračovat a mít ztráty takřka nulové. V nejhorších případech může dojít k poškození samotné tiskárny.

Cílem práce je funkční aplikace, která bude tisk snímat přistavenou kamerou a na pořízených snímcích hledat nechtěné změny, chyby a příznaky těchto chyb. V případě výskytu chyby na ni upozorní uživatele. K tomu využije rozhraní Octoprint[1] a SMTP[2] email. K detekci chyb využije algoritmy pro zpracování obrazu a strojového učení. Aplikace má pomoci primárně amatérským uživatelům při tisku na FFF[3] tiskárnách. Tento typ tiskárny je mezi neprofesionály s velkým náskokem nejrozšířenější, ať už kvůli nižší ceně tiskáren nebo používaného materiálu. Je běžné se setkat s takzvanými farmami 3D tiskáren, kdy uživatel provozuje více tiskáren, na kterých tiskne často v ten samý moment zároveň. U farem je naražení na chybu pravděpodobnější, protože majitelé těchto farem často tisknou pro třetí osoby na objednávku. Ty ale nemají s tiskem takové zkušenosti, a tak mohou zaslat model s chybou či v chybné podobě (např. ve formě vygenerovaných instrukcí pro tiskárnu, viz 2.1). Tiskař často takovou chybu nemá jak předem odhalit. Přehlédnutá chyba může znamenat

[1] https://octoprint.org
[2] SMTP – Simple Mail Transfer Protocol
[3] Fused Filament Fabrication – Výroba z tavených vláken

desítky ztracených hodin a vyplýtvání velkého množství materiálu. Chyba je navíc často násobná právě u farem tiskáren.

Práce se také obecně zabývá technikami 3D tisku, jejich rozdíly a nejčastějšími chybami, které mohou v průběhu 3D tisku nastat. Taktéž popisuje jejich příčiny a projevy. Protože jsou chyby nejčastěji pozorovatelné lidským okem, bude k jejich detekci využito počítačové vidění. Práce prozkoumává vhodné algoritmy počítačového vidění a konvolučních neuronových sítí. Byl vytvořen model využívající konvoluční neuronovou síť, a také byla vytvořena datová sada s nejčastějšími chybami, využita k natrénování modelu.

Využití aplikace Octoprint umožňuje mimo jiné i vzdálené ovládání tiskárny a při chybě zastavit tisk. Aplikace je navíc mezi tiskařskou komunitou již značně populární a pro uživatele, kteří Octoprint už využívají, je instalace výstupu této práce velmi snadná. Podobné aplikace, automaticky detekující chyby tisku, sice existují, ale většinou vyžadují připojení na externí server a pravidelný poplatek za využívání dané služby. Mnou vytvořená aplikace běží na počítači uživatele, čímž odpadá problém posílání snímků na cizí server. Využívání uživatelova počítače také značně snižuje nárok na mikropočítač, na kterém aplikace běží, jehož výpočetní kapacita může být z velké části spotřebována ovládáním tiskárny.

V kapitole 2 je představena tématika 3D tisku a chyb, které při něm mohou nastat. Jsou v ní popsány procesy a technologie používané ve 3D tiskárnách. Je zde také představeno několik řešení automatické detekce chyb. V kapitole 3 bude stručně popsáno počítačové vidění a princip konvolučních neuronových sítí, které slouží k rozpoznání obrazu. Kapitola 4 nastíní důvod, proč se práce problému věnuje a popíše návrh implementace řešení. V kapitole 5 je věnován prostor popisu samotné implementace řešení. V předposlední kapitole 6 je pak popsáno vyhodnocení a výsledky práce.

Kapitola 2

Pozadí tématiky 3D tisku

3D tisk nebo také aditivní výroba je automatizovaný proces tvorby trojrozměrného fyzického objektu z digitální předlohy. Je používaná jak v průmyslu, tak laickými uživateli. V průmyslových odvětvích je pro 3D tisk využíván název rapidní prototypování, kdy je využito tohoto postupu hlavně pro rychlé a výrazně levnější vytvoření prototypu nějakého systému nebo jeho části. Rychlé vytvoření prototypu usnadňuje testování a urychlení vývoje nové verze před zahájením sériové výroby. Pojmem 3D tisk se souhrnně označuje více druhů procesů a technologií, které fungují na principu vrstvení nebo spojování materiálu za pomocí číslicově řízeného stroje. Liší se také používaným materiálem, který může být nejčastěji na bázi plastu, foto-polymeru nebo kovových slitin [8].

3D tisk vznikl v 80. letech 20. století, kdy byl poprvé automatizovaně vytvořen fyzický objekt z digitálního modelu. Za tímto tiskem stál Charles W. Hull, který vynalezl a nechal si patentovat tiskárnu, která tisk prováděla technikou stereo-litografie. V začátcích byla technologie používána pouze firmami, kvůli tehdy vysoké ceně tiskáren, která se pohybovala kolem 300 tisíců dolarů[1]. V posledních přibližně deseti letech zaznamenal 3D tisk výrazný nárůst v popularitě také mezi běžnými amatérskými uživateli. Důvodem toho byl příchod výrazně levnějších 3D tiskáren, než bylo doposud běžné. Pád cen byl způsoben hlavně vypršením platnosti klíčových patentů v oblasti FDM (Fused Deposition Modeling, viz v podkapitole 2.1) a cena tiskáren mohla klesnout až na desetinu původních cen [9]. Vytvořila se tak značná komunita amatérských tiskařů, kteří mezi sebou sdílí poznatky a modely pro tisk. V domácích podmínkách je tisk využitelný pro tvorbu celé řady menších drobností pro domácnost jako držák pro telefon nebo na klíče. Dá se také využít pro nahrazení rozbitých plastových součástí v domácích spotřebičích. Na internetu můžeme najít mnoho repositářů s různými modely pro domácí tisk, přičemž některé z repositářů obsahují až miliony výtvorů. Kromě firem a domácích tiskařů se najde pro 3D tisk využití i například ve školství a výuce [7].

Výhodou 3D tisku je možnost vytvořit komplexní objekty se složitou geometrií, které by jiným způsobem nebylo možné vytvořit nebo by jej muselo nahradit velké množství menších částí. Přestože samotný tisk může trvat desítky hodin až jednotky dnů, je stále velmi rychlý oproti dřívějším způsobům tvorby prototypů, a kromě času snižuje i počet nezbytných kroků. Značně také snižuje množství potřebného materiálu pro tvorbu cílového objektu. Obecně se pro 3D tisk nabízí široké množství využití jak profesionálními, tak amatérskými uživateli. Ve firmách se nadále užívá nejen k rychlému prototypování, ale využívá se, nebo se experimentuje s jeho využíváním, v celé škále nových průmyslových

[1] https://3dprintingindustry.com/news/evolution-3d-printing-past-present-future-90605/

a jiných odvětví. V lékařství je to například výroba kloubů na míru pacientovi nebo tisk tkání. Firma Relativity Space, spadající do oboru kosmického inženýrství, tiskne celé trupy vesmírných raket[2]. I při výrobě samotných 3D tiskáren lze 3D tisk využít k tvorbě součástí samotné tiskárny, jako to aplikuje česká firma Prusa Research[3]. Další využití lze nalézt v automobilovém průmyslu, v oděvnictví, potravinovém průmyslu nebo například v umění. Používání 3D tisku ve více odvětvích je spíše bráněno ze stran firem, pro které mohou být pořizovací náklady tiskáren stále příliš vysoké. Firmy se také potýkají s nedostatkem pracovní síly schopné s tiskárnami efektivně zacházet [9].

Benefity 3D tisku se ale většinou ztrácí při velkoobjemové produkci, kdy je pak časově výhodnější a levnější využít jiné metody. Vytisknutý model často po dokončení potřebuje navíc ještě finální úpravy jako vyhlazení nerovností nebo tepelnou úpravu pro zajištění požadovaných fyzikálních vlastností objektu. Další nevýhodou může být, že důsledkem použitých materiálů může být větší křehkost objektu, což se děje nejčastěji u plastových materiálů, kde se vrstvy pouze pokládají na sebe. Při tisku z kovových slitin, kdy se vrstvy taví nebo spékají k sobě, tento problém tak zásadní není.

2.1 Procesy aditivní výroby

Aditivní výroba se skládá ze čtyř nezbytných kroků. Ve specializovaném programu je potřeba vytvořit CAD (Computer-aided design) model, který chceme vytisknout, případně z nějakého internetového repositáře získat už model vytvořený. Takových repositářů, za nimiž stojí tiskařská komunita, je mnoho a lze na nich najít modely jak zdarma, tak placené. CAD model je potom potřeba převést do STL souboru, kdy je model převeden do formy sítě trojúhelníků. STL je standardní formát stereolitografických softwarů[4]. V tomto kroku lze nastavit jemnost hran modelu. Převedený model je pak potřeba rozřezat na vrstvy a vytvořit z něj instrukce pro tiskárnu. K tomu slouží speciální programy jako PrusaSlicer[5] nebo Slic3r[6]. Ty kromě převodu modelu na instrukce také často dotvoří výplně a podpěry modelu, aby se předešlo co nejvíce chybám v průběhu tisku. Lze v nich také nastavit požadované vlastnosti tisku jako tloušťka stěn, rychlost pohybu tiskové hlavy, způsob výplně, teplotu podložky a žhavící hlavy atd. Výsledné instrukce se pak nahrají do tiskárny a slouží k nastavení zmíněných vlastností i k samotnému ovládání hlavy při průběhu tisknutí. Na tiskárně dochází k provádění těchto instrukcí, což má za výsledek tisknutí objektu [6].

Krájení modelu na vrstvy

Než je model zaslán do tiskárny, je potřeba naplánovat průběh jeho tisku. Součástí toho je nakrájení modelu do vrstev, přidání podpor pro potřebné přečnívající části, vytvoření kroků cesty hlavy tiskárny nebo nastavení teplot. Krájení modelu na vrstvy (slicing) je způsob převádění 3D modelu do instrukcí jdoucí do 3D tiskárny. Model je protnut rovinou pomocí čehož se získá obrys vrstvy. Z vrstev je pak vytvořena síť trojúhelníků aproximujících povrch modelu. Tato síť musí rozdělit prostor na vnitřní, který bude při tisku vyplňován materiálem, a vnější. Míra výplně rozhoduje o pevnosti objektu, přičemž nejběžněji se používá 20% výplň. Při nutnosti vyšší pevnosti může být výplň i přes 80 %. U nižší výplně

[2] https://www.relativityspace.com
[3] https://help.prusa3d.com/cs/article/tisknutelne-casti-original-prusa_1824
[4] https://www.loc.gov/preservation/digital/formats/fdd/fdd000504.shtml
[5] https://www.prusa3d.com/cs/stranka/prusaslicer_424/
[6] https://slic3r.org

je ale tisk výrazně rychlejší a levnější. Geometrie výplně má taktéž vliv na pevnost objektu, běžně se tak využívá tvaru trojúhelníků, obdélníků nebo šestiúhelníků. Některé krájecí programy umožňují měnit geometrii a hustotu výplně v průběhu tisku [16].

G-code

3D tiskárny spadají do takzvaných číslicově řízených strojů. Jsou to stroje, které jsou automatizovaně řízené počítačem namísto lidské obsluhy. Stroj je ovládán speciálními sekvenčními instrukcemi, které mohou být napsány programátorem, častěji ale bývají generované jiným softwarem, u 3D tisku takzvaným Slicerem. Příkladem takových instrukcích je právě G-code nebo M-code, které využívá většina tiskáren. G-code je využíván na většině číslicově řízených strojů. Stejné instrukce ale mohou mít jiný význam na různých strojích. Při 3D tisku G-code instrukce slouží jako způsob oznámení tiskárně, jak má určený model vytisknout. Kromě příkazů pohybu na X-Y-Z osách slouží také k nastavení rychlostí pokládání materiálu apod. M-code (M jako Miscellaneous) slouží k různým příkazům, neobsazené M-code příkazy si může sám uživatel nadefinovat. Většina M-code příkazů slouží k jiným účelům na tiskárně, než je ovládání pohybu tiskové hlavy, například nastavení teplot extruderu (příkaz `M104`) a podložky (příkaz `M140`)[7]. Jeden řádek kódu značí jednu instrukci. Zdrojový kód tak pro většinu modelů muže mít i tisíce až deseti-tisíce řádků [8].

Tisk

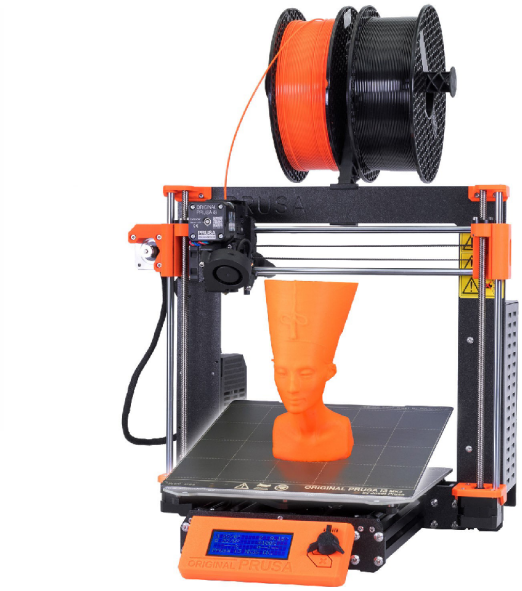
Pro samotnou aditivní výrobu, kterou se tato práce zabývá, existuje několik odlišných procesů a k nim příslušejících technologií. Druhy procesů často souvisejí s používaným materiálem. Proces, kdy je tekutý foto-polymer ozařován na vybraných místech paprskem, čímž dochází k jeho tuhnutí, se nazývá polymerizace. Využívanými technologiemi polymerizace jsou stereo-litografie (SLA) a Digital Light Processing (DLP). Stereo-litografie byla použita jako úplně první forma 3D tisku. Funguje utvrzováním foto-polymerového materiálu za pomocí ozařování laserovým paprskem. Foto-polymer je materiál, který mění své fyzikální vlastnosti při vystavení světlu o určitém světelném spektru. SLA je často využívána pro svou vysokou přesnost a kvalitu výsledného objektu. Ona kvalita a přesnost je ovlivněna zejména šířkou paprsku, hustotou polymeru a délce ozařování. Je druhou nejpoužívanější technologií mezi neprofesionálními uživateli, tou zdaleka nejpoužívanější je výroba tavením vláken (FFF)[8].

Nejrozšířenějším procesem je vytlačování materiálu a jeho nejpoužívanějších a zároveň jednou z nejlevnějších technologií je výroba tavením vláken (Fused Filament Fabrication – FFF, nadále bude využívána jen anglická zkratka, protože je běžně využívána i mezi odborníky v oboru, toto bude platit i pro další, níže zmíněné, technologie 3D tisku). Kvůli ochranným známkám je také jinak nazývána Fused Deposition Modeling (FDM), ale technologie jsou identické. Při ní postupně vstupuje do tiskárny tisková struna, takzvaného filamentu. V hlavě tiskárny je materiál taven a tryskou vytlačován po vrstvách do objektu. Průměr trysky bývá nejčastěji do 3 mm a platí, že čím menší průměr, tím vyšší je jemnost ploch a ostrost hran objektu. Tiskárny disponující touto technologií mívají celkové rozměry v desítkách centimetrů a tradičně mohou tisknout objekty o velikosti do maximálního dosahu tiskařské hlavy. Příkladem rozšířené amatérské tiskárny může být tiskárna Prusa i3

[7] https://marlinfw.org/docs/gcode/M104.html

[8] https://www.statista.com/statistics/560304/worldwide-survey-3d-printing-top-technologies/

[9] https://www.prusa3d.com/cs/produkt/3d-tiskarna-original-prusa-i3-mk3s-3/



Obrázek 2.1: FFF tiskárna Original Prusa i3 MK3S+. Převzato z webu *prusa3d*[9]

na obrázku 2.1. Existují ale tiskárny, které mají podložku ve formě posuvného pásu, kde tisknutý objekt postupně ujíždí do strany. Objekty pak mohou mít i na menší tiskárně délku teoreticky nekonečnou[10]. Tisková struna je nejčastěji z materiálu PLA a ABS. Tyto materiály se liší jak využitím, tak rozdílným potřebným nastavením tiskárny. Výtisky vyrobené z PLA jsou pevné, ale křehčí a zároveň jsou málo teplotně odolné, takže začínají měknout už při teplotě 60°C. ABS materiál je oproti tomu pružnější, ale je u něj zvýšený problém s teplotní roztažností. Výtisky z něj se tak častěji odlepují od zahřívané podložky a kroutí se [23].

