VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

DIPLOMOVÁ PRÁCE

Brno, 2020 Bc. Vojtěch Míček

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY
DEPARTMENT OF CONTROL AND INSTRUMENTATION

NEURONOVÉ SÍTĚ PRO KLASIFIKACI TYPU A KVALITY PRŮMYSLOVÝCH VÝROBKŮ

NEURAL NETWORKS FOR VISUAL CLASSIFICATION AND INSPECTION OF THE INDUSTRIAL PRODUCTS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Vojtěch Míček**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Petr Petyovský, Ph.D.**
SUPERVISOR

BRNO 2020

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní obor **Kybernetika, automatizace a měření**

Ústav automatizace a měřicí techniky

Student: Bc. Vojtěch Míček ***ID:*** 173701
Ročník: 2 ***Akademický rok:*** 2019/20

NÁZEV TÉMATU:

Neuronové sítě pro klasifikaci typu a kvality průmyslových výrobků

POKYNY PRO VYPRACOVÁNÍ:

Cílem práce je vytvořit systém pro klasifikaci typu a kontrolu kvality výrobků v průmyslu, který bude založený na zpracování obrazových dat pomocí neuronových sítí.

1. Seznamte se s problematikou nasazení neuronových sítí v průmyslových úlohách a jejich využití pro optickou klasifikaci výrobků.
2. Zvolte vhodné třídy výrobků (nejméně tři) vhodné pro optickou klasifikaci nebo detekci vad.
3. Zhodnoťte a vyberte vhodnou hardwarovou platformu pro úlohy učení a klasifikace pomocí neuronových sítí.
4. Vytvořte množiny testovacích snímků, které využijete pro učení a testování neuronových sítí.
5. Implementujte aplikaci pro optickou klasifikaci zvolených typů výrobků, která bude využívat neuronové sítě.
6. Optimalizujte natrénovanou neuronovou síť pro konkrétní hardwarovou platformu.
7. Zvolte vhodnou průmyslovou úlohu, která využívá optickou klasifikaci objektů. Pro tuto úlohu srovnejte vlastnosti a výhody i nevýhody klasického přístupu k řešení úlohy s přístupem založeným na využití neuronových sítí.
8. Zhodnoťte dosažené výsledky a navrhněte další možná rozšíření.

DOPORUČENÁ LITERATURA:

[1] WINSTON, Patrick Henry: Artificial Intelligence. 3rd ed. Addison-Wesley, 1992. ISBN: 9780201533774.

[2] ROSEBROCK, A.: Deep Learning For Computer Vision with Python, PyImageSearch, 2017. ISBN: 9781722487867.

Termín zadání: 3.2.2020 ***Termín odevzdání:*** 1.6.2020

Vedoucí práce: Ing. Petr Petyovský, Ph.D.
Konzultant: Mgr. Roman Procházka (TE Connectivity s.r.o.)

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

ABSTRAKT

Cílem této diplomové práce je umožnit posuzování kvality, nebo typu výrobku, v průmyslových aplikacích, pomocí umělých neuronových sítí. Jedná se zejména o aplikace, ve kterých je klasický přístup strojového vidění příliš komplikovaný. Takto navržený systém je dále implementován na konkrétní hardwarovou platformu a je u něj provedena optimalizace výsledného modelu, pro rychlejší běh systému.

KLÍČOVÁ SLOVA

Neuronové sítě, strojové vidění, inspekce výrobků, klasifikace objektů, Nvidia Jetson Xavier, Raspberry Pi, Intel Movidius, OpenCV, Keras, TensorFlow

ABSTRACT

The aim of this master's thesis thesis is to enable evaluation of quality, or the type of product in industrial applications using artificial neural networks, especially in applications where the classical approach of machine vision is too complicated. The system thus designed is implemented onto a specific hardware platform and becomes a subject to the final optimalisation for the hardware platform for the best performance of the system.

KEYWORDS

Neural networks, machine vision, product inspection, object classification, Nvidia Jetson Xavier, Raspberry Pi, Intel Movidius, OpenCV, Keras, TensorFlow

MÍČEK, Vojtěch. *Neuronové sítě pro klasifikaci typu a kvality průmyslových výrobků*. Brno, Rok, 66 s. Diplomová práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky. Vedoucí práce: Ing. Petr Petyovský, Ph.D

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Neuronové sítě pro klasifikaci typu a kvality průmyslových výrobků“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 1. 6. 2020 .....

podpis autora

PODĚKOVÁNÍ

Rád bych poděkoval svému vedoucímu diplomové práce panu Ing. Petru Petyovskému Ph.D. za odborné vedení, konzultace a podnětné návrhy k práci a konzultantu panu Mgr. Romanu Procházkovi za odborné rady, prosazení nákupu hardwarového vybavení a podporu při tvorbě práce.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Cílem práce bude umožnit posuzování kvality nebo kategorizaci typu výrobku v průmyslových aplikacích pomocí optické klasifikace založené na umělých neuronových sítích a takto takto vzniklé sítě optimalizovat na konkrétní hardwarovou platformu.

Bude se jednat zejména o aplikace, ve kterých je nasazení pouze klasického přístupu strojového vidění časově náročnější na vývoj, než při využití systému, který jako jeden z prostředků využívá umělé neuronové sítě. Za klasický přístup považuji ten, který je založený například na detekci hran, kružnic, rohů, vyhodnocování jasu, histogramu a podobně. Takto navržený systém bude dále implementován na konkrétní hardwarovou platformu a bude provedena optimalizace výsledného modelu pro rychlejší běh systému.

Díky navrženému systému, založenému na neuronových sítích, by tedy mělo být rentabilní řešit vizuální strojovou kontrolu a kategorizaci výrobků i v některých případech, ve kterých je to běžnými systémy velmi náročné. Jedná se zejména o kontroly s velkou variabilitou výrobků, s nemožností stabilního nasvětlení výrobků a s měnícími se vlastnosti kontrolovaného výrobku.

Ve své práci se budu nejdříve zabývat využitelností existujících systémů, které využívají neuronové sítě. Dále zvolím vhodné třídy výrobků pro optickou klasifikaci nebo detekci vad. Potom vyberu vhodnou hardwarovou platformu pro proces učení neuronových sítí a klasifikaci pomocí nich. Vytvořím množiny testovacích snímků zvolených výrobků. Tyto snímky použiji pro tvorbu modelů neuronových sítí a jejich následné testování za účelem klasifikace a detekce vad. Dále implementuji aplikaci pro optickou klasifikaci výrobků, která bude ve formě jednotlivě spustitelných skriptů. V další části této práce optimalizuji natrénovaný model pro vybranou hardwarovou platformu. Následně pro vhodnou průmyslovou úlohu srovnám vlastnosti, výhody a nevýhody klasického systému se systémem, který využívá neuronové sítě. Na závěr zhodnotím dosažené výsledky a navrhnu možné rozšíření.

Svou diplomovou práci budu konzultovat zejména se svým vedoucím práce panem Ing. Petrem Petyovským Ph.D. Dále pak s konzultantem diplomové práce s panem Mgr. Romanem Procházkou ze společnosti TE Connectivity s.r.o. Výrobní závod této společnosti v Kuřimi se zabývá zejména výrobou konektorů a kabelových svazků pro automobilový průmysl.

1 Základní pojmy

V této části definuji základní pojmy, které budou v následujících částech této práce využívány.

NOK

Z anglického „Not Okay“. V kontextu průmyslové výroby označuje kusy, které nesplňují požadované vlastnosti.

OK

Z anglického „Okay“. V kontextu průmyslové výroby označuje kusy, které splňují požadované vlastnosti.

Telecentrický objektiv

Pro danou pracovní vzdálenost jsou pro všechny vzdálenosti objektu od optiky objekty stejně velké.

Entocentrický objektiv

Vzdálenější objekty jsou zobrazeny menší, bližší větší. Jedná se o nejčastěji využívaný typ objektivu. Prakticky všechny spotřebitelské objektivy jsou tohoto typu.

Koaxiální osvětlení

Typ osvětlení, kdy je objekt osvětlován v ose objektivu. Konstrukce je většinou tvořena polopropustným zrcadlem skloněným pod úhlem 45° [41].

Low angle ring light

Jedná se o osvětlení, které se umisťuje velmi blízko výrobku. Je tvořeno několika řadami led diod umístěných do kruhu, které svítí pod úhlem 30°. Příklad tohoto typu osvětlení je na snímku 1.1.

Obr. 1.1: Příklad osvětlení typu low angle ring light [40]

Chytrá kamera

Ve vztahu k průmyslovým kamerám se jedná o kameru, která již v sobě obsahuje HW i SW pro vyhodnocování snímků a komunikaci.

Housing

Část konektoru, která neobsahuje piny, většinou vyrobena z plastu.

Učení neuronové sítě

Proces při kterém se nastavují jednotlivé váhy uvnitř neuronové sítě (dále označovány jako NN).

Trénovací, validační a testovací snímky

Ve vztahu k umělým neuronovým sítím jsou jako trénovací snímky označeny ty, které se využívají při procesu učení pro nastavování vah v modelu. Validační snímky slouží pro ověření modelu v průběhu učení. Testovací snímky se využívají pro ověření modelu až po dokončení procesu učení.

Inference

Ve vztahu k umělým neuronovým sítím se jedná o proces, kdy neuronová síť na základě dodaného vstupního vektoru a předem naučeného modelu predikuje vstupnímu vektoru konkrétní kategorii.

Architektura neuronové sítě

Popisuje, jak jsou jednotlivé neurony uspořádány do vrstev, jak jsou tyto vrstvy spojeny mezi sebou, jaké využívají aktivační funkce a metody učení.

Relu

Jedna z často využívaných aktivačních funkcí v neuronových sítích [1].

Bias

Konstanta, která je přidávána k sumě vážených jednotlivých vstupů v modelu neuronu [1].

Konvoluční vrstva

Vrstva založená na matematické operaci konvoluce. Slouží zejména pro zpracování obrazu. Při procesu učení se nastavují na místo vah, jako u klasických plně propojených vrstev, hodnoty konvolučního filtru.

Epocha

Ve vztahu k neuronovým sítím odpovídá jedna epocha, ději při kterém dojde při učicím procesu k jednomu průchodu trénovacími daty.

2 Problematika nasazení neuronových sítí v průmyslu

V současnosti se setkáváme v rozličných odvětvích průmyslu s trendem nasazování systémů, založených na neuronových sítích (dále jen NN). Bohužel se dost často vychází z představy, že jsou NN všespásné. Nicméně tomu tak není. Nejprve si je třeba uvědomit na jaké typy úloh se NN hodí a na jaké ne. NN jsou vhodné zejména pro úlohy u kterých se obtížně definuje pomocí sady jednoduchých pravidel, jak vypadá odlišnost, kterou chceme mezi výrobky detekovat. Ideálním příkladem tedy může být detekce vad například v potravinářské výrobě, kde se nedá předpokládat, že budou mít vady jasně definovaný charakter, ale budou se pokaždé lišit (spálené, nedopečené nebo neúplné pečivo). Naopak aplikace pro které NN nejsou vhodné jsou typicky měřicí aplikace.

V mém případě jsem se zabýval nasazením NN v automobilovém průmyslu, konkrétně ve společnosti TE Connectivity s.r.o., ve výrobním závodu TE Connectivity Kuřim. Tento výrobní závod se zabývá zejména výrobou konektorů a kabelových svazků pro automobilový průmysl. Tento typ průmyslu je specifický extrémně vysokými nároky na kvalitu výstupní kontroly. V případě TE 0% vadných kusů klasifikovaných jako správných a většinou méně než 1% těch, které jsou klasifikovány jako NOK kusy, které jsou ve skutečnosti v pořádku. Celá práce bude tedy směřována k tomu, aby byl výsledný systém vhodný pro potřeby detekce vad a kategorizaci výrobků výše zmíněného typu. Z nabízených a ověřených systémů pro detekci vad a klasifikaci jsem se setkal s následujícími řešeními.

2.1 Cognex VisionPro ViDi

Cognex je dnes již tradiční, americký výrobce systémů pro strojové vidění. V roce 2017 provedl akvizici společnosti ViDi Systems SA, která se zabývala systémem založeném na NN pro průmysl. V roce 2019 byl uvolněn produkt Cognex VisionPro ViDi pro systémové integrátory, a tak se mohl dostat ke koncovým zákazníkům. Jedná se o spojení původního produktu společnosti ViDi Systems SA a softwaru pro klasickou obrazovou kontrolu výrobků Cognex Designer. Celková cena za softwarovou licenci je přibližně 1 000 000 Kč bez DPH. Tato licence opravňuje ke kontrole snímků z jedné kamery systémem ViDi.

Při prezentaci produktu obchodním zástupcem společnosti Cognex byl použit pro učení modelu notebook Microsoft Surface Book 2 s dedikovanou grafickou kar-

tou NVIDIA GeForce GTX 1060 6GB. Systém vyžaduje pro učení řádově stovky fotografií každé kategorie. Bohužel při otestování na dodaných snímcích vykazoval systém nepřijatelné množství špatně určených snímků (>10%), lze to částečně přisuzovat pouze nedostatečnému nastavení při rychlé prezentaci. Při této prezentaci trvalo učení na průměrně výkonné grafické kartě, výše zmíněného notebooku, jednotky minut. Z tohoto a z dosažených výsledků usuzuji, že při učení NN neučí celý model, ale pouze přeučují několik posledních vrstev NN. Tímto přístupem, z mé zkušenosti, nelze dosáhnout lepších výsledků, než při učení celé NN. Výhodou je tedy možnost rychlého nasazení do výroby za předpokladu, že je systém dostatečně přesný pro konkrétní aplikaci. Další výhodou je potřeba malého množství snímků a potřeba pouze průměrně výkonného osobního počítače pro učení modelů. Ostatní vlastnosti vycházejí z vlastností řešení založeném na systému Cognex Designer. Je tedy zapotřebí Cognex kamery z řady CIC a průmyslového počítače Cognex VC5. Druhou variantou je využít klasického osobního počítače s příslušnými Cognex (karty CFG-8704E-200 a CIO-CC24-000). Řešení s osobním počítačem umožňuje zvolit aktuálnější, výkonnější komponenty. Do budoucna se dá předpokládat, že se objeví kompletní, plně integrované řešení v podobě chytrých kamer.

