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

SEGMENTACE SYMBOLŮ VE SKENECH PNEUMATIK S POUŽITÍM SELF-SUPERVISED LEARNING

SEGMENTATION OF SYMBOLS IN TIRE SCANS USING SELF-SUPERVISED LEARNING

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MICHAL BALOGH**
AUTHOR

VEDOUCÍ PRÁCE **doc. Ing. MICHAL ŠPANĚL, Ph.D.**
SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

163282

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Balogh Michal**
Program: Informační technologie
Název: **Segmentace symbolů ve skenech pneumatik s použitím self-supervised learning**
Kategorie: Zpracování obrazu
Akademický rok: 2024/25

Zadání:

1. Seznamte se s problematikou hlubokých neuronových sítí a jejich učení. Zaměřte se na metody segmentace obrazu využívající self-supervised a unsupervised learning.
2. Analyzujte současný stav v oblasti segmentace obrazu a rozpoznávání symbolů v obraze a seznamte se s existujícím řešením firmy Micro-Epsilon Inspection pro identifikaci znaků na snímcích pneumatik.
3. Na základě získaných poznatků vyberte vhodné metody a navrhněte řešení pro automatickou segmentaci symbolů ve snímcích povrchu pláště pneumatiky, které nebude vyžadovat náročné dotrénování pro doplnění nového symbolu.
4. Navržený postup implementujte v experimentálním prototypu.
5. Proveďte experimenty a porovnejte dosažené výsledky na připravené datové sadě reálných snímků pláště pneumatik s již existujícím řešením. Diskutujte možnosti budoucího vývoje.
6. Vytvořte stručný plakát nebo video prezentující vaši práci, její cíle a výsledky.

Literatura:

- D. Niu, X. Wang, X. Han, L. Lian, R. Herzig, T. Darrell, "Unsupervised Universal Image Segmentation", *CVPR,* 2024, https://u2seg.github.io/.
- P. Toth Vaňo, "Identifikace znaků na hloubkovém snímku pneumatiky"*, Bakalářská práce*, Vysoké učení technické v Brně, Fakulta informačních technologií, Brno, 2022, https://www.fit.vut.cz/study/thesis/25095/.

Při obhajobě semestrální části projektu je požadováno:
První tři body zadání.

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

Vedoucí práce: **Španěl Michal, doc. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2024
Termín pro odevzdání: 14.5.2025
Datum schválení: 12.11.2024

Abstrakt

Táto bakalárska práca sa zaoberá efektívnym dotrénovaním segmentačných modelov pre rozpoznávanie symbolov na bočniciach pneumatík. Súčasné riešenie používané firmou Micro-Epsilon Inspection, s.r.o. vyžaduje približne 72 hodín na adaptáciu na nové symboly. Hlavným cieľom práce bolo výrazne skrátiť tento čas pri zachovaní alebo zlepšení kvality segmentácie. Analyzované boli rôzne state-of-the-art prístupy k segmentácii obrazu vrátane few-shot učenia a kombinácií detekčných a segmentačných modelov. Na základe úvodných experimentov bola implementovaná metóda založená na modeli OneFormer v kombinácii s technikami zmiešanej presnosti a Low-Rank Adaptation (LoRA). Navrhnuté riešenie umožňuje adaptáciu na nové symboly v čase kratšom než jedna hodina, čo predstavuje 72-násobné zrýchlenie oproti pôvodnému riešeniu, pričom dosahuje porovnateľné alebo lepšie výsledky segmentácie. Práca tiež analyzuje vplyv rôznych typov segmentácie, inicializácií váh a pomerov trénovacích dát na výkon modelu.

Abstract

This bachelor's thesis addresses the efficient fine-tuning of segmentation models for recognizing symbols on tire sidewalls. The current solution used by Micro-Epsilon Inspection, s.r.o. requires approximately 72 hours to adapt to new symbols. The main goal of this work was to significantly reduce this time while maintaining or improving segmentation quality. Various state-of-the-art approaches to image segmentation were analyzed, including few-shot learning and combinations of detection and segmentation models. Based on initial experiments, a method using the OneFormer model combined with mixed precision techniques and Low-Rank Adaptation (LoRA) was implemented. The proposed solution enables adaptation to new symbols in less than one hour, representing a 72-times speedup compared to the original solution, while achieving comparable or better segmentation results. The thesis also analyzes the impact of different segmentation types, weight initializations, and training data ratios on model performance.

Kľúčové slová

Segmentácia obrazu, OneFormer, Low-Rank Adaptation (LoRA), zmiešaná presnosť, efektívne dotrénovanie, transformery, inštančná segmentácia, symboly na pneumatikách, hĺbkové snímky, kvalita segmentácie.

Keywords

Image segmentation, OneFormer, Low-Rank Adaptation (LoRA), mixed precision, efficient fine-tuning, transformers, instance segmentation, tire symbols, depth images, segmentation quality.

Citácia

BALOGH, Michal. *Segmentace symbolů ve skenech pneumatik s použitím self-supervised learning*. Brno, 2025. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce doc. Ing. Michal Španěl, Ph.D.

Segmentace symbolů ve skenech pneumatik s použitím self-supervised learning

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením doc. Ing. Michala Španěla, Ph.D. Ďalšie informácie mi boli poskytnuté spoločnosťou Micro-Epsilon Inspection, s.r.o., predovšetkým RNDr. Andrejom Lúčnym, PhD. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Michal Balogh
14.5.2025

Poďakovanie

Ďakujem vedúcemu bakalárskej práce, doc. Ing. Michalovi Španělovi, Ph.D., za odborné vedenie, cenné pripomienky a konzultácie počas vypracovania práce.

Poďakovanie patrí aj spoločnosti Micro-Epsilon Inspection, s.r.o. za sprístupnenie dátovej sady a možnosť možnosť pracovať na praktickom probléme z priemyselného prostredia. RNDr. Andrejovi Lúčnemu, PhD., ďakujem za odborné rady a konzultácie.

Obsah

Zoznam obrázkov

Kapitola 1

Úvod

Táto bakalárska práca vznikla v spolupráci s firmou Micro-Epsilon Inspection, s.r.o., ktorá sa špecializuje na vývoj inšpekčných systémov najmä pre gumárenský a automobilový priemysel. Firma poskytuje svojim zákazníkom systém kontroly kvality pneumatík, ktorého súčasťou je aj kontrola kvality symbolov na pneumatikách pomocou modelu Mask R-CNN. Súčasné riešenie však vyžaduje zdĺhavé pretrénovanie celého modelu pri zavádzaní nových symbolov, čo trvá približne 72 hodín.

Hlavným cieľom tejto práce je preskúmať a navrhnúť metódy efektívneho dotrénovania (fine-tuning) segmentačných modelov, ktoré by umožnili výrazne skrátiť čas potrebný na adaptáciu na nové symboly pri zachovaní, prípadne zlepšení kvality segmentácie dosahovanej súčasným riešením. Na dosiahnutie tohto cieľa som implementoval riešenie založené na modeli OneFormer kombinovanom s technikami zmiešanej presnosti (mixed precision training) a metódou Low-Rank Adaptation (LoRA). Navrhnuté riešenie umožňuje adaptáciu modelu na nové symboly v čase kratšom než jedna hodina miesto pôvodných 72 hodín, pričom dosahuje porovnateľné alebo lepšie výsledky segmentácie než súčasné riešenie.

Práca využíva dátovú sadu hĺbkových snímok poskytnutú firmou Micro-Epsilon Inspection, s.r.o.

Text práce je rozdelený do deviatich kapitol. Kapitola 2 popisuje problém segmentácie symbolov na bočniciach pneumatík a existujúce firemné riešenie. Kapitola 3 sa venuje poskytnutej dátovej sade. V kapitole 4 sú zhrnuté state-of-the-art prístupy k segmentácii obrazu a efektívnemu dotrénovaniu modelov, vrátane siete OneFormer, few-shot učenia, zmiešanej presnosti a metódy *Low-Rank Adaptation (LoRA)*, spolu s použitými metrikami. Kapitola 5 sa zaoberá prvotnými experimentmi. Návrh finálneho riešenia je predstavený v kapitole 6 a jeho implementácia v kapitole 7. Kapitola 8 obsahuje popis a vyhodnotenie experimentov vrátane práce s umelo generovanou dátovou sadou. Záverečná kapitola 9 sumarizuje výsledky, zdôrazňuje prínosy a načrtáva smery budúceho výskumu.

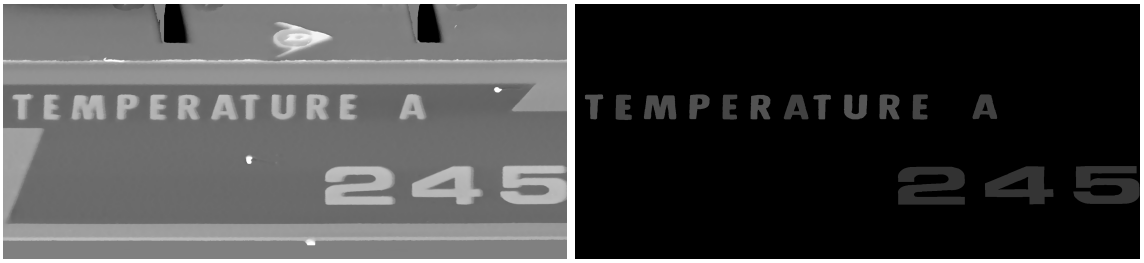
Kapitola 2

Segmentácia symbolov na bočnici pneumatiky

Segmentácia symbolov je jeden z bežných problémov počítačového videnia. Táto práca má za úlohu riešiť problém segmentácie symbolov na bočnici pneumatiky. Vznikla v spolupráci s firmou Micro-Epsilon Inspection, s.r.o., ktorá sa zaoberá výrobou a implementáciou meracích a inšpekčných vizuálnych systémov pre gumárenský a automobilový priemysel [1]. Bočnice pneumatiky obsahujú rôzne vyrezané znaky a symboly (viď obrázok 2.1), ktoré musia spĺňať určitú kvalitu, pretože obsahujú dôležité informácie, pomocou ktorých sa dá pneumatika identifikovať, napríklad dátum výroby, jedinečný identifikátor pneumatiky, výrobca a séria. Bočnica obsahuje aj informácie o fyzikálnych parametroch, ako je napríklad veľkosť, informácie o ročnom období a typoch vozovky, na ktorú je pneumatika určená, ale aj bezpečnostné varovania.

Obr. 2.1: Bočnica pneumatiky.

Preto je potrebné získať masky zo skenov pneumatík k týmto symbolom a overiť ich kvalitu. Ukážka skenu časti bočnice a korešpondujúce masky symbolov sú zobrazené na obrázku 2.2. Táto úloha sa oproti klasickej segmentácii obrazu líši a robí ju zložitejšou tým, že segmentuje vyrezané čierne symboly na čiernom pozadí pneumatiky. Preto sa z pneumatiky vytvorí 3D sken špecializovaným prístrojom, a z tohto skenu sa vygeneruje šedotónová snímka, kde intenzita šedej reprezentuje hĺbku v danom bode.



Obr. 2.2: Snímka vytvorená z 3D skenu bočnice pneumatiky a korešpondujúca maska so symbolmi.

2.1 Súčasné riešenie – segmentácia symbolov pomocou Mask R-CNN

Súčasné riešenie implementované vo firme Micro-Epsilon Inspection, s.r.o., bolo vyvinuté s cieľom automatizovať proces identifikácie symbolov vyrezaných na bočnici pneumatiky. Je postavené na architektúre Mask R-CNN [3]. Nasledujúci opis poskytuje prehľad riešenia, jeho výhod aj obmedzení. Detailný popis je uvedený v práci [23].

Segmentácia symbolov a získanie masiek

Model Mask R-CNN sa od základnej architektúry Faster R-CNN [19] líši pridaním dodatočnej vetvy, ktorá zabezpečuje predikciu masky pre každý detegovaný objekt. Proces segmentácie symbolov zo skenov bočníc pneumatík prebieha nasledovne:

Detekcia regiónov: Najprv sieť vygeneruje kandidátske regióny (regions of interest, ROI), ktoré potenciálne obsahujú symboly. Využívajú sa referenčné boxy (anchors), prispôsobené očakávaným rozmerom symbolov.

Extrahovanie príznakov: Celý obraz prechádza konvolučnými vrstvami, čím vzniká mapa príznakov. Z nej sa následne pre každý ROI extrahuje príznakový vektor pomocou vrstvy RoIAlign.

Predikcia triedy, pozície a masky: Na extrahovaných príznakoch sú paralelne aplikované tri vetvy:

- klasifikačná vetva na predikciu triedy objektov,
- regresná vetva pre spresnenie ohraničujúceho boxu,
- plne konvolučná vetva, ktorá generuje binárnu masku symbolu.

Vygenerované masky sú následne porovnávané s preddefinovanými šablónami v databáze, čo umožňuje overenie zhody a správnosti symbolov. Pri porovnávaní sa využíva algoritmus, ktorý okrem podobnosti hodnotí aj kvalitu segmentácie. Tým je zabezpečené, že výsledná identifikácia spĺňa požiadavky stanovené firmou.

Zhrnutie pôvodného riešenia

Pôvodné riešenie správne deteguje symboly a generuje masky s vysokou presnosťou na naučených dátach. Problém však nastáva pri zmene symbolov na pneumatikách alebo potrebe

segmentácie nového typu pneumatiky. V takom prípade je potrebné pretrénovať celý model Mask R-CNN, čo trvá približne tri dni.

Táto práca má za úlohu nájsť riešenie, ktoré podáva rovnaké alebo lepšie výsledky ako pôvodný model, no zároveň sa rýchlejšie dokáže adaptovať na nové symboly.

Kapitola 3

Dátová sada symbolov na bočnici pneumatiky

Dátová sada bola poskytnutá firmou Micro-Epsilon Inspection a bola použitá rovnaká ako pri tvorbe pôvodného riešenia [23], aby sa dalo nové riešenie čo najlepšie porovnať s pôvodným. Databáza snímok pneumatík je tvorená hĺbkovými snímkami s rozmermi 65 536 pixelov na šírku a 1 300 pixelov na výšku. Tieto snímky sú šedo tónové s hĺbkou 8 bitov. Snímky vznikli pomocou 3D skenu a odtieň šedej odráža hĺbku bodu na pneumatike. Celkový počet snímok v databáze snímok je 92, z toho je 21 rôznych typov. Snímky rovnakého typu obsahujú rovnaké rozloženie textov a symbolov a líšia sa len v niektorých konkrétnych hodnotách (napríklad dátume) a v tom, kde na obvode pneumatiky začínajú a končia.

Databáza snímok bola rozdelená na dve časti — sadu na trénovanie a testovanie. Snímky boli rozdelené do týchto dvoch sád tak, aby obe časti čo najlepšie obsahovali všetky typy snímok.

- Množina na trénovanie – obsahuje **19 typov** pneumatík a **66 snímok**
- Množina na testovanie – obsahuje **17 typov** pneumatík, z čoho 15 je spoločných s trénovacou množinou, 2 sú iného typu, dokopy obsahuje **26 snímok**

Ku každej snímke v databázy snímkov sú aj anotované dáta vo formáte CSV a každý symbol na danej snímke má aj samostatnú masku. Masky sú binárne obrázky o veľkosti ohraničujúceho boxu daného symbolu, kde pixel s hodnotou 0 značí pozadie a pixel s hodnotou 1 znamená, že daný pixel prislúcha symbolu.

Snímky z databázy symbolov zaberajú veľkú pamät na grafickej karte, preto bola z týchto snímok vytvorená dátová sada tak, ze každá z nich bola pomocou poskytnutých skriptov k dátovej sade rozrezaná na 128 pod snímok, každá o veľkosti 2 500 pixelov na šírku s prekryvom 2 000 pixelov so susednou snímkou a 1 300 pixelov na výšku bez prekryvu. Prekryv snímok je znázornený na obrázku 3.1.

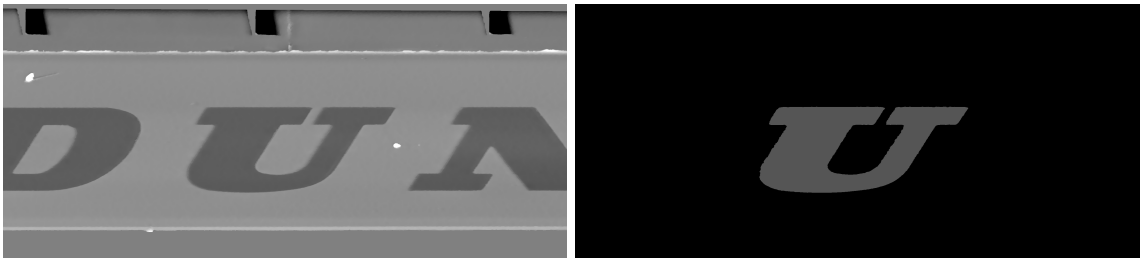
Dátová sada teda obsahuje:

- **Trénovacia sada** – 8448 snímok (66 snímok, každá rozrezaná na 128 podsnímok)
- **Testovacia sada** – 3328 snímok (26 snímok, každá rozrezaná na 128 podsnímok)

K týmto podsnímkom sa z pôvodného anotačného súboru vytvoria nové anotačné súbory a masky korešpondujúce k novo vytvoreným podsnímkom. Masky sú vytvorené len k symbolom, ktoré sa nachádzajú s viac ako určitou časťou plochy na danom podsnímku.

Obr. 3.1: Rozrezané podsnímky sa prekrývajú a majú spoločných 2000 pixelov na šírku.

Pri tvorení nového riešenia som prah nastavil na 95 % plochy symbolu. Na obrázku 3.2 je zobrazený obrázok z dátovej sady a k nemu korešpondujúca maska. Keďže iba symbol „U“ je prítomný svojou plochou viac ako 95 %, maska je vytvorená len k nemu.



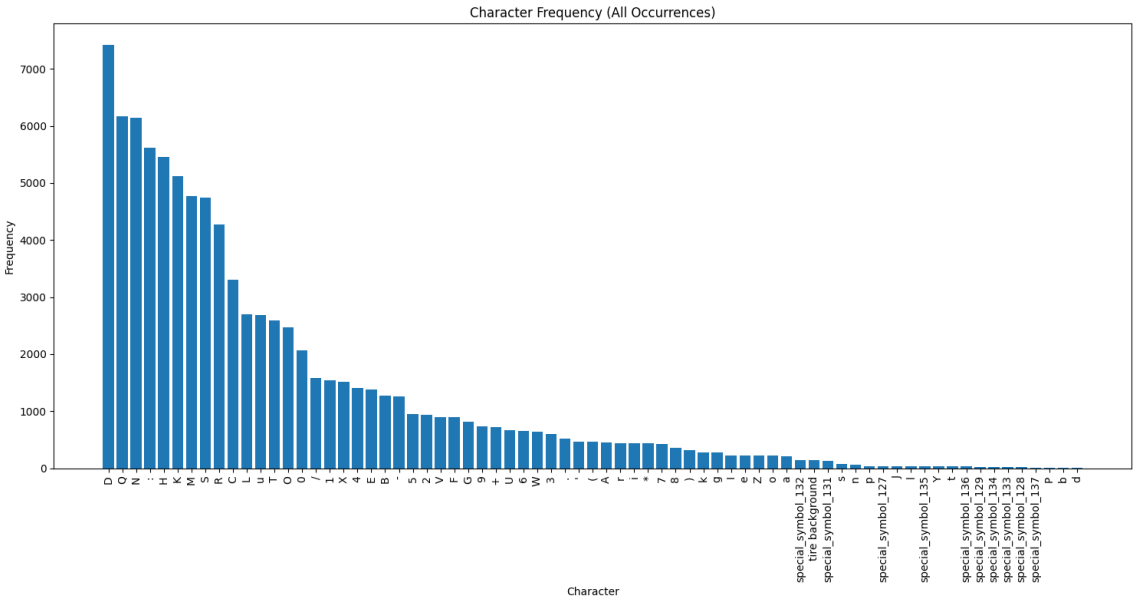
Obr. 3.2: Masky sú vytvorene len k symbolom, ktoré sú na snímke prítomne z väčšej časti ako určený prah (v tomto prípade 95%).