Proces Powder Bed Fusion (Fúze na práškovém lůžku) funguje na spojování materiálu, který je ve formě jemného prášku. Ten je rozložen jako tenká vrstva na podložce a pomocí zvýšení teploty v určeném místě spojován v pevnou část. Používané technologie tohoto procesu se odlišují hlavně ve způsobu zahřívání a typu vkládaného materiálu. Jeho často používanou technologií je Direct Metal Laser Sintering (DMLS — Přímé sintrování kovů laserem). Na podložku tiskárny se postupně nanáší jemné vrstvy prášku kovové slitiny. Po nanesení každé vrstvy je laserovým paprskem vrstva spečena k sobě na místech podle vzorové vrstvy modelu. Tento způsob tisku funguje na podobném principu jako technika Selective laser sintering (SLS — selektivní laserové sintrování). Ta jako materiál k tisku může ale používat termoplast, kovový prach nebo keramický prášek [6].

Existuje dále ještě několik nových, experimentálních nebo méně často užívaných procesů, většinou nějak upravují výše zmíněné techniky tisku. Je to například proces tryskání materiálu – Material Jetting, kdy jsou kapky materiálu ukládány na vybrané místo ve vrstvě objektu, načež jsou ozařovány k zatuhnutí podobně jako při stereo-litografii. Při Binder Jettingu dochází k vypouštění spojující tekutiny na práškové lůžko [21].

[10] https://www.creality3dofficial.com/products/cr-30-infinite-z-belt-3d-printer

Dodatečné zpracování

Při pokládání vrstvy po vrstvě je většinou po dotisknutí často ještě nutné nějaké dodatečné opracování a doladění objektu. Jde hlavně o estetické vlastnosti, ale může jít i o přímo nezbytné změny. Mezi tyto nezbytné změny patří hlavně odstranění podpor nebo pospojování odděleně vytisknutých částí. Může být také potřeba vyplnit mezery mezi vrstvami a opravit tak chybu nekonzistentního proudění materiálu (viz 2.2). Volitelnou změnou může být obroušení ploch výtisku, užívané hlavně ke zvýšení přilnavosti barvy nebo lepidla. Mezi spíše estetické změny patří leštění nebo přebarvení stěn. Způsob dodatečného zpracování záleží hlavně na typu použitého materiálu. Na amorfní látky (s nepravidelnou strukturou), jako ABS, bývá používáno vyhlazování výpary. Na polykrystalické, jako PLA, bývá použit epoxidový nátěr [21], [6].

Součásti FFF tiskárny

Nejrozšířenější technologií 3D tisku mezi amatérskými tiskaři je FFF – Fused Filament Fabrication. Jako materiál používá plastové vlákno nazývané filament. Nejdůležitějšími komponentami FFF tiskárny jsou tisková hlava, vyhřívaná podložka a řídící jednotka. Tisková hlava neboli extruder slouží k tavení a pokládání materiálu. Do hlavy putuje, zatím v pevném stavu, vlákno filamentu. V hlavě se nachází žhavící těleso, které příchozí materiál taví a tlačí ho do trysky k vypuštění. Uživatel může před tiskem našroubovat trysku o chtěném průměru, čímž ovlivní jemnost ploch výtisku i časovou délku tisku. Může také nastavit teplotu v hlavě a rychlost vstupu materiálu a jeho vytlačování. Aktuální teplotu může uživatel sledovat a případně ji za běhu tisku měnit. Materiál je vypouštěn na vyhřívanou podložku. Vyhřívání podložky má zabránit deformaci materiálu z důvodu teplotní roztažnosti. Také má za cíl lepší přilepení výtisku k podložce. U FFF tiskáren je časté, že se hlava pohybuje pouze ve dvou osách: X a Z. Pohyb po třetí ose je umožněn posouváním samotné tiskové podložky s modelem. Kvalita vytisknutí první vrstvy na podložku má výrazný vliv na konzistentní umístění objektu a předcházení jeho pádu při menších pohybech na tiskárně. Celou tiskárnu ovládá řídící jednotka. Její funkcí je čtení vstupních G-code instrukcí, nastavování teplot a ovládání motorů pohybujících s tiskovou hlavou [23].

2.2 Přehled chyb v průběhu tisku

V této podkapitole budou vypsány nejčastější chyby, které mohou v průběhu tisku nastat. Je popsána jejich častá příčina a jejich projevení se. Znalosti popsané v této kapitole byly čerpány zejména z publikace *3D Printing Failures* [1] a z článků o tomto tématu na komunitních internetových blozích o 3D tisku *Simplify3D*[11] a *All3DP*[12].

Délka doby tisku se odvíjí převážně od velikosti objektu, komplexity jeho geometrie, zvolené výšce vrstev a hustoty výplní. Bývá u jednodušších modelů desítky minut až jednotky hodin, u složitějších může jít až o desítky hodin soustavného tisku. I pokud se tiskne jednodušší model, je po potřebnou dobu velmi nepraktické, aby na tisk uživatel konzistentně osobně dohlížel. Proto uživatelé často dohlíží na tisk jenom občas, aby zkontrolovali, že k žádné chybě nedošlo. Pokud k chybě v průběhu tisku však dojde, je malá šance její nápravy, a to navíc jen pokud je odhalena chvíli po vzniku. Po delší uplynuté době a po položení několika nových vrstev se chyba často projeví na tisknutém objektu globálně a

[11] https://www.simplify3d.com/support/print-quality-troubleshooting/
[12] https://all3dp.com/1/common-3d-printing-problems-troubleshooting-3d-printer-issues/

nenávratně. Důsledkem chyby je hlavně ztráta velkého množství času. Ztracený čas je totiž nejen čas po tom, co k chybě dojde až k jejímu odhalení, ale celková doba tisku před chybou, protože i ta část tisku je ve většině případů nepoužitelná. Dalším důsledkem je množství vyplýtvaného materiálu, obzvláště znatelné při použití dražšího filamentu. V nejhorším případě pak může dojít i k mechanickému poškození samotné tiskárny. Nejčastější chyby bývají způsobené špatným počátečním nastavením tiskárny, opotřebením tiskárny časem a jejím neudržováním, nerealistickým modelem a externím strůjcem v prostředí tiskárny.

Při tisku může nastat velké množství chyb. Za chybu je považováno nechtěné odchýlení se od vzorového modelu. To se projevuje jako chybějící materiál ve vrstvách modelu a přebývající materiál mimo místa vzoru. Chyby lze rozdělit podle závažnosti na opravitelné a terminální. Opravitelné chyby je možné s menšími obtížemi po dotisknutí zapravit a není kvůli nim potřeba pozastavovat tisk. Chyby terminálního charakteru, kdy dojde k deformaci celkových dimenzí tisknutého objektu, už je vhodné co nejdříve detekovat a jejich detekci hlásit. Ve výjimečných případech může dojít k poškození nejen tisknutého objektu, ale i tiskárny.

Chyby lze také rozdělit podle jejich příčin: chyby modelu, mechanické chyby a chyby nastavení tisku. K chybě CAD modelu dochází při nerealistických fyzikálních vlastnostech modelu, například most bez přidaných podpor. Těmto chybám lze někdy předejít otočením celého modelu na tiskárně o 45°. Mechanické chyby jsou způsobené tiskárnou nebo externím činitelem v prostředí tiskárny, například nárazem do tiskárny nebo přímo do samotného výtisku. Třetím typem jsou chyby způsobené špatným nastavením tisku jako nastavením nevhodných parametrů trysky nebo teploty tiskařské podložky s ohledem k materiálu. Například při příliš vysoké teplotě nestíhá dojít po položení materiálu k jeho tuhnutí a může dojít k propadnutí stěny výtisku.

Chybám modelu a nastavení tisku dochází s narůstajícími zkušenostmi uživatele postupně méně. Často se stává, že uživatel ale netiskne objekty pro sebe, ale v případě farmy tiskáren tiskne například pro známé nebo na zakázku. K vytisknutí ale nemusí dostat model ve formě STL, ale už vytvořený G-code. V něm mohou být skryty zmíněné nereálné vlastnosti modelu nebo nesprávné příkazy pro daný typ tiskárny a po nějakém čase tisku k chybě dojde [1]. Chyby jsou často provázané a jedna může způsobit druhou. Je tedy vhodné při nalezení chyby pozastavit tisk co nejdříve, než dojde k nenávratnému poškození tisknutého objektu. Pokud uživatel pracuje s farmou tiskáren a tiskne tentýž model vícekrát najednou, chyba se v jeho nepřítomnosti může projevit na více tiskárnách. Škoda se tak znásobí, protože musí každý tisk započít nanovo. Tiskařské farmy nejsou navíc doménou jen firem nebo profesionálů, bývají součástí přivýdělku amatérských uživatelů nebo slouží k edukativním účelům, jako například na Fakultě informačních studií Vysokého učení technického[13], kde pravidelně tisknou zároveň i čtyři tiskárny.

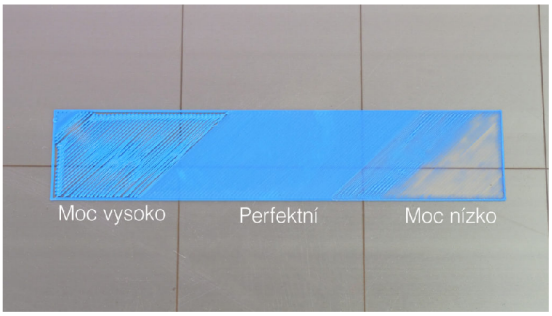
U většiny chyb platí, že se neprojeví pouze v jednom místě, ale projeví se opakovaně na více částech výtisku. Přestože je možných chyb a jejich příčin větší množství, jejich důsledky a projevení se často překrývají.

Odpojení od podložky

K odpojení od podložky může dojít, když se nepodaří správně vytisknout první vrstvu tisknutého modelu. První vrstva je při každém tisku ta nejdůležitější. Je důležité, aby byl tisknutý objekt dostatečně pevně připojen k tiskové podložce. Chyba první vrstvy bývá nejčastěji způsobena špatnou kalibrací souřadnic tiskárny, kdy se hlava při první vrstvě

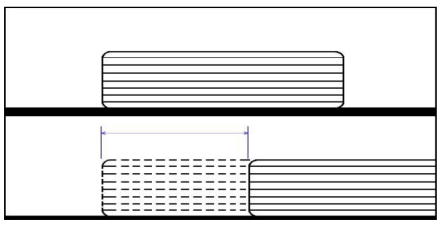
[13] https://www.fit.vut.cz/.cs

pohybuje příliš vysoko nebo příliš nízko po podložce. Pokud se hlava pohybuje příliš vysoko nad podložkou, není vrstva dostatečně přitlačena k podložce. Pokud se hlava pohybuje příliš nízko, nemá tryska dostatečný prostor k vypuštění veškerého materiálu a je tak uloženo jeho značně menší množství. Při nevyrovnané podložce může dojít k oběma druhům této chyby. Ukázku chybné vrstvy je možné spatřit na obrázku 2.2. Špatná první vrstva nedrží na podložce výtisk tak pevně, jak by měla a při běžném pohybu podložky po tiskárně může nastat úplné odpojení výtisku a jeho následný náhodný pohyb po podložce.



Obrázek 2.2: Chybný tisk první vrstvy, text v obrázku značí výšku trysky. Obrázek převzat z blogu Josefa Průši[15]

Odpojení (obrázek 2.3) může také nastat nárazem cizího objektu do výtisku. Pokud nastala jiná chyba a na modelu se nachází filament v místech, kde jej tiskárna neočekává, může do něj narazit i hlava samotné tiskárny, což způsobí odlepení a pád objektu. Důsledkem nízké teploty podložky může nastat odlepení objektu způsobené běžnými pohyby procesu tisku. Je nutné nastavit správnou teplotu podle používaného materiálu, příliš nízká teplota podložky může způsobit odlepení objektu, příliš vysoká může způsobit teplotní deformaci na hranách objektu. Uživatel může také zapomenout přidat podpory pro převislé části objektu, případně může dojít k jejich zlomení. To zapříčiní, že části, které podporu potřebovaly se bez nich odlomí nebo zapříčiní převážení a pád objektu. K odpojení může dojít v jakékoli fázi tisku. Chyba se v dalších vrstvách může projevit jako špagetová příšera nebo jako odsunutí vyšších vrstev modelu.



Obrázek 2.3: Pohyb tisku po odpojení od podložky, převzato z [3]

Ucpání trysky

K ucpání trysky může často dojít při výměně vlákna materiálu, kdy se v trysce odlomí a zasekne poslední kousek příchozího vlákna. Nové vlákno je pak blokováno v projití tryskou

[15] https://josefprusa.cz/jak-vyresit-nejcastejsi-problemy-pri-3d-tisku/

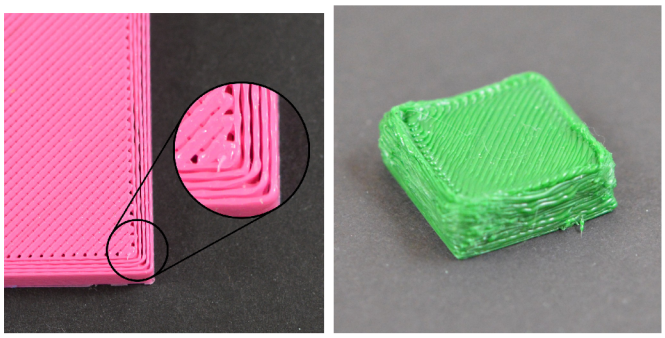
a nemusí dojít k pokládání materiálu. Chyba bývá častěji způsobena při změně typu používaného materiálu pro tisk. Nemusí dojít k úplnému zastavení přívodu materiálu, ale blokující překážka nechá projít jenom částečný tok přicházejícího materiálu. Výsledkem je nekonzistentní proudění materiálu, což se projeví na výsledné kvalitě tisku. K pokládanému materiálu mohou navíc být přilepovány malé kousky spáleného zaseknutého vlákna. Také může docházet k ukládání části materiálu u trysky hlavy. Pokud se materiálu nabalí dostatek, hlava může při přejíždění narazit vytvořeným výčnělkem do výtisku a způsobit jeho odlepení.

Dojití materiálu

Pro tisknutí je potřeba neustálý přívod materiálu. Každá tiskárna má ale uložení materiálu vyřešené jinak. U tiskáren Prusa je špulka s materiálem umístěna viditelně nad tiskárnou, proto jeho dojití bez vědomí uživatele není tak pravděpodobné. U tiskáren, kde je materiál schován v útrobách tiskárny, jako např. u tiskáren da Vinci, je šance na dojití bez kontroly před tiskem větší. U tisků, na které je potřeba využít značně větší množství materiálu, je taky obtížnější odhadnout, jestli je na špulce materiálu dostatek. Může dojít i k zamotání vlákna na špulce s materiálem nebo jeho prasknutí v místech mezi tavící hlavou a špulkou. Některé tiskárny jsou vybaveny fyzickým detektorem dojití materiálu, u některých je možné detektor přikoupit a doinstalovat k tiskárně[16].

Nekonzistentní proudění materiálu

Nekonzistentní proudění materiálu (under/over extrusion) může být způsobeno nevhodným diametrem trysky a vlákna materiálu v kombinaci s nastavením tisku. Může také dojit k dočasnému zaseknutí materiálu v příchodu do tiskárny a pozdějšímu ucpání trysky zmíněného v 2.2. Chyba se projevuje jako chybějící materiál v částech vrstvy objektu. Může také dojít k vypuštění většího množství materiálu na jiném místě. Ten pak může přečnívat a může u něj dojít k naražení hlavy, což může způsobit řadu dalších problémů. Chyba nízkého i nadměrného vytlačování je vyobrazena na obrázcích 2.4.