2.2 Opto Engineering ALBERT

Jedná se o plně integrované zařízení (snímek 2.1) o rozměru 33 x 31 x 17 cm. Je nabízeno v několika variantách, které se od sebe liší zorným polem. Nejmenší z nich má 26 x 19 cm a největší 53 x 39 cm. Všechny mají společnou minimální pracovní vzdálenost, a to 10 cm. Dále pak difuzní, bílé, zábleskové osvětlení a entocentrický objektiv. Celková cena za kompletní zařízení je přibližně 760 000 Kč bez DPH.

Obr. 2.1: Opto Engineering ALBERT [3]

Přístup k řešení společnosti Opto Engineering se značně liší od řešení společnosti Cognex. Minimální variabilita ve volbě optiky a nulová ve volbě osvětlení znemožňuje nasazení na některé typy aplikací se kterými se v TE Connectivity setkávám. Zejména se jedná o aplikace, ve kterých se kontrolují vnitřní části dutin, vyhnutí pinů a podobně. Na tyto kontroly je mnohem vhodnější telecentrický objektiv a také koaxiální osvětlení.

Liší se rovněž přístupem k učení NN. Systém se učí při běžící produkci výrobků a na základě nastavení a detekovaných anomálií vyhodnocuje chybné produkty. Vychází z předpokladu, že z prezentovaných kusů bude maximálně 20% vadných.

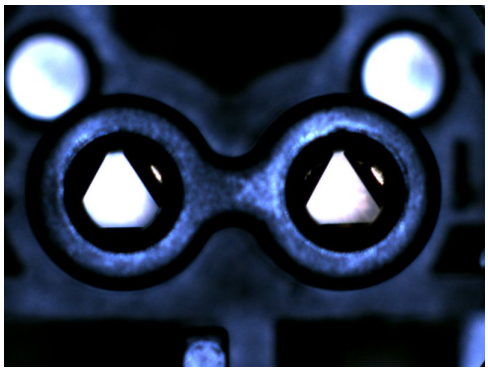
Z výše uvedených vlastností je tedy zřejmé, že toto řešení není vhodné pro automobilový průmysl, protože v něm požadujeme, aby byla detekce vad určena na základě přesně daných učicích snímcích, které byly ověřeny jiným způsobem. Není žádoucí, aby systém upravoval své rozhodování v průběhu výroby. Dalším, naprosto kritickým nedostatkem je nemožnost využít jiné osvětlení a optiku.

3 Výběr vhodné třídy výrobků

Jak již bylo zmíněno v úvodní kapitole, tato diplomová práce vychází ze zadání společnosti TE Connectivity. Při volbě vhodných tříd výrobků jsem se tedy zaměřil na stávající i budoucí projekty obrazové kontroly produktů a snažil jsem se vybrat vhodné kandidáty.

3.1 Kontrola pinů v konektoru

V rámci této úlohy se kontroluje přítomnost, stav a pozice tří přítlačných pinů v konektoru. Jedná se o projekt, který byl dlouhodobě řešen klasickým přístupem. Zdrojový snímek má rozlišení 800 x 60 px. Ukázkový zdrojový snímek 3.1

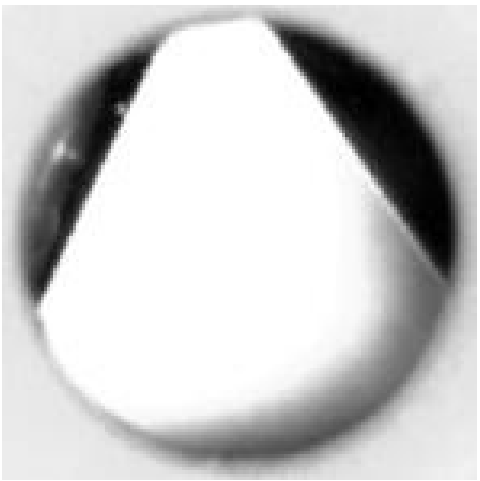


Obr. 3.1: Snímek konektoru po výměně osvětlení a objektivu

Vzhledem k tomu, že se v rámci této úlohy optické kontroly zabývám pouze kontrolou dutin, ve kterých se nachází piny, jsou pro názornost ukázány pouze výřezy. Na snímku 3.2 lze vidět ukázkovou fotografii dutiny správného konektoru. Uvnitř se nacházejí všechny tři přítlačné piny (plíšky). Jejich přítomnost poznáme tak, že se uprostřed fotografie nachází část rovnostranného trojúhelníku. Ten může být bílý, šedý, překrytý gradientem nebo na sobě mít stín. Dále je možné, aby byly v jeho rozích nedokonalosti a je nezbytné, aby se nacházely jeho hrany v povolených tolerancích od středu dutiny. Nepřípustný je taktéž deformovaný tvar dutiny jako takové.

Na snímku 3.3 vidíme konektor, který je potřeba vyhodnotit jako vadný. Spodní pin není viditelný, takže nelze vidět rovnostranný trojúhelník, který by tvořil pozadí.

Obr. 3.2: Konektor s přijatelnými piny

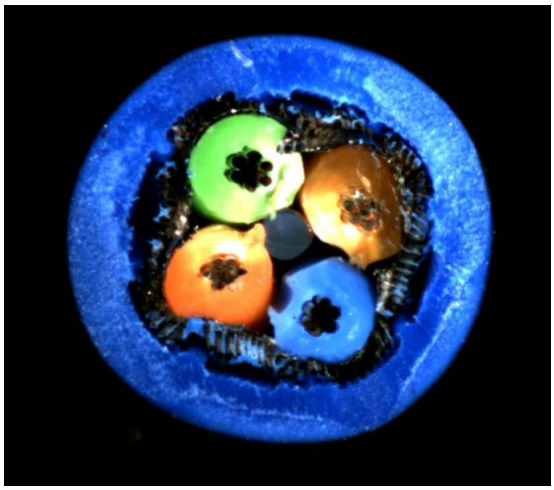


Obr. 3.3: Konektor s nepřijatelnými piny

3.2 Rozpoznání typu ustřiženého kabelu

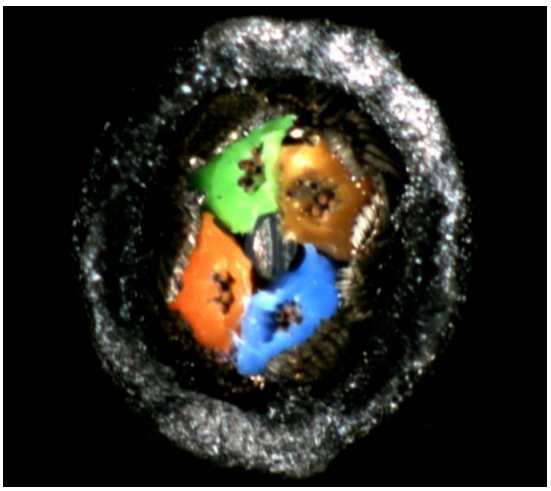
Rozpoznává se typ ustřiženého kabelu, který přichází do zařízení na další zpracování. Rozhodujeme se mezi 6 třídami, které se mezi sebou liší jak použitým kabelem, tak typem střihu. Do budoucna se dá očekávat další zvýšení počtu kategorií. Na následujících snímcích (3.4, 3.5, 3.6, 3.7, 3.8, 3.9) jsou zachyceni zástupci z jednotlivých kategorií.

Liší se následovně: nejjednodušeji rozpoznatelný je typ „blue“ (snímek 3.4), kde jednoznačně vidíme odlišnou barvu vnější izolace.



Obr. 3.4: Kabel - typ blue

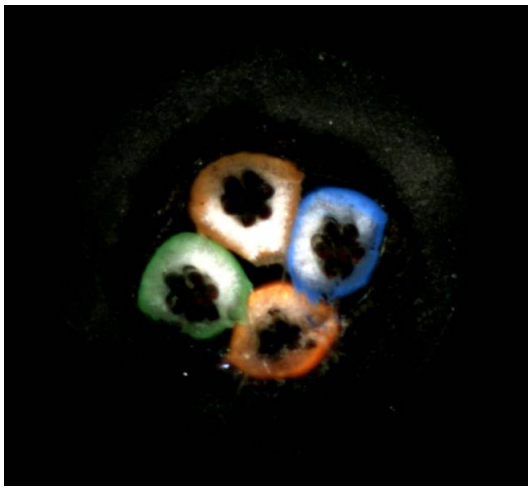
Okem jednoduše rozpoznatelný je typ „5352_man“ (snímek 3.5). Jedná se o kabel 5352 z manuálního střihu. Na rozdíl od ostatních není jeho izolace přibližně kruhová, ale spíše eliptická. Tento typ je zde spíše pro srovnání. Ve výrobě spadá do stejné kategorie jako kabel 5352_aut.



Obr. 3.5: Kabel - typ 5352_man

Typ „dacar“ (snímek 3.6) je charakteristický odlišnou vnitřní izolací kolem konkrétních žil kabelu. Na rozdíl od ostatních není izolace celobarevná, ale lze vidět, že zejména blíže ke středu je částečně bílá.

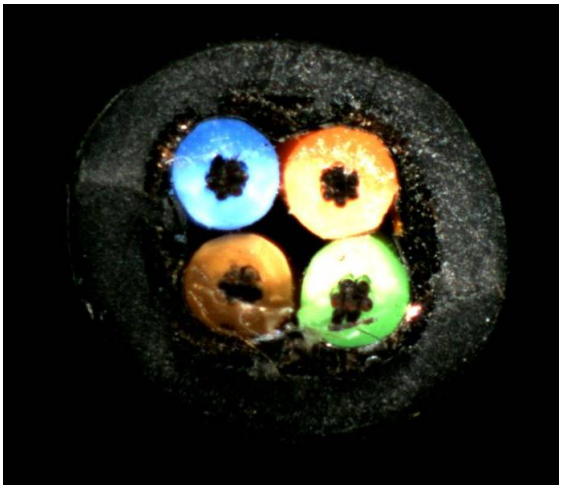
Nejobtížněji rozeznatelné od sebe jsou typy „535“ (snímek 3.7), „5352“ (snímek



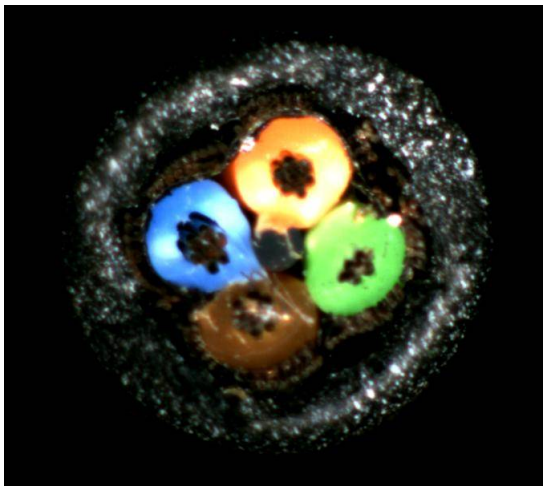
Obr. 3.6: Kabel - typ dacar

3.8) a „HT“ (snímek 3.9). Typ „5352_aut“ a „HT“ obsahují uprostřed nylonovou výztuhu, která také slouží jako separátor jednotlivých vodičů tzv. Central-strength-member. Nicméně zejména v případě „HT“ je dost často tato výztuha prakticky nerozpoznatelná. Její přítomnost se dá tedy pouze odhadovat na základě symetrie umístění jednotlivých vodičů. Dále má typ „HT“ odlišnou vnější izolaci a ve většině případů je na ní rozeznatelná jiná textura než u zbylých dvou typů.

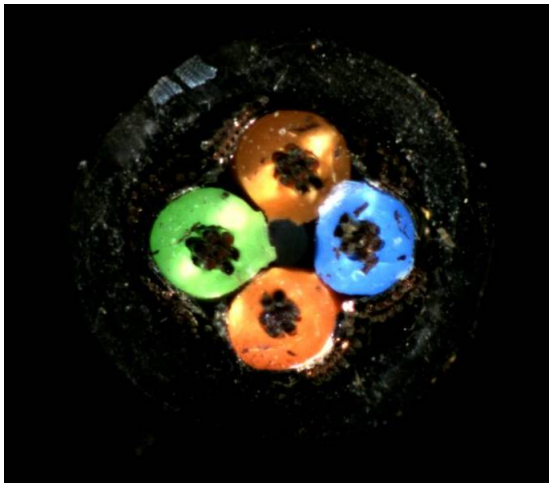
Ve výsledku se tedy spíše jedná o indikátory než o jednoznačné identifikátory typů. I v případě ruční kontroly trénovaným inspektorem kvality, dochází ke špatně identifikovaným kusům.



Obr. 3.7: Kabel - typ 535



Obr. 3.8: Kabel - typ 5352_aut



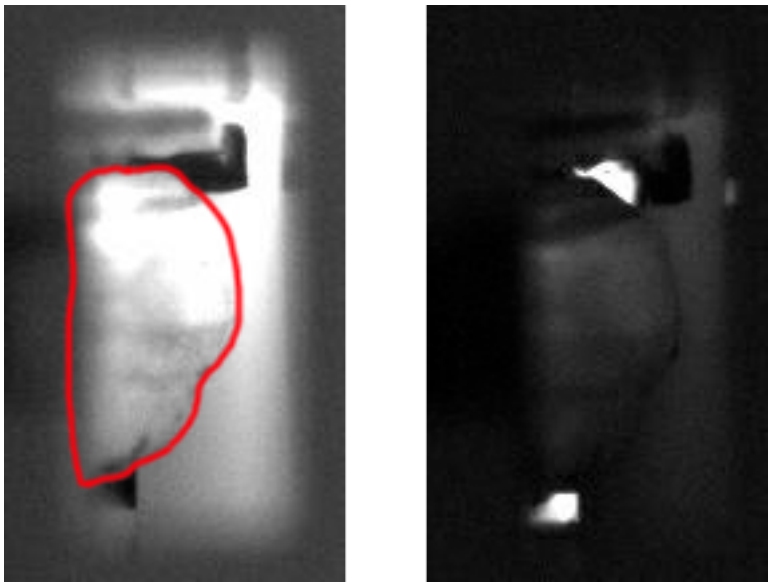
Obr. 3.9: Kabel - typ ht

Závěrem o kategorizaci kabelů můžeme říci, že i když ve většině případů je detekce typu střiženého kabelu klasickým způsobem možná, velmi často se objevují sporné fotografie, u kterých rozpoznat kategorii nezvládne ani člověk.