Tieto podsnímky sa následne využívajú ako dátová sada aj pri trénovaní a následne aj pri inferencii. Keďže sa podsnímky prekrývajú, jeden symbol sa môže vyskytovať na viacerých snímkach naraz. Toto rieši pôvodné riešenie algoritmom potlačenia nemaximálnych hodnôt (angl. *non-maximum suppression*). Detailnejší popis skladania pôvodnej snímky z podsnímkov je popísaný v práci, na ktorú nadväzujem [23].

Príklad obrázku z dátovej sady a masiek k nemu, je na obrázku 3.3 Bližší popis CSV anotácií bude uvedený v kapitole 7.1.

V dátovej sade sa nachádzajú znaky abecedy, číslice, špeciálne znaky a špeciálne symboly (napr. slnko, dažďové kvapky, snehová vločka). Dohromady je v dátovej sade 71 rôznych symbolov a pozadie. Rôzne znaky a symboly môžu mať inú veľkosť a font a niektoré sú napríklad bez výplne, len vonkajšia čiara. Frekvencia výskytu jednotlivých znakov v dátovej sade sa výrazne líši – 4 výskyty pre znaky „b“ a „d“ až po 7 423 výskytov pre znak „E“. Histogram znázorňujúci nerovnomernosť výskytu inštancii tried v dátovej sade je na obrázku 3.4.

Obr. 3.3: Ukážka snímky z dátovej sady a niektorých masiek k tejto snímke.



Obr. 3.4: Histogram výskytu inštancii tried v dátovej sade.

Kapitola 4

State-of-the-art metódy na dotrénovanie segmentačných modelov

Existujú tri druhy segmentácie:

- Inštančná – segmentuje každú inštanciu objektu samostatne, to znamená, že ak sa nachádza viac objektov z rovnakej klasifikačnej triedy v jednom obrázku, rozozná jednotlivé inštancie. Výsledkom tejto segmentácie je zoznam pixelov, ktoré patria danej inštancii objektu, a trieda danej inštancie objektu. Zameriava sa väčšinou na objekty, tzv. *things*.
- Semantická – klasifikuje každý pixel obrázka a rozoznáva len to, do akej triedy daný pixel patrí. To znamená, že viac objektov z jednej klasifikačnej triedy rozoznáva ako jeden a ten istý objekt. Okrem toho segmentuje aj pozadia, tzv. *stuff*.
- Panoptická – segmentácia kombinuje dva predchádzajúce prístupy dohromady – klasifikuje každý pixel v obrázku, no priraďuje ho k inštancii objektu, nie ku triede.

Pre našu aplikáciu je potrebná inštančná alebo panoptická segmentácia, keďže je potrebné skontrolovať každý znak, aj keď sa vyskytuje na pneumatike viackrát. Pozadie pneumatiky nie je dôležité, preto vyhovuje aj samotná inštančná segmentácia.

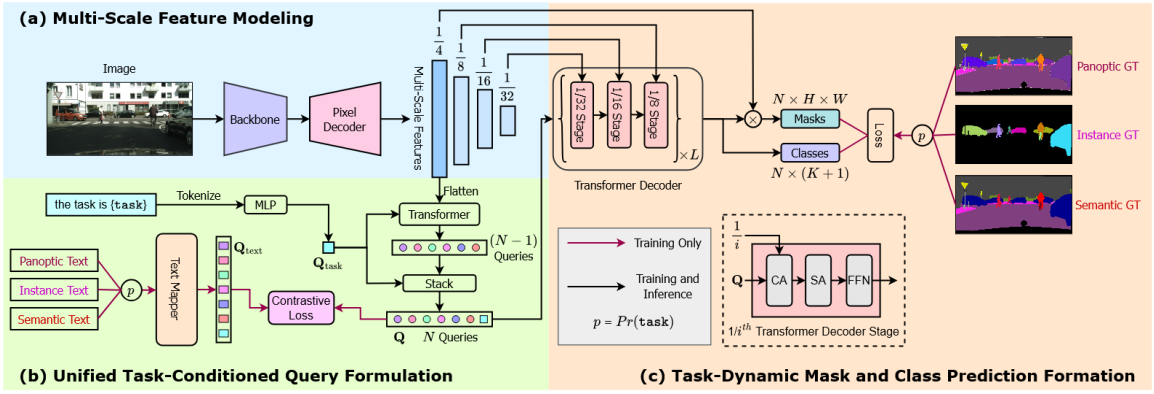
4.1 OneFormer

Ako model pre segmentáciu symbolov som zvolil jeden zo state-of-art prístupov pre túto úlohu – OneFormer. OneFormer je framework založený na vizuálnych transformeroch, ktorý unifikuje všetky hlavné typy segmentácie v jednom modeli. Jeho architektúra umožňuje trénovanie na jednej segmentačnej úlohe, pričom dosahuje state-of-the-art výsledky naprieč instančnou, sémantickou a panoptickou segmentáciou a prekonáva modely špecializované na konkrétnu úlohu. Uvedené vlastnosti a výsledky boli popísané v práci [5], z ktorej tento opis vychádza.

Architektúra

Architektúra modelu OneFormer je navrhnutá s cieľom efektívne unifikovať rôzne segmentačné úlohy v rámci jedného systému. Kľúčovými komponentmi architektúry sú: backbone

pre získavanie príznakov z obrázka, dekóder pixelov, ktorý spracúva a spája príznakové mapy, a transformerový dekóder, ktorý generuje segmentačné výstupy na základe špecifických dotazov. Architektúra modelu OneFormer je zobrazená na obrázku 4.1.



Obr. 4.1: Architektúra siete OneFormer. Prevzaté z [5].

Backbone a dekóder pixelov

Vstupný obrázok sa najprv spracováva cez pred trénovaný backbone. V spomínanej práci [5] použili Swin Transformer [11], ConvNeXt [12] alebo DiNAT [2]. Tento backbone extrahuje multi-škálové príznakové mapy, ktoré zachytávajú rôzne úrovne abstrakcie – od nízkoúrovňových detailov po vysokoúrovňové semantické informácie. Tieto informácie sú odovzdané pixelovému dekóderu, ktorý dané informácie nadvzorkuje (angl. „upsampling“) a pomocou laterálnych spojení skombinuje multi-škálové príznakové mapy do jednej, ktorá umožňuje detailnú segmentáciu.

Dekóder transformera a reprezentácie dotazov

Model využíva dva základné typy dotazov (angl. „queries“):

- **Objektové dotazy:** nový a špecifický typ tokenu, tzv. *„task token“*. Tento token explicitne informuje model o danej segmentačnej úlohe. Príklad *task tokenu* pre panoptickú segmentáciu môže vyzerať napríklad takto: „the task is panoptic“. Model podľa týchto objektových dotazov generuje daný typ predikcií. Tento token je podstatný pre OneFormer, keďže jeho odstránenie viedlo k výraznému poklesu v metrikách AP (-2.7%) a PQ (-4.5%).
- **Textové dotazy:** sú generované textovým kodérom, ktorý spracováva textové popisy. V práci je používaná šablóna, kde je stabilný popis: „a photo with a“ a nasleduje meno triedy. Tieto textové popisy slúžia ako referencia pre kontrastívne učenie, vďaka čomu model dokáže lepšie odlíšiť jednotlivé kategórie.

Tieto vstupné dotazy vstupujú do multi-škálového dekóderu transformera. Pomocou samopozornostného (angl. „self-attention“) mechanizmu a krížového pozornostného (angl. „cross-attention“) mechanizmu spája informácie z multi-škálových príznakových máp a generuje finálne predikcie – masky, a pri inštančnej a panoptickej segmentácii aj ohraničujúce boxy a skóre istoty pre jednotlivé triedy.

Spôsob trénovania: Task-Conditioned Joint Training

Keďže trénovanie sa uskutočňuje naraz pre všetky hlavné typy segmentácie, využívajú sa panoptické anotácie, ktoré okrem referenčných informácií pre panoptickú segmentáciu obsahujú informácie aj pre inštančnú a semantickú segmentáciu. Pri trénovaní sa pre každú vzorku z dátovej sady vyberie jedna z troch hlavných segmentačných úloh. To zabezpečuje, že model sa učí riešiť všetky úlohy súčasne. Počas inferencie sa potom pomocou explicitne zadaného *task tokenu* model adaptuje na zadanú úlohu.

Obrázky z dátovej sady boli pred vstupom do modelu spracované procesorom, ktorý zabezpečuje kompatibilitu vizuálnych dát s modelom. Tento procesor okrem spracovania vizuálnych dát zároveň zakóduje *task token* pomocou pred trénovaného tokenizačného procesora CLIPTokenizer pre transformerový enkóderový modul. Pri inferencii sa pomocou tohto procesora spracúvajú výstupy, ktoré sa konvertujú z logitov na predikcie, masky alebo ohraničujúce boxy – podľa toho, aký typ *task tokenu* bol modelu poskytnutý.

Chybové a optimalizačné funkcie

Pri trénovaní sa model snaží minimalizovať celkovú chybovú funkciu, ktorá je vypočítaná ako vážený súčet viacerých komponentov zodpovedajúcich jednotlivým typom segmentačných úloh:

- **Vzájomná entropia (angl. cross-entropy):** vďaka chybovej funkcii cross-entropy loss sa model učí správne kategorizovať objekty do príslušných tried.
- **Dice Loss a binárna vzájomná entropia (angl. binary cross-entropy):** tieto chybové funkcie slúžia na učenie modelu predikovať masky a zabezpečujú presnú segmentáciu objektov.
- **Kontrastívna strata (angl. contrastive loss) na dotazoch:** táto strata je dôležitá pre spojenie textových a objektových dotazov. Model porovnáva podobnosť medzi obrazom a jeho textovým popisom, vďaka čomu model jasnejšie rozoznáva jednotlivé triedy a aj jednotlivé segmentačné úlohy.

Na spárovanie predikovaných objektov a referenčných anotácií sa používa bipartitný matching, ktorý zahŕňa klasifikačnú, boxovú aj maskovú stratu. To zabezpečuje, že každá predikcia je spárovaná s najvhodnejšou referenčnou anotáciou.

4.2 Few-shot a zero-shot učenie

Na trénovanie modelov počítačového videnia sú obvykle potrebné rozsiahle sady trénovacích dát, ktoré musia byť detailne anotované. Tento problém sa snaží riešiť technika učenia few-shot [20].

Few-shot učenie je technika trénovania v strojovom učení pre modely hlbokého učenia v počítačovom videní, ktorá umožňuje modelom naučiť sa rozpoznať nové kategórie, ktoré nepoznajú, na veľmi malej vzorke dát (typicky 1 alebo 5 pre každú novú triedu). Táto technika sa používa v prípadoch, kedy dáta na trénovanie nie sú dostupné, alebo je veľmi ťažké ich získať a anotovať. Taktiež ide o snahu čo najlepšie generalizovať modely hlbokého učenia, keďže obvykle triedy, ktoré sa nenaučili počas trénovania, vôbec nerozpoznajú.

Ak nie je dostupná ani malá vzorka trénovacích dát, metóda few-shot sa zmení na zero-shot. Namiesto anotácií sa využívajú dodatočné informácie, napríklad slovné popisy alebo „word embeddings“, ktoré reprezentujú sémantické atribúty kategórií.

4.3 Trénovanie so zmiešanou presnosťou

Trénovanie so zmiešanou presnosťou (angl. mixed precision training) [22, 13] je metóda, pri ktorej sa používajú naraz 16-bitové (polovičná presnosť) a 32-bitové (jednoduchá presnosť) typy s pohyblivou desatinnou čiarkou pri trénovaní modelu. Dôsledkom použitia parametrov s presnosťou 16 bitov sú rýchlejšie výpočty a použitie menej pamäte. Zároveň sa niektoré časti modelu nekonvertujú na nižšiu presnosť a zachovajú ako 32-bitové, čím sa dosiahne stabilita a model zachová kvalitu učenia ako pri plnej presnosti. Použitie trénovania so zmiešanou presnosťou môže zrýchliť trénovanie viac ako 3-krát na moderných GPU a viac ako 2-krát na moderných CPU. Toto zrýchlenie je dôsledkom viacerých faktorov:

- 16-bitové typy sa načítajú do pamäte rýchlejšie,
- moderný hardware má špecializované operácie pre 16-bitové typy,
- 16-bitové typy zaberajú menej miesta v pamäti, takže sa môže zväčšiť veľkosť dávky (angl. batch size).

Okrem zrýchlenia trénovania sa vďaka použitiu zmiešanej presnosti môžu do pamäte zmestiť aj väčšie modely.

Trénovanie so zmiešanou presnosťou zahŕňa dva kroky:

1. pridanie škálovania stratovej funkcie (angl. loss function) na zachovanie malých hodnôt gradientov.
2. konvertovanie častí modelu na 16-bitovú presnosť tam, kde to neovplyvní kvalitu trénovania,

Škálovanie gradientov je potrebné, aby sa zabránilo ich zaokrúhľovaniu na nulu pri prevode na 16-bitovú presnosť. Bez škálovania môže zmiznúť veľké množstvo gradientov, čo vedie k divergencii chybovej funkcie. Pri úprave gradientov pred začatím spätného šírenia (angl. back propagation) sa najprv hodnoty vynásobia škálovacím faktorom S a po ich výpočte sa vynásobia naspäť inverznou hodnotou 1/S, aby sa zachovala správna škála aktualizácií. Škálovací faktor môže byť konštantná hodnota alebo sa môže upravovať dynamicky. Pri dynamickom výbere škálovacieho faktora sa začne s veľkou hodnotou S, ktorá sa postupne zvyšuje, ak nedochádza k pretečeniu. Pretečenie by sa preukázalo hodnotami Inf alebo NaN v gradientoch. Ak dôjde k pretečeniu, aktualizácia váh sa preskočí a S sa zníži.

Kvôli požiadavkám dizajnu tenzorových jadier na GPU vstupné údaje musia mať určitý tvar. Pri násobení matíc pre 16-bitový typ s pohyblivou desatinnou čiarkou musia byť všetky tri rozmery tenzoru násobkom 8 a taktiež pri konvolúcii musí byť počet vstupných a výstupných kanálov 8. Preto je dobré nastaviť ako násobok 8 nasledujúce parametre:

- veľkosť mini dávky (angl. mini-batch size),
- rozmery lineárnych vrstiev,
- počet kanálov v konvolučných vrstvách.

4.4 Low Rank Adaptation

Základným a najčastejším výpočtom v neurónových sieťach a modeloch hlbokého učenia je násobenie matíc. Autori metódy *Low Rank Adaptation* [4, 7] experimentovaním zistili,

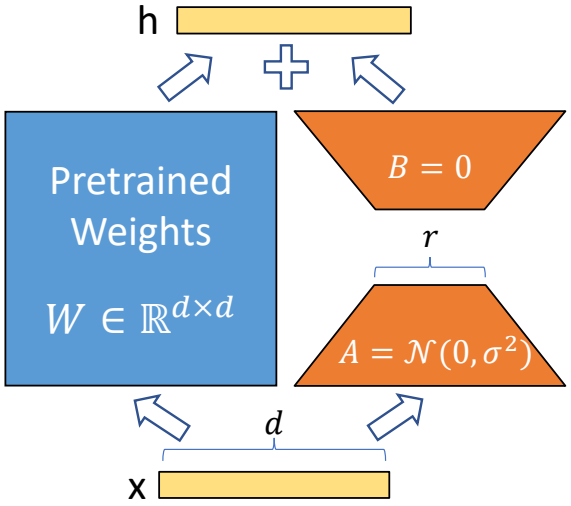
že pri adaptácii modelu na novú úlohu je potrebná len malá zmena pôvodných váhových matíc. Nech $W_0 \in \mathbb{R}^{d\times k}$ je pred trénovaná váhová matica modelu a ΔW je zmena váh vo váhovej matici pri jemnom dotrénovaní (angl. fine-tuning). Zmena váh sa dá rozložiť na matice nižšej hodnosti:

$$W_0 + \Delta W = W_0 + BA \tag{4.1}$$

kde $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$ a hodnosť $r << \min(d, k)$. Hodnosť r udáva, aké malé sú matice A a B v porovnaní s W_0. Namiesto $d \times k$ parametrov sa trénuje iba $r \times (d+k)$ parametrov. Ak by napríklad bola pôvodná matica $W_0 \in \mathbb{R}^{d\times k}$, kde $d = 1024$, $k = 4096$, ak nastavíme $r = 4$, potom namiesto ukladania a trénovania 4 194 304 (1024×4096) parametrov, by sa pracovalo len s 20 480 $(4 \times (1024 + 4096))$ parametrami.

W_0 sú zmrazené váhy modelu, ktoré sa počas adaptácie nemenia, naopak A aj B obsahujú trénovateľné parametre modelu. Výstup dopredného kroku je teda $h = W_0x + BAx$. Váhy matice A sa inicializujú Gaussovým rozdelením a váhy matice B nulami. Teda súčin BA je na začiatku nulový ($\Delta W = 0$), čo znamená, že pri začatí dotrénovania model spracúva vstupy iba s pôvodnými váhami W_0, čím sa zabezpečí, že trénovanie nezačne s veľkými nepredvídateľnými zmenami. Počas trénovania sa zmena váh ΔW škáluje pomocou faktora α/r, kde α je konštanta. Toto škálovanie zaistí stabilitu trénovania. Pri použití optimalizátora Adam, optimalizovanie α sa správa podobne ako optimalizovanie rýchlosti učenia (angl. learning rate). Vďaka tomu nie je potrebné hľadať optimálnu hodnotu α, ale stačí ju nastaviť na prvú hodnotu r, čo uľahčuje manuálne ladenie hyperparametrov.

Princíp metódy LoRA (Low Rank Adaptation) je zobrazený na obrazku 4.2.



Obr. 4.2: Princíp metódy LoRA: namiesto aktualizácie celej matice váh (znázornené modrým štvorcom) sa pridáva nízkorozmerová korekcia. Takto sa výrazne znižuje počet trénovateľných parametrov. Prevzaté z [4].

Metóda LoRA sa líši od klasického doladenia tým, že nie všetky váhy modelu sa aktualizujú. Zvýšením hodnosti r sa metóda LoRA približuje k plnému klasickému jemnému doladeniu.

LoRA sa dá aplikovať na akúkoľvek plne prepojenú vrstvu. Pôvodne bola navrhnutá pre transformery s veľkým množstvom parametrov na prirodzené spracovanie jazyka, ako napríklad GPT alebo BERT. LoRA sa najčastejšie v transformeroch aplikuje na vrstvy

dotaz (angl. query) a hodnota (angl. value) v samopozornostnom (angl. self-attention) mechanizme, pretože tieto vrstvy majú najväčší vplyv na adaptáciu modelu na novú úlohu. Pre vizuálne transformery sa dá LoRA použiť podobne ako v jazykových transformeroch.