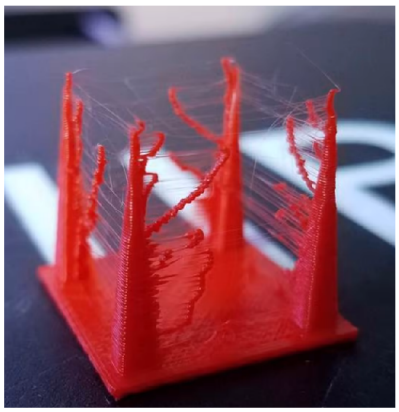


Obrázek 2.4: Nedostatečné (vlevo) a nadměrné (vpravo) vytlačování materiálu. Obrázky převzaty z blogu *Simplify3D*[18]

[16] https://www.prusa3d.com/cs/produkt/ir-filament-senzor/
[18] https://www.simplify3d.com/support/print-quality-troubleshooting/

Stringování

Stringování jsou kapky na vrstvách výtisku a z nich tyčící se tenké vlásky materiálu. Projevuje se tak, že tryska po odjetí z hrany za sebou nechává tenká vlákna materiálu. Chyba je častěji vídána při tisknutí z ohebných materiálů. Bývá způsobena špatným nastavením zatahováním vlákna a příliš vysokou teplotou v tavící hlavě. Chyba je pak obzvláště viditelná, pokud hlava musí přejíždět větší vzdálenosti bez extruze při tisku vrstvy. V neposlední řadě může být chyba také způsobena ucpáním trysky hlavy, zmíněném v 2.2. Příklad této chyby lze vidět na obrázku 2.5.



Obrázek 2.5: Chyba typu stringing. Obrázek převzat z blogu *all3Dp*[19]

Špagetová příšera

Pokud z libovolného důvodu dojde k pokládání nového materiálu do vzduchu, vzniká takzvaná špagetová příšera či krátce špagety. Špagetová příšera nastává, pokud tiskárna pokládá materiál do vzduchu, protože v daných místech očekávala poslední vrstvu výtisku, na kterou by byl nový materiál položen. To se stává i při menším pohybu výtisku po podložce. Chybu může také způsobit špatné nakrájení STL modelu. Při špagetové příšeře už je pozdě na nápravu a tisk je nutné započít znova [2]. Chyba je vyobrazena na obrázku 2.6.

Obrázek 2.6: Chyba typu špagety. Obrázek převzat z blogu *Prusa Knowledge Base*[20]

[19] https://all3dp.com/2/3d-print-stringing-easy-ways-to-prevent-it/

Deformace objektu

Důsledek některé z předchozích chyb, případně jejich kombinace, může způsobit, že tisk nepůjde podle představ uživatele. Deformace může být zapříčiněna špatnou volbou materiálu s nevhodnými fyzikálními vlastnostmi potřebnými pro daný model. Výtisku se kvůli teplotní roztažnosti materiálu mohou v místech od sebe odlepit vrstvy nebo se mohou propadnout celé zdi z vrstev. Časové opotřebení tiskárny nebo její špatné seřízení může také vést k nechtěným výsledkům. Uvolněné šrouby nebo řemen, ovládající pohyb hlavy, mohou způsobit nepřesnost pohybu hlavy po prostorových osách, což vede k odchylkám při pokládání vrstev a výraznému odjetí hran u vyšší části objektu. Platí, že s přibývajícími zkušenostmi uživatele chyb ubývá, jelikož ví, jakými preventivními kroky se jim může vyhnout. Stále však zůstává nemalé množství chyb, které mohou nastat a nelze je předvídat [19]. Obzvláště pokud uživatel tiskne model, který mu byl zaslán, nebo když sdílí tiskárnu s dalšími uživateli.

2.3 Současná řešení detekce chyb

Pro monitorování 3D tisku existuje řada zdarma dostupných aplikací. Většinou se ale zabývají jen přenosem obrazu od tiskárny k uživateli a nezajímá je jakékoli hlubší porozumění obrazu a toho, co se při tisku děje. V této části je popsáno několik studií a komerčních řešení, které se pokoušejí řešit podobný okruh problémů jako tato práce. Zaměřují se na užší výběr chyb i na komplexní spojitosti více chyb a jejich příčin. Využívají k tomu převážně počítačové vidění a strojové učení, některé pracují i se vstupním CAD modelem. Řešení zde pracují se vstupem z jedné i dvou kamer i se speciálními snímači.

Odborné práce

Studie s názvem *Development of a low-cost monitoring system for open 3d printing* [10], snažící se vyvinout monitorovací systém pro 3D tisk, porovnává referenční obraz vytvořený z G-code instrukcí s obrazem z kamery. Dokáže tak za běhu poznat chybějící a přebývající materiál v konkrétních místech. Po každé vrstvě umístí tiskařskou hlavu do rohu a kamerou umístěnou nad tiskárnou vyfotí tisknutý objekt. Fotografii pak porovná s příslušnou vrstvou modelu. Nedokáže ale chyby více rozlišit.

Práce *Open source computer vision-based layer-wise 3D printing analysis* [19], analyzující vrstvy 3D tisku pomocí počítačového vidění, rozlišuje 26 druhů chyb, z nichž některé se snaží rovnou opravit vygenerováním nových G-code příkazů. Z kamery umístěné nad tiskárnou si aplikace maticově přepočítá obraz na pohled z hora a pohled z boku, přes které pak lépe zjišťuje deviace v textuře. Z důvodu odlišného nasvícení částí tisknutého objektu používá metodu učení bez učitele, konkrétně algoritmus GMM (The Gaussian Mixture Model). Pro část chyb se snaží generovat nové instrukce pro tiskárnu ve snaze opravit detekovanou chybu.

Publikace *Real-Time 3D Printing Remote Defect Detection (Stringing) with Computer Vision and Artificial Intelligence* [17] využívá počítačové vidění a detekci objektů k detekci chyby stringování. K detekci objektů využívá konvoluční neuronovou síť architektury VGG16, konkrétně modul Single Shot Detector. V detekci dosáhli přesnosti (precision) 44 % a při metrice Recall, která vyjadřuje poměr detekovaných pozitivních případů vůči všem pozitivním případům (viz 3.4), dosáhli 69% úspěšnosti na testovacím datasetu.

[20] https://help.prusa3d.com/article/spaghetti-monster_1999

Komerční řešení

Existuje několik komerčních řešení. Nejčastěji používaným je aplikace Spaghetti Detective[21]. Ta funguje jako rozšíření do aplikace OctoPrint a za pomocí umělé inteligence detekuje volně pokládaný filament, takzvané špagety. Aplikace má otevřený zdrojový kód. Z aplikace OctoPrint zasílá snímky z tiskárny na server, kde je analyzuje. Uživateli je umožněno spustit server u sebe nebo využít jejich cloudových služeb a zasílat snímání tam. Detekuje, pouze pokud se chyba projeví jako špagetová příšera.

PrintWatch[22] je nová aplikace, vydaná v lednu 2022, fungující také jako plugin do aplikace Octoprint. Vysílá snímky tisku na cloudový server, kde je analyzuje pomocí strojového učení a detekce objektů. Existuje pouze v placené variantě založené na měsíčním předplatném. U žádného z těchto komerčních řešení nebylo možné nalézt přesnost nebo jiné metriky vyhodnocení pro jimi používané modely detekce.

[21] https://www.thespaghettidetective.com/
[22] https://printpal.io/

Kapitola 3

Využití počítačové vidění a neuronové sítě pro detekci chyb

Protože je většina chyb, které mohou v průběhu 3D tisku nastat, viditelná lidským okem, nabízí se pro jejich detekci využít počítačové vidění. Počítačové vidění má za cíl umožnit počítači porozumět digitálním obrázkům nebo videím. Umělé neuronové sítě jsou jednou z architektur využívaných pro strojové učení. Byly koncipovány ve 40. letech 20. století, koncept jejich trénování vznikl o 20 let později s příchodem algoritmu zpětné propagace chyby. Byly ale příliš výpočetně náročné na implementaci. Jejich rozmach tak nastal až v několika posledních letech díky výkonnějším počítačům, optimalizovanému hardwaru (GPU, TPU) nebo dostupným velkým datasetům. Konvoluční neuronové sítě jsou potom propojení počítačového vidění se strojovým učením.

Pakliže nebude uvedeno jinak, většinu informací v této kapitole jsem čerpal z knih *Advanced Applied Deep Learning* [15], *Deep Learning for Computer Vision with Python* [4] a *Deep Learning for Computer Vision with Python* [22].

3.1 Reprezentace obrazu v počítači

Základní stavební jednotkou snímku je pixel. Pixely jsou nejmenší, nedělitelnou součástí snímku. Snímky mohou být barevné nebo v odstínech šedi. V šedo-tónovém snímku má každý pixel jednu hodnotu od 0 do 255. Každá hodnota mezi těmito dvěma značí odlišný odstín šedi, přičemž krajní hodnota 0 je černá, hodnota 255 je bílá. U barevného obrázku má každý pixel hodnoty 3. Jejich význam se odlišuje podle používaného barevného modelu. Nejpoužívanějším barevným modelem je RGB (Red, Green, Blue), kde hodnoty označují intenzitu těchto jednotlivých barev, které se poté opticky zdají jako jedna výsledná barva. Protože každá ze tří barev v pixelu může mít jednu z 256 intenzit, jejich smícháním můžeme vytvořit až 256^3, tedy přes 16 milionů různých barev [22].

3.2 Umělé neuronové sítě

Umělé neuronové sítě jsou třídou algoritmů strojového učení, které se specializují na rozpoznávání opakovaných vzorů. Jsou inspirovány neurony, stavebními jednotkami mozku a jejich propojeními – synapsemi. Umělé neuronové sítě jsou strukturované grafy, kde každý uzel (neuron) provádí nějaký jednoduchý výpočet. Umělý neuron bere vstupní signál, zpracuje ho a zašle výsledek k němu připojeným neuronům. Každý neuron má váhu, která

zvyšuje nebo snižuje jeho výstupní signál. Neurony jsou řazeny do vrstev, kdy neurony na jedné vrstvě se navzájem neovlivňují. Každá další vrstva transformuje vstup na vyšší úroveň jeho abstrakce.

Cílem trénování neuronové sítě je minimalizace chybové funkce na trénovacím datasetu. Nejpoužívanějším algoritmem pro učení neuronových sítí je algoritmus zpětného šíření chyby (backpropagation). Spočívá v minimalizaci chyby na výstupu úpravou vah neuronů podle záporného gradientu chybové funkce. Algoritmus postupuje od výstupní vrstvy ke vstupní vrstvě. Aby bylo možné gradient spočítat, je nutné, aby aktivační funkce neuronů byla spojitá a bylo ji tak možné derivovat [12]. Pro rozpoznání obrazu se používá speciální druh neuronové sítě – konvoluční neuronová síť. Dalším speciálním typem může být rekurentní neuronová síť používaná k rozpoznání sekvenčního signálu nebo textu [13].

Aktivační funkce

Aktivační funkce generuje typicky nelineární hodnotu na základě vstupní hodnoty. Nelinearita je důležitá pro zjištění vztahu mezi vstupními a výstupními hodnotami, je nezbytná pro řešení nelineárních problémů, což je v běžném užívání většina. Vypočtené výsledky předává další vrstvě. Používá se ve skrytých vrstvách a ve výstupní vrstvě sítě. Ve skrytých vrstvách aktivační funkce určuje, jestli neuron předá nebo nepředá svou hodnotu do další vrstvy. Příkladem může být ReLU (usměrněná lineární funkce), která je nejčastěji používanou aktivační funkcí. Je definována jako $Relu(x) = max(0, x)$. Jsou používány také upravené verze této funkce, jako exponenciální ELU nebo děravá (leaky) ReLU. Dalšími aktivačními funkcemi mohou být funkce binárního kroku nebo Gaussovská funkce [11].

Klasifikace obrazu

Klasifikace obrazu je přiřazení kategorie (labelu) k obrázku z předdefinované množiny kategorií. Rozpoznání obrazu je velmi jednoduché pro člověka, ale velmi obtížné pro počítač, počítač totiž vidí obrázek pouze jako matici hodnot pixelů. Dobrý model rozpoznání musí rozpoznat variace detekovaného objektu. Mezi možné variace patří změna úhlu pohledu a osvícení, zakrytí části objektu nebo jeho deformace a variace v rámci třídy [22]. Pro klasifikaci obrazu je využívána konvoluční neuronová síť. Ta je nejčastěji natrénovaná, pro co nejlepší výsledek, na menší množinu kategorií týkajících se specifického problému.

3.3 Konvoluční neuronové sítě

Konvoluční neuronové sítě jsou speciálním druhem umělých neuronových sítí. V tradičních neuronových sítích jsou běžně používány plně propojené vrstvy kdekoli v jejich architektuře, v konvolučních sítích jsou ale používány pouze v poslední vrstvě. Plně propojená vrstva je totiž nahrazena za konvoluční vrstvu. Každá vrstva v této síti používá jinou sadu filtrů. Těmito filtry získává informace o hranách, rozích a tvaru objektů v obrázku a z těchto informací vytváří vyšší abstrakce jako postavu člověka nebo okno domu. Poslední, plně propojená vrstva, slouží k přiřazení předpovídané kategorie k obrázku [22].

Konvoluční vrstva

Slouží k extrakci vlastností (features). Započítává prostorovou strukturu vstupu a zkoumá, jestli hodnoty blízko sebe spolu nějak souvisí. Vstupem je dvourozměrná matice např. obrázek a výstupem je dvourozměrná mapa vlastností. Pomocí posuvného okna provádí kon-

voluci na vstupní matici. Neurony v konvoluční vrstvě nejsou propojeny s celým vstupem, ale vidí pouze menší obdélníkovou oblast, která se nazývá receptivní pole. Díky tomu model získá spíše více nízko-úrovňových informací o vstupu. Bylo by navíc téměř nemožné natrénovat síť na snímcích s velkými prostorovými rozměry [22].

Pooling vrstva

Snižuje vzorkování vstupních dat. Pokouší se zmenšit problém, kdy je mapa vlastností citlivá na polohu vlastností na vstupu. Snížením rozlišení jsou mapy vlastností odolnější na změnu poloh vlastností. Pokud například detekujeme na větším objektu hrany, zůstane nám na plochách velké množství místa s nulovou hodnotou. Místa s nulovou hodnotou ukládají pouze poziční informaci relevantních vlastností. Pooling tyto místa zmenšuje. Nejpoužívanější typy poolingu jsou Max Pooling a Average Pooling. Jak názvy napovídají, MaxPooling ponechá největší hodnotu nacházející se právě v posuvném okně. Average Pooling uloží průměrnou hodnotu okna [15].

Plně propojená vrstva

Také nazývaná jako hustá (dense) vrstva bývá umístěna jako jedna z posledních vrstev architektur modelu sloužící ke zhuštění získaných vlastností pro rozdělení do tříd k predikci. Každý její uzel je propojen se všemi uzly její nadcházející vrstvy a každé propojení má svou váhu. Každý vstupní neuron má tak vliv na výstupní hodnotu [11].

Posuvné okno

Je využíváno při konvoluci a poolingu. Nebylo by praktické získávat vlastnosti v jediném kroku přes celý obrázek. K tomu slouží dvourozměrná matice nazývaná posuvné okno. Při konvoluci se okno nazývá jádrem (kernel). Okno námi určené velikosti je přikládáno místo vedle místa na zvolený obrázek a je na něm aplikována chtěná operace, jako detekce hran nebo získání maximální hodnoty.

Jeho parametry jsou jeho velikost, velikost kroku a okraj. Velikost okna určuje, jak velkou oblast chceme při jednom kroku pokrýt. Krokem můžeme nastavit, že okno se nebude přesouvat pixel po pixelu, ale bude jich několik přeskakovat. Při pokládání okna ke hranám vstupu se snižují dimenze výsledné vrstvy. Tomu můžeme zabránit nastavením, jak pracovat s pixely za okraji vstupu. Můžeme jim nastavit nulovou hodnotu, dát jim hodnotu původního okraje nebo zvolit jinou. Posuvné okno je pak přiloženo i přes nové okrajové pixely a prostorové velikosti zůstanou zachovány [15].

3.4 Trénování CNN

Při trénování neuronové sítě se datová sada nejčastěji rozdělí na tři podmnožiny. Největší část, kolem 70 % se použije k natrénování modelu. Dalších zhruba 20 % se pak použije k validaci modelu. Zbylá část se použije k testování. Při menším datasetu může vyčlenění testovací části výrazně negativně ovlivnit kvalitu modelu, proto se může, s vědomím rizik, přeskočit. Validační i testovací sada se modelu při trénování skryje, aby nebylo testování ovlivněno, a trénování probíhá pouze na trénovací podmnožině datasetu. Vyzkoušení modelu na validační sadě nám pak pomůže odhadnout kvalitu modelu. Získané výsledky kvality pak můžeme překontrolovat a případně potvrdit na testovací sadě.