3.3 Kontrola přítomnosti plastové západky

V tomto případě je potřeba detekovat absenci plastové západky na plastové části konektoru. Z důvodu výrobního tajemství nelze uvést přesnější informace o výrobku jako takovém. I u této úlohy je detekce standardním způsobem obtížná. Zásadním problémem jsou extrémně se měnící světelné podmínky z důvodu různorodosti

vstupního materiálu a jeho nekonzistentní lubrikace. Z těchto důvodů vznikají na fotografiích odlesky na různých místech a o různé intenzitě. Na fotografiích 3.10 lze vidět tentýž typ plastové části konektoru se západkou, pouze s jinými odlesky a odrazivostí materiálu. Na snímku 3.11 se nalézá vadný, nedokončený kus bez západky. Zkoumaná západka je zaznačena červenou barvou na prvním snímku 3.10.



Obr. 3.10: Housing se západkou, světlá a tmavá varianta

Obr. 3.11: Housing bez západky

4 Hardwarová platforma pro učení a klasifikaci

Ač mají neuronové sítě mnohé výhody, pokud chceme NN učit nebo pomocí nich klasifikovat, je zapotřebí dostatečně výkonný hardware.

4.1 Hardware pro učení modelů a předzpracování dat

V první řadě je zapotřebí rozhodnout jak výkonný hardware potřebujeme a zda se finančně vyplatí investovat do vlastního řešení pro učení modelů nebo využít jedné z cloudových služeb pro učení NN, jako je například Google Colab, Google Caggle, Microsoft Azure nebo Amazon AWS.

V případě, že řešíme pouze konkrétní projekt a neplánujeme učit neuronové sítě do budoucna, je zřejmé, že bychom měli zvolit cloudové služby.

4.1.1 Google Colab

Společnost Google umožňuje omezené využití svého výpočetního výkonu pro uživatele zdarma. Omezení spočívá v tom, že jakýkoliv kód může na grafické kartě běžet pouze 8-12 hodin. V případě učení neuronové sítě je tedy zapotřebí ukládat tzv. checkpointy, což jsou modely, které se uloží po zadaném množství dokončených epoch nebo při splnění jiných zadaných parametrů. Pokud se tedy nestihne kompletní model vypočítat v daném časovém oknu, je možné manuálně navázat na poslední checkpoint a spustit učení od něj znovu. Dalším omezením je velikost dostupné grafické paměti. Ta je sdílena mezi více uživateli a ve většině případů se pohybuje okolo 12 GB. Nicméně je možné narazit i na hodnotu 500 MB. Maximální teoreticky dostupná paměť je dána použitou grafickou kartou. Konkrétně je použita NVIDIA T4. Podle článku [10] je pro samotné načtení modelu architektury Inception V3 (výsledná, použitá architektura NN, popsáno dále v kapitole 5) zapotřebí alokovat minimálně 0,72 GB grafické paměti. Vzhledem k měnící se přidělené paměti tedy nelze zaručit, že bude možno požadovaný model vůbec učit.

Google Colab zpracovává kód v jazyku Python. Kód se zapisuje v prostředí Jupyter. Nabízí integrované knihovny *PyTorchPyTorch, TensorFlow, Keras* a *OpenCV*. Data se ukládají na Google Drive.

4.1.2 Google Kaggle

Je obdobná služba jako je Google Colab. Nicméně je v něm náročnější pracovat. Poskytuje pouze 6 hodin pro nepřetržitý chod kódu a nabízí menší výkon.

4.1.3 Placené cloudové služby

Pro výpočet náročnějších modelů lze využít jednu z placených služeb. Specializované řešení pro učení NN poskytuje zejména Google, Amazon a Microsoft. Bohužel jejich vzájemné srovnání je velmi náročné. Google a Amazon AWS neposkytují veřejně ceník, Microsoft Azure ho sice poskytuje, ale platí se za konkrétní operace, takže záleží na konkrétní úloze. Bylo by tedy nejprve potřeba definovat testovací úlohu a následně buď poptat cenové nabídky a odhadnout počty konkrétních požadavků pro naučení a dle nabízeného výkonu vypočítat dobu trvání učení, nebo poptat jednotlivé cloudové služby a následně na každé z nich úlohu spustit. Teprve potom by je bylo možné porovnat.

4.1.4 Konfigurace vlastní pracovní stanice

Vzhledem k tomu, že se plánuji zabývat NN v TE Connectivity dlouhodobě, rozhodl jsem se, že bude nejvýhodnější učit modely NN na vlastním hardwarovém řešení. Parametrem, který mě nejčastěji omezoval byla velikost grafické paměti. Jednou z variant je použít profesionální grafické karty. Vzhledem k poměru ceny a grafické paměti a zachování aktuální architektury Turing, nejlépe vychází NVIDIA Quadro RTX 5000 s 16 GB grafické paměti. Aktuálně se nabízí přibližně za 45 000 Kč bez DPH. Alternativní variantou je zvolit herní grafickou kartu. V úvahu připadají dva modely, NVIDIA Titan RTX za cenu přibližně 59 000 Kč bez DPH s 24 GB grafické paměti a GeForce RTX 2080 Ti za 25 000 Kč bez DPH s 11 GB grafické paměti.

Další variantou pak může být spojení dvou grafických karet. V případě aktuální generace se využívá NVIDIA NVLink. Problémem tohoto řešení je nedostatečná podpora napříč využívaným softwarem a operačními systémy. Nelze tedy jednoduše prohlásit, že se budou dvě grafické karty chovat jako jedna s dvojnásobnou pamětí, protože záleží na konkrétním využití. NVIDIA kompatibilitu oficiálně neuvádí, je tedy nutné vše ověřit experimentálně nebo spoléhat na kusé zkušenosti třetích stran.

Dalším z důležitých parametrů, který omezuje rychlost učení neuronových sítí je rychlost komunikace mezi grafickou pamětí, operační pamětí a úložištěm. Pokud budeme vyžadovat maximální rychlost v těchto ohledech, pak jednoznačně zvolíme

aktuální architekturu AMD Ryzen 3000, která podporuje PCI Express nejnovější generace, tedy PCI-E 4.0.

Vysledoval jsem, že při učení modelů NN jsou vytěžovány maximálně 2 vlákna procesoru. Při spuštění kódu pro učení se nejprve vytěžuje pouze jedno vlákno procesoru na 100%. V této fázi se načítají dynamické knihovny. Kapitola 6.2 popisuje o které knihovny se jedná. V další fázi se přesouvají data do a z grafické karty. V této fázi se vytěžuje opět jedno vlákno trvale na 100%, nicméně se k němu občas přidá jedno další. Tyto závěry potvrzují články [11] a [12]. Dále se v nich uvádí, že to, že jsou ostatní vlákna nevytížena je dáno tím, že pro to nejsou uzpůsobeny využívané knihovny. Hledal jsem tedy procesor s nejvyšším výkonem pro aplikace využívající pouze 2 vlákna. V době stavby počítače to byl AMD Ryzen 9 3900X (přibližná cena je 11 554 Kč bez DPH). Takže jsem si opět potvrdil, že je vhodné zvolit systém postavený na procesoru od AMD.

Proto, aby nebyl procesor brzděn rychlostí operační paměti, je zapotřebí vybrat paměti s jak dostatečně vysokou frekvencí, tak s dostatečně malou CL latencí. Zvolil jsem dvě sady od HyperX HX434C16FB3AK2/32 (celkově 64 GB RAM, cena za jednu sadu je přibližně 4 000 Kč bez DPH). Tyto moduly nabízí rychlost 3466 MHz se zapnutým automatickým taktováním, latencí CL 16 a jsou plně podporovány na většině základních desek s čipsetem AMD X570. Pro správnou funkci procesoru i při dlouhodobém zatížení je zapotřebí, aby byl dostatečně chlazen. V balení dodávaný vzduchový chladič je sice pro běžné používání přijatelný, nicméně na dlouhodobou velkou zátěž a případné přetaktování se nehodí. Zvolil jsme tedy set vodního chlazení od společnosti NZXT Kraken X52. Jedná se o běžný set uzavřeného vodního chlazení. Blok pro procesor již obsahuje pumpu. Radiátor je osazen dvěma ventilátory o průměru 12 cm.

Jak už bylo zmíněno výše, kritickou částí řetězce je rychlé úložiště, toto platí obzvláště pro velké datasety snímků. Vzhledem k dostupnosti PCI Express slotů nové, 4. generace, se nabízí využít SSD, které tuto sběrnici využívají. Z běžně dostupných se jedná zejména o ADATA XPG GAMMIX S50, M.2 - 1TB (přibližná cena 5 000 Kč bez DPH). Alternativou by bylo starší, ale přesto velmi výkonné SSD Intel Optane 905P, který nemá konkurenci při čtení menších souborů. Vzhledem k jeho ceně (přibližně 29 000 Kč bez DPH) jsem zvolil ADATA XPG GAMMIX S50.

Výsledná konfigurace osobního počítače pro učení modelů neuronových sítí je zobrazena v následujícím seznamu.

- procesor - AMD Ryzen 9 3900X
- chladič procesoru - NZXT Kraken X52
- základní deska - MPG X570 GAMING EDGE WI-FI
- paměť RAM - 2x HyperX HX434C16FB3AK2/32
- úložiště - ADATA XPG GAMMIX S50, M.2, 1TB
- grafická karta - MSI GeForce RTX 2080 Ti VENTUS 11G OC
- zdroj - Seasonic Prime Ultra Gold - 850W
- počítačová skříň - Fractal Design Meshify C Blackout TG
- ventilátory - Noctua NF-S12A PWM chromax.black.swap

Celková sestava je na fotografii 4.1. Z několika měsíčního používání vzešel požadavek na překrytí zapínacího tlačítka a reset tlačítka. Reset tlačítko bylo odpojeno od základní desky. Kolem zapínacího tlačítka byl vytištěn kryt, který zabraňuje náhodnému vypnutí počítače pouhým opřením nebo manipulací.

Obr. 4.1: Osobní počítač pro učení NN a předzpracování dat

4.2 Hardwarová platforma pro inferenci

Hardwarová platforma pro inferenci slouží v případě této diplomové práce konkrétně jako zařízení, na kterém probíhá rozhodování o zařazení snímků do jednotlivých kategorií. Při výběru této hardwarové platformy záleží na mnoha faktorech. Předně záleží na tom, jak velký bude naučený model na základě kterého se bude predikovat výsledná kategorie, dále pak na potřebné rychlosti inference a důležitým faktorem je také požadavek na digitální a analogové vstupy a výstupy z platformy.

4.2.1 Osobní počítač

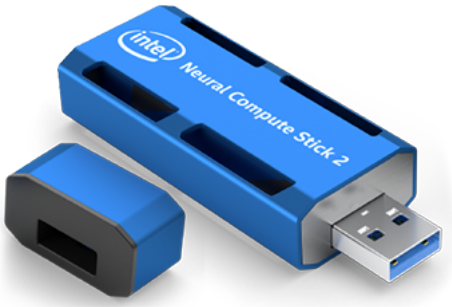
Základní premisa zůstává podobná jako v případě osobního počítače pro učení. Jediným rozdílem je to, že pro většinu aplikací není nutné, aby byl počítač pro inferenci stejně výkonný, ale stačí u něj výkon podstatně menší. Hlavní výhodou použití osobního počítače pro inferenci je možnost použít velký rozsah digitálních a analogových vstupů a výstupů do zařízení, protože jej lze rozšířit pomocí příslušných PCI-E karet. V případě, kterým jsem se zabýval, se jednalo o dvě karty do PCI-E slotu. První z nich je frame grabber, který umožňuje zaznamenávání obrazu až ze 4 GigE kamer COGNEX CIC. Použitý model má kódové označení CFG-8702E-200. Další z nich je karta pro vstupně-výstupní operace CIO-CC24-000. Nabízí 8 digitálních vstupů, 16 digitálních výstupů, 4 vstupy pro enkodér a Ethernet port, který podporuje protokoly Profinet a Mitsubishi SLMP.

4.2.2 Jednodeskové počítače a specializovaný hardware

Další možností jak provádět inferenci je využít jednodeskové počítače nebo specializovaný hardware. Pro velmi jednoduché aplikace jako je například rozpoznávání textu mělkými neuronovými sítěmi, je možno použít i běžné jednodeskové počítače jako je Raspberry Pi 3 Model B+. V praxi ovšem většinou narazíme na to, že nám jeho výkon nedostačuje.

Tento problém lze řešit dodáním akceleračního USB zařízení přímo určeného pro neuronové sítě. Jedná se zejména o akcelerační jednotky Intel Movidius - Neural Compute Stick 2 (snímek 4.2) a Google Coral - USB Accelerator (snímek 4.3). Oba zmíněné jsou velmi elegantním řešením, jak zvýšit výkon pro inferenci při minimálních investicích. Intel Movidius i Google Coral stojí přibližně 1 800 Kč bez DPH. K rychlostnímu porovnání u Intel Movidius - Neural Compute Stick 2 je inference rychlejší přibližně 4x [28]. Nevýhodou je, že se naučené modely musí upravovat, aby

byly kompatibilní s Intel Movidius a inference byla možná. Nutné úpravy modelu jsou popsány v oficiální dokumentaci [14]. U Google Coral je zrychlení až 10násobné [28]. Výhodou je, že model není třeba nijak upravovat.



Obr. 4.2: Intel Movidius - Neural Compute Stick 2 [4]

Obr. 4.3: Google Coral - USB Accelerator [5]

Další variantou je využít specializovaný hardware. Zde je již zapotřebí si stanovit, jaký typ modelů budeme pro inferenci využívat.