Použitie metódy LoRA pre riešenie problému dlhého dotrénovania nových symbolov na bočnici pneumatiky by mohlo mať za následok rýchlejšie dotrénovanie vďaka menším výpočtovým nárokom a úsporám pamäte, čo má za dôsledok zvýšenie veľkosti dávky a rýchlejšie trénovanie. Napríklad, u GPT-3 so 175 biliónmi parametrov sa dosiahlo zrýchlenie o 25% [4]. Okrem priameho rýchlejšieho dotrénovania má LoRA aj iné benefity. Úspora VRAM na veľkých transformeroch môže dosiahnuť až 2/3, keďže nie je potrebné uchovávať stavy optimalizátora pre zmrazené váhy. Pri trénovaní GPT-3 sa vďaka LoRA znížila spotreba VRAM z 1.2 TB na 350 GB [4]. Tým, že netreba uchovávať stavy zmrazených váh, zníži sa aj veľkosť kontrolných bodov (angl. checkpoint). Pri GPT-3 to bolo $10,000\times$ – z 350 GB na 35 MB [4].

4.5 Metriky

Na ohodnotenie kvality riešení boli použité typické metriky využívané pri evaluácii segmentačných modelov. Tieto metriky poskytujú rôzne pohľady na kvalitu segmentácie, a preto sa často používajú zároveň, vďaka čomu sa získa komplexný prehľad o výkone modelu.

Mean Intersection over Union

Mean Intersection over Union (mIoU) je jednou z najčastejšie používaných metrík pri segmentácii obrazu. Pre každú triedu sa vypočíta ako pomer prieniku a zjednotenia medzi predikovanou a skutočnou maskou:

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{4.2}$$

kde A je množina pixelov predikcie a B je množina pixelov ground truth. Mean IoU sa následne získa spriemerovaním IoU cez všetky triedy. Vyššie hodnoty mIoU znamenajú presnejšiu segmentáciu. Nadobúda hodnoty od 0 do 1.

Dice Score/F1 Score

Dice skóre, alebo aj F1 skóre, je metrika podobná IoU, ale kladie väčší dôraz na prekryv. Používa sa väčšinou v dátových sadách s medicínskymi skenmi, kde je veľká časť obrázku pozadie, a objekty tvoria len málu časť. Podobný prípad je to aj v dátovej sade so skenmi bočníc pneumatík, kde symboly tvoria len málu časť zo skenu. Je definované ako:

$$\text{Dice} = \frac{2|A \cap B|}{|A| + |B|} \tag{4.3}$$

Panoptic Quality

Panoptic Quality (PQ) je metrika navrhnutá pre hodnotenie panoptickej segmentácie. PQ spája detekciu objektov a kvalitu segmentácie do jedného čísla, preto je ju vhodné použiť, keďže pri segmentovaní symbolov na skenoch pneumatík je dôležitá aj detekcia objektov, aj kvalita masiek. Táto metrika hodnotí, ako dobre model rozpoznal jednotlivé objekty

a zároveň, ako presne ich segmentoval. Vypočíta sa ako:

$$\mathrm{PQ} = \frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP| + \frac{1}{2}(|FP| + |FN|)} \tag{4.4}$$

kde TP je množina správne detekovaných a segmentovaných inštancií, FP sú falošné pozitíva a FN sú falošné negatíva.

Falošne pozitívne pixely sú tie, ktoré model označil ako objekt, ale v referenčnej maske patria inej triede. Falošne negatívne pixely sú pixely, ktoré v referenčnej maske patria objektu, ale model ich nerozpoznal.

Average Precision

Average Precision (AP) je štandardná metrika používaná hlavne pri detekcii objektov, ale používa sa aj pri inštančnej segmentácii. Dá sa počítať buď pomocou ohraničujúcich boxov inštancií objektov, alebo pomocou masiek objektov. AP vyhodnocuje ako plocha pod krivkou Precision-Recall pre jednotlivé triedy.

Precision (presnosť) je definovaná ako podiel správne identifikovaných pozitívnych príkladov k celkovému počtu predikovaných pozitívnych príkladov:

$$\mathrm{Precision} = \frac{TP}{TP + FP} \tag{4.5}$$

kde TP sú pravdivé pozitíva a FP sú falošné pozitíva.

Recall (citlivosť) je definovaná ako podiel správne identifikovaných pozitívnych príkladov k celkovému počtu skutočných pozitívnych príkladov:

$$\mathrm{Recall} = \frac{TP}{TP + FN} \tag{4.6}$$

kde FN sú falošné negatíva.

Výsledné AP sa často udáva pri rôznych prahoch IoU (napr. AP@0.5, AP@0.75, AP@0.95) a následne sa môže spriemerovať:

$$\mathrm{mAP} = \frac{1}{N} \sum_{i=1}^{N} \mathrm{AP}_i \tag{4.7}$$

kde N je počet tried. Vyššie hodnoty AP indikujú, že model dobre identifikuje a segmentuje jednotlivé objekty.

Kapitola 5

Prvotné experimenty so self-supervised modelmi

V tejto kapitole sú zaznamenané prvotné experimenty s modelmi, na základe ktorých sa uberalo tvorenie riešenia problému pomalej adaptácie na nové symboly.

5.1 Využitie metódy few-shot a zero-shot

Few-shot učenie by mohlo byť vhodné pre dotrénovanie nových symbolov hneď z niekoľkých dôvodov:

- firma väčšinou nedisponuje rozsiahlymi dátami nových znakov a symbolov na pneumatikách,
- keďže vzorka nových dát je veľmi malá, dotrénovanie na týchto dátach by prebehlo rýchlo,
- ak by sa použil prístup zero-shot, ušetril by sa čas anotovaním dát,
- rozličné znaky a symboly na pneumatikách majú podobné vizuálne príznaky oproti rozdielnym triedam v klasických datasetoch pre počítačové videnie (napr. COCO [10], ImageNet [21]).

Metódu few-shot učenia som skúšal na modeloch hlbokého učenia OpenSeeD [24] a Personalized Segment Anything Model [25].

OpenSeeD

OpenSeeD je jednoduchý framework pre open-vocabulary segmentáciu a detekciu, ktorý dosahuje state-of-the-art výsledky na rôznych datasetoch pre open-vocabulary inštančnú a panoptickú segmentáciu.

Open-vocabulary semantická segmentácia je technika v počítačovom videní, ktorá umožňuje modelom rozpoznávať a segmentovať objekty, aj keď neboli zahrnuté v trénovacej sade. To znamená, že model dokáže identifikovať a správne označiť objekty, ktoré nepoznal počas trénovania, na základe ich textového popisu alebo iných reprezentácií [9].

Najlepšie výsledky, ktoré sa mi podarilo dosiahnuť bez dotrénovania, boli s modelom, ktorý bol pred trénovaný na datasete ADE20k [27, 26] a ako backbone používal vizuálny

transformer Swin-L [11]. Ukážka je zobrazená na obrázku 5.1. Na inferenciu obrázkov s neznámymi triedami je potrebné poskytnúť tieto nové neznáme triedy. Triedy a ich rôzne kombinácie, ktoré som skúšal, sú tieto:

> "number 5", "letter o", "character", "letter", "text", "word", "number", "digit", "symbol", "letter-like", "logo".

Triedy v typických datasetoch pre počítačové videnie majú príliš rozličné príznaky oproti znakom a symbolom na pneumatike a zároveň ich textový popis modelom hlbokého učenia nepridáva signifikantné informácie, napríklad o tvare alebo podobnosti s inými triedami. Preto dosiahnuté výsledky bez dotrénovania na datasete so skenmi bočníc pneumatík sú veľmi nedostatočné.

Obr. 5.1: OpenSeeD, výsledky dosiahnuté na pred trénovanom modeli.

PerSAM

Personalized Segment Anything Model je prístup trénovania Segment Anything Modelu (SAM) [8], ktorému stačí jeden obrázok a korešpondujúca referenčná maska na to, aby dokázal vysegmentovať vizuálne koncepty z tohto obrázka aj z iných obrázkov a videí. Taktiež predstavujú variantu s dotrénovaním (PerSAM-F), ktorá dosahuje lepšie výsledky. Táto varianta zmrazí všetky váhy v modeli SAM okrem dvoch parametrov, ktoré sa dotrénujú do 10 sekúnd. Výsledky z tohto experimentu sú na obrázku 5.2.

Kľúčovým prvkom, ktorý umožňuje učenie na takej malej vzorke dát, je väčšinou práve prepojenie medzi vizuálnymi príznakmi tried a jazykovými informáciami. Tento prístup sa preto v riešeniach, ktoré som skúšal, neosvedčil, keďže slovné popisy znakov nedávajú modelom hlbokého učenia dostatočné informácie na prepojenie s vizuálnymi informáciami. Navyše, riešenie pre firmu nekladie nárok na rozpoznávanie tried bez učenia, keďže segmentácia symbolov nepracuje v dynamickom prostredí, kde by pribúdali rôzne neznáme triedy za behu. Z týchto dôvodov som few-shot učenie vo finálnom riešení nepoužil.

Obr. 5.2: Ukážka výsledku po 10-sekundovom dotrénovaní znaku "4"pomocou PerSAM.

5.2 Dotrénovanie modelu na detekciu a klasifikáciu symbolov a následnú extrakciu masiek pomocou segmentačného modelu

Návrh tohto riešenia spočíva v kombinácii dvoch modelov — jedného na detekciu a klasifikáciu symbolov a druhého na ich segmentáciu. Prvý model deteguje symboly v obraze a určí ich triedu, pričom výstupom je ich poloha (napríklad vo forme ohraničujúcich boxov) a príslušná trieda. Tento výstup následne slúži ako vstup pre segmentačný model, ktorý vytvorí presnú masku symbolu.

Konkrétne bol na detekciu a klasifikáciu symbolov použitý model YOLOv11 (You Only Look Once, verzia 11) [6], a na následnú segmentáciu model SAM2 (Segment Anything Model 2) [18]. YOLO pracuje na princípe rozdelenia vstupného obrazu na mriežku a pre každú jej časť predikuje ohraničujúce boxy a pravdepodobnosti príslušnosti k jednotlivým triedam. SAM je univerzálny segmentačný model vyvinutý spoločnosťou Meta AI, ktorý dokáže segmentovať akýkoľvek objekt na základe vstupného promptu — napríklad bodu, rámčeka alebo masky. SAM2 vylepšuje pôvodnú verziu o lepšiu generalizáciu a vyššiu presnosť aj pri komplikovaných a neštandardných tvaroch objektov.

Tento návrh vznikol na základe pozorovania, že model SAM2 dokázal vytvárať kvalitné masky aj na objektoch, na ktorých nebol trénovaný, ak mal k dispozícii správny prompt (bod alebo box) označujúci, kde sa objekt nachádza. Detekčné a klasifikačné modely sú vo všeobecnosti menej náročné na trénovanie aj na výpočtové zdroje. Model YOLOv11 bol natrénovaný na malej podmnožine dát (128 snímok) za približne 15 minút. Už za takýto krátky čas sa model dokázal naučiť detegovať a klasifikovať väčšinu symbolov. Model SAM2 nebol dotrénovaný vôbec — a napriek tomu dokázal vo väčšine prípadov vytvoriť kvalitné segmentácie.

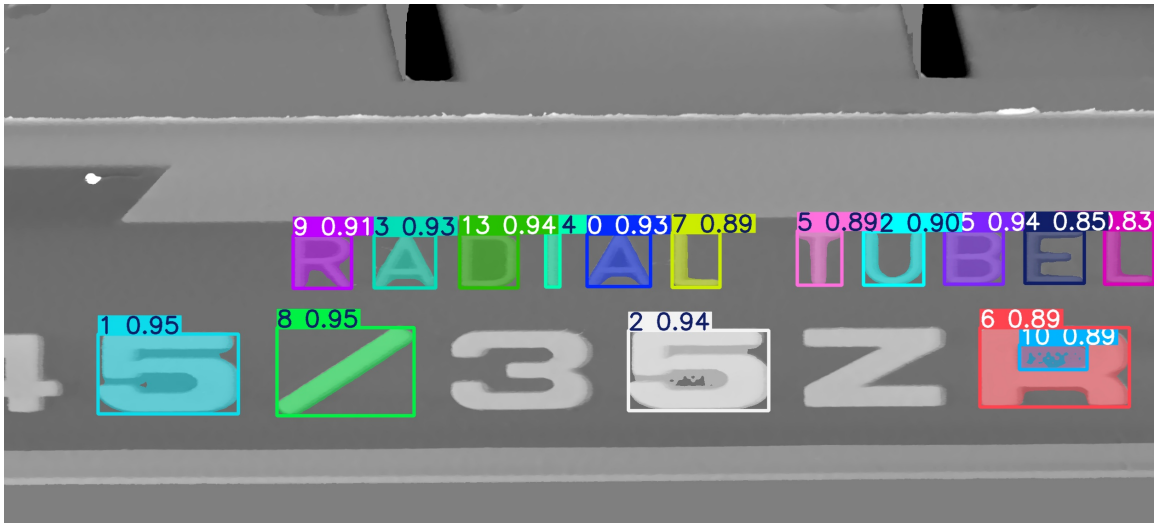
Ukážka výstupov kombinácie modelov YOLOv11 a SAM2 je zobrazená na obrázku 5.3.

Niektoré symboly však zostávali problematické a ich presné segmentovanie by si vyžadovalo dotrénovanie aj modelu SAM2, čo by už znižovalo výhodnosť tohto riešenia. Z tohoto dôvodu som ďalej s týmto riešením nepokračoval.

Zhrnutie úvodných experimentov

Na základe úvodných experimentov som dospel k týmto zisteniam:

- *Few-shot* a *zero-shot* metódy sa ukázali ako nevhodné pre túto úlohu. Hoci sú tieto prístupy efektívne v dynamických prostrediach vyžadujúcich rýchlu adaptáciu, v prípade symbolov na pneumatikách zlyhávajú. Hlavným dôvodom je, že slovné popisy



Obr. 5.3: Výstup kombinácie dotrénovaného modelu YOLOv11 a nedotrénovaného modelu SAM2.

symbolov neposkytujú modelom dostatočné informácie na vytvorenie kvalitných vizuálnych asociácií. Navyše, pre našu aplikáciu nie je prioritou rozpoznávanie neznámych tried za behu, ale presná a kvalitná segmentácia.

- Kombinácia detekčného a segmentačného modelu (YOLOv11 + SAM2) preukázala sľubné výsledky pri kratšom čase trénovania. Zatiaľ čo detekčný model sa dokázal rýchlo naučiť lokalizovať symboly, niektoré zložitejšie tvary by vyžadovali dodatočné dotrénovanie aj segmentačného modelu SAM2. Implementácia takéhoto riešenia stráca výhodu trénovania iba detekčného modelu, ktorý je menej náročný na zdroje.

Na základe týchto zistení som vo finálnom riešení nepoužil ani jednu z uvedených metód. Namiesto toho som sa rozhodol zamerať na efektívne dotrénovanie jedného zo state-of-the-art modelov na segmentáciu – OneFormer.

Kapitola 6

Návrh finálneho riešenia problému pomalej adaptácie na nové symboly

Na úlohu segmentácie symbolov na bočniciach pneumatík som zvolil architektúra OneFormer. Dôvody prečo som zvolil túto architektúru sú nasledovné:

- dosahuje kvalitné výsledky v oblasti panoptickej a inštančnej segmentácie na viacerých verejných dátových sadách,
- vďaka univerzálnosti siete OneFormer bolo možné experimentovať s rôznymi typmi segmentácii.

Model OneFormer [5] slúži ako základ pre experimentovanie s metódami efektívneho dolaďovania (fine-tuning) a adaptácie na nové symboly.

Na základe úvodných experimentov sa metódy *zero-shot* a *few-shot* učenia neosvedčili. Tieto prístupy sú vhodnejšie pre dynamické prostredia, ako napríklad robotika, kde je prioritou flexibilita a okamžitá adaptácia na nové situácie. V prípade segmentácie symbolov na bočniciach pneumatík sú však kľúčovými požiadavkami presnosť a kvalita segmentačných masiek, ktoré majú vyššiu prioritu než okamžitá schopnosť rozpoznávať nové symboly.

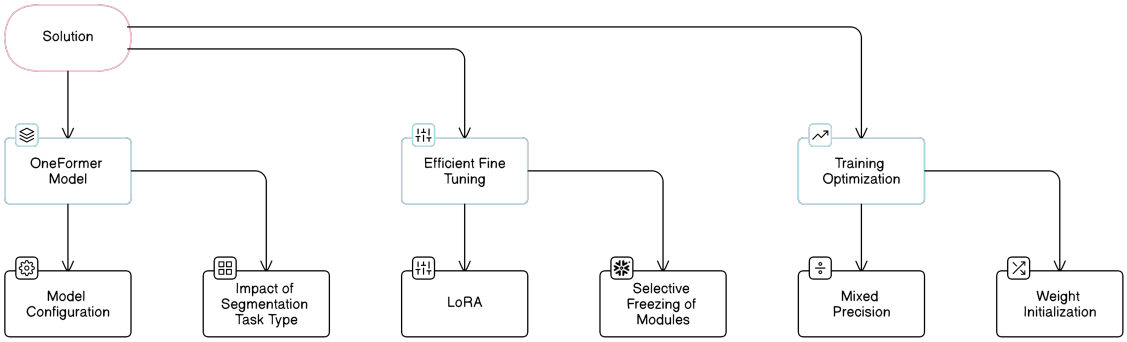
Preto boli vybrané metódy tréningu so zmiešanou presnosťou a Low-Rank Adaptation (LoRA), ktoré výrazne znižujú výpočtové nároky a čas potrebný na dolaďovanie modelu, pričom zachovávajú vysokú kvalitu segmentácie.

Hlavné komponenty návrhu finálneho riešenia zahŕňajú:

- **OneFormer model** – univerzálny segmentačný model založený na vizuálnych transformeroch, ktorý dosahuje state-of-the-art výsledky naprieč rôznymi typmi segmentácie (inštančná, sémantická, panoptická). Funguje ako primárny model pre segmentáciu symbolov na bočniciach pneumatík.
- **Formulácia úlohy** — porovnanie inštančnej, sémantickej a panoptickej segmentácie na určenie najvhodnejšieho prístupu vzhľadom na našu dátovú sadu.
- **Inicializácia váh** — testovanie rôznych stratégií (napr. Xavier [16, 17], Kaiming [14, 15]) na zlepšenie konvergencie a kvality segmentácie.
- **Metóda LoRA [4]** – umožňuje efektívne doladiť len vybrané časti modelu — konkrétne lineárne vrstvy — pomocou nízkorozmerných maticových dekompozícií. Vzhľadom na to, že lineárne vrstvy sú súčasťou takmer každej segmentačnej siete, je táto metóda široko aplikovateľná aj mimo architektúry siete OneFormer.

- **Zmrazenie modulov** — selektívne zmrazenie častí modelu (napr. backbone), čím sa zrýchli adaptácia a predíde katastrofickému zabúdaniu.
- **Trénovanie so zmiešanou presnosťou** – univerzálna technika umožňujúca zrýchliť učenie modelu bez negatívneho dopadu na presnosť výsledkov. Z tohto dôvodu je výhodné ho nasadiť vo všetkých prípadoch, kde je to technicky možné.

Tieto prístupy sú pre prehľadnosť znázornené v diagrame 6.1.



Obr. 6.1: Schéma znázorňujúca komponenty navrhnutého riešenia segmentácie symbolov: model OneFormer, metódy efektívneho doladenia (LoRA a selektívne zmrazenie modulov) a techniky optimalizácie trénovania, ktoré spolu umožňujú rýchlu adaptáciu na nové symboly.

Cieľom navrhovaného riešenia je zabezpečiť univerzálnosť a flexibilitu, aby bolo možné zvolené metódy aplikovať nielen na sieť OneFormer, ale aj na iné modely s podobnou architektúrou.