Vyhýbání se přetrénování

Při trénování neuronové sítě je běžné, že model vidí tatáž vstupní data opakovaně. Přetrénováni (overfitting) je jev, kdy se model příliš detailně naučí vlastnosti vstupních dat, včetně jejich šumu a odchylek. Model má pak výborný výsledek na trénovacích datech, ale právě přetrénování se negativně projeví na značně nepřesném výsledku modelu na nových, neviděných datech. Čím více dat poskytneme našemu algoritmu strojového učení, tím lepší a efektivnější budou jeho výsledky. U klasifikace obrazu se ale často setkáváme s nedostatečným počtem vstupních dat [18]. Nízký počet dat velmi zvyšuje pravděpodobnost přetrénování sítě.

Pro vyvarování se přetrénování na malé datové sadě existuje několik metod. Na úrovni samotné sítě existuje metoda Dropout, která pravidelně náhodně zablokuje několik neuronů na vrstvě sítě, a tím snižuje šanci koadaptace. Koadaptace je stav, kdy několik neuronů na téže vrstvě extrahují stejná nebo velmi podobná data a jejich signifikance je tím pak znásobena. Náhodným vypouštěním neuronů se výskyt tohoto jevu snižuje [22]. Snahou tohoto přístupu je, aby síť dosahovala dobrých výsledků i s jen náhodnou podmnožinou neuronů. Pomáhá také se šumem v trénovací sadě [11].

Při trénování je pravidelně kvalita modelu zjišťována na validačním datasetu. Pokud je zjištěno, že přesnost detekce se stále zvyšuje na trénovacích datech, ale přesnost na validačních datech se začíná snižovat, je možné trénování předčasně ukončit. Tím se zamezí většímu přetrénování.

Další metodou prevence přetrénování je transfer learning, tedy přenos učení. Je to technika, kdy se použije předem natrénovaná síť a spodní část se nastaví jako nenatrénovatelná. Uživatel pak trénuje na svém specifickém problému jen několik posledních vrstev. Využijeme tak uloženou znalost a aplikujeme ji na jiném, ale obdobném problému. Proto se často používají sítě natrénované na milionech vstupních příkladů, jejichž architektura se projevila jako nejlepší na světových soutěžích pro neuronové sítě.

V neposlední řadě lze použít metodu normalizace dávek. Metoda normalizuje vrstvy a umožňuje zpracovávat i normalizované váhy. Krom snížení šance na přetrénování také síť zrychluje a dělá ji stabilnější. Téměř vždy ale platí, že více vstupních dat poskytne lepší výsledky. Při nedostatku dat je možné další data uměle z dostupných dat vytvořit a rozšířit tak trénovací datovou sadu.

Rozšíření datové sady

Asi nejlepším způsobem, jak vylepšit model strojového učení, je natrénovat jej na více datech. Pokud ale není možné získat přístup k více datům, je možné sadu rozšířit uměle. Rozšíření datové sady může probíhat vytvářením úprav sady, která je k dispozici nebo Generativní soupeřící sítí. Nejběžněji se dnes datová sada rozšiřuje o upravené snímky tradičními transformacemi [5]. Může se jednat o geometrickou transformaci jako zrcadlení podle jedné nebo obou os, změnu perspektivy nebo rotaci. Lze také upravit barevnou informaci v datech jako posunem barevného spektra, změnou kontrastu nebo ostrosti. Nejčastěji se používají kombinace těchto úprav. Je ale důležité změny provádět rozumně a s opatrností, aby zůstaly zachovány podstatné informace, které chceme model naučit. Například není možné provádět zrcadlení podle horizontální osy, pokud má model rozpoznávat ručně psaná čísla, došlo by totiž k záměně čísel 6 a 9.

Rozšíření počtu dat je běžně prováděno za běhu, tedy změny provádíme s jistou náhodností, až když je předkládáme v dávkách do sítě při jejím trénování. K tomu slouží rozšiřující vrstva sítě. Je možné aplikovat rozšiřující transformace přímo na dataset, ale

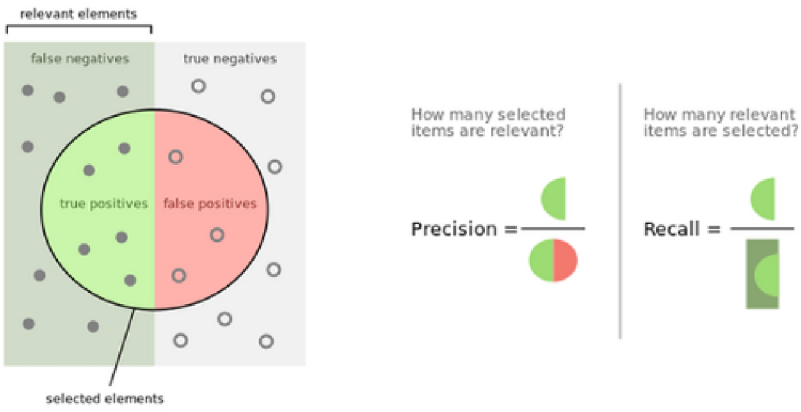
častěji se změny aplikují inline za běhu trénování. Změny a jejich velikost se pak vybírají náhodně a při každém přístupu k obrázku v rámci várky může být změna jiná.

Nové snímky můžeme také vytvořit pomocí generativních soupeřících sítí. Tyto sítě vytvářejí nová data aplikováním určeného stylu na snímky v datasetu. Nejedná se ale o pouhé jednodušší instagramové filtry. GAN zachytí charakteristiky vybrané kolekce obrázků a ty se pak pokusí přenést na náš vložený snímek [18]. Datovou sadu tak můžeme zdvojnásobit použitím původních snímku a jejich stylově transformovaných kopií. Příkladem takové sítě může být DeepDream od společnosti Google [25].

Vyhodnocení vytvořeného modelu

Pro vyhodnocení kvality modelu je možné využít celou řadu metrik. Metriky často využívají rozdělení predikcí na true positives, false positives, false negatives a true negatives. True positive značí, pokud model správně předpoví přítomnou vlastnost. False positive nastává, když model předpoví přítomnou vlastnost i přestože tam není. False negative značí, kdy model nerozezná vlastnost, i přestože se tam nachází. True negative je nerozeznání vlastnosti v případě, že je opravdu nepřítomná. Z poměru těchto predikcí na testovacím datasetu je možné získat kvality modelu.

Pro různá použití modelu se vhodné metriky liší. Někdy nejsou přípustné false positive předpovědi, ale také na nich nemusí tak záležet. Metrika Precision (preciznost) udává poměr true positive predikcí oproti všem predikcím. Recall neboli senzitivita udává poměr detekovaných pozitivních oproti celkovému počtu pozitivních případů [20]. Grafické znázornění těchto metrik ukazuje obrázek 3.1. Accuracy (přesnost) popisuje, jak často souhlasí predikce se skutečnou hodnotou [14].



Obrázek 3.1: Znázornění metrik Recall a Precision. Převzato z webu *Towards Data Science*[2]

3.5 Detekce objektů

Rozdíl mezi klasifikačním algoritmem a algoritmem detekce objektů je ten, že druhý jmenovaný se pokouší vrátit i lokaci nalezeného objektu a případně kolem něj vykreslit vyznačující ohraničení nebo jinak objekt zvýraznit. Také dokáže rozpoznat více druhů tříd na stejném

[2] https://towardsdatascience.com/whats-the-deal-with-accuracy-precision-recall-and-f1-f5d8b4db1021

snímku, na rozdíl od klasifikace, která rozpozná jen ten nejpravděpodobnější. Příkladem algoritmu detekce mohou být algoritmy R-CNN, Fast R-CNN, Faster R-CNN, YOLO nebo SSD, přičemž Fast a Faster R-CNN jsou vylepšené verze téhož algoritmu. Verze algoritmu R-CNN používají dva kroky. První pomocí Region Proposal Net (RNN) vygenerují předpokládanou oblast a pak na ní provedou klasifikaci a regresi. Verze algoritmu YOLO (You Only Look Once) stojí hlavně na regresní detekci objektu a algoritmu rozpoznávání, který používá pouze jednu síť a vrací krajní meze ohraničujícího boxu detekovaného objektu. YOLO bývá rychlejší, ale mívá problém s detekcí menších objektů a o něco nižší průměrnou preciznost (Mean Average Precision) [24].

Kapitola 4

Návrh

Protože je značně nepraktické, aby uživatel neustále kontroloval několikahodinový proces tisknutí na 3D tiskárně, nabízí se vytvořit aplikaci, která to bude dělat za něj. Pokud aplikace nějakou chybu detekuje, vyšle uživateli upozornění, podle kterého se pak sám rozhodne, jestli v tisku pokračovat nebo začít tisk nanovo. Při dostatečně vysoké jistotě přítomnosti chyby by aplikace mohla tisk přerušit i sama. Protože je většina chyb, které při 3D tisku nastávají, viditelná lidským okem, má smysl pro detekci daných chyb využít počítačového vidění. Vytvořený systém, který bude snímky zároveň pořizovat i analyzovat, by měl být co možná nejjednodušeji instalovatelný a zároveň co možná nejméně finančně nákladný na pořízení. Z těchto důvodů jsem se rozhodl využít možnost vytvoření aplikace jako rozšíření do aplikace Octoprint. Ta umožňuje využít její rozhraní pro ovládání a monitoring tiskárny, ke které je Octoprint připojen. Aplikace Octoprint běží na mikropočítači Raspberry Pi připojeném k tiskárně. Uživatelské rozhraní aplikace je dostupné z prohlížeče přes bezdrátovou lokální síť.

Pro tisknutí na 3D tiskárně bylo tradičně potřeba nahrát chtěný model na SD kartu a tu pak vložit do slotu na kartu v tiskárně. Tomu se aplikace Octoprint vyhýbá tak, že je nainstalovaná jako operační systém na mikropočítači Raspberry Pi, který je připojen USB kabelem k tiskárně a přes WiFi k domácí síti. Poskytuje uživatelské prostředí, přes které mohou uživatelé nahrát své modely do tiskárny vzdáleně a stejně tak započít jejich tisk. Při zapojení kamery také umožňuje tvorbu časosběrných záznamů. K aplikaci je možné vytvořit zásuvné moduly a její možnosti ještě rozšířit. Aplikace modulům nabídne zjednodušený přístup k ovládání tiskárny, čtení aktuálních parametrů tisku nebo přístup ke snímkům kamery při tvorbě časosběrného videa. Pro svoje možnosti se stala velmi populární a běží na desítkách tisíců 3D tiskáren. Pro uživatele, kteří tuto aplikaci používají, je pak instalace mého systému otázkou prakticky jednoho kliknutí, a to díky správci rozšíření přítomného v aplikaci. Přes něj je také možné rozšíření aktualizovat nebo odinstalovat. Při vytvoření nového vydání na GitHub repositáři sám nabídne možnost aktualizace.

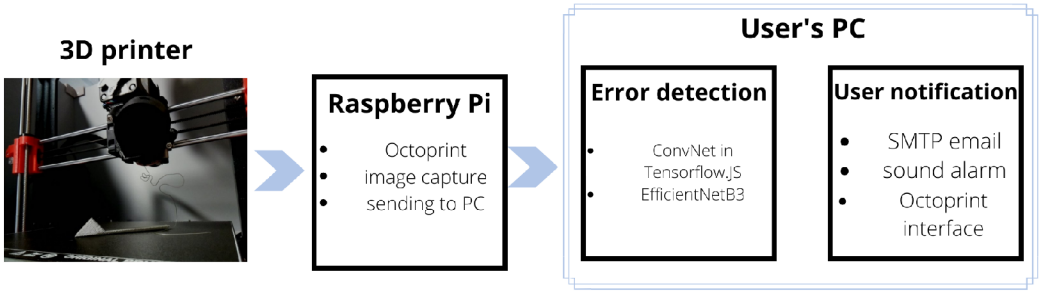
V repositáři zásuvných modulů, které aplikace Octoprint nabízí, se nachází dva detektory chyb využívající počítačové vidění. Ty ale zasílají získaný obraz na vzdálený server, kde jej analyzují a do uživatelského rozhraní vrací pouze nalezenou chybu. Obě aplikace navíc fungují na bázi měsíčního předplatného. To nemusí být pro každého dostupné, navíc jejich využití závisí vysoce na připojení k internetu. Jeden z těchto modulů – Spaghetti Detective umožňuje spuštění serveru s detekcí i u uživatele, který se tak vyhne předplatnému i zasílání svých dat. Toto spuštění není ale moc přívětivé pro běžné uživatele. Navrhl jsem tak aplikaci, která všechny kroky bude dělat lokálně na počítači uživatele. Bude navíc

pracovat ve webovém prohlížeči, aby odpadla závislost na konkrétním operačním systému a nutnost cokoli navíc instalovat.

4.1 Návrh systému

Většina chyb, které mohou nastat při 3D tisku, jsou detekovatelné lidským okem. Protože je ale nepraktické kontrolovat tisk po celou jeho dlouhou dobu člověkem, je na místě použít počítačové vidění. Cílem práce bylo využít možnosti počítačového vidění k automatické identifikaci častých chyb v průběhu 3D tisku.

Přišlo mi nepraktické vytvořit aplikaci, kterou by uživatel musel ručně spouštět při započetí každého tisku. Také jsem chtěl, aby celý systém mohl běžet lokálně u uživatele a jeho implementace do tiskařského systému uživatele byla co možná nejméně náročná, nic v systému nenarušila a pouze rozšířila jeho možnosti. Proto jsem se rozhodl k tomu využít aplikaci Octoprint. Tu využívá velký počet amatérských tiskařů. Octoprint umožňuje za běhu komunikovat s tiskárnou, získávat z ní aktuální pozici hlavy, upravovat nahrávaný G-code a získávat snímky z připojené kamery. Funguje také jako server přístupný z lokální sítě. Uživatel tak může ovládat tiskárnu z uživatelského rozhraní ve webovém prohlížeči. Snímky je možné pořizovat jak v pravidelném časovém intervalu, tak při posunu hlavy po ose Z, tedy po dokončení jedné vrstvy a posunu hlavy na další. Navrhovaný systém bude získávat snímky tisku, detekovat na nich chyby a bude poskytovat výstup, který upozorní uživatele na chybu a případně jej přiměje, aby tisk přerušil nebo zcela zastavil. Implementace systému do pracovního postupu není navíc ani nijak náročná, mnoho nových tiskáren je už vybaveno kamerou a případná pořizovací cena kamery je v řádu několika stokorun. Zjednodušené schéma navrhovaného systému je znázorněno na obrázku 4.1.



Obrázek 4.1: Zjednodušený návrh systému

Práce je cílená na FFF typ tiskárny, který je mezi amatérskými uživateli nejrozšířenější. Bylo by možné rozšířit použití i na SLA tiskárny, ale bylo by nutné rozšířit datovou sadu o snímky z tisku na nich a úspěšnost detekce nebude pravděpodobně tak vysoká jako u FFF tiskáren. Při tisknutí na SLA tiskárně je z důvodu vytváření jedovatých výpar tisknutý

objekt zakryt hůře průhledným plastovým víkem, které u těchto tiskáren zabraňuje úniku výpar do okolí.

Tím, že je Octoprint spuštěn na Raspberry Pi, využívá velkou část jeho výpočetní kapacity pro ovládání tiskárny, pořizování snímků a podobně. To je znatelné hlavně na výpočetně slabších řadách Raspberry Pi. Klasifikace obrazu je ale poměrně výpočetně náročná. Bude tedy zapotřebí provádět velkou část výpočtu na jiném zařízení. K tomu lze využít uživatelské rozhraní Octoprintu. Přístup k němu je ale možný jen na lokální síti. To je možné rozšířit například pomocí zásuvného modulu OctoEverywhere[1], který odstraňuje limitaci lokální sítě a umožňuje přístup odkudkoli. Výpočetní kapacita externího počítače bude pravděpodobně mnohem vyšší než používaného Raspberry. Rozhodl jsem se tak spouštět část s klasifikací obrazu v prohlížeči počítače, ze kterého se k Octoprintu přistupuje.