V případě, že se zaměřujeme zejména na menší modely a ideálně na ty, které jsou optimalizovány na mobilní zařízení, to v praxi nejčastěji znamená modely s architekturou NN MobileNet V1 a V2, bude zajímavým řešením jednodeskový počítač Google Coral Dev Board (snímek 4.4). Jeho cena je přibližně 3 500 Kč bez DPH.

Pokud předpokládáme, že budeme využívat modely NN s různorodými architekturami a nebo vyžadujeme vyšší výkon, řešení nabízí společnost NVIDIA. V aktuální

Obr. 4.4: Google Coral - Dev Board [6]

produktové řadě se nachází 3 produkty [15]. Všechny na rozdíl od řešení Google vycházejí z architektury grafických karet. Nejprve Jetson Nano Developer Kit, který je přibližně srovnatelně výkonný jako Google Coral - Dev Board (záleží na konkrétní aplikaci) [13]. Jeho přibližná cena je 2 300 Kč bez DPH. Následuje Jetson TX2, který výkonově již předčí Google Coral - Dev Board, jeho cena je přibližně 11 000 Kč bez DPH. Posledním v této řadě je Jetson AGX Xavier. Ten je ze všech nejvýkonnější a z tohoto důvodu jsem zvolil právě toto řešení. Více o této hardwarové platformě následuje v další sekci.

4.2.3 NVIDIA Jetson AGX Xavier

Platforma NVIDIA Jetson AGX Xavier (snímek 4.5) je založena na osmijádrovém procesoru ARM v8.2 64-bit CPU, 8MB L2 + 4MB L3, dále na specializované 512jádrové grafické kartě, která je založena na architektuře Volta a obsahuje Tensor Cores, což jsou jádra určeny speciálně pro výpočty s NN. Dále se v zařízení nalézá několik podpůrných jednotek pro práci s obrazem. Zařízení obsahuje 16 GB operační paměti a úložiště o velikosti 32 GB typu eMMC 5.1. Ze vstupů a výstupů je to pak HDMI 2.0, Gigabit Ethernet ve formě konektoru RJ45, NVMe M.2 slot pro SSD disk, USB 3.1 porty a CSI-2 Lanes pro přímé připojení kamer. Podrobnější informace se nalézají na oficiálních stránkách výrobce [16], nicméně je potřeba upozornit, že existují dvě revize téhož zařízení. Aktuální obsahuje již 32 GB operační paměti. Výrobce na existenci dvou verzí zařízení neupozorňuje a obě mají shodné označení. Z fyzických parametrů pak rozměry 10,5 x 10,5 x 6,5 cm. Tyto rozměry umožňují umístění například do rozvaděče výrobní linky. Velikost klasického osobního počítače je v tomto směru již značně omezující a v případě některých výrobních linek by jeho využití

nebylo vůbec možné.

Před samotným používáním zařízení jsem rozšířil vnitřní úložiště SSD diskem WD Blue SN500 NVMe SSD 500GB. Tento disk nabízí rychlost čtení až 1 700 MB/s a zápis 1 400MB/s. Je tedy řádově rychlejší než úložiště integrované. Dále jsem u zařízení požadoval připojení Wi-Fi a Bluetooth. Původním plánem bylo využít klasický USB Wi-Fi adaptér. Nicméně v době zkoušení neexistovaly pro žádný z nich ovladače a nebyly tak použitelné. Přidal jsem tedy kombinovanou Wi-Fi a Bluetooth kartu Intel 7265NWG (fotografie 4.6). Tato sice nebyla na jednočlenném seznamu podporovaných Wi-Fi karet [17], nicméně fungovala bez jakýchkoliv problémů. Připojení antén ke kartě je realizováno konektory U.FL. Tuto informaci výrobce neuvádí v žádných z oficiálních zdrojů. Bylo tedy zapotřebí vycházet ze vzhledu a rozměrů, čímž se omezí výběr pouze na konektory U.FL a MHF4.

Obr. 4.5: NVIDIA Jetson AGX Xavier

Následujícím krokem, ještě před spuštěním zařízení jako takového, je instalace NVIDIA JETPACK SDK verze 4.2.0 [18]. Tento balíček obsahuje jak vývojové nástroje, tak zejména operační systém, ovladače a běžné knihovny. Při jeho instalaci je zapotřebí osobní počítač s nainstalovaným operačním systémem Ubuntu v aktuální

Obr. 4.6: Wi-Fi karta Intel 7265NWG

verzi. Následně je zapotřebí dodržet instalační návod od výrobce. Častým problémem je, že se software aktualizuje rychleji než je doplňována technická dokumentace a návody. V takovém případě je zapotřebí vycházet ze starších informací a vyzkoušet, zda jsou stále platné. Alternativou je vznést dotaz u výrobce a čekat na oficiální vyjádření.

5 Tvorba množiny testovacích snímků

Jak již bylo zmíněno v kapitole 3, zabýval jsem se kontrolou tří typů produktů. Množiny testovacích snímků jsem tvořil tak, že jsem využil již instalované systémy, které záznam snímků umožňují. Všechny tři byly postaveny na systémech od společnosti Cognex, konkrétně pak na programech Cognex In-Sight Spreadsheet. Dohromady jsem získal přibližně 200 000 unikátních snímků. Tyto snímky byly postupně získávány v časovém rozsahu přibližně tři měsíce, bylo zapotřebí sledovat plánování výroby a záznam snímků koordinovat s pracovníky, kteří zodpovídají za konkrétní výrobní linky. Takto dlouhý časový interval byl zapotřebí, protože v případě rozpoznávání typu ustřiženého kabelu se nevyrábí všechny typy výrobků současně. V případě kontroly pinů v konektoru výrobní linka vyrábí pouze velmi malé množství vadných kusů (přesnou hodnotu nemohu z důvodu utajení firemních informací uvést) a v případě kontroly přítomnosti plastové západky se jednalo o výrobní linku, která nebyla ještě zcela uvedena do provozu.

Část takto získaných snímků bylo potřeba roztřídit na jednotlivé kategorie. Nejprve jsem na základě snímků testovacích kusů a technické dokumentace k nim, analyzoval, jak se na snímcích výrobků projevují zaznamenané typy vad, abych byl schopný rozlišit OK a NOK kusy, v případě rozpoznávání typu ustřiženého typu kabelu jsem se zabýval odlišností mezi jednotlivými typy. Dále jsem navštívil konkrétní výrobní linky a ověřil jsem tyto informace na fyzických výrobcích.

Roztříděním do jednotlivých kategorií jsem získal 1 000 unikátních snímků od každé kategorie. Tento počet jsem vybral z časových důvodů. Vzhledem k tomu, že část snímků musela být zpracována ručně a zařazení některých snímků do správné kategorie není triviální, jedná se o časově velmi náročný proces. Uvědomuji si, že v případě průmyslového nasazení by bylo zapotřebí takto rozřadit snímků více. Zejména se jedná o snímky, které by sloužily pro testování. Nicméně pokud by ke skutečnému nasazení mělo dojít, nejspíše by bylo možné využít pracovníky, kteří pracují na oddělení kontroly kvality a jsou pro rozpoznávání vad a kategorií jednotlivých výrobků speciálně školeni.

V případě kontroly pinů v konektoru jsem anotoval 2 000 snímků, v případě rozpoznávání typu ustřiženého kabelu pak 6 000 snímků a v případě kontroly přítomnosti plastové západky 2 000 snímků. Jednotlivá kategorizace probíhala tak, že jsem částečně vycházel z apriorních informací a informací ze stávajících systémů. Přibližně třetina snímků musela být roztříděna ručně. Při ručním třídění jsem v případě kontroly pinů v konektoru konzultoval své rozhodování s pracovníky, kteří

se přímo zabývají kontrolou tohoto typu výrobků, protože se často vyskytují sporné případy. Takto získané snímky byly dále upraveny.

V případě kontroly pinů v konektoru byl využit Cognex In-Sight Spreadsheet. V programu In-Sight Spreadsheet byly vyříznuty pouze dvě oblasti zájmu, které zachycují dutinu konektoru (snímek 3.1). Oříznutí bylo provedeno tak, že se ve snímku hledaly v předpokládaném místě 2 dominantní kružnice. Na základu jejich pozice byla oblast vyříznuta a uložena. U rozpoznání typu ustřiženého kabelu se měnilo rozlišení na 800 x 600 px. Taktéž snímky z kontroly plastové západky byly pouze zmenšeny na rozlišení 128 x 216 px.

Ukázkové snímky jsou obsaženy v kapitole 3. V následujících tabulkách 5.1, 5.2 a 5.3 je uveden přehledný výčet roztříděných snímků v jednotlivých kategoriích a jejich rozdělení na snímky trénovací, validační a testovací. Rozřazení snímků na trénovací, validační a testovací proběhlo pomocí skriptu *random_vyber_ze_slozky.py* napsaném v jazyce Python 3.7. Pro správnou funkčnost skriptu je zapotřebí zadat množství snímků k přesunutí, cesty ke vstupní a výstupní složce. Následným spuštěním pak skript začne přesouvat zadané množství pseudonáhodně vybraných snímků ze zdrojové složky do složky cílové. Skript využívá knihovnu *random*, konkrétněji funkci *random.choice()*, která umožňuje pseudonáhodný výběr z listu.

	Trénovací	Validační	Testovací
Kategorie OK	700	200	100
Kategorie NOK	700	200	100

Tab. 5.1: Tabulka rozdělení snímků pro úlohu kontroly pinů v konektoru [počty snímků]

	Trénovací	Validační	Testovací
Kategorie OK	700	200	100
Kategorie NOK	700	200	100

Tab. 5.2: Tabulka rozdělení snímků pro úlohu kontroly přítomnosti plastové západky [počty snímků]

	Trénovací	Validační	Testovací
Kategorie 535	700	200	100
Kategorie 5352_aut	700	200	100
Kategorie 5352_man	700	200	100
Kategorie blue	700	200	100
Kategorie dacar	700	200	100

Tab. 5.3: Tabulka rozdělení snímků pro úlohu rozpoznávání typu ustřiženého kabelu [počty snímků]

Ve výsledku jsem tedy pořídil přibližně 200 000 unikátních snímků a část z nich jsem roztřídil do jednotlivých kategorií pro jednotlivé úlohy. Takto roztříděný dataset čítá 10 000 unikátních snímků. Následovalo pseudonáhodné rozřazení na snímky trénovací, validační a testovací v poměru 7:2:1.

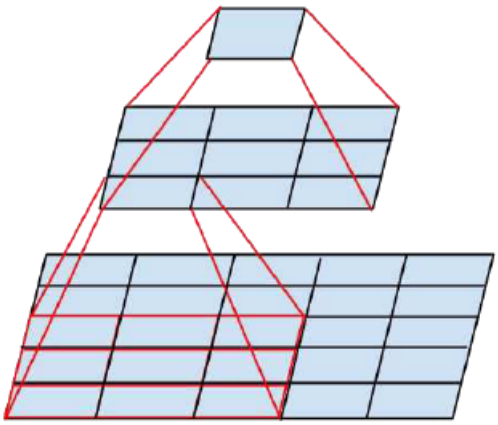
6 Implementace aplikace pro optickou klasifikaci výrobků

Nejdříve jsem experimentoval s vlastními architekturami konvolučních NN. Vyzkoušel jsem si klasickou úlohu rozpoznávání ručně psaných číslic s využitím volně dostupného MNIST datasetu [19], dále rozpoznávání psů a koček. Následně jsem zkoušel rozpoznávání obličejů. Vycházel jsem z veřejně dostupných tutoriálů od Harrison Kinsley [23] a placeného výukového kurzu PyImageSearch Gurus od Adrian Rosebrock [24]. Zde jsem došel k závěru, že pro složitější klasifikace bude vhodnější využít jednu z již existujících architektur. Po několika pokusech a na základě výsledků benchmarku [10] jsem zvolil architekturu NN Inception-V3.

6.1 Architektura neuronové sítě Inception-V3

Inception-V3 je architektura konvoluční neuronové sítě. Vychází z architektury Google net [25]. Architektura Inception-V3 obsahuje přibližně 24 000 000 parametrů, má 310 vrstev. Nabízí několik inovativních řešení.

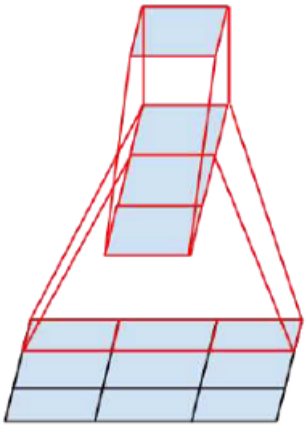
Jedním z nic je faktorizace velkých konvolučních filtrů [20] [21] do několika menších. Znázorněno na nákresu 6.1. Tímto dochází ke snížení počtu parametrů, v případě konvoluční vrstvy z 25 (5 x 5) na 18 ($3 \times 3 + 3 \times 3$).



Obr. 6.1: Faktorizace konvoluční vrstvy [21]

Další z nich je asymetrická faktorizace. Ta funguje obdobně jako faktorizace symetrická, ale využívá nesymetrické konvoluční masky. V případě, kdy chceme snížit

počet parametrů pro masku 3 x 3, lze ji rozdělit na dvě masky 1 x 3 a 3 x 1. Tímto krokem snížíme původní počet parametrů z 9 (3 x 3) na 6 ($3 \times 1 + 1 \times 3$). Ukázáno na následujícím nákresu 6.2.