Augmentácia dát

Augmentácia dát je technika používaná pri trénovaní modelov počítačového videnia, ktorá umelo rozširuje trénovaciu dátovú sadu pomocou náhodných transformácií obrázkov. Medzi bežné augmentácie patrí zmena veľkosti, orezanie, otočenie, prevrátenie, úprava jasu, kontrastu, rozmazanie, pridanie šumu alebo geometrické deformácie.

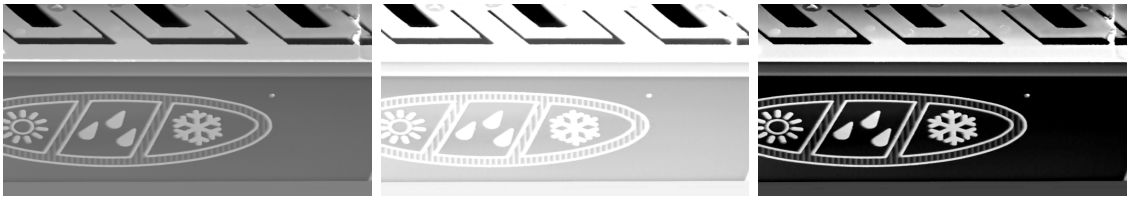
Pri použití augmentácií na dátovú sadu so skenmi bočníc pneumatík nie je vhodné používať veľmi „agresívne“ augmentácie. Dôvody, prečo niektoré augmentácie nie sú vhodné:

- Písmená aj symboly majú pevný tvar a na skenoch sa nikdy nenachádzajú otočené. Napríklad, ak by sa použila rotácia o 90 stupňov, z písmena `U` by sa mohlo stať písmeno `C` a naopak. Pri horizontálnom prevrátení by sa z písmena `p` mohlo stať písmeno `d`.
- Skeny sú šedotónové snímky, takže farebné augmentácie (napr. zmena odtieňa, saturácie) sa nedajú efektívne využiť.
- Orezanie nie je vhodné, keďže cieľom je, aby model rozpoznával iba symboly, ktoré na skene zaberajú plochu väčšiu ako určitý prah. Orezaním by sa mohol tento prah zmeniť.
- Rôzne geometrické transformácie (napríklad skosenie), môžu spôsobiť, deformáciu znakov do nečitateľnej alebo nesprávnej podoby, čo vedie k nesprávnemu učeniu modelu.

Aj napriek tomu, že agresívne augmentácie môžu byť nevhodné, určité mierne transformácie môžu modelu pomôcť generalizovať a lepšie rozonávať poškodené skeny. Medzi takéto augmentácie patria napríklad:

- **Pridanie mierneho šumu** -– napríklad Gaussov šum môže simulovať malé nedokonalosti v skene alebo rôzne artefakty,
- **Mierne rozmazanie (angl. „blur“)** – môže pomôcť modelu zvládať rozostrené skeny,
- **Úprava kontrastu alebo jasu** — keďže skeny sú hĺbkové snímky pneumatiky, zmena jasu alebo kontrastu znamená, že symboly by boli na pneumatike v inej výške
- **Horizontálne prevrátenie (zrkadlenie)** — môže byt vhodné, ak ide o špeciálne symboly

Experimentovanie s umelou dátovou sadou (viď. ďalšia podkapitola 6) ukázalo na dôležitosť kontrastu symbolov a pozadia. To viedlo k experimentovaniu s kontrastom, ktorý bol aplikovaný na každú snímku a v rovnakej miere. Väčší kontrast medzi pozadím a symbolmi by mohol modelu zjednodušiť rozoznávanie symbolov. Experimentoval som s rôznymi hodnotami parametrov jasu a kontrastu, pričom sa ukázalo, že najlepšie výsledky priniesla kombinácia zníženého jasu a výrazne zvýšeného kontrastu. Tieto hodnoty boli aplikované rovnomerne na všetky snímky. No oproti pôvodným snímkam bez úpravy jasu a kontrastu dosahovala tato možnosť nanajvýš rovnaké výsledky (-0.6% Dice Score a -0.1% PQ oproti neupraveným snímkam). Ukážka niektorých experimentálnych snímok je na obrázku 6.2



Obr. 6.2: Pôvodná snímka a snímky s upraveným jasom a kontrastom.

Použitie výrazných augmentácií dát pri experimentovaní malo za následok výrazný pokles modelu v učení sa. Použitie miernejších augmentácii nepreukázalo zlepšenie v učení sa modelu a preto neboli použité.

Umelo generovaná dátová sada

Ako bolo spomínané v kapitole 3, rozloženie tried v dátovej sade je veľmi nerovnomerné (nerovnomernosť tried je zobrazená na histograme 3.4), z čoho vyplýva, že niektoré triedy sa model naučí dobre, no niektoré skoro vôbec, pretože s nimi príde do styku počas jednej epochy málo krát. Tento problém som riešil pomocou umelo generovaných snímok.

Ako prvé sa pomocou anotácií a masiek extrahujú z pôvodných snímok všetky symboly. Tie sa uložia do slovníka. Ukážka zopár extrahovaných symbolov je na obrázku 6.3. Tieto symboly nie sú dokonalé, keďže masky presne nekopírujú tvar symbolov – oproti originálnym symbolom majú ostrejšie hrany.

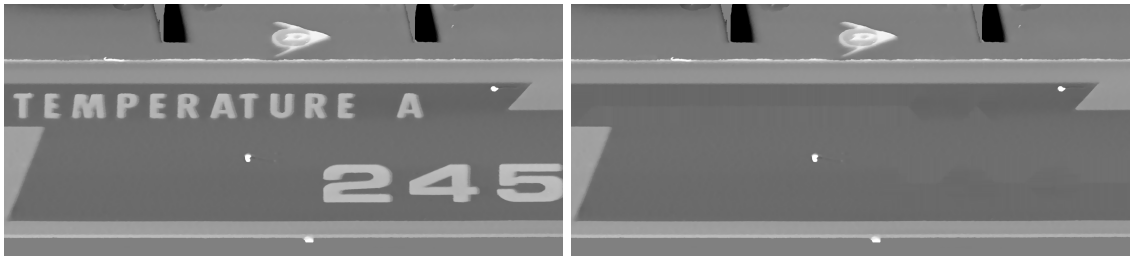
Následne sa vyberie pozadie. Pozadie je vhodné zvoliť také, ktoré je komplikovanejšie a zaujímavejšie, obsahujúce rôzne útvary, pretože v opačnom prípade tieto útvary model

Obr. 6.3: Ukážka extrahovaných symbolov.

niekedy nesprávne rozoznáva ako symboly. Zo vstupného snímku s vizuálne zaujímavým pozadím sa extrahujú (vyrežú) oblasti obsahujúce symboly, pričom tieto vyrezané regióny sa následne rekonštruujú pomocou techniky *inpainting* založenej na Navier–Stokesových rovniciach.

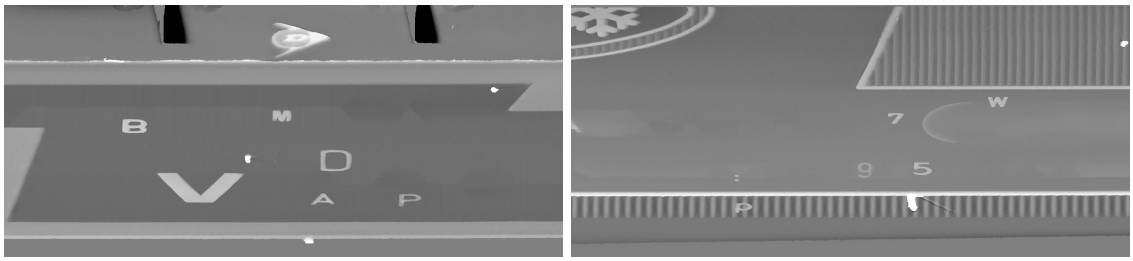
Inpainting je metóda spracovania obrazu, ktorá slúži na dopĺňanie chýbajúcich alebo poškodených častí obrazu tak, aby výsledok pôsobil vizuálne konzistentne s okolím. Varianta metódy *inpainting* založená na Navier–Stokesových rovniciach simuluje šírenie textúr a štruktúr pomocou vektorového poľa, ktoré sa riadi fyzikálnymi princípmi prúdenia kvapalín. Cieľom je, aby boli okraje a farebné prechody v rekonštruovanej oblasti čo najplynulejšie a konzistentné s okolitými štruktúrami.

Ukážka takto separovaného pozadia so zaujímavými a rušivými útvarmi je zobrazená na obrázku 6.4.



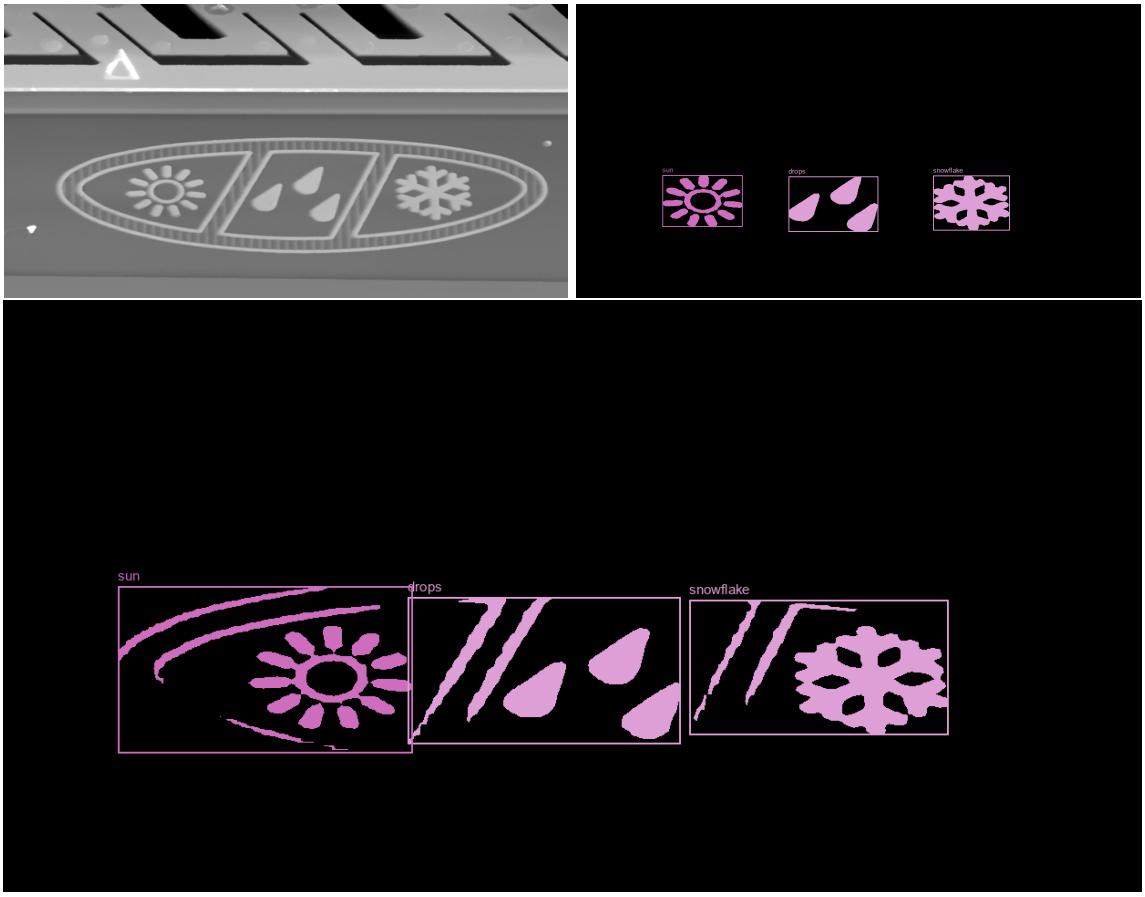
Obr. 6.4: Ukážka upraveného pozadia bez symbolov (vpravo).

Keď už sú pripravené pozadia a slovník so symbolmi, dajú sa generovať umelé snímky. Symboly sa väčšinou nachádzajú na určitej pozícii – viac v strede, kde je medzi nimi a pozadím aj väčší kontrast. Preto ohraničujem oblasť, do ktorej môžu byť symboly vygenerované, pomocou ohraničujúceho obdĺžnika. Na jednotlivé symboly s určitou pravdepodobnosťou aplikujem dve augmentácie: zväčšenie alebo zmenšenie a skosenie. Ukážka vygenerovaných umelých snímok je na obrázku 6.5.



Obr. 6.5: Ukážka umelých snímok generovaných zo symbolov a umelého pozadia.

Model sa na tejto umelej dátovej sade učí len o trochu pomalšie ako na originálnej, no zároveň s niektorými symbolmi vzniká problém, ak majú okolo seba rôzne útvary, ktoré nepatria žiadnej triede, no model ich aj tak klasifikuje ako triedy. Ako príklad uvádzam špeciálne symboly „sun“, „snowflake“ a „drops“, okolo ktorých sú na pneumatikách oválne útvary. Model tieto ovály zahŕňa k maskám týchto tried, ako je možné vidieť dole na obrázku 6.6.



Obr. 6.6: Ukážka správnej segmentácie (hore) a problému pri trénovaní na umelej dátovej sade (dole), kde model nesprávne klasifikuje niektoré útvary ako súčasť masiek.

Vďaka umelo generovanej dátovej sade sa dá jednoducho dosiahnuť rovnomerne rozloženie tried v dátovej sade. Okrem toho, veľa snímkov v pôvodnej dátovej sade je prázdnych – bez žiadnych symbolov, a veľa snímkov sa opakuje, keďže niektoré nápisy na pneumatikách sú rovnaké. Preto som vygeneroval menšiu dátovú sadu iba s 574 snímkami, ktorá má rovnomerne rozložené všetky symboly z pôvodnej dátovej sady a 10 rôznych pozadí. Na jednej snímke je 3 až 7 symbolov a každý symbol je obsiahnutý v dátovej sade 50 krát.

Kapitola 7

Implementácia riešenia

Riešenie je implementované v programovacom jazyku Python. Na implementáciu som využil framework HuggingFace, kde, okrem iných transformerov, bol implementovaný aj OneFormer. HuggingFace je postavený na frameworku PyTorch, ktorý som taktiež využil na modifikáciu modelu, efektívnu prácu s dátovou sadou, implementáciu trénovacieho cyklu a evaluáciu. Okrem toho som použil knižnicu NumPy na prácu s maticami, knižnice Pillow a OpenCV na spracovanie obrázkov z dátovej sady a vizualizácie, Pandas na prácu s anotáciami vo formáte CSV a knižnicu TorchMetrics na evaluáciu modelu pomocou rôznych metrík.

Na získavanie informácií počas trénovania a sledovanie správania sa modelu počas trénovania som využil platformu Weights & Biases. Pomocou logovania som do nej zaznamenával informácie o chybovej funkcii, dĺžke trénovania, metrikách a výsledkoch segmentácie. Taktiež som si na tejto platforme uchovával informácie o použitej konfigurácii modelu, čo mi umožnilo jednoduchšie sledovať vhodné konfigurácie.

7.1 Transformácia dátovej sady

Pri riešení boli použité rôzne modely hlbokého učenia a frameworky, a väčšina z nich požadovala dáta v kompatibilnom formáte. Taktiež som experimentoval s rôznymi typmi segmentácie, kde napríklad semantická segmentácia vyžaduje iný formát dát ako inštančná segmentácia.

Ako základ bola použitá dátová sada rozdelená na podsnímky pomocou poskytnutých skriptov. Dátová sada má pevnú štruktúru — rozdelenie na trénovaciu a testovaciu časť, pričom v každej sa nachádzajú priečinky reprezentujúce jednotlivé skeny bočníc pneumatík. Každý z týchto priečinkov obsahuje podsnímky, k nim prislúchajúce masky a anotácie.

Pre experimenty boli vytvorené dve verzie dátovej sady. Prvá reprezentuje pôvodné dáta a obsahuje všetky triedy okrem špecificky vyfiltrovaných. Druhá obsahuje len podsnímky s vyfiltrovanými triedami a simuluje nové dáta, na ktoré sa má model dodatočne adaptovať.

`HeightMapDatabase`

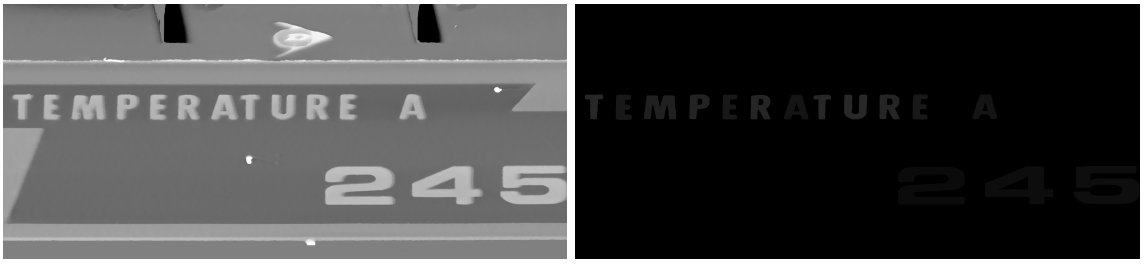
Trieda `HeightMapDatabase` predstavuje základný nástroj na prácu s dátovou sadou. Využíva statickú triedu `MeiDatasetDirectoryProcessor`, ktorá slúži ako wrapper na callback funkcie aplikované na každú snímku v dátovej sade podľa definovanej štruktúry.

`HeightMapDatabase` poskytuje metódy na získavanie informácií o dátovej sade, frekvencii výskytu tried, spracovanie pôvodných anotácií vo formáte CSV, mapovanie identifikátorov na názvy tried a ďalšie pomocné funkcie.

Na inicializáciu je potrebné zadať cestu k pôvodnej, už rozdelenej dátovej sade. Trieda predpokladá, že štruktúra zodpovedá výstupu poskytnutých skriptov.

Dátová sada pre semantickú segmentáciu

Formát pre semantickú segmentáciu vyžaduje ku každej snímke zodpovedajúcu šedotónovú masku, kde každý pixel reprezentuje triedu podľa svojej intenzity. Hodnota `0` zodpovedá pozadiu, ďalšie hodnoty reprezentujú jednotlivé triedy, napr. hodnota `20` môže reprezentovať veľké písmeno `A`. Ukážka je na obrázku 7.1. Identifikátory tried musia tvoriť postupnosť inkrementujúcu sa o jeden so začiatkom v nule, preto sú niektoré symboly tak tmavé.



Obr. 7.1: Ukážka snímky a masky z dátovej sady pre semantickú segmentáciu modelu OneFormer.

`HeightMapSemanticDataset`

Trieda `HeightMapSemanticDataset` implementuje verziu dátovej sady určenú pre semantickú segmentáciu a dedí z triedy `HeightMapDatabase`. Oproti rodičovskej triede umožňuje pri inicializácii špecifikovať triedy, ktoré majú byť vyfiltrované do samostatnej dátovej sady, čím sa simuluje príchod nových dát.

Masky pre semantickú segmentáciu sa generujú načítaním anotácií vo formáte CSV, z ktorých sa získavajú informácie o polohe, veľkosti a triede symbolov na snímke. Na základe týchto údajov sa vytvára nová maska, kde každému pixelu sa priradí hodnota podľa triedy z pôvodného CSV, pričom sa mapujú unikódové identifikátory symbolov na súvislú číselnú postupnosť začínajúcu nulou.