Octoprint umožňuje vytvořit zpětné volání (callback) funkcionality rozšíření při mnoha událostech nastávající na tiskárně. Logika mojí aplikace tak bude provázána s tvorbou časosběrného videa a volána při každém úspěšném pořízení snímku výtisku. Interval pořizování snímků bude nastavitelný uživatelem před započetím tisku. Při zavolání se zaznamená aktuální čas a spolu s ním se snímek zašle na frontend běžící na počítači. V backendu je také implementováno uložení nastavení prvků uživatele jako email, heslo, práh pro jistotu detekce a údaje pro SMTP službu.

Tím, že část detekce běží v prohlížeči, může uživatel nechat otevřenou záložku s běžící instancí detektoru v pozadí a počítač nadále využívat. Zatížení počítače se bude odvíjet převážně od frekvence pořizování snímků a volání logiky detekce. K detekci ale bude použit co možná nejméně náročný model neuronové sítě. Tím, že je logika vázaná na příchod snímků, může uživatel využívat detekci na starších počítačích s tím, že pouze sníží frekvenci pořizování a tím tak prodlouží čas pro výpočet na jednom snímku. Nižší frekvence také umožní bezproblémovější ostatní práci s počítačem, na kterém detekce běží.

4.2 Detekce chyb

Pro co nejrychlejší odhalení chyb se vhodné algoritmy liší podle druhu chyby. Pro detekci pohybu výtisku jako posunu po podložce nebo jeho pádu je nutné pozorovat změny v čase, a proto využít jednodušší algoritmy pro zpracování obrazu. Pro odhalení chyb na základě jejich typického vizuálního vzhledu se nabízí použít algoritmu strojového učení, konkrétně konvoluční neuronovou síť, která pro detekci chyby využije klasifikaci obrazu. Systém je tak rozdělen do několika částí, kdy každá část vykonává určené kroky pro monitorování tisku, záznam obrazu, jeho následné analýzy a přístup uživatele. Jako chyby změn v čase počítám pohyb výtisku po podložce nebo pád výtisku a dojití materiálu. Pohyb po podložce by měl být detekovatelný jako větší změna v obraze oproti několika předešlým snímkům. Při pohybu po podložce nemá uživatel výtisk jak znova upevnit a je nucen začít tisk od začátku. Dojití materiálu by se mělo projevit naopak jako absence změn. Při detekci dojití tiskárna nebude tisknout na prázdno, ale tisk bude přerušen a uživatel bude upozorněn na doplnění materiálu. Po doplnění je možné pokračovat v tisku od místa, kdy došlo k jeho přerušení. Chyb, které jsou pozorovatelné podle vzhledu, je více. Je to například špagetová příšera, stringing, kapky materiálu, nekonzistentní proudění materiálu a praskání vrstev. Tyto chyby by v případě dostatečně velkého datasetu mělo být možné rozpoznat za pomoci umělé inteligence.

[1] https://plugins.octoprint.org/plugins/octoeverywhere/

Detekce podle změn v čase

Úspěšnost této metody závisí vysoce na umístění kamery a tiskárny. Je nutné, aby kamera snímala výhradně tiskovou plochu a pozadí tiskárny bylo pokud možno jednotně vypadající. Je také nezbytné, aby tiskařská hlava pokaždé při snímání kamerou odjela ze záběru na vyhrazené a vždy to stejné místo. K tomu by bylo možné využít rozšíření Octolapse[2] fungující v aplikaci Octoprint. To po každé vrstvě zašle G-code do tiskárny, který odsune hlavu do domácí pozice v rohu tiskárny. Hlavu nechá stát několik sekund v domovské pozici a v této době pořídí snímek výtisku, potom tisk pokračuje další vrstvou. Výsledné časosběrné video pak vypadá, že se tiskárna vůbec nepohybuje a výtisk jakoby sám od sebe rostl. Důvodem využití tohoto odsunu je to, že pokud by byl dán pevný časový interval pořizování snímků, hlava tiskárny bude pokaždé na jiném místě či bude překážet v záběru na výtisk. I kdyby tak docházelo k odfiltrování pozadí, nebude pokaždé výtisk na snímku kompletní.

Správné fungování této metody také záleží na úrovni nasvícení scény výtisku. I při použití pluginu Octolapse v obraze zůstává velký počet pohyblivých částí tiskárny, které detekování změn ztěžují. Experimentoval jsem s několika filtry odstraňující pozadí z obrázků. Pokud ale uživatel tiskne z filamentu nevýrazné barvy, výsledek filtrování se výrazně liší snímek od snímku. To vytváří velké množství změn při odečtení dvou po sobě jdoucích snímků a algoritmus tak vydává poměrně mnoho falešně pozitivních detekcí. Nepodařilo se mi najít univerzální řešení, ve kterém by uživatel nemusel u každého tisknutí zdlouhavě přenastavovat úroveň prahování a dalších operací nad obrazem. Je ale časté, že barva použitého filamentu je dostatečně výrazná a odlišná od pozadí, takže je možné ji zdařile pravidelně odfiltrovávat pomocí barevného rozsahu a pozorovat změny pouze na výtisku. V e-shopech s materiálem do tiskáren je takřka polovina všech špulek s filamentem výrazné, křiklavé barvy. Je možné tak alespoň v některých případech této detekce využít. V pozadí by se ale nemělo vyskytovat cokoli pohyblivého podobné barvy.

Jako první je uživatel vyzván k označení tisknutého objektu na snímku. Z pozice, kam uživatel klikne, si systém převezme pozici ve snímku a barvu výtisku. Barva je pak převedena do HSV (Hue-Saturation-Value) barevného modelu. To je z důvodu, že v tomto barevném modelu mají podobné barvy blízkou hodnotu a je tak možné nastavit okolí barev, které budou brány jako součást výtisku. Okolí je zvětšeno z důvodu možného rozdílného nasvícení jednotlivých částí objektu nebo přítomnosti stínů. Na snímek je aplikován bilaterální filtr. Ten vyhladí plochy, ale zároveň ponechává hrany. To má za cíl vyhladit vrstvy výtisku, které při této detekci nejsou podstatné, ale mohly by zhoršit pozdější maskování. Ze snímku se vezmou všechny pixely patřící do zvoleného okolí barvy. Z těch je vytvořena maska, podle které se odstraní okolí výtisku. Takto upravený obrázek se porovná s předchozím takto upraveným. Menší změny se Gaussovským filtrem odstraní, větší změny budou diletací posíleny. Vypočítá se plocha takto upraveného rozdílu těchto dvou snímků. Pokud je vyšší, než je nastavená tolerance, a zároveň průměrná chyba na posledních několika snímcích je vyšší, než je tolerance, je hlášena chyba. Kontrola na průměr posledních snímků má za cíl nebrat jako chybu případný výpadek kamery. Počítá se s tím, že po odlepení výtisk nezůstane nehybný, ale bude se, důvodem pohybů samotné tiskárny, výrazně pohybovat po tiskařské podložce nebo z ní úplně sjede. Pokud je plocha rozdílu menší, než je hranice pro dojití filamentu, je oznámena tato chyba.

[2] https://plugins.octoprint.org/plugins/octolapse/

Model	Přesnost	Délka první klasifikace	Průměrná rychlost klasifikace
MobileNetV3	91 %	1204 ms	208 ms
EfficientNet-B1	91 %	5000 ms	341 ms
EfficientNet-B3	92 %	5000 ms	370 ms
EfficientNet-B5	94 %	7000 ms	1300 ms

Tabulka 4.1: Rychlost klasifikace v prohlížeči, přesnost je měřena na validačním subsetu. MobileNetV3 předčil v rychlosti i přesnosti verzi 2, proto je v tabulce vynechán.

Detekce pomocí neuronové sítě

Úlohu detekce staticky pozorovatelných chyb bude mít na starosti konvoluční neuronová síť provádějící klasifikaci obrazu. Se znalostmi popsanými v kapitole 3 jsem experimentoval s vlastními návrhy sítě. Z důvodu ale příliš malého datasetu se mi nedařilo překonat přesnost vyšší než 65 procent, což je pro vydání aplikace a používání skutečnými uživateli značně nedostačující. S malým datasetem se ale v praxi při rozpoznávání obrazu musí pracovat poměrně často a používaným řešením tohoto problému je využití přenosu učení.

Vytvořený model tak bude využívat přenos učení s případným jemným doladěním přeneseného modelu. K přenosu učení existuje celá řada architektur a modelů. V rámci Tensorflow je možné využít předem natrénované modely od společnosti Google. Důležitým faktorem při výběru byla pro mou práci rychlost predikce a samozřejmě také přesnost. K experimentování jsem si tak vybral architektury MobileNetV2, MobileNetV3 a EfficientNetV1, které byly vytvořeny se záměrem rychlého výsledku na slabších procesorech. Vybraný základnový model bude architektury EfficientNetV1 a bude se jednat o model EfficientNet-B3, který se při experimentech ukázal jako dostatečně rychlý a přesný. Výsledky testování rychlosti je možné vidět v tabulce 4.1. Rychlost klasifikace jsem měřil na svém počítači Acer Swift SF315-52 s procesorem Intel(R) Core(TM) i5-8250U CPU @ 1.60GHz 1.80GHz.

Při klasifikaci obrazu bývá často využíváno předem zhotovených velkých datových sad. Neexistuje ale vhodná datová sada, který by pokrývala problém detekování chyb při 3D tisku, a proto bude muset být speciálně pro tuto práci vytvořena. Pro základnu modelu EfficientNet bude ale využita datová sada imagenet. Ta obsahuje několik milionů snímků rozdělených do 1000 tříd jako stolní lampa, tygr, holičství a další. Tento dataset je využit k získání vah na nižších vrstvách základnového modelu. K natrénování vah hlavy modelu a doladění základny bude využita nově vytvořená sada snímků tiskových chyb.

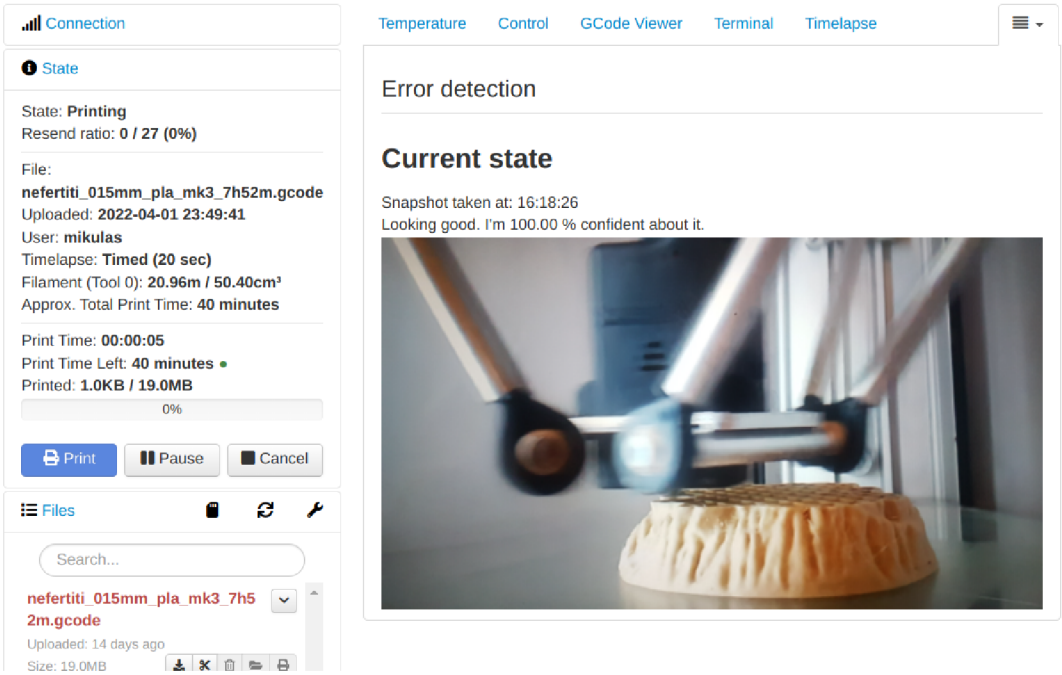
Síť bude detekovat chyby typu špagetové monstrum, stringing a kapek. Je nakročeno k rozpoznání nekonzistentního proudění materiálu a prasknutí vrstev, ale jejich aktuální úspěšnost detekce není vysoká a konzistentní. Nepodařilo se mi totiž získat dostatečný počet snímků těchto chyb pro natrénování modelu k dostatečné přesnosti. K zjištění přítomnosti chyb používá model klasifikaci obrazu. Model vrací pravděpodobnost přítomnosti chyb, z nichž se vybere ta s největší jistotou, která se zašle na výstup. Případně model vrátí informaci, že je tisknutý objekt v pořádku.

4.3 Uživatelské rozhraní

Uživatelské rozhraní má za cíl vytvořit pro uživatele přívětivou formu pro interakci s aplikací. Mělo by myslet i na uživatele s nižším technickým vzděláním. Proto jsem zavrhl

spouštění aplikace z příkazové řádky. Namísto toho bude vytvořeno grafické uživatelské rozhraní. Rozhraní je tvořeno s myšlenkou, že cílem aplikace je, aby u ní uživatel strávil co nejméně času a mohl se věnovat smysluplnějším aktivitám. Kdyby se uživatel místo kontroly tisku musel věnovat sledování aplikace, ztrácela by svůj smysl. Při zjištění chyby tak bude potřeba uživatele upozornit hlavně v případech, kdy se nenachází v blízkosti tiskárny. Je ale myšleno i na to, že se v blízkosti nachází, ale nevěnuje tisku pozornost, a díky upozornění si tak může všimnout chyby dříve. Zřetel není tak brán hlavně na vizuální styl jako na co nejsrozumitelnější výstup a konfiguraci před začátkem detekce.

V aplikaci Octoprint si může každý plugin vytvořit záložku na hlavním okně uživatelského rozhraní a v jeho nastavení, čehož využiji. Uživatel se tak nemusí seznamovat s úplně novým rozložením grafických prvků. V nastavení budou k zadání pouze nezbytné vstupy od uživatele jako email, poskytovatel SMTP služby a práh jistoty detekce. Výstupy aplikace budou tři. V aplikaci Octoprint bude v hlavním okně vytvořena záložka pro toto rozšíření. Bude zobrazovat výstup predikce spolu s posledním analyzovaným snímkem a časovým razítkem jeho pořízení. Uživatel může ověřit, že nastavení kamery a osvětlení je dostačující a v prostředí tiskárny se nenachází nic, co by klasifikaci obrazu mátlo. Přes tento výstup si tak uživatel bude moci v samotné webové aplikaci ověřit aktuální stav tisku či aktivnosti detekce. Zároveň ale může pomocí něj pozorovat tisk a jeho stav vzdáleně, pokud se nachází v dosahu signálu WiFi, na kterou je připojeno Raspberry Pi s tiskárnou. Tento výstup je možné vidět na obrázku 4.2.



Obrázek 4.2: Záložka vytvořeného rozšíření v hlavním rozhraní aplikace Octoprint

Dalším výstupem je krátká zvuková výstraha vycházející z připojeného počítače. Ta se spustí v případě nalezení chyby. Má za cíl uživatele upozornit v případě, že se nachází na téže WiFi síti jako tiskárna a má počítač, na kterém detekce probíhá u sebe, ale nevěnuje tisku plnou pozornost. Zvuková výstraha bude mít formu zvukového alarmu, později může být případně výstupem text-to-speech api, které vysloví detailnější informace o chybě.

Třetím výstupem je zaslání emailu s chybou. Tímto způsobem muže být uživatel upozorněn bez ohledu na to, kde se nachází. Stačí mu pouze zařízení připojené k internetu s nastaveným emailovým klientem, na který má upozornění dojít. Pro možnost zasílání emailu s výpisem o chybě je potřeba použít účet na některé službě umožňující SMTP připojení. Tuto službu poskytuje zdarma například Gmail od společnosti Google nebo Outlook od společnosti Microsoft. Heslo není možné zcela bezpečně na zařízení uložit, proto bude doporučeno vytvořit nový, postradatelný účet. Heslo k účtu je zašifrováno pouze pomocí algoritmu base64, aby nebylo uloženo jen jako prostý text. Uživateli bude zaslán email s detailem chyby, jejím popisem, časovým razítkem a pořízeným snímkem tisku, aby si mohl ověřit, že k chybě opravdu došlo a nejedná se o chybně pozitivní (false positive) výsledek. Při ověření chyby pak může pomocí již zmíněného pluginu OctoEverywhere tisk dálkově přerušit nebo pozastavit. OctoEverywhere nabízí další možnosti notifikace například formou SMS, přes aplikaci Discord nebo Slack. Bylo nabídnuto autorem modulu OctoEverywhere, že by mnou vytvořená aplikace mohla využívat možnosti notifikací jeho modulu, čehož v budoucnosti nejspíše využiji.