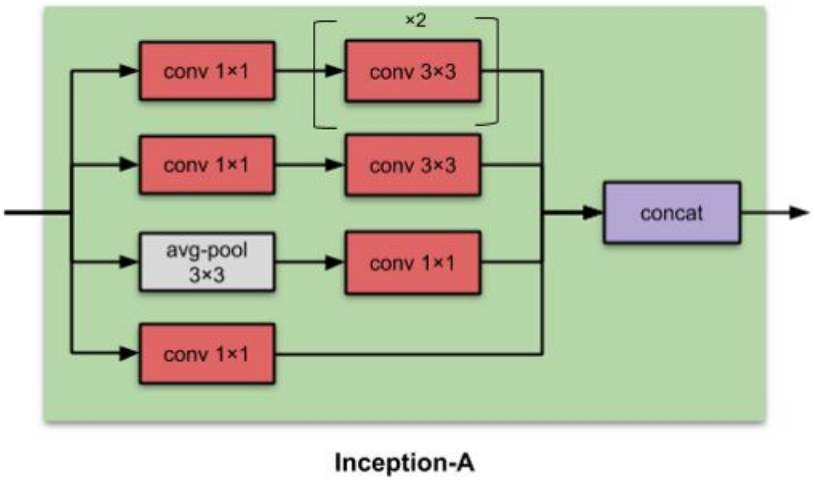


Obr. 6.2: Faktorizace asymetrické konvoluční vrstvy [21]

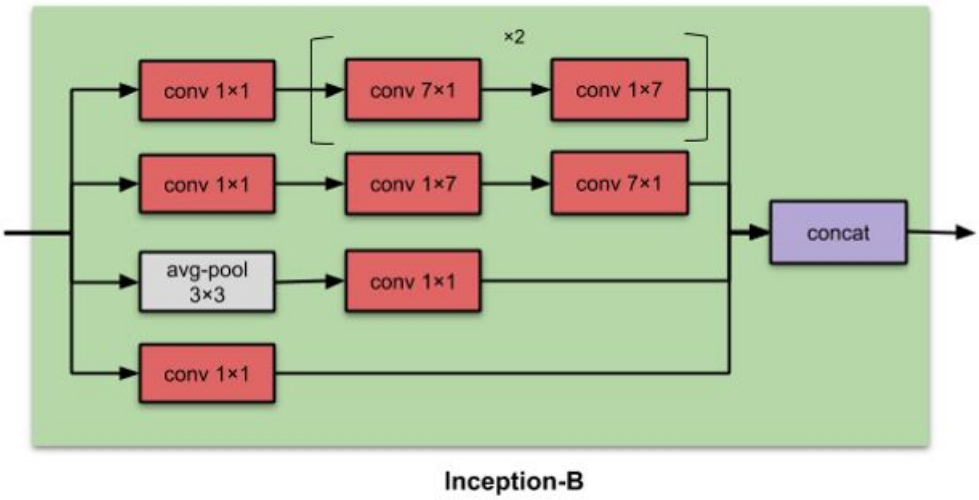
Takto poskládané konvoluční filtry se dále skládají do tzv. Inception modulů. Inception modul je složen z několika paralelních konvolučních filtrů o různých velikostech. V architektuře inception verze 3 se nalézají 3 typy inception modulů. Základním z nich je blok Inception typ A. Skládá ze 3 paralelních konvolučních filtrů o různých velikostech. Konkrétně 1 x 1, 3 x 3 a 5 x 5. Díky tomu je schopný detekovat jak znaky, které jsou na malé ploše, tak ty, které se projevují na velké ploše. Příkladem může být detekce psa. Pes se na snímku může nalézat jak v pozadí, tak v popředí a být tedy přes celý snímek [22]. Přeneseno na detekci vad v průmyslu, znečištění lubrikantem může být jak na malé, tak i na velké ploše povrchu testovaného výrobku. Tento blok je konstrukčně starší, vychází z první verze této architektury, optimalizováno pomocí symetrické faktorizace. Znázorněno na následujícím nákresu 6.3.

Další dva typy bloků, tj. blok B se zabývá detekcí větších znaků. Využívá asymetrické faktorizace. Základem jsou konvoluční filtry o velikosti 7 x 7 (přesná velikost záleží na implementaci), kdy v jedné větvi jsou dva spojené sériově a ve druhé se nalézá pouze jeden. Znázorněno na následujícím nákresu 6.4. Blok C se zabývá propagací vyšších dimenzí.

Takto vytvořené bloky se skládají za sebe. Znázorněno na následujícím nákresu 6.5.

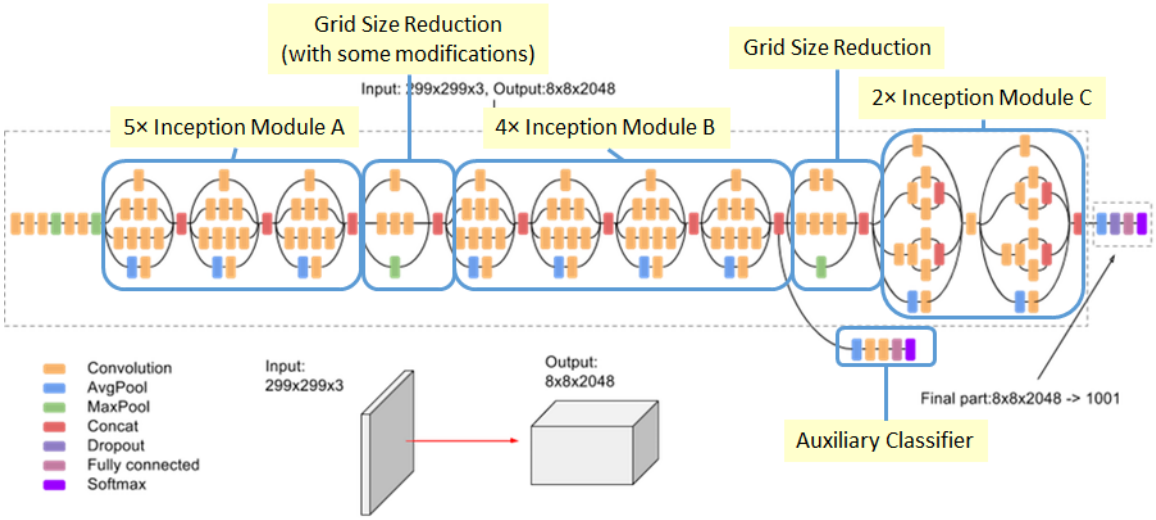


Obr. 6.3: Blok Inception typ A [26]



Obr. 6.4: Blok Inception typ B [26]

Na tomto nákresu 6.5 je taktéž vidět další inovativní prvek. Je to tzv. auxiliary classifiers [25], a to pouze před poslední vrstvu. Ten slouží v průběhu učení. Častým problémem u velmi hlubokých neuronových sítí je to, že se učí velmi pomalu tzv. Vanishing Gradient Problem. Gradient se zmenšuje při postupování hlouběji do sítě při učení. Tomu lze částečně zamezit zavedením dalšího výstupu, který vychází zevnitř sítě a ne až z jejího konce. Tento výstup je pak součástí učícího procesu. Dalším prvkem, který byl obsažen, aby urychlil učení je tzv. Batch Normalization, detailně popsáno v původním článku [27].



Obr. 6.5: Nákres architektury Inception-V3 [21]

6.2 Instalace softwaru pro učení a inferenci NN

V případě, že chceme využít pro učení modelu grafickou kartu NVIDIA z pracovní stanice (část 4.1.4), je zapotřebí nainstalovat značné množství podpůrného softwaru.

6.2.1 Knihovna NumPy

NumPy je knihovna spadající pod BSD licenci. Umožňuje práci s vektory, maticemi a vícerozměrnými poli. V této diplomové práci ji využívám ve skriptech pro inferenci. Slouží pro úpravu vstupních snímků do tvaru, který je možno použít jako vstup do neuronové sítě [34].

6.2.2 Knihovna OpenCV

OpenCV je open source knihovna pro počítačové vidění. Obsahuje přibližně 2 500 optimalizovaných algoritmů pro zpracování obrazu. Nicméně v rámci této diplomové práce je využita pro měření času. Využívá se pro zjištění délky trvání načítání modelu neuronové sítě a doby, která je potřebná pro infernci [31].

6.2.3 Platforma NVIDIA CUDA

NVIDIA CUDA je hardwarová i softwarová platforma, díky které je možné využívat grafickou kartu pro spouštění programů napsaných v programovacích jazycích *C, C++, Python* a dalších. Hlavní výhodou spouštění programů na grafické kartě je, že architektura grafické karty umožňuje chod násobně více paralelních výpočtů než je tomu u procesorů. [29]

6.2.4 Knihovna NVIDIA CUDNN

Knihovna *NVIDIA CUDNN* využívá platformu *NVIDIA CUDA*. *CUDNN* obsahuje základní primitiva pro hluboké neuroné sítě, poskytuje vysoce optimalizované implementace běžných postupů jako jsou například konvoluce a aktivační vrstvy. Pomocí knihovny *CUDNN* je optimalizovaný chod frameworků pro umělou inteligenci jako je například *TensorFlow, Cafe, Matlab, Pytorch* a další [30].

6.2.5 Knihovna TensorFlow

TensorFlow je open source knihovna, která zjednodušuje vývoj aplikací založených na neuronových sítích. Je vyvíjena výzkumným týmem Google Brain. Podporuje programovací jazyky *Python, JavaScript, C++, Java* a *Go*. V rámci této diplomové práce se využívá na více místech, předně pro nadřazené rozhraní *Keras* a dále při optimalizaci modelu neuronové sítě (popsáno v další části této práce, v kapitole 7), kde je pracováno s knihovnou přímo [32].

6.2.6 Rozhraní Keras

Keras je programovací rozhraní, které zjednodušuje práci s knihovnou *TensorFlow*. Podporuje programovací jazyk *Python*. Je oficiálně podporované vývojáři knihovny *TensorFlow*. V této práci je využíván ve skriptech pro učení i inferenci NN [33].

6.2.7 Knihovna EasyGui

EasyGui je knihovna, která umožňuje využití jednoduchých prvků ze systémového grafického rozhraní, jako jsou například dialogová okna pro zvolení souboru. Tohoto okna využívám v případě skriptu pro optimalizaci modelu [35].

6.2.8 Kompatibilita jednotlivých verzí softwaru

Pro správnou funkčnost jednotlivých skriptů, použitých v této diplomové práci, je zapotřebí nainstalovat jednotlivý software ve verzích, které jsou vzájemně kompatibilní. Tato oblast bohužel není velmi dobře dokumentovaná u oficiálních zdrojů a občas jsou tyto informace dokonce v rozporu s reálnou funkčností, a tak je potřeba kompatibilitu jednotlivých verzí ověřit experimentálně.

Zvolena byla tato stabilní kombinace:

- NumPy 1.17.4
- Keras 2.3.1
- TensorFlow 2.0
- Cuda 10
- CUDNN 7.6.0
- OpenCV 4.1.2.30
- EasyGui 0.98.1

6.3 Učení neuronové sítě architektury Inception-V3

6.3.1 Skript pro učení neuronových sítí architektury Inception-V3

Jak již bylo zmíněno v předchozí kapitole, skript pro svoji funkci využívá platformu *CUDA* a na ni navázanou knihovnu *CUDNN*, kterou využívá knihovna *TensorFlow*, se kterou pracuje rozhraní *Keras*. Skript je napsán v programovacím jazyce Python 3.7. Při psaní tohoto skriptu jsem vycházel z příkladu, který je uveden v oficiální dokumentaci k rozhraní Keras [36] a z knihy „Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API“. Samotný skript je přiložený v přílohách této práce pod názvem *Inceptionv3_learning.py*

6.3.2 Spuštění skriptu pro učení neuronových sítí architektury Inception-V3

Základem je dostatečně výkonný hardware, ten zajišťuje pracovní stanice z části 4.1.4, zejména grafická karta NVIDIA GeForce RTX 2080 Ti. Následuje nastavení

vstupních informací skriptu. Jedná se zejména o název výstupního modelu, zaznaménávaných logů s informacemi o průběhu učení (logování umožňuje *TensorBoard*, což je část knihovny *TesnorFlow*), dále je zapotřebí zadat cestu k trénovacím a testovacím snímkům a názvy jednotlivých kategorií. Ve skriptu je také možné upravovat parametry učení.

Výčet jednotlivých úloh pro rozpoznávání kategorií a rozdělení na trénovací a validační snímky je podrobně rozepsán v předešlých tabulkách 5.1, 5.2 a 5.3.

Následně je již možné učení spustit. Nejprve dojde k inicializaci jednotlivých knihoven, následně se incializuje architektura neuronové sítě, v případě Inception-v3 se jedná o 310 jednotlivých vrstev. Poté dochází k nahrávání modelu a obrázků z datasetu do grafické paměti. Dále již nastává samotný proces učení. Učení končí po vykonání zadaného počtu etap. Výsledkem jsou uložené modely. Model se ukládá po každé ukončené etapě (lze změnit nastavením parametrů ve skriptu). V případě použití největších obrázků (800 x 600 px) z kontroly typu ustřiženého kabelu, kterých je zároveň díky šesti kategoriím nejvíce, trvá výpočet jedné etapy přibližně 4,5 minuty. Pro dvě ze tří úloh jsem nastavil 200 epoch a pro jednu zbývající 150 epoch.

6.4 Výběr optimálního naučeného modelu

Po dokončení učení jsem v grafech z *TensorBoard* zjistil průběh acuracy a los funkce pro trénovací a validační množinu a na základě těchto dat jsem se rozhodl pro konkrétní model. Jednotlivé grafy jsou obsaženy v příloze A. Základním ukazatelem pro mě bylo to, že se neuronová síť nepřeučila. To se pozná tak, že zatímco stoupá accuracy u tréninkové množiny snímků, accuracy pro validační množinu snímků klesá. Dále lze z grafů zjistit, zda jsme nenastavili příliš malé množství epoch. To poznáme tak, že se hodnoty jednotlivých parametrů ustálí. Při žádném učení pro jednotlivé úlohy nenastalo ani přeučení ani neustálení parametrů. Můžeme tedy přejít k výběru optimálního modelu. Model vybíráme na základě dvou hledisek. Je žádoucí, aby měl vybraný model co největší hodnotu accuracy pro tréninkovou i testovací množinu snímků. To znamená, že u tohoto modelu nastal nejmenší počet chybných predikcí. Dalším parametrem, který sledujeme je loss funkce. Ta udává o jakou hodnotu jsme se v případe chybné predikce zmýlili. Hodnota loss funkce by tedy měla být co nejmenší pro trénovací i validační množinu snímků, což značí, že v případě, že došlo k chybě, byla tato chyba malá. Hodnoty accuracy a loss funkce pro vybrané, naučené modely, pro jednotlivé úlohy, jsou znázorněny v následujících tabulkách 6.1, 6.2 a 6.3.