Formát YOLO

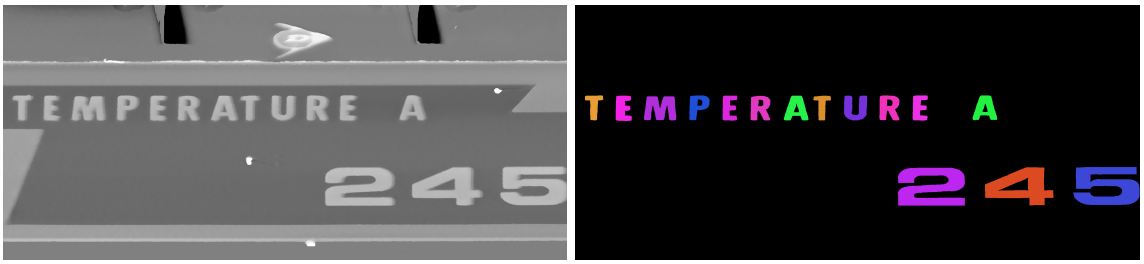
Formát dátovej sady YOLO bol použitý pri experimentovaní s trénovaním klasifikátora YOLO. Tento formát má namiesto masiek k snímkam textové anotácie. Ku každej snímke existuje textový súbor (formát *.txt). V tomto textovom súbore jeden riadok predstavuje jeden objekt. Riadok začína identifikátorom triedy a nasledujú ohraničujúce súradnice vo formáte: $x_1 y_1 x_2 y_2 \ldots x_n y_n$. Tieto súradnice musia byť normalizované do intervalu 0 až 1. Každý objekt musí mať minimálne 3 body, teda 6 súradníc. Dátová sada v YOLO formáte taktiež vyžaduje jeden YAML súbor s metadátami o dátovej sade.

Formát COCO

Formát COCO patrí medzi najpoužívanejšie formáty pre panoptickú segmentáciu. Použitý bol pri trénovaní modelu OneFormer (4.1). Pre každý obrázok je potrebný JSON súbor s anotáciami jednotlivých objektov a referenčná maska. Anotácie obsahujú nasledovné položky:

- `id`: jedinečný identifikátor inštancie objektu,
- `category_id`: identifikátor kategórie, do ktorej objekt patrí,
- `bbox`: ohraničujúci box objektu,
- `area`: plocha objektu v pixeloch,
- `iscrowd`: binárna hodnota, určujúca či ide o prekrývajúce sa objekty.

Referenčná maska má oproti semantickej segmentácii odlišnú funkciu — hodnoty pixelov reprezentujú jednotlivé inštancie, nie triedy. Pre lepšiu vizuálnu kontrolu dát boli pri generovaní masiek jednotlivým inštanciám priradené odtiene farieb na základe kategórie. Ukážka je na obrázku 7.2.



Obr. 7.2: Ukážka snímky a masky z dátovej sady pre panoptickú segmentáciu. Inštancie patriace do rovnakej triedy majú podobné farby (napríklad trieda „A“)

`HeightMapInstanceDataset`

Trieda `HeightMapInstanceDataset` implementuje inštančnú dátovú sadu vo formáte COCO a dedí z triedy `HeightMapDatabase`. Oproti rodičovskej triede spracováva anotácie vo formáte JSON, kde sú masky uložené pomocou RLE (Run-Length Encoding). Tento formát nahrádza binárne masky sekvenciami počtov opakujúcich sa hodnôt, čím výrazne šetrí veľkosť anotačných súborov a zodpovedá požiadavkám COCO štruktúry.

`HeightMapPanopticDataset`

Trieda `HeightMapPanopticDataset` rozširuje spracovanie pre panoptickú segmentáciu vo formáte COCO. Masky sa generujú podobne ako pri semantickej segmentácii, no každá trieda dostáva základnú farbu (v HSV priestore) a jednotlivé inštancie tejto triedy jej variácie. Tento prístup zvyšuje prehľadnosť pri vizualizácii a zjednodušuje kontrolu dátovej sady.

7.2 Riešenie založené na modeli OneFormer

Ako model som použil pred trénovaný model z frameworku HuggingFace, kde bol OneFormer aj implementovany.

Dátová sada pre sieť OneFormer

Dátová sada bola implementovaná pomocou knižnice `datasets` z frameworku HuggingFace. Vlastná trieda dátovej sady dedí z triedy `Dataset` a pri inicializácii prijíma generátor snímok, odvodený z triedy `GeneratorBasedBuilder`.

Pri získavaní prvku z dátovej sady (metóda `__getitem__`) sa jednotlivé položky zakódujú do formátu požadovaného sieťou OneFormer. Tento formát obsahuje:

- `pixel_values` — 2D pole hodnôt pôvodnej snímky,
- `pixel_mask` — maska, kde každá hodnota reprezentuje inštanciu objektu,
- `class_labels` — pole obsahujúce kategórie všetkých inštancií na snímke,
- `mask_labels` — pole binárnych masiek pre jednotlivé inštancie, zoradené podľa `class_labels`,
- `text_inputs` — zakódované textové vstupy,
- `task_inputs` — špecifikovaný *task token*,
- `inst_cat_map` — mapovanie medzi inštanciami a kategóriami,
- `segments_info` — informácie o jednotlivých objektoch na snímke: identifikátor inštancie, identifikátor kategórie, plocha, ohraničujúci box a údaj `iscrowd`. V prípade snímok bočníc pneumatík je `iscrowd` vždy `False`, keďže sa symboly neprekrývajú.

Pri tvorbe dávok (angl. batches) môže nastať problém, ak majú jednotlivé snímky rozdielny počet kategórií, čo spôsobí nezhodu tvaru tenzorov `class_labels` a `mask_labels`. Tento problém je riešený vyplnením chýbajúcich pozícií nulami tak, aby mali všetky položky v dávke rovnaký tvar podľa celkového počtu kategórií v dátovej sade.

Spracovanie vstupných a výstupných dát siete OneFormer

Trieda `OneFormerDataPipeline` zabezpečuje komplexnú prácu so vstupnými a výstupnými dátami siete OneFormer. Slúži ako centrálny bod pre rôzne transformácie, vizualizácie a extrakcie požadovaných formátov dát.

Pri inicializácii prijíma vstupné dáta modelu a voliteľné parametre pôvodnej veľkosti snímky. Interne ukladá a spracováva vstupy z datovej sady.

Trieda `OneFormerDataPipeline` poskytuje sadu metód pre prácu s dátami:

- Metódy pre presun dát medzi zariadeniami (CPU/GPU),
- Príprava post-processingu výstupov modelu,
- Ukladanie výsledkov segmentácie pomocou,

- Rozsiahle možnosti transformácie veľkosti (resize) snímok a masiek s rešpektovaním špecifických vlastností rôznych typov dát,
- Extrakcia vizualizovateľných snímok, referenčných masiek a sémantických máp,
- Generovanie normalizovaných segmentácií,
- Výpočet ohraničujúcich boxov (bounding boxes) z predikovaných segmentov aj referenčných masiek,
- Tvorba máp prepájajúcich inštancie a kategórie objektov pre predikované aj referenčne dáta.

Na evaluáciu poskytuje táto trieda metódy, ktoré formátujú dáta do podoby vhodnej pre výpočet metrík ako *Average Precision*, *Dice Score* a *Panoptic Quality*. Tieto metódy vytvárajú štruktúrované slovníky obsahujúce rôzne reprezentácie dát — ohraničujúce boxy, skóre, kategórie a binárne masky.

Dôležitou vlastnosťou je schopnosť práce s dávkami (batches) snímok, kde všetky operácie sú implementované tak, aby efektívne spracovali viacero snímok naraz.

Trénovací cyklus

Tréning a vyhodnocovanie siete OneFormer je realizované pomocou triedy `OneFormerTrainer`, ktorá zabezpečuje celý proces učenia, validácie a ukladania modelu počas trénovania.

Pri inicializácii triedy `OneFormerTrainer` je potrebné poskytnúť všetky potrebné komponenty pre trénovanie: model, optimalizačnú funkciu, trénovacie a validačné dávkovače (anlg. dataloader), procesor vstupov, metriky, mapovanie identifikátorov tried, zariadenie (CPU/GPU), počet epoch, voľba použitia zmiešanej presnosti (mixed precision), stratégia vyhodnocovania (po epochách alebo po krokoch), interval ukladania modelu a typ úlohy (panoptická, semantická alebo inštančná segmentácia).

Vyhodnocovanie modelu počas trénovacieho cyklu je implementované v metóde, ktorá akumuluje a priemeruje výsledky naprieč dávkami a pripraví vizualizácie segmentačných výsledkov pre ďalšiu analýzu.

Model

Architektúra modelu aj s pred trénovanými váhami je dostupná vo frameworku HuggingFace. Použil som túto implementáciu a pri experimentovaní som model mierne modifikoval. Experimenty s modelom a modifikácie sú opísané v kapitole 8.

Kapitola 8

Experimenty a ich vyhodnotenie

Podstatou úlohy bolo vyriešiť pomalú adaptáciu modelu na nové dáta. Aby som tento problém mohol simulovať, určil som zopár tried, a všetky snímky, ktoré tieto triedy obsahovali, som z dátovej sady odfiltroval. Z týchto odfiltrovaných snímok som vytvoril novú dátovú sadu, ktorá reprezentuje nové dáta. Vybral som špeciálne symboly, keďže model je už naučený na štandardné písmená a číslice a ak by nové dáta obsahovali iné texty, mal by ich byť stále schopný rozpoznať. Jediný problém by mohol nastať, ak by sa font starých a nových písmen a číslic príliš líšil.

Navrhol som štyri experimenty, ktoré mali za cieľ otestovať rôzne aspekty navrhovaného riešenia:

- **Základné konfigurácie modelu** – porovnávanie vplyvu rôznych typov backbone a inicializácií váh na kvalitu segmentácie, s cieľom určiť optimálnu konfiguráciu siete OneFormer pre ďalšie experimenty.
- **Porovnanie typov segmentačných úloh** – overenie vplyvu formulácie úlohy (sémantická, inštančná alebo panoptická segmentácia) na kvalitu získaných segmentačných masiek.
- **Trénovanie na umelo generovanej dátovej sade** – testovanie, či umelo vygenerovaná dátová sada, ktorá rieši nevyvážené rozloženie tried môžu pomôcť zlepšiť schopnosti modelu.
- **Dotrénovanie nových symbolov** – porovnanie rôznych prístupov k adaptácii modelu na nové symboly (dotrénovanie celého modelu, vybraných častí alebo s využitím metódy LoRA) z hľadiska rýchlosti a zachovania schopnosti rozpoznávať pôvodné triedy.

Trénovanie prebiehalo na stroji s GPU s 16 GB VRAM. Dotrénovanie a experimenty prebiehali na systéme s NVIDIA GeForce RTX 4060 Laptop GPU (8 GB VRAM), 48 GB RAM a operačným systémom Windows.

8.1 Inicializácia váh siete OneFormer

Experimentoval som s váhami pred trénovanými na dátových sadách ADE20k [26] a COCO. Tieto pred trénované váhy zahŕňali aj váhy pre Swin [11] transformer backbone rôznych veľkostí – menovite Swin-T (28 miliónov parametrov) a Swin-L (197 miliónov parametrov).

S ohľadom na veľkosť modelu OneFormer sa na 16GB GPU okrem samotného modelu zmestila maximálna veľkosť dávky 4 pri použití Swin-T ako backbone a veľkosť dávky 2 pri použití Swin-L ako backbone s rozmermi obrázkov 512×512 pixelov.

Pri jemnom dolaďovaní modelov sa zvykne „zmraziť“ určitá časť modelu s cieľom urýchliť trénovanie a zároveň zachovať niektoré naučené schopnosti. Toto zmrazenie váh sa dosiahne tak, že v určitých častiach modelu sa deaktivuje počítanie gradientu a tým pádom sa tieto váhy neaktualizujú. Typicky sa zmrazuje backbone alebo iné fundamentálne časti modelu. Experimentovaním som dospel ku konfigurácii, v ktorej je zmrazený celý backbone Swin transformera. Ten poskytuje všeobecné vizuálne príznaky extrahované z iných dátových sád. Tieto príznaky vstupujú do transformerového modulu zodpovedného za enkódovanie vstupov, ktorý je zmrazený tiež. Dôsledkom tohoto sa trénovali hlavne špecifické segmentačné a klasifikačné hlavy modelu.

Pri použití Swin-T ako backbone bolo trénovaných 23,29 milióna parametrov, čo predstavovalo približne 34,1 % z celkového počtu parametrov modelu. Pri použití Swin-L je trénovateľných 23,66 milióna parametrov, čo tvorí približne 10 % zo všetkých parametrov – zvyšok modelu, najmä väčší backbone, je totiž zmrazený. Takýmto prístupom je dosiahnutá efektívna adaptácia modelu na špecifické úlohy s minimálnou potrebou pretrénovania rozsiahlych častí siete.

V tabuľke 8.1 sú zobrazené rozdiely v metrikách pri trénovaní siete OneFormer s využitím danej backbone. Najlepšie výsledky dosahuje model na pred trénovaných dátach z dátovej sady ADE20k [26], a z nich ten väčší backbone, pretože dokáže extrahovať viac vizuálnych informácií zo vstupných obrázkov. Experimenty boli uskutočnené na podmnožine dátovej sady.

Tabuľka 8.1: Tabuľka s rozdielmi výkonnosti siete OneFormer pri použití danej pred trénovanej backbone na metrikách Dice Score a Panoptic Quality.

Model	**Dataset**	**Dice Score**	**Panoptic Quality**
Swin-L	ADE20k	**0.9343**	**0.9283**
Swin-L	COCO	0.8857	0.9049
Swin-L	Cityscapes	0.8131	0.8520
Swin-T	ADE20k	0.8222	0.8464

Na grafe 8.1 je tiež možné pozorovať, že sieť OneFormer pred trénovaná na dátovej sade ADE20k začína s najnižšou chybovou funkciou.

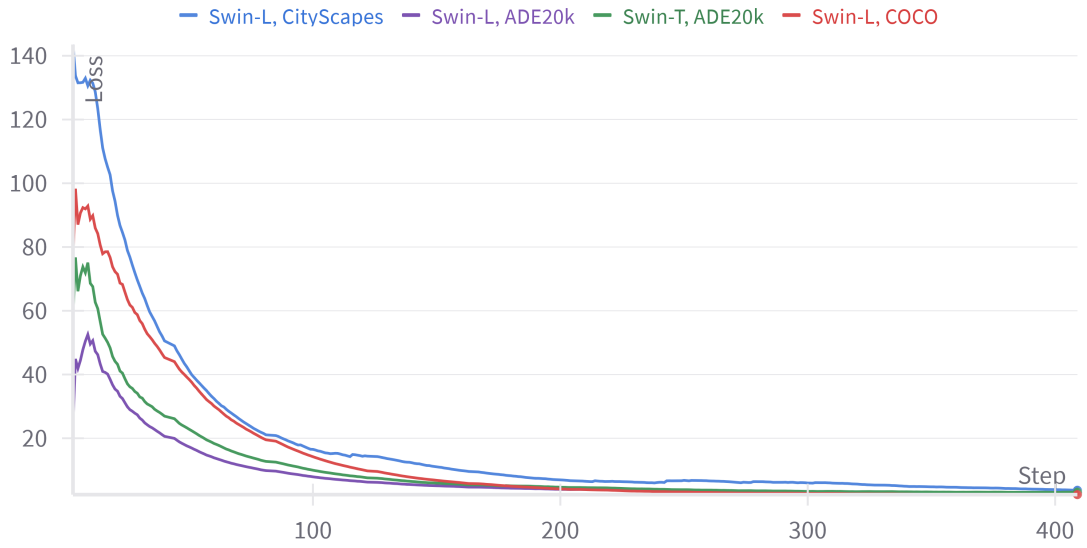
Dátová sada symbolov na bočnici pneumatiky má iný počet tried ako dátové sady, na ktorých bol model pred trénovaný. Segmentačné a klasifikačné hlavy modelu treba adaptovať na nový počet tried. Skúšal som dva prístupy:

1. Prepísal som pôvodné triedy novými – napríklad v prípade dátovej sady ADE20k, ktorá obsahuje 150 tried, som prvých 72 pôvodných tried, prepísal triedami z našej dátovej sady.

2. Nahradil som pôvodné moduly transformera závislé od počtu tried (`decoder` a `queries embedder`) novými.

V prípade prvého prístupu som chcel využiť už naučené informácie v prospech ďalšieho učenia. V druhom prístupe som využil na inicializáciu váh normálne rozdelenie.

Experimentoval som s pred trénovaným modelom na dátovej sade ADE20k a ako backbone som použil aj Swin-L, aj Swin-T. Model bol trénovaný na podmnožine našej dátovej



Obr. 8.1: Chybová funkcia pri trénovaní na podmnožine dátovej sady pre rôzne pred trénované verzie siete OneFormer.

sady. Výsledky metrík Dice Score a Panoptic Quality viacerých experimentov sú zobrazené na grafoch 8.2. V legende označenie *pretrained* značí prvý prístup – teda pôvodné triedy, na ktorých bol model pred trénovaný, boli prepísané triedami z našej dátovej sady.

Experimenty ukázali, že aj pre menší, aj väčší Swin transformer backbone model dosahuje lepšie výsledky, ak sa moduly siete OneFormer, ktoré sú závislé na počte tried, nahradia novými modulmi a ich váhy sa inicializujú normálnym rozdelením.

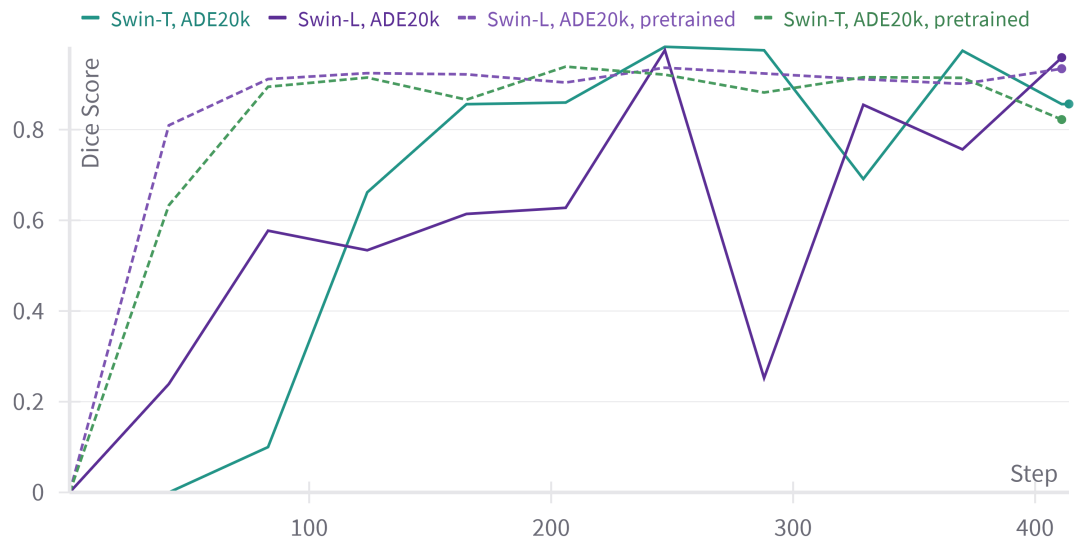
V tabuľke 8.2 sú zobrazené percentuálne rozdiely výkonnosti modelu na metrikách Dice Score a Panoptic Quality pri nahradení pôvodných modulov oproti ponechaniu pred trénovaných váh.

Tabuľka 8.2: Percentuálne zlepšenie metriky Dice Score (DS) a Panoptic Quality (PQ) pre modely s nahradením pôvodných modulov novými oproti pretrénovaným verziám. Zlepšenie je uvedené v zátvorkách.