4.4 Datová sada

Protože součástí systému je konvoluční neuronová síť, je zapotřebí vytvořit datovou sadu, na které se síť naučí rozpoznávat konkrétní druhy chyb. Datová sada je sbírka anotovaných dat. Tyto sady bývají používány k trénování, validaci a testování neuronových sítí a dalších metod strojového učení. Při klasifikaci obrazu se nejčastěji používá již vytvořená datová sada. Podařilo se mi ale najít pouze jednu sadu[3], která se zabývala stejným problémem, jaký řeší tato práce. Anotace dat v ní byla dělená pouze na tisky v pořádku a s chybou. V rámci asi 600 snímku zabírala pouze malý počet tisků v jeho různých fázích a při jejím použití v mém modelu neuronové sítě hrozilo riziko přetrénování nechtěných vlastností. V takové formě nebyl pro mé účely dataset použitelný a bylo tedy nutné vytvořit zcela nový.

Na Google Images jsem nashromáždil snímky 3D výtisků, které jsem rozčlenil do složek podle druhu chyby, případně do složky obsahující bezchybně vytisknuté objekty. Hledal jsem také snímky nedokončených tisků, kdy je tisk v počáteční fázi nebo jeho výplň stále viditelná. Podařilo se mi nasbírat dostatek snímků k povedeným výtiskům a dále pak chybám typu špagety a stringování. Nejvíce se mi podařilo shromáždit chyby typu špagetová příšera, pro které mám 112 příkladů. Pro chyby stringování jsem získal 86 záznamů, a pro povedené výtisky 187 záznamů. Dále jsem vyčlenil záznamy pro chyby praskání objektu a nekonzistentní extruze, nicméně pro každou z těchto chyb se mi povedlo získat méně než 20 záznamů, což se projevilo, i po jejich umělém rozšíření, jako nedostatečné. Pro přesnější klasifikaci jsem vynechal záznamy obsahující více než jednu chybu najednou. Svou datovou sadu jsem nadále rozšířil o snímky ze zmiňovaného nevhodného datasetu. Snímky jsem ale ručně probral a vybral z nich menší, asi desetinou část, která byla pro mé účely použitelná.

Snímky v datové sadě pocházejí převážně z dobře nasvícených prostředí, což může zapříčinit nižší kvalitu rozpoznávání při horších světelných podmínkách. Snahou tento fakt vykompenzovat je použití náhodné změny kontrastu snímků při předzpracování v rámci modelu. Všechny snímky jsou v barevném model RGB a mají různé rozlišení. To bude před vstupem do modelu sníženo a sjednoceno. Sadu jsem předem nijak nedělil na subsety, na-

[3] https://www.kaggle.com/datasets/ssharkov/3d-printing-defects

místo toho využiji algoritmy knihovny Keras, která automaticky datovou sadu rozdělí na části pro trénování, validaci a testování náhodně v poměru podle zadaných parametrů.

Datovou sadu snímku také při trénování neuronové sítě uměle rozšířím pomocí k tomuto účelu používaných algoritmů. Protože u rozpoznání těchto chyb by nemělo záležet na tom, z jaké strany jsou výtisky sledovány, jestli jsou tisknuty obráceně nebo jaký model je tisknut, nastavuji při předzpracování také náhodné otočení podle vertikální a horizontální osy, zkosení stran a otočení podle středového bodu. Předzpracování je implementováno jako vrstva modelu, což znamená výrazně rychlejší trénování při použití akcelerace na grafické kartě. Pro případné pozdější využití datasetu pro detekci objektů jsem snímky anotoval pomocí nástroje Make Sense[4], kde jsem označil místa, na kterých došlo na snímcích k chybám. Při anotaci také použiji snímky, na kterých je přítomno více druhů chyb, což detekce objektů umožňuje.

V systému bude implementováno také dobrovolné zasílání snímků pro rozšíření datasetu chyb a zvýšení přesnosti modelu. Dále jsem nashromáždil časosběrná videa pro vývoj detekce chyb na základě změn v čase. Tyto videa jsem získal převážně z internetového portálu YouTube a tvorbou na tiskárnách Prusa přítomných na fakultě vysoké školy. Z těchto videí jsem také vytvořil snímky pro dataset chyb vytvořením snímku obrazovky v části videa, kdy se chyba projevila. Datovou sadu jsem zvěřejnil na službě Kaggle[5].

[4] https://www.makesense.ai/

[5] https://www.kaggle.com/datasets/mikulhe/3d-print-errors-appended

Kapitola 5

Implementace

Tato kapitola popisuje postup tvorby aplikace pro automatickou detekci nejčastějších chyb v průběhu 3D tisku. Výsledná aplikace je implementována jako zásuvný modul pro aplikaci OctoPrint. To umožňuje ovládání tiskárny přes rozhraní Octoprintu, získávání snímků tisku v jeho průběhu a tvorbu uživatelského rozhraní. Backend běžící na Raspberry Pi je napsán v Pythonu, frontend využívá Javascript a šablonovací systém Jinja2. Pro detekci chyb jsou využity knihovny OpenCV, OpenCV.js, Tensorflow a Tensorflow.js. Při vývoji neuronové sítě bylo použito cloudových grafických karet a TPU[1] za pomocí internetové služby Kaggle.

Octoprint od počátku umožňuje rozšíření zásuvnými moduly. Pro co nejsnazší zapojení pluginu do aplikace Octoprint jsem si vývojářskou funkcionalitou Octoprintu nechal vygenerovat základní kostru pluginu. V té je již implementováno propojení s hlavním rozhraním aplikace, objekty pro komunikaci s tiskárnou a kamerou a také připojení do repositáře pro všechna rozšíření. Propojení s hlavním rozhraním Octoprintu umožňuje vytvoření vlastních záložek v nastavení a hlavním menu aplikace. V nich si pak rozšíření může vytvořit vlastní grafické rozhraní pomocí vložení HTML elementů.

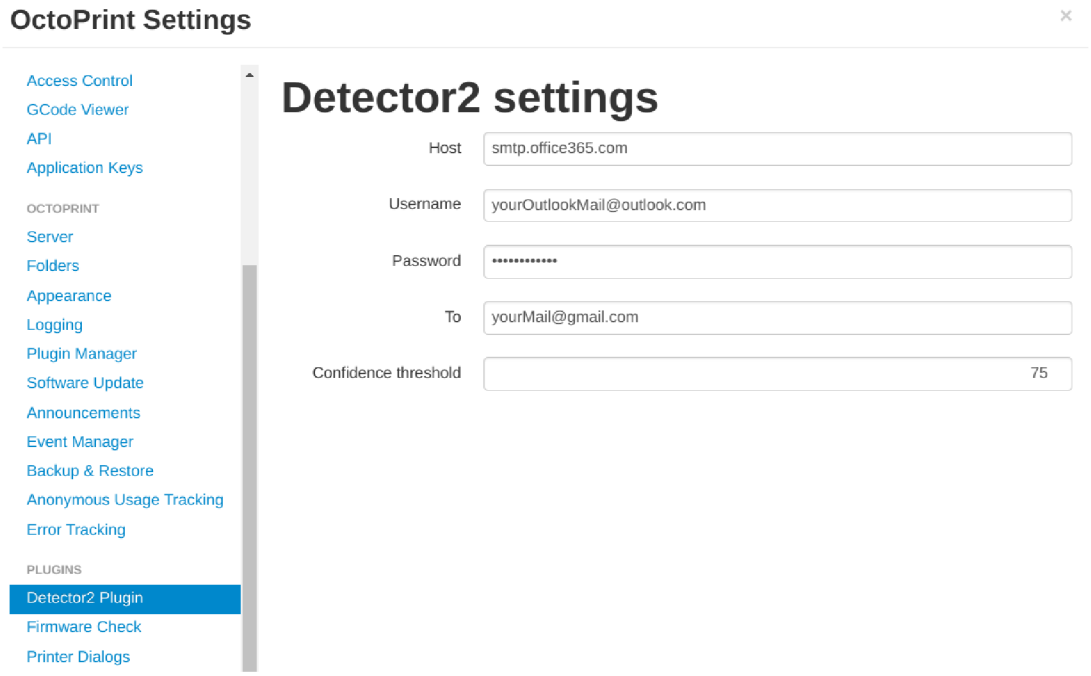
Vytvořený plugin je přímo propojen se správcem zásuvných rozšíření. Pro správnou funkci je před vygenerováním kostry nutné zadat údaje jako url GitHub repositáře, jméno pluginu, autora, verzi a potřebné knihovny pro správné fungování. Vytvoří také nezbytnou složkovou strukturu. Autor rozšíření tak svoji aplikaci píše do podsložky s názvem pluginu. V tomto případě se jedná o složku `octoprint_detector2/`. Část běžící na Raspberry je napsána v souboru `__init__.py`. Ve složce `static/` se nachází zdrojové kódy pro část běžící v prohlížeči jako CSS styly nebo Javascriptové skripty a model konvoluční sítě.

5.1 Propojení s tiskárnou a aplikací Octoprint

V souboru `__init__.py/`. je vytvořena třída `Detector2Plugin`, která dědí od základní třídy `octoprint.plugin`. Z ní přebírá funkce a zpětná volání k přístupu do nastavení tiskárny, frontendu a obsluhu událostí. Vlastní logika je vložena přepsáním funkcí třídy, které jsou napojené na určité systémové události. Při načtení pluginu při zapínání Octoprintu probíhá kontrola na přítomnost potřebných dat uživatele. Pokud je uživatel nenastavil, volá se metoda `get_settings_default`. V ní se nastaví hodnoty jako uživatelské jméno, heslo, email k zasílání notifikace, práh detekce a hostitelský server smtp služby. Tyto hodnoty slouží jako zástupná hodnota (placeholder) při zadání vlastních údajů uživatelem v záložce nastavení pluginu.

[1] https://cloud.google.com/tpu/docs

Metoda `on_settings_save` je volána při uložení zadaných údajů v záložce nastavení pluginu. Dostává objekt `data` se změněnými hodnotami. Nová data jsou zasílána na frontend, kde je vytvořena kopie všech údajů uživatele pro možnost zaslání emailu o chybě. Metoda `on_settings_load` je volána při snaze získat aktuální data uživatele. Metoda `get_template_configs` umožňuje konfiguraci uživatelského rozhraní pluginu. V tomto případě je vytvořena záložka v nastavení (obrázek 5.1) a záložka v hlavním okně uživatelského rozhraní aplikace Octoprint (obrázek 4.2). Je pro ně nastaveno základní připojení k aplikaci. Metoda `get_assets` umožňuje přístup k vypsaným zdrojovým souborům z frontendu. Soubory, které jsou na frontendu využívány, jsou `detector2.js` obsahující logiku aplikace na frontendu a tensorflow model neuronové sítě ve formátu JSON. Dále jsou frontendem využívány soubory napsané v šabloně Jinja2, které jsou uložené ve složce `templates/`.



Obrázek 5.1: Záložka nastavení detekce

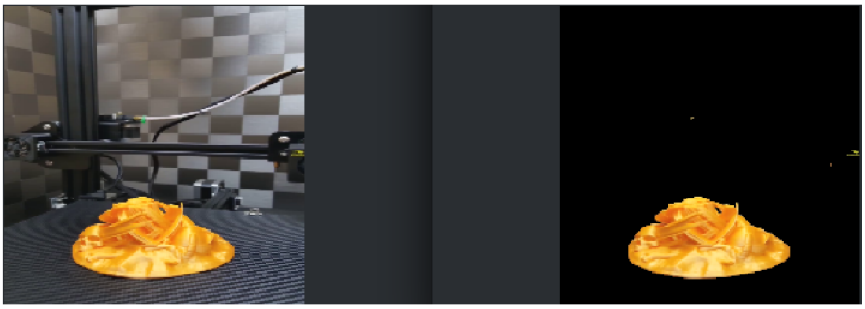
Aplikace Octoprint vysílá oznámení pluginům o řadě událostí o komunikaci s tiskárnou, zpracování G-code příkazů nebo o pořizování snímků z kamery. K těmto událostem je možné připojit vlastní metody. Při načítání pluginu je vytvořena globální proměnná `__plugin_hooks__` obsahující strukturu slovník. Klíčem v něm je vždy název události, hodnotou je odkaz na volání příslušné metody pluginu. Slovník v mé v implementaci detekce obsahuje odkazy na dvě metody. Metoda `get_update_information` byla vytvořena při generování kostry a slouží k automatické aktualizaci pluginu pokaždé, co je na GitHub repositáři vydána nová verze. Druhá metoda se jménem `capture_post_handler` je napojena na událost `octoprint.timelapse.capture.post`, což značí vytvoření snímku připojenou kamerou. Jejím vstupem je cesta k vytvořenému snímku a boolean znak úspěchu tvorby snímku. Při úspěšném pořízení je snímek zaslán na frontend spolu s časovým razítkem pořízení. K zaslání je použita metoda `send_plugin_message`, která pošle vložená data pomocí protokolu WebSocket. Není možné ale přes metodu zaslat objekt modulu PIL, používaným

v Pythonu k práci s obrázky. Pořízený snímek je tak konvertován za pomocí base64 po bajtech do ASCII podoby a v této podobě zaslán.

5.2 Detekce sledováním změn v obraze

Detekce změn v čase byla vytvořena za použití OpenCV knihovny v jazyce Python. Pro běh v prohlížeči ji ale bylo nutné převést do jazyka Javascript. OpenCV v jazyce Javascript ale není tak rozšířené, a ne všechny funkce dostupné v jazyce Python jsou dostupné v Javascriptu. Použití OpenCV v Javascriptu bylo také značně výpočetně náročnější a způsobovalo problémy s klientem. Detekce změn tak nebyla do výsledného pluginu implementována. Již vytvořený fungující algoritmus v Pythonu bude ale v pozdější verzi přidán k možnému spuštění v backendu na výkonnějších modelech Raspberry Pi.

Obraz je převeden do HSV barevného modelu pro segmentaci na základě barvy. Protože systém má fungovat univerzálně, nezávisle na druhu tiskárny a pozadí za tiskárnou, nechám uživatele na prvním záznamu zvolit, který objekt v obraze se má detekovat. Z toho získám informaci o barvě a pozici objektu. V HSV obrázku ponechám pouze pixely, jejichž hodnota spadá do okolí vybrané barvy. Pomocí ponechaných pixelů odmaskuji pozadí tiskárny a ostatní zbytečné vlastnosti obrazu. Výsledek tohoto maskování je možné spatřit na obrázku 5.2. Toto ale funguje výrazně hůře na nevýrazných barvách materiálu. Pomocí funkce `absdiff` získám změny na předchozím, stejně upraveném obrázku. Z HSV pixelů snímku ponechám jenom hodnoty Value, z čehož vznikne šedo-tónový obrázek. Ostatní hodnoty již nejsou potřeba. Výhodou šedo-tónového modelu je ponechání pouze užitečných informací jako hrany, rohy nebo kontrast prvků. Změny pomocí Gaussova filtru rozostřím a prahováním podle hodnoty 20 převedu pixely do binární podoby. Na nich provedu diletaci a pomocí ploch výsledků funkce `findContours` spočítám množství změn. Jako práh změn pro zaznamenání pohybu výtisku byla zvolena hodnota 1000 pro obrázek o rozměrech 500 na 400 pixelů. Pro práh dojití filamentu byla zvolena hodnota 10 u obrázku o stejných rozměrech. Množství změn porovnám s oběma prahy a v případě splnění jedné z podmínek oznámím daný druh detekované chyby.



Obrázek 5.2: Maskování podle barvy výtisku

5.3 Detekce chyby neuronovou sítí

Pro spuštění detekce v prohlížeči bylo nutné ji implementovat v javascriptu pomocí knihovny Tensorflow.js. Trénování modelu ale probíhalo v Pythonu. Ten díky možnosti využít NumPy[2] pole umožňuje výrazně rychlejší trénování. Jistou nevýhodou neuronových sítí je jejich velká časová náročnost pro natrénování. Lze však využít akceleraci na grafických kartách, ale musejí být značky nVidia a podporovat architekturu CUDA. K trénování tak byla použita služba Kaggle umožňující spouštění a akceleraci trénovacího skriptu na cloudových grafických kartách a jednotkách TPU.