	Trénovací snímky	Validační snímky
Loss funkce [-]	0,0145	0,0303
Accuracy [-]	0,9945	0,9783

Tab. 6.1: Tabulka loss funkce a accuracy pro trénovací a validační snímky pro úlohu kontroly pinu v konektoru - model z epochy 139

	Trénovací snímky	Validační snímky
Loss funkce [-]	0,0020	0,0014
Accuracy [-]	0,9998	1,0000

Tab. 6.2: Tabulka loss funkce a accuracy pro trénovací a validační snímky pro úlohu rozpoznávání typu ustřiženého kabelu - model z epochy 145

	Trénovací snímky	Validační snímky
Loss funkce [-]	0,0015	0,0000
Accuracy [-]	1,0000	0,9999

Tab. 6.3: Tabulka loss funkce a accuracy pro trénovací a validační snímky pro úlohu kontroly přítomnosti plastové západky - model z epochy 138

6.5 Ověření výsledků klasifikace

Po naučení modelu NN je zapotřebí model ověřit na snímcích, které nebyly využity při učení. Tyto snímky jsou označeny jako testovací (při procesu učení se využívá trénovacích a validačních snímků). Postup tohoto ověření je následující. Spustíme inferenci pro jednotlivé testovací snímky a výsledky ukládáme, abychom je pak mohli následně vyhodnotit.

6.5.1 Skript pro inferenci neuronových sítí architektury Inception-V3

Skript využívá stejných knihoven a podpůrného softwaru jako skript pro učení. Pouze se přidává knihovna *OpenCV* pro měření času a *NumPy* pro úpravu snímků. Při tvorbě skriptu jsem vycházel z oficiální dokumentace k rozhraní Keras [36] a z knihy „Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API“. Samotný skript je přiložený v přílohách této práce pod názvem *Inceptionv3_predict.py*. Skript umožňuje, jak spouštění na grafické kartě pro vyšší výkon, tak vynucení spuštění pouze na procesoru. Je zapotřebí zadat cestu k modelu, pomocí kterého chceme inferenci uskutečnit, dále cestu ke složce se snímky, které chceme klasifikovat, cestu a název

souboru, do kterého chceme zapsat výsledky inference a zadat rozlišení vstupních snímků a vypsat kategorie, do kterých bude skript snímky rozřazovat.

Po spuštění proběhne načtení snímků ze zadané složky do seznamu snímků. Následně se inicializují knihovny a ostatní podpůrný software. Následuje načtení modelu, vypsání seznamu názvů načtených snímků a samotný běh inference. Poté, co inference skončí skript postupně vypíše jednotlivé predikované kategorie v podobě číselného indexu. 0 reprezentuje první kategorii. Pořadí kategorií je určeno při učení modelu. Následně skript vypíše počet proběhlých predikcí a počty snímků, které byly predikovány do konkrétní kategorie. Dále pak průměrný čas, který byl potřebný na jednu predikci. Například:

```
0 0 0 0 0 1 0 0 0 0

Pocet predikci: 10

Label: 0 Pocet: 9
Label: 1 Pocet: 1

Prumerny cas na jednu predikci [ms]: 13
```

V rámci skriptu jsou výsledné predikce ve dvojím formátu, a to jak ve formě listu číselných hodnot, tak ve formě listu, kde jsou jednotlivé hodnoty uvedeny jako znak. Je tedy snadné s nimi následně pracovat. Výsledek inference se dále uloží do textového souboru, který byl zadán před spuštěním.

6.6 Výsledky klasifikace natrénovaných neuronových sítí

Jak již bylo zmíněno dříve, pro ověření se využívá snímků z datasetu určeného pro testování. Testovací dataset obsahuje pro každou kategorii jednotlivých úloh 100 unikátních snímků. Postupným spouštěním skriptu pro jednotlivé úlohy a kategorie získáme výsledky klasifikace, které se vynesou do tzv. confusion matrix. Buňky matice obsahují četnosti toho, kolikrát došlo na zkoumané datové množině k dané kombinaci skutečné a predikované kategorie. Správné četnosti se nachází na hlavní diagonále této matice, protože se schoduje predikovaná se skutečnou kategorií. Z této matice se dále dá spočítat accuracy a presision [2]. Accuracy udává poměr počtu správně klasifikovaných snímků ku celkovému počtu snímků v testovacím datasetu. Precision je specifická pro každou kategorii samostatně, udává nám poměr

mezi počtem správně klasifikovaných snímků konkrétní kategorie a celkovým počtem snímků, kterým byla tato konkrétní kategorie predikována.

6.6.1 Výsledky klasifikace pro úlohu rozpoznávání typu ustřiženého kabelu

Výsledná confusion matrix 6.4 ukazuje, že pro testovací dataset predikuje tato neuronová síť bezchybně. Všech 600 snímků bylo klasifikováno správně. Z této tabulky taktéž vyplývá, že accuracy pro tento testovací dataset je rovna 100% a precision pro všechny kategorie taktéž rovna 100%

	predikováno neuronovou sítí					
	535	5352_aut	5352_man	blue	dacar	HT
535 skutečné	100	0	0	0	0	0
5352_aut skutečné	0	100	0	0	0	0
5352_man skutečné	0	0	100	0	0	0
blue skutečné	0	0	0	100	0	0
dacar skutečné	0	0	0	0	100	0
HT skutečné	0	0	0	0	0	100

Tab. 6.4: Confusion matrix pro úlohu rozpoznávání typu ustřiženého kabelu [počet snímků]

6.6.2 Výsledky klasifikace pro úlohu kontroly přítomnosti plastové západky

Výsledná confusion matrix 6.5 ukazuje, že pro testovací dataset predikuje tato neuronová síť bezchybně. Všech 200 snímků bylo klasifikováno správně. Z této tabulky taktéž vyplývá, že accuracy pro tento testovací dataset je rovna 100% a precision pro všechny kategorie taktéž rovna 100%

	predikováno neuronovou sítí	
	OK	NOK
OK skutečné	100	0
NOK skutečné	0	100

Tab. 6.5: Confusion matrix pro úlohu kontrolu absence plastové západky [počet snímků]

6.6.3 Výsledky klasifikace pro úlohu kontroly pinů v konektoru

Tato úloha je podle výsledků nejproblematičtější. Nasvědčovaly tomu již parametry modelu při učení, které byly horší než u modelů pro ostatní úlohy (tabulka 6.1). Výsledná confusion matrix 6.6 to potvrzuje. V případě predikce pro snímky z testovacího datasetu, které by měly být klasifikovány jako OK, byly klasifikovány 3 chybně. Pro snímky, které měly být klasifikovány jako NOK kusy proběhla klasifikace správně. Toto je naštěstí pro průmysl výhodnější stav, protože nenastane situace, že by byl špatný výrobek klasifikován jako správný, pouze se kus, který mohl být prodán, zařadí mezi zmetky. Accuracy je v tomto případě 98,5%, precision pro je OK kategorii je 97,1% a pro NOK kategorii 100%.

	predikováno neuronovou sítí	
	OK	NOK
OK skutečné	97	3
NOK skutečné	0	100

Tab. 6.6: Confusion matrix pro úlohu kontrolu pinů v konektoru [počet snímků]

Existuje několik variant, proč neuronová síť pro testovací dataset neklasifikuje 100%. Jako nejpravděpodobnější se mi jeví ta varianta, že je chyba v anotaci snímků pro tuto úlohu. Jak jsem již uvedl v kapitole 5 mnohé snímky byly sporné. Bylo by sice možné tyto snímky z datasetu vyřadit, a namísto toho umístit snímky, u kterých je jejich rozřazení jasné, nicméně pro nasazení ve výrobě by to bylo takovéto řešení nepřijatelné. Další z variant je nedostatečně velký trénovací a validační dataset snímků. Tuto možnost nemám z časových důvodů jak ověřit, nicméně předpokládám, že se jí budu zabývat v budoucnosti. Snímky, které byly chybně klasifikovány jsou

následující *piny_0175.jpg*, *piny_1372.jpg* a *piny_1202.jpg*. U čísla 1202 si troufám říci, že se jedná o hraniční případ. U zbylých dvou se sice nejedná o ideální případy OK kusů, nicméně pracovník kvality by je rozpoznal správně.

7 Optimalizace natrénovaného modelu neuronové sítě pro NVIDIA Jetson Xavier

Pro zvýšení rychlosti inference na již naučeném modelu neuronové sítě je možné tento model optimalizovat. V případě zařízení od společnosti NVIDIA je možno využít jejich knihovnu *TensorRT*.

7.1 TensorRT

TensorRT [7] je optimalizační knihovna pro NVIDIA grafické karty a NVIDIA zařízení z řady Jetson. Knihovna podporuje optimalizaci modelů, vytvořených ve většině knihoven pro neuronové sítě. Jsou podporovány modely NN z knihoven *TensorFlow, Caffe, PyTorch, MXNet* a dalších.

Základem této knihovny je, že optimalizuje model neuronové sítě na konkrétní hardwarovou platformu. Optimalizované modely tedy nejsou přenosné mezi různými modely zařízení. Stejně tak jsou optimalizované modely vázány na konkrétní verzi *TensorRT*.

Samotná optimalizace probíhá tak, že se načte podporovaný model, odstraní se vrstvy a operace, které nemají žádný efekt na výsledek inference. Následně spojí konvoluci, bias a ReLu operace do bloků. Dále v závislosti na použité hardwarové platformě zvolí optimální algoritmy a datovou strukturu. Také minimalizuje potřebnou velikost paměti tím, že zřetězí výstupy z některých vrstev, a pokud používá stejná data více operací zamezí jejich duplikaci. Dalším krokem je optimalizace změnou velikosti jednotlivých proměnných. Pro zařízení Jetson Xavier je nejvýhodnější zvolit INT16 (popsáno dále v části 7.5.1).

Pro zachování funkčnosti modelu i pro optimalizaci a pro zlepšení výkonu, knihovna využívá toho, že nejprve vygeneruje testovací data a iterativně testuje na skutečném hardwaru, jaké nastavení a jaké optimalizace fungují nejlépe při zachování přesnosti modelu.

Podrobnější popis toho, jak optimalizace pomocí knihovny *TensorRT* funguje je zachycen na přednášce od Dmitry Korobchenko. Dmitry Korobchenko je manažer vývojového týmu společnosti NVIDIA, který se specializuje na deep learning. Přednáška proběhla v rámci konference Data Summer Conf 2018. Záznam je dostupný

na [39].

Co se týče instalace knihovny TensorRT, tak tuto knihovnu není potřeba instalovat, neboť se nainstaluje automaticky při instalaci Jetpack SDK do zařízení Jetson AGX Xavier. V mém případě se nainstalovala verze TensorRT 5.0.6. Tato verze je plně kompatibilní s ostatním podpůrným softwarem z kapitoly 6.2.8 s výjimkou rozhraní *Keras*.

Modely, které vzniknou prostřednictvím rozhraní *Keras* totiž v případě architektury NN Inception-V3 obsahují některé z funkcí, které nejsou knihovnou *TensorRT 5.0.6* podporovány. Seznam podporovaných funkcí je dostupný na [38]. Abychom mohli model z rozhraní Keras využít, je zapotřebí ho převést na model, který odpovídá modelům, které vznikají při použití samotné knihovny *TensorFlow*. Dále je nutno pojmenovat výstupní vrstvu modelu již při jeho prvotním učení v rozhraní *Keras*. Pokud je výstupní vrstva bezejmenná nijak to neovlivňuje využití modelu k inferenci. Zamezí to však tomu, aby mohl být model převeden na model, který odpovídá modelu, který vznikl v samotném *TensorFlow*.

7.2 Skript pro optimalizace modelu NN

Pokud máme připravený model, který vznikl v rozhraní *Keras*, můžeme přistoupit k samotnému procesu optimalizace pomocí navrženého skriptu *OptimizeKerasModelToTRT.py*. Při tvorbě skriptu jsem vycházel z oficiální dokumentace [7]. Skript využívá knihoven *OpenCV* pro měření času, dále *Tensorflow* a *TensorRT* pro samotnou práci s modely a jejich optimalizaci, dále *EasyGui* pro prvky grafického rozhraní.

Optimalizace jako taková je náročná na velikost operační paměti zařízení. Pro modely, se kterými pracuji, je vyžadováno přibližně 20 GB. Zařízení NVIDIA Jetson AGX Xavier první revize má však pouze 16 GB operační paměti. Při pokusu o optimalizaci s nedostatečnou pamětí, proces využije veškerou operační paměť a zařízení přestane reagovat. Je tedy zapotřebí aktivovat funkci *SWAP*, která umožní odkládání dat z operační paměti na úložiště. Pro Jetson AGX Xavier se tato funkce aktivuje zadáním následujících příkazů do terminálu.

```
sudo fallocate -l 10G /media/nvidia/SSD/swapfile
sudo chmod 600 /media/nvidia/SSD/swapfile
sudo mkswap /media/nvidia/SSD/swapfile
sudo swapon /media/nvidia/SSD/swapfile
```

Po zadání těchto příkazů se aktivuje swap funkce. Virtuálně jsme rozšířili operační paměť o 10 GB. Data z operační paměti jsou odkládána na instalované SSD úložiště. Výhodou je, že je instalovaný SSD disk připojen přes podporované rozhraní M.2 NVMe, úložiště má díky tomu kratší přístupovou dobu než v případě staršího rozhraní M.2 SATA.

Po spuštění nás skript vyzve ke zvolení cesty vstupního *Keras* modelu. Po načtení je model otevřen, jsou z něj vyčteny naučené váhy, architektura a název výstupní vrstvy a je uložen ve formátu tzv. frozen grafu *TensorFlow*. Tento frozen graf je následně načten a optimalizován knihovnou *TensorRT*. Po dokončení optimalizace je uložen a je vypsáno jak dlouho optimalizace trvala. V případě modelů, které vznikly podle popisu z předchozích kapitol, trvá optimalizace přibližně 4 minuty.