Backbone	**Váhy**	**DS** [%]	**PQ** [%]
Swin-L	predtrénované	91,80	91,57
Swin-L	novo inicializované	**94,43** (+2,63)	**92,75** (+1,18)
Swin-T	predtrénované	88,09	89,83
Swin-T	novo inicializované	**92,22** (+4,13)	**91,57** (+1,74)

Inicializácia váh je kľúčová pre neurónové siete, pretože výrazne ovplyvňuje rýchlosť konvergencie a stabilitu pri učení. Experimentoval som teda s tromi typickými metódami: normálna distribúcia, Xavier (Glorot) inicializácia a Kaiming (He) inicializácia.

Pri experimentovaní s rôznymi inicializáciami váh segmentačnej hlavy som používal pred trénovaný model na dátovej sade ADE20k, ktorý z predchádzajúcich experimentov vykazoval najlepšie výsledky, s backbone Swin-L. Výsledky experimentov sú zobrazené na grafoch 8.3 a 8.4.



Obr. 8.2: Porovnanie výsledkov metrík Dice Score pri ponechaní pred trénovaných tried a nahradení modulov s novo inicializovanými váhami. Rozdiely pre jednotlivé verzie nie sú až tak výrazné, ale modely s novo inicializovanými váhami podávajú lepší výkon.

V tabuľke 8.3 sú zobrazené rozdiely vo výkonnosti modelu pre metriky Dice Score a Panoptic Quality pri použití rôznych inicializácii váh. Z grafov je vidieť, že model konverguje najrýchlejšie pri inicializácia váh pomocou uniformnej Kaiming inicializácie.

Tabuľka 8.3: Percentuálne rozdiely metriky Dice Score (DS) a Panoptic Quality (PQ) rôznych inicializácií voči inicializácii normálnou distribúciou.

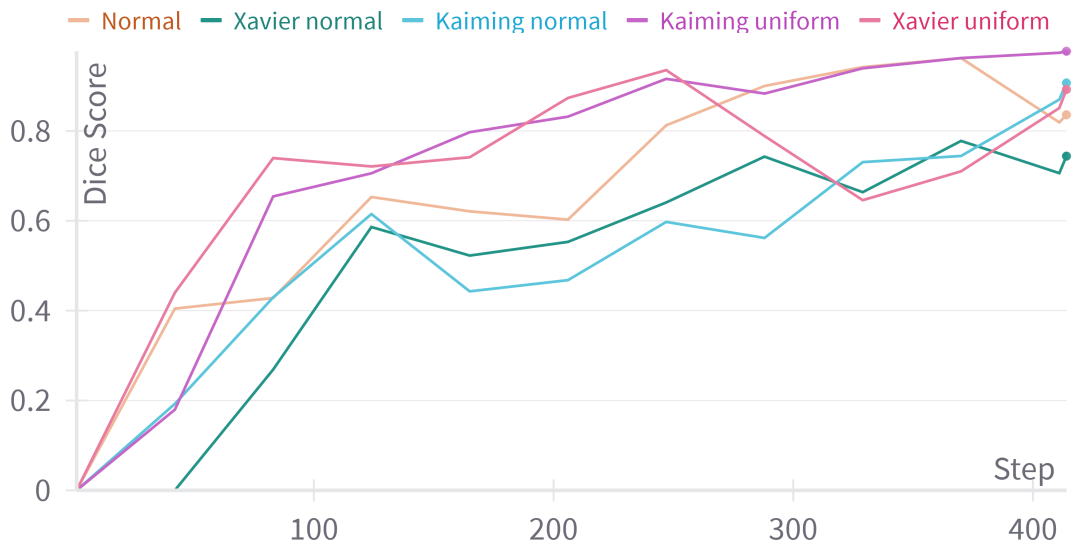
Inicializácia	**Δ DS [%]**	**Δ PQ [%]**
Kaiming uniform [15]	**+13.78%**	**+9.38%**
Kaiming normal [14]	+12.06%	+9.39%
Xavier uniform [17]	+11.28%	+8.31%
Normal (základ)	0.00%	0.00%
Xavier normal [16]	–6.54%	–3.26%

V nasledujúcich experimentov boli podla týchto výsledkov využité na inicializáciu uniformná Kaimingova distribúcia váh a ako backbone bol použitý Swin-L pred trénovaný na dátovej sade ADE20k.

Konfigurácia trénovania základného modelu pre ďalšie experimenty

Pri trénovaní siete OneFormer som použil stroj s GPU s kapacitou 16 GB. Na tomto stroji bol model trénovaný 8 epoch, čo zabralo 11 hodín. Výsledky po natrénovaní na pôvodnej dátovej sade sú zobrazené v tabuľke 8.4.

Tieto výsledky sú porovnateľné s výsledkami pôvodného riešenia 2.1. Porovnanie s pôvodným riešením je v tabuľke 8.5, ktorá zobrazuje podrobné rozdiely v metrike *Average Precision*. Označenia *@50* a *@75* predstavujú prah presnosti IoU (50 % a 75 %) a kategórie



Obr. 8.3: Výsledky metriky Dice Score pri použití rôznych inicializácii váh.

Tabuľka 8.4: Výsledky metrík modelu trénovaného na pôvodnej dátovej sade.

Mean DS	**PQ**	**mAP**
0.9438	0.9087	0.7793

small, *medium*, *large* označujú veľkosť objektov. Z výsledkov je zrejmé, že model OneFormer výrazne prekonáva pôvodný model Mask-RCNN najmä pri segmentácii malých objektov.

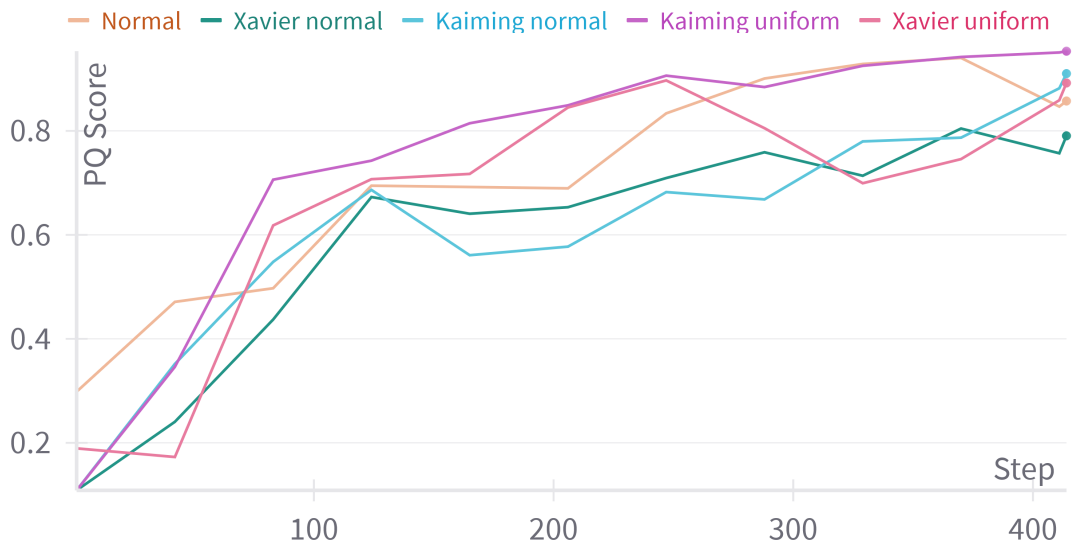
Tabuľka 8.5: Porovnanie výsledkov metrík modelu OneFormer oproti pôvodnému riešeniu.

Metrika	**OneFormer**	**Pôvodný model**	**Rozdiel**
mAP	**0.7793**	0.7380	+0.0413
mAP@50	**0.9602**	0.9520	+0.0082
mAP@75	0.8658	**0.8670**	-0.0012
mAP (large)	**0.9571**	0.8270	+0.1301
mAP (medium)	**0.7665**	0.7290	+0.0375
mAP (small)	**0.7075**	0.3580	+0.3495

Na trénovanie siete OneFormer boli použité rôzne prístupy z kapitoly 6. Veľkosť dávky (angl. *batch size*) bola nastavená na 2, keďže väčšia veľkosť sa už nezmestila na jednu 16 GB GPU. Veľkosť vstupných dát bola podľa odporúčaní autorov článku o sieti OneFormer [5] zmenená na 800×800 pixelov. Experimentovanie s rôznymi veľkosťami nepreukázalo dopad na schopnosti modelu.

Ako základ na trénovanie boli použité pred trénované váhy na dátovej sade *ADE20k* a ako backbone bol použitý transformer Swin-L. Na inicializáciu váh segmentačnej a klasifikačnej hlavy bola použitá uniformná verzia Kaiming inicializácie. Váhy backbone transformeru Swin-L boli zmrazené.

Model sa trénoval prvé 3 epochy s rýchlosťou učenia 1×10^{-3} pre segmentačnú a klasifikačnú hlavu a 1×10^{-4} pre zvyšné váhy modelu. Ďalšie 3 epochy sa model trénoval



Obr. 8.4: Výsledky metriky Panoptic Quality pri použití rôznych inicializácii váh.

s 10-krát menšou rýchlosťou učenia. Posledné 2 epochy sa trénoval opäť s 10-krát menšou rýchlosťou učenia ako predchádzajúce epochy. Ako optimalizačná funkcia bola použitá `AdamW` s úpadkom váh (angl. *weight decay*) 1×10^{-5}.

Model sa trénoval s využitím zmiešanej presnosti (angl. *mixed precision accuracy*). Jedna epocha bez použitia zmiešanej presnosti na celej trénovacej dátovej množine (8410 snímok) trvá 105 minút a s použitím zmiešanej presnosti sa tento čas skráti na 77 minút.

8.2 Experiment: Sémantická, inštančná alebo panoptická segmentácia

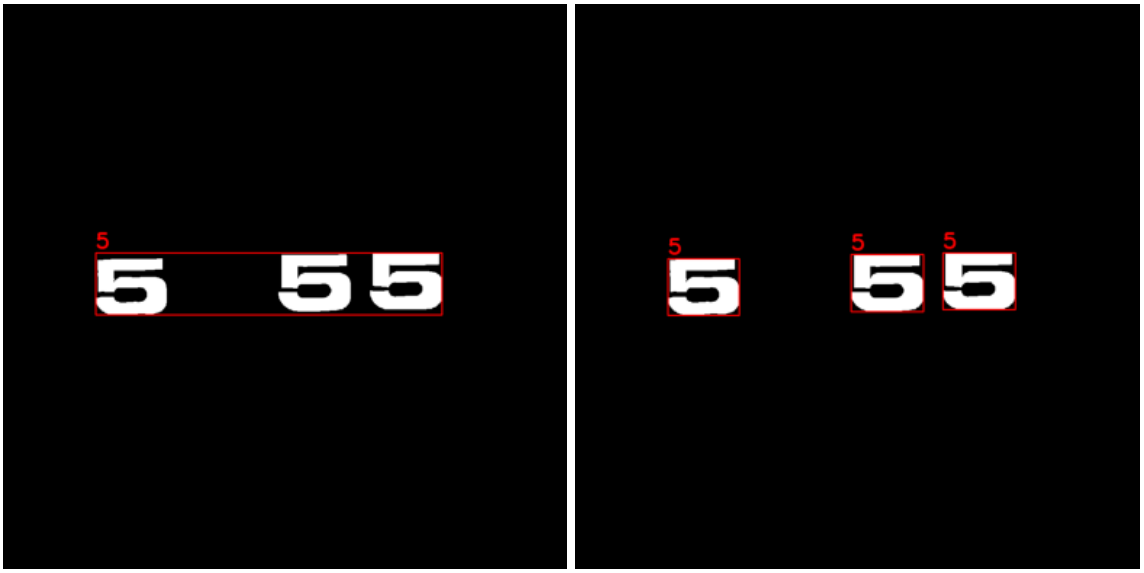
Ako už bolo spomínané v kapitole 4.1, OneFormer je model pre univerzálnu segmentáciu. V tejto časti porovnám trénovanie, vstupy a výsledky podľa toho, ako zadefinujeme úlohu segmentácie.

Ako prvé som vytvoril dátovú sadu pre sémantickú segmentáciu 7.1. Vychádzal som z toho, že symboly sa na skenoch bočníc pneumatík neprekrývajú, takže nie je potrebná panoptická alebo inštančná segmentácia na identifikáciu jednotlivých inštancií a je potrebné získať kvalitné masky. Následne som na tejto dátovej sade trénoval sieť OneFormer s tým, že typ úlohy som pomocou *task tokenu* nastavil na sémantickú segmentáciu.

Tento prístup mal dva problémy. Pri sémantickej segmentácii je výstupom priradenie každého pixela na obrázku do nejakej triedy. Nerozoznáva však jednotlivé inštancie, preto ak sa na jednom obrázku vyskytovalo dva a viac symbolov z rovnakej triedy, správali sa ako jeden objekt. Toto sa dá vyriešiť jednoducho napríklad tak, že zadefinujeme objekt ako určitú plochu ohraničenú napríklad pozadím. Takýmto spôsobom vieme získať jednotlivé inštancie objektov, ak sa neprekrývajú (čo je pri symboloch na bočnici pneumatiky splnené).

Druhý problém súvisí s prvým — model sa horšie učil tvar symbolov, pretože ak na jednom obrázku boli dva symboly z rovnakej triedy a na inom len jeden, pre model to

znamená jednu a tú istú triedu, aj keď ide o dve inštancie (viď obrázok 8.5). To viedlo k nesprávnej klasifikácii či nižšej kvalite masiek.



Obr. 8.5: Sémantická segmentácia nerozlišuje jednotlivé inštancie (vľavo) na rozdiel od panoptickej alebo inštančnej segmentácie (vpravo).

Dôsledkom týchto zistení som vytvoril dátové sady pre inštančnú aj panoptickú segmentáciu. Model vykazoval podobné výsledky, no tvorcovia siete OneFormer odporúčajú trénovať model na dátovej sade pre panoptickú segmentáciu, preto bol využitý tento typ dátovej sady. Inferencia následne prebieha ako inštančná segmentácia, keďže segmentovať pozadie na skenoch pneumatík nie je potrebné.

Pri použití tohto typu dátovej sady model okrem naučenia sa rozlišovať jednotlivé inštancie symbolov, ako výstup dáva kvalitnejšie masky symbolov. Masky pri špecifikovaní úlohy ako inštančná alebo panoptická segmentácia výrazne zvyšujú kvalitu a sú oproti maskám sémantickej segmentácie celistvejšie. Je to pravdepodobne dôsledok toho, že model chápe jednotlivé inštancie objektov ako celok, a teda nepriraďuje každý pixel v obrázku ku triede, ale zoskupuje pixely, ktoré pravdepodobne patria danému objektu. Ukážka niektorých menej presných masiek (hlavne pre veľké symboly), ktoré boli výstupom modelu učiaceho sa na dátovej sade pre sémantickú segmentáciu sú na obrázku 8.6.

Obr. 8.6: Nekvalitné a necelistvé masky pre veľké symboly, ktoré sú výstupom modelu trénovaného na sémantickej segmentácii (vľavo) a na porovnanie výstup instančnej segmentácie (vpravo).

Sieť OneFormer vykazuje výrazne odlišné výsledky pri použití rôznych typov *task tokenov* na tej istej dátovej sade. Rozdiel v metrikách toho istého modelu, ktorý bol trénovaný na dátovej sade pre panoptickú segmentáciu, pri použití rôznych *task tokenov* v porovnaní s *task tokenom* typu `instance` je uvedený v tabuľke 8.6.

Tabuľka 8.6: Výsledky jednotlivých prístupov s vyznačeným zlepšením oproti sémantickej segmentácii.

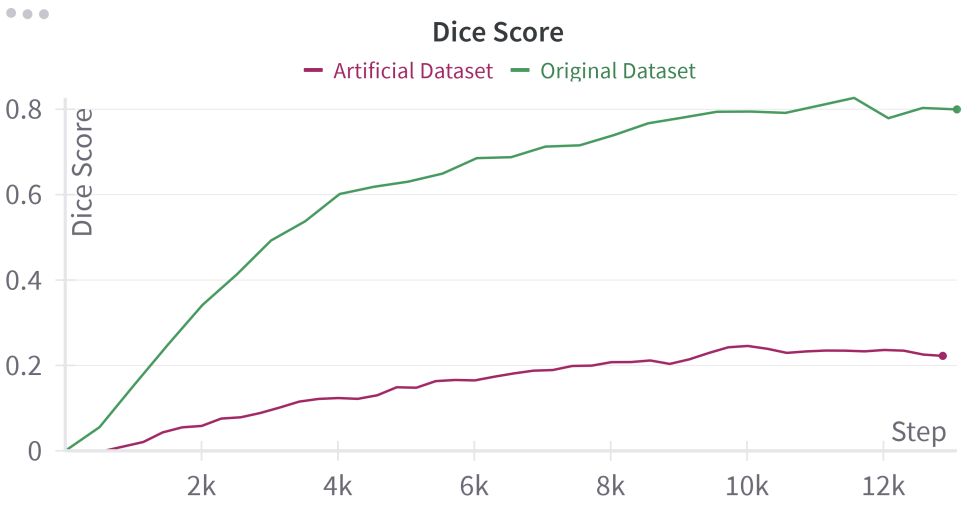
Task token	**DS** (%)	**PQ** (%)	**AP** (%)
`semantic`	76.1	67.9	53.0
`panoptic`	88.3 (+12.2)	82.6 (+14.7)	68.8 (+15.8)
`instance`	**94.4** (+18.3)	**90.9** (+23.0)	**77.9** (+24.9)

Z experimentov vyplýva, že je najvhodnejšie úlohu segmentácie symbolov na skenoch bočnice pneumatiky definovať ako inštančnú segmentáciu.

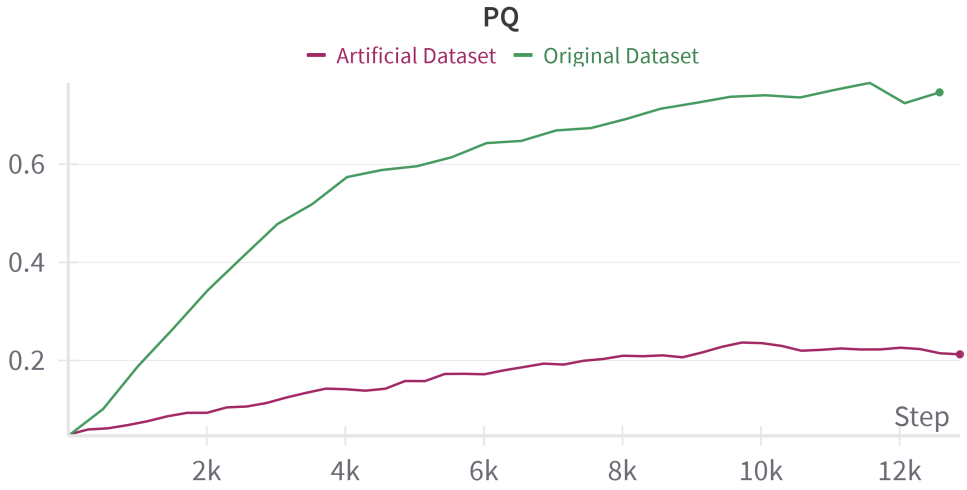
8.3 Trénovanie na umelej dátovej sade

Na trénovanie na umelej dátovej sade bola použitá rovnaká konfigurácia modelu a hyperparametrov ako v predchádzajúcej časti, kde bol model trénovaný na pôvodných dátach. Model sa na tejto umelej trénovacej množine učil rovnako rýchlo.