Datová sada byla při každém spuštění trénování náhodně rozdělena na trénovací a validační subset v poměru 8:2 což u mého datasetu činilo 307 snímků pro trénování a 76 k validaci. Nebyl vytvořen testovací subset, protože vytvořený dataset obsahuje poměrně málo příkladů, a odebrání dalších z trénovacího subsetu už výrazně snižovalo jeho přesnost. Ke tvorbě datasetové objektu byla využita funkce knihovny Keras `image_dataset_from_directory`. Ta umožňuje automatické rozdělení datasetu na subsety a vytvoří strukturu s kategoriemi odvozenými z názvů složek se snímky. Kategorie jsou uložené jako integery pro pozdější snadnější nasazení nových tříd. Várky jsou vytvořeny po 16 snímcích. Pro lepší výkon jsou datasety přednačteny, počet přednačtených snímků je dynamicky měněn pomocí `td.data.AUTOTUNE`. Vstupní velikost snímků byla nastavena jako 300 pixelů na výšku a 300 na šířku, což byla používaná velikost snímků při trénování modelu využitého pro přenos učení.

Při rozpoznávání pomocí strojového učení bylo vytvořeno několik typů neuronových sítí. První síť se učila takzvaně od nuly, ale z důvodu malého datasetu se její přesnost nedokázala přehoupnout přes 65 %. Později tak bylo využit i přenos učení z předem natrénovaného modelu EfficientNet od společnosti Google.

Rozšíření dat je zajištěno pomocí sekvenční skupiny Keras vrstev sloužící k předzpracování obrazu. Nejdříve je náhodně změněn kontrast, poté je provedeno otočení snímku podle horizontální a vertikální osy, dále je nastavena rotace a posun obrázku. Rotace je nastavena na faktor 0.2, což značí rotaci v rozsahu od $-0.2 * 2\Pi$ do $0.2 * 2\Pi$. Intenzita těchto změn je nastavena u každé várky náhodně. Použitým základnovým modelem je EfficientNet-B3, který na vstup bere hodnoty v rozsahu -1, 1. V rámci předzpracování je tak hodnota každého pixelu vydělena 127,5 a posunuta o -1, čímž snímky dostaneme do požadovaného rozsahu. K tomuto předzpracování je použita funkce pro předzpracování vstupu patřící právě k modelu EfficientNet. Jak bylo zmíněno, použitý základnový model je architektury EfficientNet. Před vstupem do základnového modelu je vstup upraven rozšiřující vrstvou a hodnoty snímků převedeny do požadovaného rozsahu. Váhy uzlů jsou převzaty z natrénování na datasetu Imagenet. K modelu není připojena jeho hlava (vrchní klasifikační vrstva), protože k tomu bude použita vlastní sekvence tří vrstev. První z nich je `GlobalAveragePooling2D`, která převede 2D obrázek do pole o jedné dimenzi. Následuje `Dropout` vrstva, u které je míra blokovaných neuronů nastavena na 20 %. Vrstva má za cíl snížit přetrénování. Poslední vrstvou je `Dense` (plně propojená vrstva), která vrátí předpověď jedné ze tří tříd. Tato předpověď ale musí být ještě převedena pomocí funkce `Softmax` na jistotu předpovědi jednotlivých kategorií.

Při kompilaci modelu je zvolen jako optimalizátor algoritmus Adam. Chybová funkce je použita `SparseCategoricalCrossentropy` z důvodu nebinárního počtu tříd a uložení kategorií ve formě integerů. Model je natrénován na 20 epochách. Jeho přesnost se po natrénování pohybuje na validačním datasetu kolem 92 % a chyba kolem 0,3.

[2] https://numpy.org

Natrénovaný model je poté uložen. Je ale nutné ho posléze převést do JSON souboru pro použití v Javascriptu. K tomu je využit nástroj `Tensorflowjs_converter`. Model je konvertován do grafového modelu. To je nezbytné pro správné fungování při využití vlastních Keras vrstev. Vytvořený grafový model ale nemá už možnost dalšího dotrénování, mám tedy uložen i původní model.

Při spuštění uživatelského prostředí Octoprintu je model načten. K tomu je využita funkce knihovny Tensorflowjs, která získá potřebná data z html elementu obrázku. Detekce je spouštěna po úspěšném načtení obrázku. Při vstupu obrázku je nutné data přepočítat na formu vhodnou na vstup modelu. Obrazová data jsou tak zmenšena na požadovaných 300 na 300 pixelů. Vrací pole hodnot, jehož obsahem jsou hodnoty předpovědí ke každé třídě. Pomocí funkce funkce `Softmax` je výsledek předpovědi převeden na její jistotu. Je použit index s nejvyšší hodnotou. Vynásobením hodnoty 100 získám jistotu v procentech, kterou vypíšu do uživatelského rozhraní.

Je zapotřebí při každé predikci vypočítat, jak moc si je model výsledkem jist. V případě, že jistota bude menší, než uživatel nastavil, bude výsledek ignorován a chyba nijak oznámena nebude. To má zamezit oznamování falešně pozitivních předpovědí.

Model: OctoDetector2

Layer (type)	Output Shape	Param #
input_2 (InputLayer)	[(None, 300, 300, 3)]	0
sequential (Sequential)	(None, 300, 300, 3)	0
efficientnetb3 (Functional)	(None, 10, 10, 1536)	10783535
global_average_pooling2d	(None, 1536)	0
dropout (Dropout)	(None, 1536)	0
dense (Dense)	(None, 3)	4611
Total params: 10,788,146		
Trainable params: 4,611		
Non-trainable params: 10,783,535		

Tabulka 5.1: Shrnutí vrstev modelu s použitým základnovým modelem EfficientNet-B3

5.4 Uživatelské rozhraní

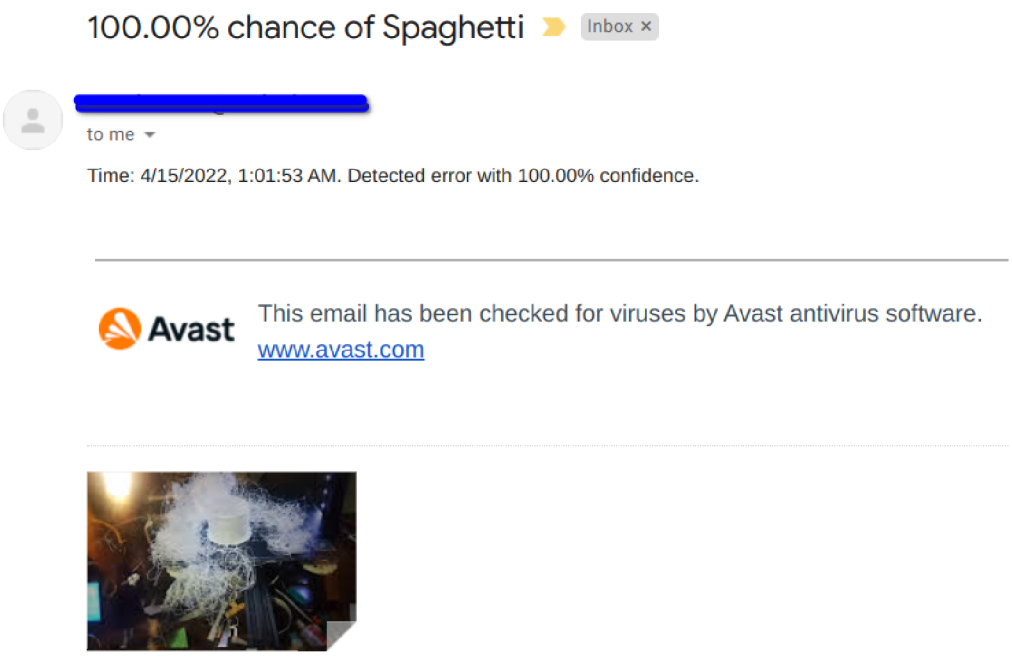
Vytvořil jsem jednoduché uživatelské rozhraní využívající aplikaci Octoprint. Rozhraní se skládá ze záložky nastavení a záložky v hlavním uživatelském rozhraní aplikace. Soubory s uživatelským rozhraním jsou načteny při každém novém obnovení záložky s aplikací Octoprint v prohlížeči. Do zdrojového souboru pro záložku nastavení jsem tak vložil url adresy skriptů, které jsou potřebné pro správné fungování detekce, k importování. Importovanými skripty jsou TensorFlow.js a SmtpJs. SmtpJs umožňuje zasílání emailů pomocí Javascriptu přímo v prohlížeči od klienta. Tensorflow.js je nezbytný pro spouštění převedených modelů strojového učení do JSON formátu. Pod importovanými skripty se nachází HTML formulář pro vyplnění požadovaných uživatelských hodnot (obrázek 5.1). Jednotlivé vstupy jsou provázány s Python proměnnými přes `data-bind` atribut. Změněné hodnoty jsou po odeslání zaslány na backend a jsou zpracovány výše zmíněnou funkcí `on_settings_save`. V záložce na hlavním uživatelském rozhraní jsou vytvořeny dva `<div>` elementy. Mezi ty se budou vázat výstupy detekce pomocí funkcí pro přístup k objektovému modelu HTML dokumentu.

Konkrétně se jedná o funkce `getElementById` na jejímž výstupu se zavolá `appendChild`, čímž se příchozí snímek vloží viditelně pro uživatele.

Upozornění na chybu

Jak bylo zmíněno, upozornění probíhá třemi cestami. Pro možnost zasílání emailu s výpisem o chybě je potřeba použít účet na některé službě umožňující SMTP připojení. Heslo není možné opravdu bezpečně uložit, proto je doporučené vytvořit nový, postradatelný účet. Heslo pro něj je zašifrováno pouze pomocí base64, aby nebylo uloženo jen jako prostý text. Je uložen příznak poslání, který se při detekci kontroluje, aby email s chybou nebyl zaslán více než jednou. Mail je posílán z frontendu pomocí služby SmtpJs. Ta vyžaduje objekt s uživatelským jménem, adresou hostitelského SMTP serveru, heslem a údaji k zasílanému emailu jako email odesilatele a adresáta, předmět a tělo. Ne všichni emailoví klienti umožňují zasílat obrázky v těle emailu, například služba Roundcube to umožňuje, používanější Gmail však ne. Proto je snímek s chybou zaslán jako příloha mailu, což je zobrazeno na obrázku 5.3. Po odeslání je na jeho úspěch upozorněno javascriptovou funkcí `alert` a vypsána chybová hláška nebo úspěch odeslání.

Zvuk alarmu je uložen jako mp3 soubor. Je načten a při detekci spuštěn. Je ale nutné, aby uživatel nějak se stránkou interagoval, jinak není možné spustit zvukový výstup. Zvukový signál je spuštěn nanejvýš třikrát. Při nalezení chyby je také upraven text elementů v rozhraní s příslušnou chybovou hláškou.



Obrázek 5.3: Upozornění na chybu emailem

5.5 Použité technologie

Projekt je implementován v jazycích Python a Javascript. Python je vysokoúrovňový programovací jazyk podporující více programovacích paradigmat. Za pomocí různých knihoven jako například knihovny NumPy se dokáže rychlostí přiblížit i programovacím jazykům nižší úrovně. Je také nejčastěji používaným jazykem pro práci se strojovým učením. Zvolil jsem jej, protože je podporován nejpoužívanějšími frameworky pro strojové učení Pytorch a Tensorflow a existuje pro něj řada dostupných nástrojů pro práci právě na strojovém učení.

Jednou ze služeb, které jsem nejčastěji pro vývoj využíval, je platforma Kaggle. Umožňuje spouštění Python notebooků v prohlížeči, pro které poskytuje vymezený čas na grafických kartách a Tensor Processing Units, vyvinutých speciálně pro strojové učení, které jsou obzvlášť užitečné pro konvoluční neuronové sítě. Zároveň obsahuje návody k jednotlivým krokům strojového učení a fórum týkající se strojového učení. Služba je vlastněná společností Google a dominantními frameworky používanými na službě jsou Tensorflow a Keras.

OpenCV

OpenCV (Open Source Computer Vision Library) je knihovna zaměřená na počítačové vidění a zpracování obrazu. Je využitelná v jazycích C, C++, Octave a Python a částečně v Javascriptu. Umožňuje GPU akceleraci pro zpracování obrazu v reálném čase. Pro zpracování obrazu má implementovánu celou řadu algoritmů umožňující morfologické transformace, prahování, detekci hran, změnu barevných modelů a další nástroje k vyššímu porozumění zpracovávaného obrazu. Umí číst jak z obrázku, tak zpracovávat video, o kterém zvládne získat metadata a číst jej snímek po snímku. V Pythonu využívá knihovny NumPy, takže práce s daty je několikanásobně rychlejší než s obyčejnými Python poli.

Obsahuje také několik algoritmů strojového učení, ale rozsahem a použitelností se nevyrovná knihovnám jako PyTorch nebo Tensorflow, které se na strojové učení zaměřují primárně. Mezi algoritmy, které se v knihovně nachází patří například algoritmus Decisiontree, k-nearest algoritmus nebo algoritmus náhodného lesa.

Tensorflow a Keras

Tensorflow je otevřená knihovna vyvíjená společností Google k práci se strojovým a hlubokým učením. Umožňuje trénování hlubokých neuronových sítí, pro jehož jednotlivé kroky poskytuje řadu funkcí a algoritmů. Obsahuje předpřipravené aktivační funkce jako relu, elu, softmax nebo sigmoid. Umožňuje nízko-úrovňově operace na CPU, grafických procesorech a TPU. TPU neboli Tensor Processing Unit je speciální výpočetní jednotka vyvinuta společností Google, jejíž primárním účelem je strojové učení s knihovnou Tensorflow, pro což je také optimalizována. Oproti GPU je rychlejší hlavně při práci s konvoluční neuronovou sítí. Knihovna umožňuje také tvorbu modelů pro aplikaci v internetovém prohlížeči nebo v chytrých telefonech a zařízeních internetu věcí. Obsahuje také předem natrénované modely pro klasifikaci obrazu, textu a zvuku, detekci objektů, číslic a znaků a jiné. Tyto modely lze využít ve formě „jak jsou“ nebo je možné je přetrénovat.

Keras je otevřená knihovna poskytující vysoko-úrovňové rozhraní ke knihovně Tensorflow. Dříve podporovala více backend knihoven pro strojové učení jako PyTorch od společnosti Facebook nebo Cognitive Toolkit od Microsoftu. Zjednodušuje syntax pro práci s funkcemi z knihovny Tensorflow. Obsahuje nejčastěji používané nástroje pro tvorbu vrstev neuronových sítí, aktivační funkce, optimizátory. Pro pracování s obrazem obsahuje také

nástroje pro tvorbu konvoluční neuronové sítě a funkce pro její optimalizaci jako maximum pooling, filtrování obrazu, parametry posuvného okna. Umožňuje také rozšíření datasetu prostřednictvím rozšiřovací vrstvy. Toho docílí například změnou kontrastu nebo jasu, posunem, otočením kolem os nebo natočením snímku. Umožňuje také náhodně rozdělit dataset na trénovací, validační a testovací. Při trénování také umí dataset rozdělit na menší várky a nastavit, kolikrát má být každá várka viděna. Zvládne také vyhodnotit kvalitu natrénovaného modelu.

Octoprint

Je populární open-source nástroj pro monitorování 3D tisku přes webové rozhraní a vzdáleného ovládání tiskárny. Bývá spuštěn na mikropočítači Raspberry Pi, který je připojen k tiskárně a internetu. Funguje jako server a vysílá po lokální síti aktuální stav tisku spolu s posledním pořízeným snímkem. Umožňuje vzdáleně zobrazovat přenos obrazu z kamery na tiskárně, zobrazovat teplotu podložky a tavící hlavy a ovládat průběh tisku. Kromě svých vlastních funkcí je možné rozšířit jeho použitelnost o řadu zásuvných modulů. Octoprint s pluginy počítá a usnadňuje jejich vývoj vygenerováním kostry nezbytné pro komunikaci s hlavním rozhraním. Umožňuje také zasílání modelů do tiskárny přes internet, které by jinak musel uživatel ručně nahrávat přes USB disk.