7.3 Skript pro inferenci NN po optimalizaci knihovnou TensorRT

Abychom mohli provést inferenci modelu, který byl optimalizován pomocí knihovny TensorRT, je pro to zapotřebí napsat samostatný skript. Skript z kapitoly 6.5.1 využít nelze. Při tvorbě skriptu *InceptionV3_predict_Jetson_TRT.py* jsem opět vycházel z oficiální dokumentace [7]. Skript využívá knihovnu *NumPy* pro převod snímku do formátu, který lze použít jako vstup do neuronové sítě, dále pak *OpenCV* pro měření časových intervalů, *Tensorflow* a *TensorRT* pro inferenci pomocí neuronové sítě.

Před samotným spuštěním skriptu je do něj zapotřebí zadat cestu ke složce se snímky, pro které proběhne predikce kategorií a cestu odkazující na optimalizovaný model. Po spuštění skriptu, skript nejprve načte jednotlivé knihovny. Následně předpřipraví prázdný model a do něj poté načte hodnoty z modelu uloženého. Následně načte vstupní a výstupní vektor z modelu. Tímto je model načten. Samotné načtení modelů, se kterými pracuji trvá přibližně 3 minuty. Dále následuje inference pro jednotlivé snímky. Na závěr skriptu se vypíše průměrný čas inference.

7.4 Skript pro inferenci neoptimalizovaných NN

Tento skript *InceptionV3_predict_Jetson.py* je téměř shodný se skriptem z části 6.5.1, liší se pouze tím, že se odlišně přidávají knihovny při inicializaci programu.

7.5 Výsledky optimalizace modelů pomocí knihovny TensorRT

Postupným spouštěním inference pro optimalizované i neoptimalizované verze modelů z kapitoly 6.4 a zaznamenáváním jednotlivých časů jsem zjistil následující hodnoty, zaznamenáno v tabulce 7.1.

	Čas inference před optimalizací [ms]	Čas inference po optimalizaci [ms]
Model pro kontrolu pinů v konektoru	71	17
Model pro rozpoznávání typu ustřiženého kabelu	114	37
Model pro kontrolu přítomnosti plastové západky	67	23

Tab. 7.1: Tabulka časů potřebných pro inferenci před a po optimalizaci

Využitím optimalizace pomocí knihovny *TensorRT* lze tedy snížit délku trvání inference v závisloti na konkrétním modelu o 65-76%. Pro srovnání uvádím, že inference modelu bez optimalizace z úlohy rozpoznávání typu ustřiženého kabelu trvá na pracovní stanici z kapitoly 4.1.4 v případě výpočtu na procesoru 118 ms a v případě výpočtu na grafické kartě 26 ms.

7.5.1 Vliv velikosti proměnných v optimalizovaném grafu NN

Před spuštěním inference umožňuje knihovna *TensorRT* nastavit velikost jednotlivých proměnných ve výsledném optimalizovaném grafu. Nabízí 3 varianty osmibitový Int8, šesnáctibitový Int16 a třicetidvoubitový FP32 s pohyblivou řádovou čárkou. Ověřil jsem jaké má toto nastavení vliv na čas inference. Využil jsme model z úlohy rozpoznávání typu ustřiženého kabelu. V případě nastavení Int8 trvala inference 4,2 s, jak již bylo zmíněno, pro Int16 trvala 37 ms a pro FP32 přestane zařízení reagovat a je nutno ho restartovat. Je tedy vidět, že pro zařízení Jetson AGX Xavier je nejvýhodnější zvolit proměnné typu INT16.

8 Srovnání optické klasifikace klasickým způsobem a pomocí NN pro úlohu kontroly pinů v konektoru

8.1 Popis původního řešení kontroly pinů v konektoru - HW

Původní řešení vizuální kontroly tohoto konektoru bylo dodáno společností EOLA s.r.o. Bylo založeno na chytré kameře Cognex 7200C. Maximální rozlišení této kamery je 800 x 600 px. K této kameře byl připojen objektiv s pevným ohniskem. Protože nebyla zachována doporučená vzdálenost výrobku od objektivu, bylo zapotřebí upravit hloubku ostrosti. Úprava hloubky ostrosti byla provedena montáží distančního kroužku mezi kameru a objektiv. Toto řešení se ukázalo být nevhodné. Distanční kroužky sice přesunou ostřicí vzdálenost blíže k objektivu, nicméně razantně snižují hloubku ostrosti objektivu jako takovou, což znamená, že při pohybu kontrolovaného dílu nahoru nebo dolů (způsobeno nedokonalým úchytem konektoru v kleštinách výrobní linky) vznikají rozostřené snímky a dále vlivem typu standardního entocentrického objektivu se mění i rozměry jednotlivých prvků na snímku. Toto zkreslení je přílišným přiblížením výrobku a širokým ohniskem objektivu o to více razantní. Dalším problémem bylo zvolené osvětlení, protože piny, které se nacházejí v dutině byly nerovnoměrně a pouze slabě nasvětlené. Teoreticky bylo sice správně zvoleno koaxiální osvětlení, nicméně nebylo nejspíše provedeno dostatečné ověření v optické laboratoři. Snímky z původního řešení bohužel již nejsou k dispozici.

8.2 Popis úpravy původního řešení kontroly pinů v konektoru - HW

Pro zlepšení funkce tohoto systému bylo vyměněno osvětlení za tzv. low angle ring light. Dále byl vyměněn objektiv za telecentrický. Ukázkový snímek se nalézá v kapitole Výběr vhodné třídy výrobků 3.1.

8.3 Popis původního řešení kontroly pinů v konektoru - SW

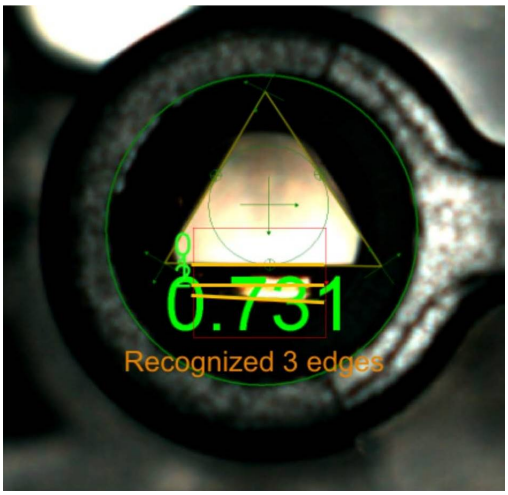
Další problematickou částí tohoto projektu byla programová část. Jak již bylo zmíněno dříve, systém kontroly je založen na chytré kameře Cognex 7200C. Společnost Cognex k těmto kamerám dodává dvě varianty programovacího prostředí a to Cognex In-Sight EasyBuilder, které je určeno pro jednodušší kontroly a dále pak Cognex In-Sight Spreadsheet pro náročnější inspekce. Společnost EOLA s.r.o. navrhovala své programové řešení v systému In-Sight Spreadsheet verze 4.09.01. Problémem bylo to, že dodané řešení nepracovalo spolehlivě i po úpravě nasvětlení a optiky.

8.4 Popis úpravy původního řešení kontroly pinů v konektoru - SW

Softwarové řešení bylo tedy upraveno. Využívá se toho, že se nedetekuje pouze jedna hrana pro konkrétní pin, ale více výrazných hran v oblasti. Těmto hranám se následně přiřadí skóre. Toto skóre je dáno kombinací jednotlivých hodnotících parametrů. Zvažuje se výraznost hrany. Výraznost je vypočítána na základě histogramu. Dalším parametrem je blízkost k další nejbližší hraně, penalizuje se více hran blízko u sebe, což obvykle značí, že se detekují hrany na odlesku nebo stínu. Posledním sledovaným parametrem je úhel detekovaných hran. Z popisu konektoru víme, že má být uprostřed dutiny část rovnostranného trojúhelníku. Vizualizace takto upravené kontroly je na snímku 8.1.

Na tomto snímku lze vidět, že program detekoval tři hrany, z čehož jedna je správně určena na hraně pinu, další dvě jsou chybně určeny v odlesku. Na základě skórovacího systému se pak následně určí, která z nich se má využít pro další zpracování.

I přes tyto úpravy je tato optická kontrola stále problematická, protože se liší vstupní suroviny v závislosti na konkrétní dodávce, čímž se mění optické vlastnosti jako je například odrazivost. V důsledku takto se měnících podmínek, je zapotřebí, aby pracovník, který spravuje tuto optickou kontrolu, pravidelně upravoval parametry v nastavení optické kontroly.



Obr. 8.1: Ukázkový snímek upravené kontroly v programu In-Sight Spreadsheet

8.5 Samotné srovnání klasického systému a systému, který využívá NN

Úpravou tohoto programu jsem docílil toho, aby bylo možno využít dataset testovacích snímků vytvořených pro ověření neuronové sítě. Program je přiložen v přílohách pod názvem *piny_v_konektoru_jeden kontakt.job* Následně jsem nechal program zkontrolovat 100 NOK a 100 OK snímků z testovacího datasetu. Výsledky jsem zaznamenal do následující confusion matrix 8.1.

	predikováno v systému Cognex In-Sight Spreadsheet	
	OK	NOK
OK skutečné	99	2
NOK skutečné	1	98

Tab. 8.1: Confusion matrix pro úlohu kontrolu pinů v konektoru pomocí systému Cognex In-Sight Spreadsheet [počet snímků]

Z tabulky vidíme, že systém chybně predikoval 3 snímky z celkového počtu 200 snímků. Problematický je zejména ten, který byl klasifikován jako OK a přitom byl ve skutečnosti vadný, tedy NOK. V takovémto případě by bez další kontroly mohlo dojít k prodání vadného kusu zákazníkovi, což by zapříčinilo finančně náročnou reklamaci celé dodávky. Accuracy je pro tuto tabulku 94,2%, precision pro NOK kusy

je 98,0% a pro OK 99,0%.

Pokud srovnáme tuto optickou kontrolu založenou na klasickém přístupu s kontrolou založenou na neuronových sítích, můžeme říci, že v případě tohoto testovacího datasetu snímků je vhodnější použít neuronovou síť, protože nezařadila žádný NOK snímek mezi OK snímky. Nicméně se domnívám, že pro plnohodnotné srovnání by bylo zapotřebí použít mnohem více snímků. Pro tento testovací dataset si troufám říci, že jsou obě metody přibližně srovnatelné.

Jako velká výhoda klasického přístupu se mi jeví možnost zpětně zjistit na základě čeho systém vyhodnotil konkrétní snímek daným způsobem. U neuronové sítě je zpětné zhodnocení bez použití pokročilých metod nemožné [42].

Nevýhodou klasického přístupu je případná velká časová náročnost. Na obrazové kontrole pinů v konektoru pracovala odborná firma a následně pracovníci z oddělení obrazové kontroly společnosti TE Connectivity dohromady po dobu delší než jeden rok, i přes to, je zapotřebí systém neustále udržovat. Po získání dostatečně velkého anotovaného datasetu snímků je vytvoření systému, který by využíval neuronové sítě otázkou měsíce, a to včetně ověření. Dokonce lze využít i stávajících Cognex systémů, protože kamera umožňuje odesílání pořízeného snímku přes ftp a zpětnou komunikaci výsledků zajišťuje komunikace TCP/IP.

Co se týče rychlostního srovnání, tak na pracovní stanici z kapitoly 4.1.4 trvá vyhodnocení jednoho snímku 13 ms, tatáž kontrola trvá na této pracovní stanici pomocí neoptimalizované neutronové sítě 9 ms. Na NVIDIA Jetson AGX Xavier pak 17 ms při využití optimalizovaného modelu.

Závěr

Cílem této diplomové práce bylo umožnit posuzování kvality nebo typu výrobku v průmyslových aplikacích pomocí umělých neuronových sítí. Zejména u těch aplikací, u kterých by byl klasický přístup strojového vidění příliš náročný. Systém založený na neuronových sítích byl dále implementován na konkrétní hardwarovou platformu. Zároveň byla u něj provedena optimalizace výsledného modelu pro rychlejší běh systému.

Nejdříve jsem popsal dvě exitující řešení, které využívají NN pro průmyslové úlohy. První z nich je Cognex ViDi, druhé Opto Engineering Albert. Zjistil jsem, že žádné z nalezených řešení, které by využívalo umělé neuronové sítě, nevyhovuje požadavkům kontroly kvality ve společnosti TE Connectivity s.r.o., pro kterou je projekt určen. Jak řešení od společnosti Cognex, tak od společnosti Opto Engineering nedosahují požadované přesnosti.

Následně jsem se zabýval výběrem vhodné třídy výrobků, pro které by byl systém založený na neuronových sítích vhodný. Z aktuálně fungujících i plánovaných typů kontrol jsem vybral následující. Jedná se o vizuální kontrolu přítomnosti a pozice pinů v konektoru, rozpoznávání typu ustřiženého kabelu a kontrolu přítomnosti plastové západky. Všechny zmíněné kontroly mají společné to, že se pro ně obtížně navrhuje systém založený na klasickém přístupu strojového vidění.

Dále jsem se zabýval výběrem hardwarové platformy pro učení a platformy určené pro detekci vad a klasifikaci.

V případě platformy pro učení jsem začal zhodnocením dostupných cloudových služeb a následně jsem se zabýval volbou komponentů pro vlastní pracovní stanici. Nakonec jsem zvolil osobní počítač, ve kterém primární výpočetní výkon pro učení neuronových sítí zajišťuje grafická karta NVIDIA GeForce RTX 2080 Ti.

V případě hardwarové platformy pro detekci vad a klasifikaci jsem zhodnotil dostupné řešení v podobě osobního počítače i specializovaného hardwaru, a to jak už v podobě USB akcelerátorů, tak jednodeskových počítačů. Z dostupných možností jsem zvolil tu nejvýkonnější ze specializovaného hardwaru, a to NVIDIA Jetson AGX Xavier.

Dalším krokem bylo zprovoznění této hardwarové platformy a následná instalace potřebných softwarových nástrojů, ovladačů, knihoven a také rozšíření tohoto zaří-

zení o větší úložiště v podobě 500GB SSD disku a o Wi-Fi modul.