Ako testovacia množina bola použitá rovnaká sada ako pri trénovaní modelu na pôvodných dátach. Výsledky ukázali výrazný rozdiel v schopnosti modelu rozpoznávať symboly na testovacej množine obsahujúcej pôvodné snímky v porovnaní so symbolmi na umelých snímkach. Na grafoch 8.7 a 8.8 je možné vidieť rozdiely vo výsledkoch metrík Dice Score a Panoptic Quality medzi modelom trénovaným na pôvodnej a na umelej dátovej sade.



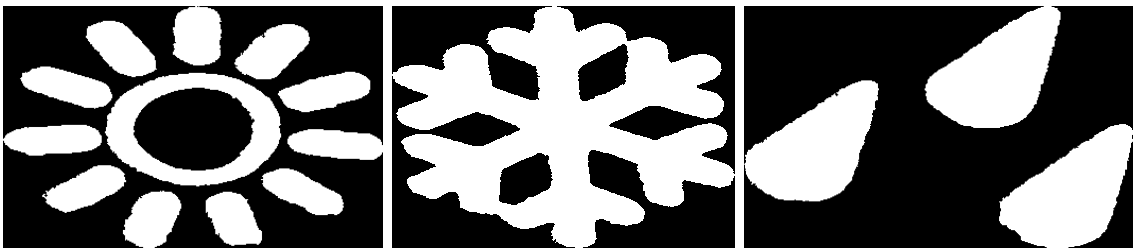
Obr. 8.7: Porovnanie metriky Dice Score modelu trénovaného na originálnych a umelých dátach. Model trénovaný na umelých dátach zle detekuje a segmentuje reálne dáta.



Obr. 8.8: Porovnanie metriky Panoptic Quality (PQ) modelu trénovaného na originálnych a umelých dátach. Model trénovaný na umelých dátach zle detekuje a segmentuje reálne dáta.

8.4 Dotrénovanie nových symbolov

Symboly, ktoré som vybral na dotrénovanie, sú *sun*, *snowflake* a *drops*. Vhodné sú kvôli tomu, že je reálne, že by sa vyskytli ako nové dáta, a na snímkach s nimi sa vyskytujú výlučne len tieto tri triedy a žiadne iné. Ukážka masiek symbolov je na obrázku 8.9.



Obr. 8.9: Odfiltrované symboly *sun* (slnko), *snowflake* (snehová vločka) a *drops* (dažďové kvapky).

Po natrénovaní modelu na základnej dátovej sade som model dotrénoval na nových (odfiltrovaných) dátach. Trénovacia dátová sada s odfiltrovanými symbolmi obsahuje 38 snímok, ktoré obsahujú iba symboly `sun`, `snowflake` a `drops`. V tejto časti uvádzam výsledky rôznych experimentov s dotrénovaním modelu na tejto dátovej sade. Skúmané boli nasledujúce prístupy:

- dotrénovanie celého modelu,
- dotrénovanie vybraných častí modelu,
- dotrénovanie prostredníctvom techniky LoRA.

Dotrénovanie celého modelu

Pri tomto dotrénovaní boli upravené všetky váhy modelu, žiadne neboli zmrazené.

Model sa pri učení na nových dátach dokáže efektívne naučiť klasifikovať a segmentovať nové triedy, avšak dochádza pritom k výraznej strate schopnosti rozpoznávať pôvodné triedy.

Už krátky trénovací proces trvajúci len 4 epochy (približne 60 sekúnd) postačuje na to, aby model dosiahol uspokojivú presnosť na nových dátach, no zároveň dochádza k výraznému úpadku výkonu na pôvodnej dátovej sade. Tento jav, známy ako katastrofické zabúdanie (catastrophic forgetting), predstavuje bežný problém v oblasti strojového učenia, najmä pri postupnom (inkrementálnom) učení, kde model pri adaptácii na nové úlohy stráca informácie o predtým naučených vedomostiach. Výsledky modelu po 4 epochách na pôvodných dátach a rozdiely oproti pôvodným výsledkom sú v tabuľke 8.7.

Tabuľka 8.7: Výsledky a rozdiely oproti pôvodným metrikám po 4 epochách trénovania. Model zabúda väčšinu pôvodných vedomostí.

Metrika	**Hodnota**	**Rozdiel**
Mean Dice	0.64	-0.30
PQ	0.62	-0.28
mAP	0.35	-0.42

Dotrénovanie vybraných častí siete

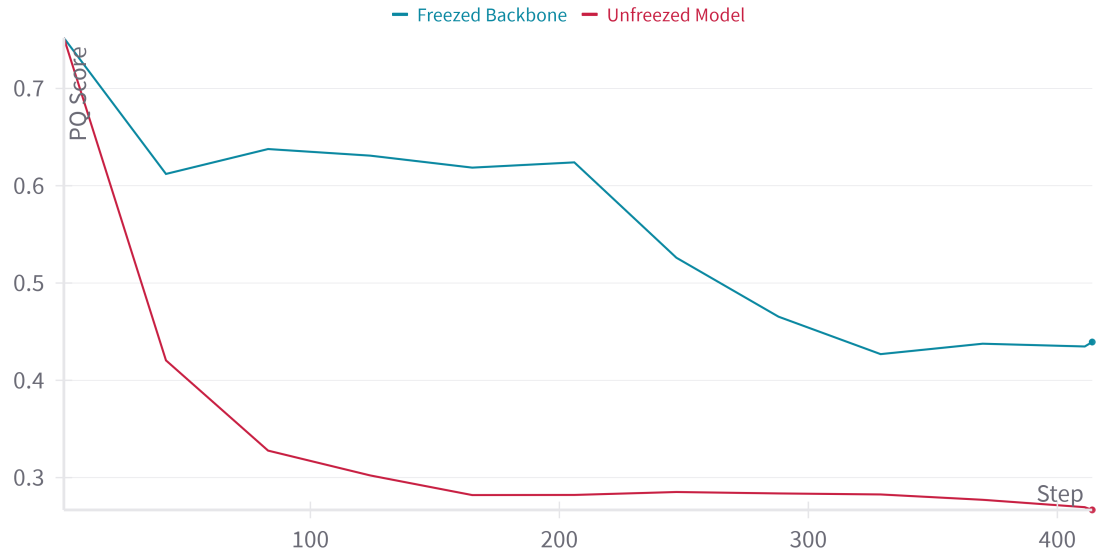
Klasickým prístupom pri doladení modelov je zmrazenie váh, ktoré obsahuje backbone siete a nižších vrstiev modelu, ktoré zabezpečujú schopnosť modelu rozpoznávať všeobecné vizuálne informácie. Učia sa len vyššie vrstvy, ktoré majú za úlohu priraďovať tieto vizuálne informácie k jednotlivým triedam.

Už len jednoduchým zmrazením váh Swin transformera, ktorý slúži ako backbone siete OneFormer, sa radikálne zlepšilo dotrénovanie modelu. Backbone Swin-L má približne 192 miliónov parametrov a tvorí približne 83 % všetkých váh modelu. To znamená, že oproti prvému trénovaniu je trénovateľných len 17 % parametrov.

Porovnanie klesania metriky Dice Score na pôvodných dátach počas desiatich epoch medzi trénovaním celého modelu a trénovaním modelu, ktorý ma zmrazené váhy backbone je zobrazené na obrázku 8.10.

Aj malá manipulácia s pôvodnými váhami vedie k zníženiu všetkých hodnotených metrík približne o 4–5 %. Mierne zlepšenie nastáva v prípade, ak sa zmrazia aj váhy transformérového modulu siete OneFormer a trénuje sa primárne segmentačná a klasifikačná hlava spolu s *queries embedderom*. Táto konfigurácia je efektívna, pretože model si zachováva väčšinu pôvodných vedomostí, pričom sa sústreďuje na klasifikáciu nových tried a ich následnú segmentáciu. Týmto spôsobom sa trénuje iba 23,66 milióna parametrov, čo predstavuje približne 10 % z celkového počtu parametrov pri použití transformera *Swin-L* ako backbone. Napríklad, po trénovaní modelu na nových dátach počas jednej epochy model dosahuje rozdiel vo výsledkoch, ako je uvedené v tabuľke 8.8.

Aby sa predišlo katastrofickému zabúdaniu, model bol dotrénovaný popri nových dátach aj na pôvodných snímkach. Trénovanie prebiehalo s veľmi nízkou hodnotou rýchlosti učenia. Pre segmentačné a klasifikačné hlavy modelu bola nastavená rýchlosť učenia na 1×10^{-4}, pričom pre zvyšné váhy bola táto hodnota desaťnásobne nižšia. V rámci experimentu boli



Obr. 8.10: Model pri zmrazení backbone zabúda pomalšie naučené informácie pri trénovaní na nových dátach.

Tabuľka 8.8: Rozdiely v percentách oproti pôvodným metrikám po jednej trénovacej epoche.

Metrika	**Rozdiel** [%]
Mean Dice	-2.95
PQ	-3.68
mAP	-3.29

testované rôzne pomery snímok zo starej a novej dátovej sady. Model bol trénovaný po dobu 10 epoch. Výsledky jednotlivých experimentov sú uvedené v tabuľke 8.9.

Tabuľka 8.9: Výsledky pre staré dáta s rozdielmi oproti pôvodnému modelu (v zátvorke).

Nastavenie	**Mean Dice** [%]	**PQ** [%]	**mAP** [%]
Pôvodný model	**94,38**	**90,87**	**77,93**
1:6	**94,38** (0,00)	**90,87** (0,00)	**77,93** (0,00)
1:3	94,06 (–0,32)	**90,87** (0,00)	77,32 (–0,61)
1:2	93,71 (–0,67)	89,81 (–1,06)	76,43 (–1,50)
1:1	93,13 (–1,25)	89,64 (–1,23)	75,57 (–2,36)

Z výsledkov je zrejmé, že model sa naučí segmentovať a klasifikovať nové symboly viacmenej rovnako, bez ohľadu na pomer nových a pôvodných snímok v trénovacej množine. Naopak, čím väčší podiel pôvodných snímok trénovacia množina obsahuje, tým lepšie si model zachováva svoje pôvodné vedomosti. Pri pomere nových snímok k pôvodným snímkam 1:6 (38 nových a 228 pôvodných snímok) model už nestráca svoje pôvodné schopnosti a dosahuje porovnateľnú presnosť metrík ako pôvodný model, pričom zároveň spoľahlivo a kvalitne segmentuje a klasifikuje aj nové symboly. Proces dotrénovania v tomto prípade trval iba **25 minút**.

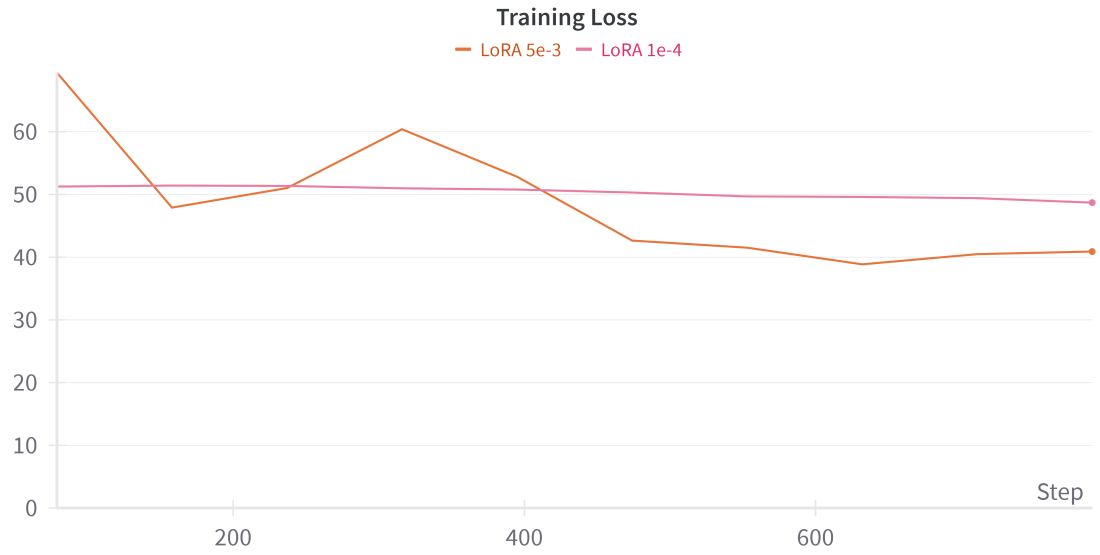
Tabuľka 8.10: Výsledky pre nové dáta.

Nastavenie	Mean Dice	PQ	mAP
1:6	0.9740	0.9502	0.9000
1:3	0.9760	0.9538	0.9167
1:2	0.9754	0.9528	0.9167
1:1	0.9754	0.9528	0.9000

Dotrénovanie pomocou techniky LoRA

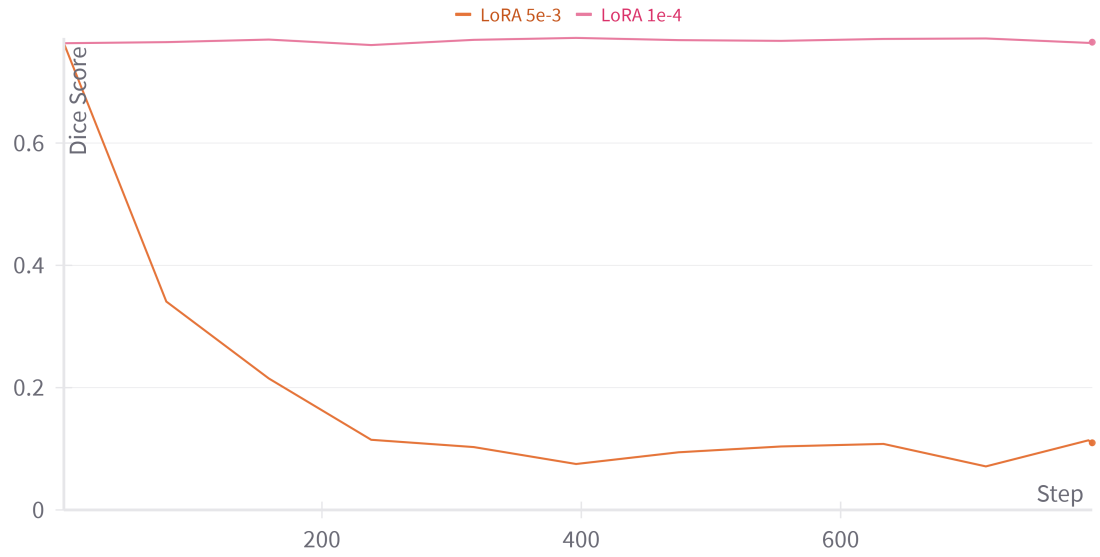
Technika *Low Rank Adaptation* bola popísaná v kapitole 4. Pri experimentovaní bola aplikovaná na `Query` a `Value` časti transformera, keďže všeobecne dosahujú najlepšie výsledky [4]. Všetky ostatné parametre a váhy modelu sa zmrazili. Aplikovaním metódy LoRA na pred trénovanú sieť OneFormer na pôvodných dátach sa pridali k modulom `Query` a `Value` nové LoRA moduly, ktoré sú trénovateľné. Pri tejto konfigurácii metódy LoRA je trénovateľných len 231 tisíc parametrov, čo je 0,1 % parametrov celej siete pri použití *Swin-L backbone*.

Experimentoval som s rôznymi konfiguráciami hyperparametrov, no tým pádom, že niektoré kľúčové moduly ako napríklad segmentačná a klasifikačná hlava neboli trénované, model sa buď neučil nové dáta pri malej rýchlosti učenia (červená krivka na grafe 8.11), alebo nastala explózia gradientov pri vyšších rýchlostiach učenia (oranžová krivka na grafe 8.11). Chybová funkcia počas 10 epoch s rýchlosťou učenia 1×10^{-4} a 5×10^{-3} je na obrázku 8.11.



Obr. 8.11: Chybová funkcia po aplikovaní LoRA modulov len na `Query` a `Value` časti siete OneFormer. Model sa neučil pri malej rýchlosti učenia nič, a pri väčších nastala explózia gradientov.

Na obrázku 8.12 je priebeh metriky Dice Score počas dotrénovania na nových dátach. Tieto metriky boli vyhodnocované na dátach, ktoré obsahovali aj pôvodné aj nové snímky. Pri nízkej rýchlosti učenia sa model nič neučí a pri vyšších zabúda aj pôvodné triedy.



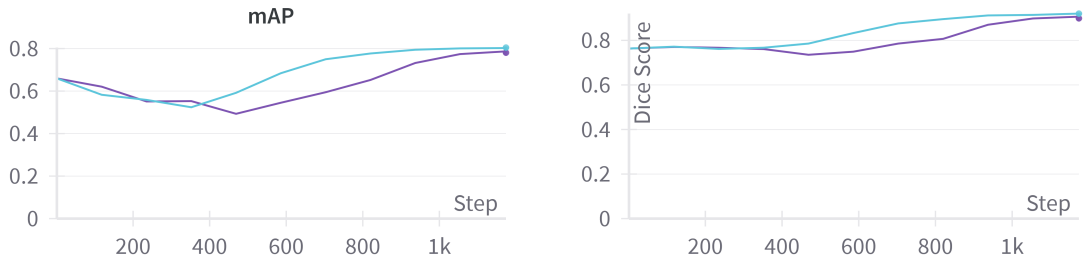
Obr. 8.12: Vývoj Dice Score po aplikovaní LoRA modulov len na `Query` a `Value` časti siete OneFormer.

Pri ďalšom experimente som zachoval kľúčové moduly ako trénovateľné a LoRA moduly som aplikoval rovnako ako v predchádzajúcom experimente.

Takto nastavená sieť OneFormer sa dotrénovávala na zmiešaných dátach počas 10 epoch. Rýchlosť učenia bola nastavená na 1×10^{-4}. Na grafoch je znázornené porovnanie metrík a chybových funkcií pre dotrénovanie modelu metódou LoRA a zmrazením váh niektorých častí siete. Sieť OneFormer bola v obidvoch prípadoch trénovaná na zmiešanej trénovacej množine obsahujúcej nové dáta a pôvodné dáta v pomere 1:2 a 1:4. Priebeh trénovania pre tieto pomery je zobrazený na obrázkoch 8.13 a 8.15. Použitie metódy LoRA dosahuje veľmi podobné výsledky (viď grafy na obrázkoch 8.14 a 8.16) ako prístup opísaný v predchádzajúcej časti.

Takáto konfigurácia v prípade siete OneFormer obsahuje 65,71 miliónov parametrov, čo predstavuje 27,8 % zo všetkých parametrov siete. Oproti priamemu zmrazeniu vybraných častí modelu, ako bolo opísané v predchádzajúcej časti, je v tomto prípade trénovateľných viac parametrov. Napríklad ak sa použije Swin-T ako backbone, pri použití metódy LoRA je trénovateľných len 19,65 % parametrov a pri zmrazení určitých modulov je to 34,1 %. Z toho vyplýva, že pri menšej veľkosti backbone môže byť výhodnejšie z pohľadu času a zdrojov použiť metódu LoRA.

Obr. 8.13: Porovnanie priebehu chybovej funkcie pri trénovaní s použitím techniky LoRA a selektívneho zmrazenia modulov. Grafy ukazujú podobný priebeh konvergencie pre oba prístupy pri pomere nových a pôvodných dát 1:2.



Obr. 8.14: Porovnanie metrík mAP a Dice Score pri trénovaní s použitím techniky LoRA a selektívneho zmrazenia modulov pri pomere nových a pôvodných dát 1:2. Z grafov vidno len malý rozdiel medzi týmito metódami.