Kapitola 6

Vyhodnocení

Testování je důležitou součástí vývoje, proto bylo provedeno několik druhů testování. Aplikace byla vyvíjena ve vývojářském prostředí Octoprintu. Neběžela tak na žádném mikropočítači, ale byla spouštěna v Linuxu pomocí příkazové řádky. Vytvořená aplikace byla proto testována v domácím prostředí i nahráním na Raspberry Pi Zero. V rámci testování uživateli bylo prováděno akceptační a integrační testování. Model klasifikace byl vyhodnocen standardními metrikami poskytovanými knihovnou Tensorflow. S testováním na skutečných tiskárnách jsem se musel spolehnout na ostatní uživatele. V domácích podmínkách mi však vystačila alternativa použití chytrého telefonu jako kamery a nepřerušených záznamů tisku dostupných na portálu Youtube. Příklad spouštění klasifikace při vývoji lze spatřit na obrázku 6.1.

Obrázek 6.1: Testování v domácích podmínkách bez tiskárny, použité video je dostupné na portálu Youtube[2]

[2] https://www.youtube.com/watch?v=MV8pONAHTwA

6.1 Testování uživateli

Aplikace byla zveřejněna v repositáři pluginů aplikace Octoprint[3]. Testování se tak mohl zúčastnit kdokoli, kdo aplikaci Octoprint na své tiskárně používá. O vydání pluginu byli informování všichni uživatelé pomocí automatické notifikace v rozhraní aplikace. Pro získání zpětné vazby bylo vytvořena záložka `Discussion` u GitHub repositáře[4] vytvořeného rozšíření. Uživatelé mohli vytvářet `Issues` a spojit se se mnou přes email uvedený u pluginu. Zpětná vazba probíhala na dobrovolné bázi. Během prvního týdne od vydání si aplikaci stáhlo 454 uživatelů. Na konci druhého týdne od vydání byla aplikace aktivně používána na 620 tiskárnách. Přestože 3D tisk stále nabírá na popularitě, ještě není tak běžně rozšířený mezi lidmi a znám jen jednu osobu, která 3D tiskárnu vlastní. Osobně jsem tak aplikaci mohl řešit pouze se správcem fakultních 3D tiskáren na Fakultě informačních studií VUT. Získávání zpětné vazby tak probíhalo hlavně přes internet.

Zpětná vazba od uživatelů

Zpětnou vazbu prostřednictvím fóra poskytlo dohromady 14 uživatelů. Zpětná vazba byla poskytnuta i ke zdrojovému kódu při vytvoření `Pull Request` pluginu k repositáři aplikace Octoprint. V ní bylo upozorněno na zbytečné opakované volání pro získání snímku tisku a nepoužití šifrování při uložení hesla k emailu. Tyto nedostatky tak byly ještě před vydáním opraveny. Na další chyby bylo upozorněno přes `Issues` na GitHubu a v rámci vytvořené diskuze. Při užívání aplikace nejvíce chybělo uživatelům více cest pro upozornění při chybě. K tomuto tématu byla nejvíce navrhována integrace s aplikací Telegram. Dalšími požadovanými způsoby notifikace byly za využití aplikace Discord, Pushover a Octopod.

Uživatelé by také chtěli možnost otestování správného připojení se SMTP serverem. Byla vytýkána chyba při instalaci pluginu přes správce rozšíření v aplikaci, která nastávala některým uživatelům z důvodu nenainstalované potřebné knihovny PIL pro práci s obrazem v Pythonu. Bylo také navrhnuto, aby mohla detekce běžet na samotném Raspberry v případě, že uživatel disponuje výkonnějším modelem tohoto mikropočítače. Bylo doporučeno vytvořit možnost logování událostí v záložce pluginu pro snadnější ověření uživatelem o správném chodu detekce. Také bylo navrhnuto cloudové řešení detekce, podobně jako pracuje rozšíření Spaghetti Detective. Využití tohoto řešení bylo odůvodněno údajnou vyšší přesností detekce mého rozšíření. To ale, kvůli absenci vlastní 3D tiskárny a nulové zkušenosti se zmíněnou aplikací, nemám možnost sám posoudit. Provozování v cloudu by navíc muselo být placeno. V prvním týdnu od vydání jsem obdržel 3 nová časosběrná videa tisku a dva snímky chyb.

Vyhodnocení modelu neuronové sítě

Aby se uživatelé mohli na detekci spolehnout i v průběhu několika hodinových tisknutí, je důležitá vysoká přesnost modelu. Protože aplikace se snaží vyhnout opakovanému upozorňování na stejnou chybu, posílá email s chybou pouze jednou při jejím prvním nálezu. Proto je velmi důležité, aby detektor neoznamoval falešně pozitivní výsledek. Pokud aplikace zašle email s chybou, která není přítomná, nemá už jak upozornit, pokud k chybě opravdu dojde později v tisku. K vyhodnocení modelu byla použita jako metrika řídká kategorická přesnost (Sparse Categorical Accuracy). Byla využita z důvodu používání číselných hod-

[3] https://plugins.octoprint.org/plugins/detector2/
[4] https://github.com/mikulash/Octoprint-detector2/discussions

not pro označení kategorií. Tato metrika udává, jak často se předpověď modelu shoduje se skutečnou kategorií snímku.

Bylo vytvořeno více modelů s odlišnými základovými modely architektury EfficientNet. Ty, s kterými jsem nakonec nejvíce pracoval, byly na základě verzí B3 a B5. Na modelu se základem o verzi B5 bylo provedeno trénování na 25 epochách, po nichž byla přesnost modelu 94 %. Pro verzi se základem B3 byla na trénovacím datasetu po trénování na 20 epochách přesnost zhruba 94 % a chyba 0,2197. Na validačním datasetu byla přesnost 92 %, ale chyba vyšší: 0,3003. Při trénování na dalších epochách se přesnost modelu začala snižovat. Verze používající B5 se ale ukázala jako v praxi příliš pomalá, a tak byla využita verze B3. Graf vytvořený z historie trénování modelu je vyobrazen na obrázku 6.2. Detailnější průběh testování je k nahlédnutí na platformě Kaggle[5].

Obrázek 6.2: Průběh trénovaní modelu se základnovým modelem EfficientNet-B3

6.2 Možnosti rozšíření

Při testování bylo navrhnuto několik vylepšení, které by stálo za to v delším časovém úseku k aplikaci přidat. Po dodatečném rozšíření datasetu by mělo být přidáno větší množství detekovatelných chyb. Tím, že je k přenosu učení použit úsporný model architektury EfficientNet je možné jej znova využít i na slabších zařízení. Po návrhu, získaném při testování, bude přidána možnost spouštění detekce i na samotném mikropočítači. Uživatelé, kteří mají mikropočítač dostatečně výkonný a nechtějí používat detekci v prohlížeči, se budou moci rozhodnout, na které části má klasifikace probíhat. Vytvořený model by bylo také možné převést na mobilní zařízení. Pokud uživatel má farmu tiskáren a jeho počítač nedokáže obsluhovat detekci snímků přicházející ze všech tiskáren, bylo by možné vytvořit jednoduchou mobilní aplikaci, která by snímala tisk kamerou telefonu a zároveň prováděla na stejném

[5] https://www.kaggle.com/code/mikulhe/latesteffnetb3

zařízení analýzu obrazu. Uživatel by tak mohl ke každé tiskárně, kterou nestíhá detekcí obsluhovat, přistavit starší chytrý mobilní telefon a používat k detekci něj. Při umístění na obchod s aplikacemi navíc zůstává jednoduchost instalace jako je při využití správce rozšíření Octoprintu, ale odpadá nutnost vlastnictví mikropočítače. Bylo by vhodné také implementovat automatické zasílání snímků od uživatelů, které by bylo možné využít k vylepšení modelu detekce.

Kapitola 7

Závěr

Cílem této práce bylo vytvořit automatizovaný systém pro detekci nejčastějších chyb, které mohou nastat při 3D tisku. Tento cíl byl splněn. Systém je napojen jako zásuvný modul do aplikace Octoprint. Má sloužit převážně amatérským tiskařům a usnadnit jim tisk časově náročnějších projektů, o které by se museli jinak v průběhu pravidelně starat. Chce nabídnout snadnější, levnější a lokálně běžící alternativu k některým zavedeným službám, které fungují v rámci měsíčního předplatného.

Bakalářská práce představuje principy a nejrozšířenější technologie 3D tisku. Popisuje nejčastější chyby, které mohou v průběhu tisku nastat, jejich příčiny a následky. Bylo zkoumáno využití počítačového vidění a strojového učení. Bylo vyzkoušeno několik architektur strojového učení a porovnány jejich výsledky.

Hlavním výstupem je funkční aplikace, pracující jako rozšíření do aplikace Octoprint. Je veřejně dostupná od poloviny dubna 2022 a na začátku května 2022 byla používána na více než 600[1] tiskárnách. Aplikace je napsána v jazycích Python a Javascript a šablonovém systému Jinja2. Využívá frameworky Tensorflow, TensorflowJS a Keras. Model konvoluční neuronové sítě, který aplikace využívá k rozpoznání chyb, dosáhl přesnosti 92 % při predikci na neviděných snímcích chyb.

Uživatelé mohou sledovat postup i stav tisku v aplikaci Octoprint v záložce pluginu nebo mohou být vzdáleně upozorněni přes email nebo zvukovou výstrahu. Aplikace dokáže rozpoznat tři nejčastější druhy chyb: takzvané špagety, stringing a s ní provázané kapky materiálu. Byl vytvořen jednoduchý systém, kde uživatelé mohou nahrát záznamy tisku vytvořené na svých tiskárnách a zaslat mi je. Tyto záznamy pomohou v rozšíření a testování nových funkcionalit.

Po vydání aplikace mi byla nabídnuta spolupráce na propojení s pluginem OctoEverywhere, který by umožňoval výrazně větší množství možností upozornění spolu s následným ovládáním tiskárny za účelem přerušení nebo zastavení tisku. Byly navrhnuty možnosti rozšíření a možné kroky pro zlepšení současných vlastností vytvořené aplikace. Ve vývoji aplikace bych chtěl nadále pokračovat a implementovat navrhovaná rozšíření.

[1] https://plugins.octoprint.org/plugins/detector2/

Literatura

[1] Aranda, S. *3D Printing Failures*. 2. vyd. Independently published, 2021. ISBN 979-8784041258.

[2] Bach, M. *Jak vyřešit nejčastější problémy při 3D tisku* [online]. 2018 [cit. 2022-23-04]. Dostupné z: https://josefprusa.cz/jak-vyresit-nejcastejsi-problemy-pri-3d-tisku/.

[3] Baumann, F. a Roller, D. Vision based error detection for 3D printing processes. *MATEC Web of Conferences*. 1. vyd. Květen 2016, sv. 59, s. 06003. DOI: 10.1051/matecconf/20165906003.

[4] Brownlee, J. *Deep Learning for Computer Vision: Image Classification, Object Detection, and Face Recognition in Python*. 1. vyd. Machine Learning Mastery, 2019. Dostupné z: https://books.google.cz/books?id=DOamDwAAQBAJ.

[5] DeVries, T. a Taylor, G. W. *Dataset Augmentation in Feature Space*. arXiv, 2017. DOI: 10.48550/ARXIV.1702.05538. Dostupné z: https://arxiv.org/abs/1702.05538.

[6] Gibson, I., Rosen, D. a Stucker, B. *Additive Manufacturing Technologies*. 2. vyd. Springer, 2015. ISBN 978-1-4939-2112-6.

[7] Horvath, J. a Cameron, R. *3D Printed Science Projects Volume 2*. 1. vyd. Apress, 2017. ISBN 978-1484226940.

[8] Horvath, J. a Cameron, R. *Mastering 3D Printing in the Classroom, Library, and Lab*. 1. vyd. Apress, 2018. ISBN 978-1484235003.

[9] Jordan, J. M. *3D Printing*. 1. vyd. The MIT Press, 2019. ISBN 978-0262536684.

[10] Khandpur, M. S., Galati, M., Minetola, P., Marchiandi, G., Fontana, L. et al. Development of a low-cost monitoring system for open 3d printing. *IOP Conference Series: Materials Science and Engineering*. 1. vyd. IOP Publishing. jun 2021, sv. 1136, č. 1, s. 012044. DOI: 10.1088/1757-899x/1136/1/012044. Dostupné z: https://doi.org/10.1088/1757-899x/1136/1/012044.

[11] Kinsley, H. a Kukieła, D. *Neural Networks from Scratch in Python: Building Neural Networks in Raw Python*. 1. vyd. Harrison Kinsley, 2020. Dostupné z: https://books.google.cz/books?id=Ll1CzgEACAAJ.

[12] Kneusel, R. *Math for Deep Learning: What You Need to Know to Understand Neural Networks*. 1. vyd. No Starch Press, 2021. ISBN 9781718501904. Dostupné z: https://books.google.cz/books?id=05yAzgEACAAJ.

[13] Long, L. *Beginning Deep Learning with TensorFlow*. 1. vyd. APress, 2022. ISBN 9781484279144.

[14] Menditto, A., Patriarca, M. a Magnusson, B. Understanding the meaning of accuracy, trueness and precision. *Accreditation and Quality Assurance*. 1. vyd. Říjen 2007, sv. 12, s. 45–47. DOI: 10.1007/s00769-006-0191-z.

[15] Michelucci, U. *Advanced Applied Deep Learning*. 1. vyd. APress, 2019. ISBN 978-1-4842-4976-5.

[16] Minetto, R., Volpato, N., Stolfi, J., Gregori, R. a Silva, M. da. An Optimal Algorithm for 3D Triangle Mesh Slicing. *Computer-Aided Design*. 1. vyd. Červenec 2017, sv. 92. DOI: 10.1016/j.cad.2017.07.001.

[17] Paraskevoudis, K., Karayannis, P. a Koumoulos, E. P. Real-Time 3D Printing Remote Defect Detection (Stringing) with Computer Vision and Artificial Intelligence. *Processes*. 1. vyd. Listopad 2020, sv. 8, s. 1464. DOI: 10.3390/pr8111464.

[18] Perez, L. a Wang, J. *The Effectiveness of Data Augmentation in Image Classification using Deep Learning*. arXiv, 2017. DOI: 10.48550/ARXIV.1712.04621. Dostupné z: https://arxiv.org/abs/1712.04621.

[19] Petsiuk, A. L. a Pearce, J. M. Open source computer vision-based layer-wise 3D printing analysis. *Additive Manufacturing*. 1. vyd. 2020, sv. 36, s. 101473. DOI: https://doi.org/10.1016/j.addma.2020.101473. ISSN 2214-8604. Dostupné z: https://www.sciencedirect.com/science/article/pii/S2214860420308459.

[20] Powers, D. M. W. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. 1. vyd. arXiv. 2020. DOI: 10.48550/ARXIV.2010.16061. Dostupné z: https://arxiv.org/abs/2010.16061.

[21] Redwood, B., Schöffer, F. a Garret, B. *The 3D Printing Handbook*. 1. vyd. 3DHubs, 2017. ISBN 978-90-827485-0-5.

[22] Rosebrock, A. *Deep Learning for Computer Vision with Python*. 1. vyd. PyImageSearch, 2017. Dostupné z: https://books.google.cz/books?id=9Ul-tgEACAAJ.

[23] Stříteský, O., Průša, J. a Bach, M. *Základy 3D tisku s Josefem Průšou* [online]. 2019 [cit. 2022-12-04]. Dostupné z: https://is.muni.cz/el/ped/jaro2021/TI9009/111101390/zaklady-3d-tisku.pdf.

[24] Tan, L., Huangfu, T., Wu, L. a Chen, W. Comparison of YOLO v3, Faster R-CNN, and SSD for Real-Time Pill Identication. 1. vyd. arXiv. 2021. DOI: https://doi.org/10.21203/rs.3.rs-668895/v1. Dostupné z: https://assets.researchsquare.com/files/rs-668895/v1_covered.pdf?c=1631875157.

[25] Zhu, J.-Y., Park, T., Isola, P. a Efros, A. A. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. 1. vyd. arXiv. 2017. DOI: 10.48550/ARXIV.1703.10593. Dostupné z: https://arxiv.org/abs/1703.10593.