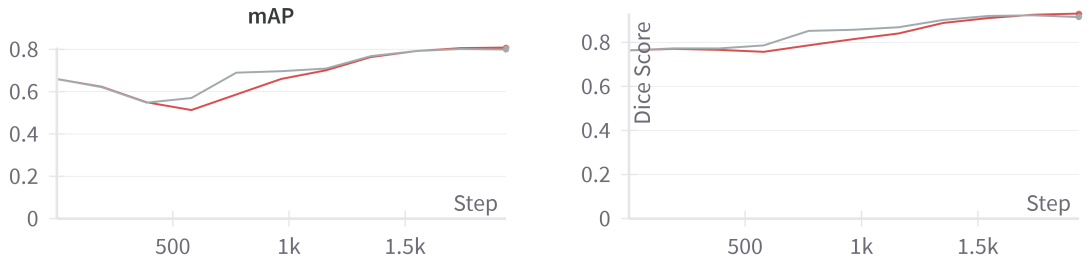
Porovnanie prístupov dotrénovania

Na základe vykonaných experimentov sú v tabuľke 8.11 zhrnuté výsledky metriky Panoptic Quality na dátach zo starej aj novej dátovej sady. Na dotrénovanie bola použitá sieť OneFormer s backbone Swin-L.

Tabuľka 8.11: Porovnanie jednotlivých prístupov dotrénovania na metrike Panoptic Quality.

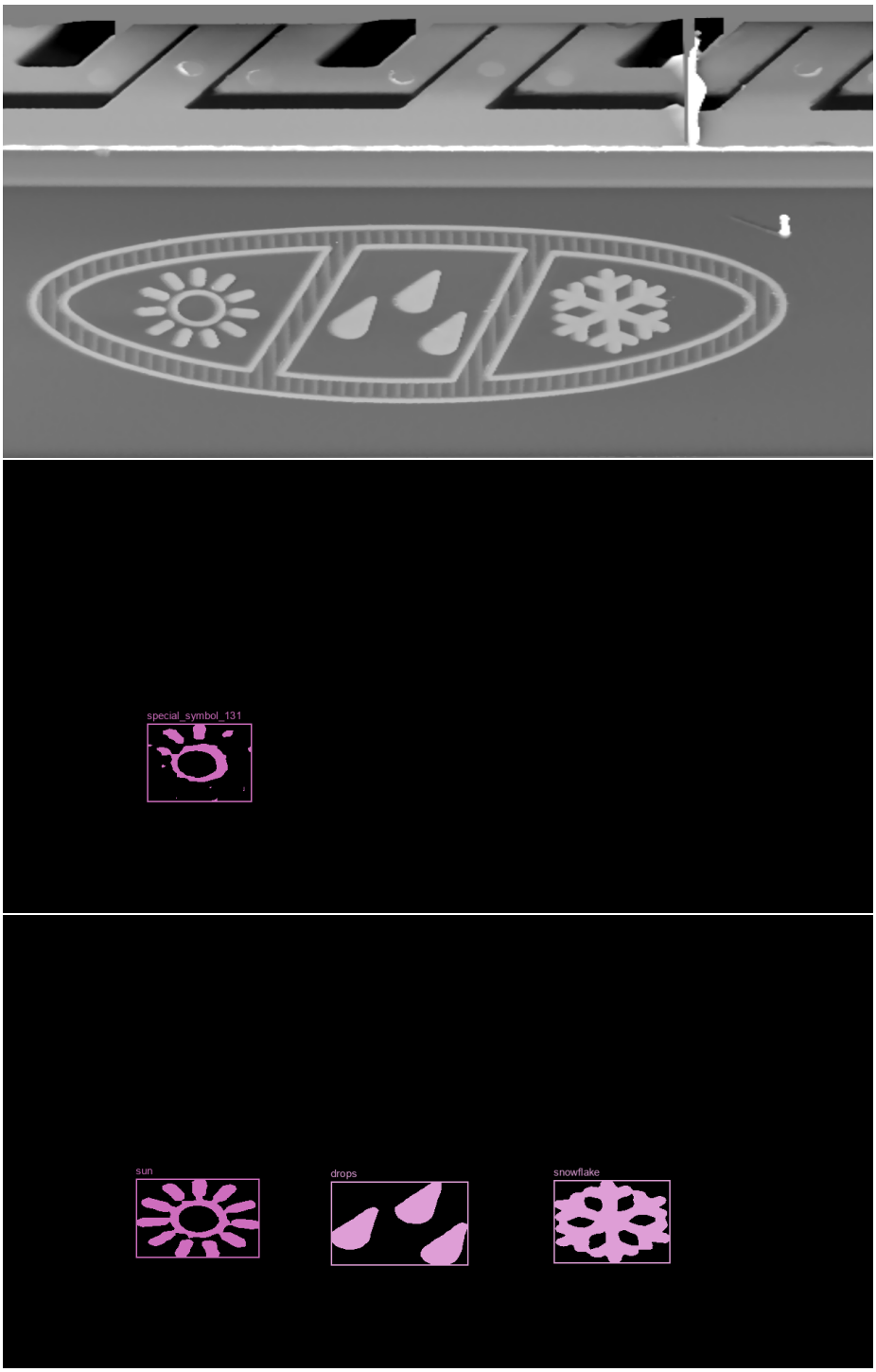
Prístup	**Staré dáta**	**Nové dáta**	**Trénovateľné parametre**
Celý model	0.8791	0.9077	100 %
Zmrazenie backbone	0.8728	0.9152	∼ 17%
Časti siete	**0.9087**	**0.9502**	∼ 10%
LoRA	0.9044	0.9496	∼ 28%

Obr. 8.15: Porovnanie priebehu chybovej funkcie pri trénovaní s použitím techniky LoRA a selektívneho zmrazenia modulov. Grafy ukazujú podobný priebeh konvergencie pre oba prístupy pri pomere nových a pôvodných dát 1:4.



Obr. 8.16: Porovnanie metrík mAP a Dice Score pri trénovaní s použitím techniky LoRA a selektívneho zmrazenia modulov pri pomere nových a pôvodných dát 1:4. Z grafov vidno len malý rozdiel medzi týmito metódami.

Ukážka schopnosti modelu segmentovať a klasifikovať symboly pred dotrénovaním a po dotrénovaní na vyfiltrovaných dátach pomocou spomínaných metód v tejto kapitole je na obrázku 8.17.



Obr. 8.17: Výsledok segmentácie snímky z vyfiltrovanej dátovej sady (hore) pred dotrénovaním na vyfiltrovaných dátach (v strede) a výsledok po dotrénovaní modelu aj na vyfiltrovaných dátach (dole).

Kapitola 9

Záver

Cieľom tejto práce bolo vytvoriť riešenie na segmentáciu symbolov na bočniciach pneumatík, ktoré zachová kvalitu výsledkov pôvodného riešenia, no zároveň umožní rýchlejšiu adaptáciu na nové symboly. Práca porovnáva rôzne state-of-the-art metódy efektívneho dotrénovania a analyzuje vplyv typu segmentácie na výkon modelu.

Riešenie

Výkon self-supervised modelov neprekonal iné vyskúšané postupy a preto neboli využité. Na riešenie úlohy bola použitá technika trénovania so zmiešanou presnosťou, metóda *Low Rank Adaptation* a prístup trénovania iba vybraných častí modelu. Výsledkom je výrazné skrátenie času adaptácie na nové symboly z pôvodných 72 hodín na **menej než jednu hodinu** (v závislosti od veľkosti novej dátovej sady). Navyše, toto dotrénovanie nemá vysoké nároky na zdroje — experimenty prebiehali na grafickej karte s kapacitou 8 GB.

Použité metódy a prístupy sú univerzálne a dajú sa aplikovať na rôzne modely a siete. Okrem rýchlej adaptácie na nové dáta otvárajú tieto metódy možnosti aj pre individuálnejší prístup k rôznym typom pneumatík (a teda aj zákazníkom), s cieľom zvýšiť presnosť segmentácie na konkrétnych typoch pneumatík.

Toto riešenie bolo prezentované zadávateľom úlohy a bolo prijaté ako zaujímavé a zlepšujúce ich súčasný stav.

Možnosti ďalšieho výskumu

Táto práca nerieši problém nevyváženosti tried v dátovej sade. Experimentovanie s umelou dátovou sadou, ktorá tento problém rieši, ukázalo, že model sa na tejto sade vie dobre učiť. Stále však pretrváva výrazný rozdiel medzi umelo vytvorenými snímkami a pôvodnými snímkami, čo sa prejavilo v zníženej úspešnosti modelu trénovaného na umelej dátovej sade pri segmentácii a klasifikácii na pôvodnej sade. Vytvorenie kvalitnejšej umelej sady, ktorá bude vernejšie reprezentovať reálne snímky, by mohlo pomôcť pri dotrénovaní modelu — najmä v prípadoch, keď sa niektoré symboly nevyskytnú v malej vzorke nových dát určených na dotrénovanie, čo môže znížiť kvalitu segmentácie týchto neprítomných symbolov.

Pôvodné riešenie (MaskRCNN) používa dvojitú segmentačnú vetvu za účelom zvýšiť kvalitu komplikovanejších masiek. Je možné, že podobná modifikácia architektúry siete OneFormer by mohla priniesť zlepšenie u niektorých symbolov.

Samotná sieť OneFormer použitá pri experimentoch s rôznymi metódami je náročná na výpočtové zdroje, podobne ako väčšina transformerových architektúr. Na strojoch, kde

sa aktuálne používa existujúce riešenie, sú tieto zdroje k dispozícii, no nasadenie menšieho modelu by zjednodušilo integráciu tohto riešenia aj na stroje s nižšou výpočtovou kapacitou.

Literatúra

[1] *Prezentácia firmy MiroEpsilon-Inspection* online. 2025. Dostupné z: https://www.micro-epsilon.com/company/company-group/me-inspection-sk/?sLang=en. [cit. 2025-05-04].

[2] Hassani, A. a Shi, H. *Dilated Neighborhood Attention Transformer* online. 2023. Dostupné z: https://arxiv.org/abs/2209.15001. [cit. 2025-05-04].

[3] He, K.; Gkioxari, G.; Dollár, P. a Girshick, R. *Mask R-CNN* online. 2018. Dostupné z: https://arxiv.org/abs/1703.06870. [cit. 2025-05-04].

[4] Hu, E. J.; Shen, Y.; Wallis, P.; Allen Zhu, Z.; Li, Y. et al. *LoRA: Low-Rank Adaptation of Large Language Models* online. 2021. Dostupné z: https://arxiv.org/abs/2106.09685. [cit. 2025-05-04].

[5] Jain, J.; Li, J.; Chiu, M.; Hassani, A.; Orlov, N. et al. *OneFormer: One Transformer to Rule Universal Image Segmentation* online. 2022. Dostupné z: https://arxiv.org/abs/2211.06220. [cit. 2025-05-04].

[6] Khanam, R. a Hussain, M. *YOLOv11: An Overview of the Key Architectural Enhancements* online. 2024. Dostupné z: https://arxiv.org/abs/2410.17725. [cit. 2025-05-04].

[7] Khanna, S.; Irgau, M.; Lobell, D. B. a Ermon, S. *ExPLoRA: Parameter-Efficient Extended Pre-Training to Adapt Vision Transformers under Domain Shifts* online. 2025. Dostupné z: https://arxiv.org/abs/2406.10973. [cit. 2025-05-04].

[8] Kirillov, A.; Mintun, E.; Ravi, N.; Mao, H.; Rolland, C. et al. *Segment Anything* online. 2023. Dostupné z: https://arxiv.org/abs/2304.02643. [cit. 2025-05-04].

[9] Liang, F.; Wu, B.; Dai, X.; Li, K.; Zhao, Y. et al. *Open-Vocabulary Semantic Segmentation with Mask-adapted CLIP* online. 2023. Dostupné z: https://arxiv.org/abs/2210.04150. [cit. 2025-05-04].

[10] Lin, T.-Y.; Maire, M.; Belongie, S.; Bourdev, L.; Girshick, R. et al. *Microsoft COCO: Common Objects in Context* online. 2015. Dostupné z: https://arxiv.org/abs/1405.0312. [cit. 2025-05-04].

[11] Liu, Z.; Lin, Y.; Cao, Y.; Hu, H.; Wei, Y. et al. Swin Transformer: Hierarchical Vision Transformer using Shifted Windows. In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)* online. 2021. [cit. 2025-05-04].

[12] Liu, Z.; Mao, H.; Wu, C.-Y.; Feichtenhofer, C.; Darrell, T. et al. *A ConvNet for the 2020s* online. 2022. Dostupné z: https://arxiv.org/abs/2201.03545. [cit. 2025-05-04].

[13] NVIDIA Corporation. *Train With Mixed Precision* online. 2023. Dostupné z: https://docs.nvidia.com/deeplearning/performance/mixed-precision-training/index.html. [cit. 2025-05-04].

[14] PyTorch Team. *Torch.nn.init.kaiming_normal_* online. PyTorch, 2024. Dostupné z: https://pytorch.org/docs/stable/nn.init.html#torch.nn.init.kaiming_normal_. [cit. 2025-05-04].

[15] PyTorch Team. *Torch.nn.init.kaiming_uniform_* online. PyTorch, 2024. Dostupné z: https://pytorch.org/docs/stable/nn.init.html#torch.nn.init.kaiming_uniform_. [cit. 2025-05-04]. Accessed: 2025-04-05.

[16] PyTorch Team. *Torch.nn.init.xavier_normal_* online. PyTorch, 2024. Dostupné z: https://pytorch.org/docs/stable/nn.init.html#torch.nn.init.xavier_normal_. [cit. 2025-05-04]. Accessed: 2025-04-05.

[17] PyTorch Team. *Torch.nn.init.xavier_uniform_* online. PyTorch, 2024. Dostupné z: https://pytorch.org/docs/stable/nn.init.html#torch.nn.init.xavier_uniform_. [cit. 2025-05-04]. Accessed: 2025-04-05.

[18] Ravi, N.; Gabeur, V.; Hu, Y.-T.; Hu, R.; Ryali, C. et al. SAM 2: Segment Anything in Images and Videos. *ArXiv preprint arXiv:2408.00714* online, 2024. Dostupné z: https://arxiv.org/abs/2408.00714. [cit. 2025-05-04].

[19] Ren, S.; He, K.; Girshick, R. a Sun, J. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks* online. 2016. Dostupné z: https://arxiv.org/abs/1506.01497. [cit. 2025-05-04].

[20] Ren, W.; Tang, Y.; Sun, Q.; Zhao, C. a Han, Q.-L. *Visual Semantic Segmentation Based on Few/Zero-Shot Learning: An Overview* online. 2022. Dostupné z: https://arxiv.org/abs/2211.08352. [cit. 2025-05-04].

[21] Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S. et al. ImageNet Large Scale Visual Recognition Challenge. *International Journal of Computer Vision (IJCV)* online, 2015, zv. 115, č. 3, s. 211–252. Dostupné z: https://doi.org/10.1007/s11263-015-0816-y. [cit. 2025-05-04].

[22] TensorFlow Authors. *Mixed Precision* online. 2025. Dostupné z: https://www.tensorflow.org/guide/mixed_precision. [cit. 2025-05-04]. Accessed: 2025-03-29.

[23] Toth, P. V. *Identifikace znaků na hloubkovém snímku pneumatiky.* Brno, CZ, 2022. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Dostupné z: https://www.fit.vut.cz/study/thesis/25095/.

[24] Zhang, H.; Li, F.; Zou, X.; Liu, S.; Li, C. et al. A Simple Framework for Open-Vocabulary Segmentation and Detection. *ArXiv preprint arXiv:2303.08131* online, 2023. [cit. 2025-05-04].

[25] Zhang, R.; Jiang, Z.; Guo, Z.; Yan, S.; Pan, J. et al. *Personalize Segment Anything Model with One Shot* online. 2023. Dostupné z: https://arxiv.org/abs/2305.03048. [cit. 2025-05-04].

[26] Zhou, B.; Zhao, H.; Puig, X.; Fidler, S.; Barriuso, A. et al. Scene Parsing through ADE20K Dataset. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* online. 2017. [cit. 2025-05-04].

[27] Zhou, B.; Zhao, H.; Puig, X.; Xiao, T.; Fidler, S. et al. Semantic understanding of scenes through the ade20k dataset. *International Journal of Computer Vision* online. Springer, 2019, zv. 127, č. 3, s. 302–321. [cit. 2025-05-04].

Zoznam príloh

Príloha A

Plagát

FAST ADAPTATION OF SEGMENTATION MODELS FOR TIRE SIDEWALL SYMBOLS

Michal Balogh
Supervisor: **doc. Ing. Michal Španěl, Ph.D**

In collaboration with Micro-Epsilon Inspection, this work addresses the current industry solution, which requires approximately 72 hours to retrain when new symbols are introduced. My approach combines state-of-the-art segmentation architecture (OneFormer) with efficient fine-tuning techniques including Low-Rank Adaptation (LoRA) and mixed precision training. The results show comparable segmentation quality to the current solution while reducing adaptation time from 72 hours to less than one hour — in extreme cases, fine-tuning a single symbol in just 5 minutes.

introduction & problem statement

Tire sidewalls of tires contain important information through various symbols and text that must be accurately segmented for quality control. The current industrial solution uses Mask R-CNN, but requires extensive retraining when new symbols are introduced:

current state

- 72-hour retraining time for new symbols
- Industrial requirement: Fast adaptation to new symbols while maintaining segmentation quality

dataset

- Depth images of tire sidewalls (65,536 × 1,300 pixels)

used hardware

- GPU with 16GB memory for training base model, 8GB for fine-tuning on new symbols

model architecture

OneFormer with Swin Transformer backbone

- Unified framework for semantic, instance, and panoptic segmentation
- Task-conditional joint training capability

optimization techniques

- Mixed precision training
- Low-Rank Adaptation (LoRA)
- Selective freezing of network components
- Kaiming uniform weight initialization

fine-tuning strategy

- Using 10% of model parameters
- Mixed dataset approach (blending new and original data)
- Preventing catastrophic forgetting through balanced sampling

new:old dataset ratio comparison

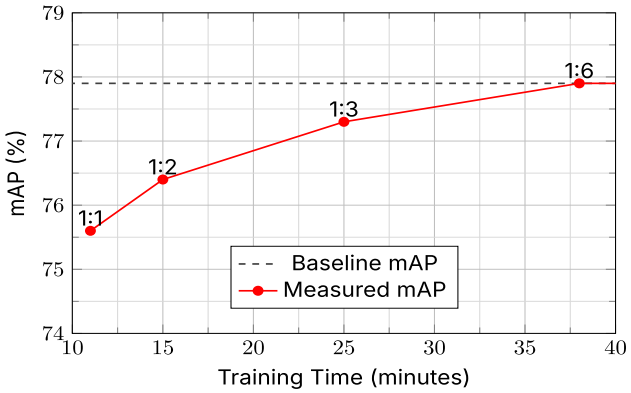


figure 1: preserving performance by leveraging old data.

segmentation type comparison

Task	F1 (%)	PQ (%)	AP (%)
Instance	94.4	90.9	77.9
Panoptic	−6.1	−8.3	−9.1
Semantic	−18.3	−23.0	−24.9

table 1: Impact of task type on F1, PQ, and mAP metrics.

fine-tuning approaches

- Full model fine-tuning - severe catastrophic forgetting
- Backbone freezing - moderate preservation of knowledge
- Selective parameter updating - best balance
- Low-Rank Adaptation - comparable to freezing but with less parameters