Následovala tvorba samotného datasetu. Zaznamenal jsme přibližně 200 000 unikátních snímků. Část z těchto snímků byla následně anotována. Ve výsledku jsem tedy získal 1 000 snímků z každé kategorie konkrétní kontroly. Celkově tedy 10 000 anotovaných snímků. K záznamu jsem využíval softwarové řešení od společnosti Cognex, konkrétně In-Sight Spreadsheet. Snímky byly následně upraveny tak, aby vyhovovaly pro zpracování neuronovými sítěmi. Byly oříznuty, aby obsahovaly pouze danou zónu zájmu a měly rozlišení zpracovatelné na dostupném hardwaru.

Dále jsem hledal vhodnou architekturu neuronové sítě pro kontrolu kvality a klasifikaci. Zvolil jsem Inception-v3.

Vzhledem k tomu, že jsem chtěl pro učení využít grafické karty z pracovní stanice, bylo zapotřebí nainstalovat podpůrný software a knihovny. Kvůli nedostatečné dostupnosti validních zdrojů informací jsem musel experimentovat s různými verzemi knihoven a podpůrného softwaru abych určil stabilní kombinaci, na které je možné učení provádět. Učení jako takové, je založeno na knihovně *TensorFlow* a rozhraní *Keras*.

I přes využití výkonné grafické karty trvá učení jedné epochy Inception-v3 pro rozpoznávání typu kabelu přibližně 4,5 minuty. Modely byly učeny na 200 nebo 150 epoch. Na základě grafů, které znázorňují průběh učení jsem následně vybral optimální modely. Po dokončení učení bylo zapotřebí otestovat správnou funkčnost. Toto ověření probíhá tak, že se naučené neuronové síti postupně předkládají snímky, se kterými se tato neuronová síť nikdy nesetkala a kontroluje se správnost určení kvality a klasifikace.

V úloze pro rozpoznávání typu ustřiženého kabelu neuronová síť správně predikovala typ kabelu pro všech 600 snímků z testovacího datasetu. U úlohy kontroly přítomnosti plastové západky byla správně predikována kategorie OK nebo NOK pro všech 200 snímků z testovacího datasetu. Úloha kontroly pinů v konektoru byla nejproblematičtější. Neuronová síť správě predikovala kategorie OK nebo NOK pro 197 z 200 snímků z testovacího datasetu. Všechny tři špatně predikované snímky byly zařazeny do kategorie NOK přesto, že se jednalo o OK snímky.

Výsledkem tohoto testování jsou jednotlivé confusuin matrix a vypočtené hodnoty precision a accuracy.

V následující kapitole jsem se zabýval optimalizací již dříve natrénovaných modelů pomocí knihovny *NVIDIA TensorRT*. V první části popisuji princip fungování této optimalizační knihovny. Dále se zabývám optimalizací konkrétních modelů. Zjistil jsem, že pro to, aby mohl být model, který vznikl pomocí rozhraní *Keras*, optimalizován, je zapotřebí ho nejprve převést do formátu, který odpovídá modelům z knihovny *TensorFlow*. Model nelze optimalizovat přímo z důvodu nepodporovaných funkcí v modelu z rozhraní *Keras*.

Následně popisuji samotný skript pro optimalizaci a skript pro inferenci. Dále jsem pomocí skriptu pro optimalizaci optimalizoval již dříve natrénované modely pro zařízení NVIDIA Jetson AGX Xavier.

V další části hodnotím výsledky této optimalizace. Zjistil jsem, že díky ní probíhá inference o 65 - 76% rychleji. Nejsložitější model pro rozpoznávání typu ustřiženého kabelu trvá před optimalizací 114 ms a po optimalizaci 17 ms. Pro srovnání uvádím časy inference pro tentýž model z násobně dražší pracovní stanice. Dále jsem zhodnotil vliv nastavení velikosti proměnných pro optimalizovaný graf. Pro zařízení NVIDIA Jetson AGX Xavier je nejvýhodnější zvolit INT16.

V poslední kapitole jsem se zabýval srovnáním optické klasifikace klasickým způsobem a pomocí NN. Srovnával jsem tyto dva postupy pro úlohu kontroly pinů v konektoru. Nejprve jsem popsal jakým způsobem funguje klasické řešení a uvedl jsem historii vývoje tohoto systému ve společnosti TE Connectivity s.r.o. Následně jsem tento systém využil a otestoval testovací množinu snímků. Zjistil jsem, že klasický způsob predikuje obdobně jako systém založený na neuronových sítích. Uvedl jsem výhody a nevýhody jednotlivých systémů a srovnal jsem časovou náročnost na jejich vývoj. Dále uvádím rychlostní srovnání. Na stejné pracovní stanici trvá predikce standardním způsobem 13 ms, pomocí neoptimalizovaného modelu na téže pracovní stanici pak 9 ms.

V rámci dalšího rozšiřování této práce bych chtěl provést srovnání přesností predikce a časů inference i pro další architektury neuronových sítí. Zejména pak pro novější verzi architektury Inception, tedy verzi v4.

Hlavním a nejnáročnějším plánovaným rozšířením této práce je však nasazení na reálné úloze optické kontroly v průmyslu. Vzhledem k výsledkům predikce předpokládám, že by po dalším testování a doplnění predikčního skriptu o komunikaci bylo možné mnou natrénované neuronové sítě využít pro úlohu kontroly typu ustřiženého kabelu a pro kontrolu přítomnosti plastové západky. S tímto úzce souvisí, rozšíření

datasetu snímků, který je potřebný pro důkladnější a statisticky více významné testování.

Za svého působení ve firmě TE Connectivity s.r.o. jsem zjistil, že klasické způsoby optické kontroly není možno pro některé výrobní linky nasadit z důvodu přílišné časové náročnosti na vývoj. Domnívám se, že závěry z mé diplomové práce by mohly přispět ke tvorbě nových systémů optické kontroly výrobků založených na neuronových sítích pro některé výrobní linky, pro které to doposud nebylo ekonomicky výhodné, a že díky navrženému postupu optimalizace by bylo možné využít systémy založené na neuronových sítích i pro vysokorychlostní obrazové kontroly.

Literatura

[1] *WINSTON, Patrick Henry: Artificial Intelligence. 3rd ed. Addison-Wesley, 1992. ISBN: 9780201533774.*

[2] *ROSEBROCK, A.: Deep Learning For Computer Vision with Python, PyImageSearch, 2017. ISBN: 9781722487867.*

[3] ALBERT, 2020. In: Opto Engineering [online]. [cit. 2020-02-01]. Dostupné z: https://www.opto-e.com/albert

[4] *Intel® Neural Compute Stick 2(Intel® NCS2):*[online].[cit. 1. 12. 2019]. Dostupné z URL: <`https://software.intel.com/en-us/neural-compute-stick`>.

[5] *Google Coral - USB Accelerator:*[online].[cit. 1. 12. 2019]. Dostupné z URL: <`https://coral.ai/products/accelerator`>.

[6] *Google Coral - Dev Board:*[online].[cit. 1. 12. 2019]. Dostupné z URL: <`https://coral.ai/products/`>.

[7] *TensorRT 7.0.0 Developer Guide:*[online].[cit. 1. 12. 2019]. Dostupné z URL: <`https://docs.nvidia.com/deeplearning/sdk/tensorrt-developer-guide/index.html`>.

[8] *Keras Documentation:*[online].[cit. 1. 12. 2019]. Dostupné z URL: <`https://keras.io/documentation/`>.

[9] *TensorFlow API Documentation:*[online].[cit. 1. 12. 2019]. Dostupné z URL: <`https://www.tensorflow.org/api_docs`>.

[10] BIANCO, Simone, Remi CADENE, Luigi CELONA a Paolo NAPOLETANO, 2018. Benchmark Analysis of Representative Deep Neural Network Architectures. IEEE Access [online]. 6, 64270-64277 [cit. 2020-04-20]. DOI: 10.1109/ACCESS.2018.2877890. ISSN 2169-3536. Dostupné z: https://ieeexplore.ieee.org/document/8506339/

[11] LAKIS, Phillip. A Starter Hardware Guide to Deep Learning. In: Medium [online]. 7. 12. 2017 [cit. 2020-03-25]. Dostupné z: https://medium.com/@philliplakis/a-starter-hardware-guide-to-deep-learning-19e8ddd34d0e

[12] DETTMERS, Timm, 2018. A Full Hardware Guide to Deep Learning. In: Tim Dettmers [online]. [cit. 2020-03-25]. Dostupné z: https://timdettmers.com/2018/12/16/deep-learning-hardware-guide/

[13] Jetson Nano: Deep Learning Inference Benchmarks, 2020. In: NVIDIA DEVELOPER [online]. [cit. 2020-04-15]. Dostupné z: https://developer.nvidia.com/embedded/jetson-nano-dl-inference-benchmarks

[14] Guidance for Compiling TensorFlow Networks, 2019. In: Intel Movidius Neural Compute SDK [online]. [cit. 2020-02-18]. Dostupné z: https://movidius.github.io/ncsdk/tf_compile_guidance.html

[15] Compare NVIDIA Jetson Module Specifications, 2020. NVIDIA [online]. [cit. 2020-01-8]. Dostupné z: https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/

[16] Jetson AGX Xavier Developer Kit, 2019. NVIDIA [online]. [cit. 2019-10-26]. Dostupné z: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit

[17] NVIDIA Jetson AGX Xavier Series Supported Component List. In: NVIDIA AUTONOMOUS MACHINES [online]. 2020-01-15 [cit. 2020-02-2]. Dostupné z: https://developer.nvidia.com/embedded/downloads

[18] JetPack SDK [online]. In: . [cit. 2020-02-26]. Dostupné z: https://developer.nvidia.com/embedded/jetpack

[19] LECUN, Yann, Corinna CORTES a Christopher J.C. BURGES. THE MNIST DATABASE of handwritten digits [online]. [cit. 2020-01-23]. Dostupné z: http://yann.lecun.com/exdb/mnist/

[20] SZEGEDY, Christian, Vincent VANHOUCKE, Sergey IOFFE, Jon SHLENS a Zbigniew WOJNA, 2016. Rethinking the Inception Architecture for Computer Vision. In: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2016, s. 2818-2826 [cit. 2020-05-29]. DOI: 10.1109/CVPR.2016.308. ISBN 978-1-4673-8851-1. Dostupné z: http://ieeexplore.ieee.org/document/7780677/

[21] TSANG, Sik-Ho, 2018. Review: Inception-v3 — 1st Runner Up (Image Classification) in ILSVRC 2015 [online]. In: . [cit. 2020-02-29]. Dostupné z: https://medium.com/@sh.tsang/review-inception-v3-1st-runner-up-image-classification-in-ilsvrc-2015-17915421f77c

[22] GoogLeNet, 2018. AGGARWAL, Charu C. Neural Networks and Deep Learning A Textbook. Gewerbestrasse 11, 6330 Cham, Switzerland: Springer International Publishing, s. 345-347. ISBN 978-3-319-94463-0.

[23] KINSLEY, Harrison. Sentdex. In: YouTube [online]. [cit. 2020-01-05]. Dostupné z: https://www.youtube.com/channel/UCfzlCWGWYyIQ0aLC5w48gBQ

[24] ROSEBROCK, Adrian. PyImageSearch Gurus: A course and comunity designed to take you from computer vision beginer to expert. PyImageSearch [online]. [cit. 2020-02-20]. Dostupné z: https://www.pyimagesearch.com/pyimagesearch-gurus/

[25] SZEGEDY, Christian, WEI LIU, YANGQING JIA, et al., 2015. Going deeper with convolutions. 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) [online]. IEEE, 2015, 1-9 [cit. 2020-02-29]. DOI: 10.1109/CVPR.2015.7298594. ISBN 978-1-4673-6964-0. Dostupné z: http://ieeexplore.ieee.org/document/7298594/

[26] KARIM, Raimi, 2019. Illustrated: 10 CNN Architectures. Towards Data Science [online]. [cit. 2020-01-23]. Dostupné z: https://towardsdatascience.com/illustrated-10-cnn-architectures-95d78ace614d

[27] IOFFE, Sergey a Christian SZEGEDY. Batch Normalization: Accelerating Deep Network Training byReducing Internal Covariate Shift [online]. 2015, 11 [cit. 2020-02-24]. arXiv:1502.03167. Dostupné z: https://arxiv.org/abs/1502.03167

[28] ALLAN, Alasdair. Benchmarking Edge Computing. In: Medium [online]. 2019 [cit. 2020-01-29]. Dostupné z: https://medium.com/@aallan/benchmarking-edge-computing-ce3f13942245

[29] EBERSOLE, Mark. What Is CUDA? In: NVIDIA [online]. 2012 [cit. 2020-04-29]. Dostupné z: https://blogs.nvidia.com/blog/2012/09/10/what-is-cuda-2/

[30] NVIDIA cuDNN, 2020. In: NVIDIA DEVELOPER [online]. [cit. 2020-04-29]. Dostupné z: https://developer.nvidia.com/cudnn

[31] OpenCV [online], 2020. [cit. 2020-04-29]. Dostupné z: https://opencv.org/about/

[32] TensorFlow [online], 2020. [cit. 2020-04-30]. Dostupné z: https://www.tensorflow.org/overview/

[33] Keras [online], 2020. [cit. 2020-04-30]. Dostupné z: https://keras.io/Keras [online], 2020. [cit. 2020-04-30]. Dostupné z: https://keras.io/

[34] NumPy [online], 2020. [cit. 2020-01-12]. Dostupné z: https://numpy.org/

[35] EasyGui [online], 2014. [cit. 2020-01-12]. Dostupné z: https://pythonhosted.org/easygui/index.html

[36] Keras Applications, 2019. In: Keras [online]. [cit. 2019-10-10]. Dostupné z: https://keras.io/api/applications/

[37] Deep Learning with TensorFlow 2 and Keras: Regression, ConvNets, GANs, RNNs, NLP, and more with TensorFlow 2 and the Keras API, 2nd Edition, 2019. Birmingham, UK: Packt Publishing. ISBN 978-1838823412.

[38] TensorRT Support Matrix. NVIDIA [online]. 2019 [cit. 2020-03-25]. Dostupné z: https://docs.nvidia.com/deeplearning/tensorrt/archives/tensorrt-506/tensorrt-support-matrix/index.html

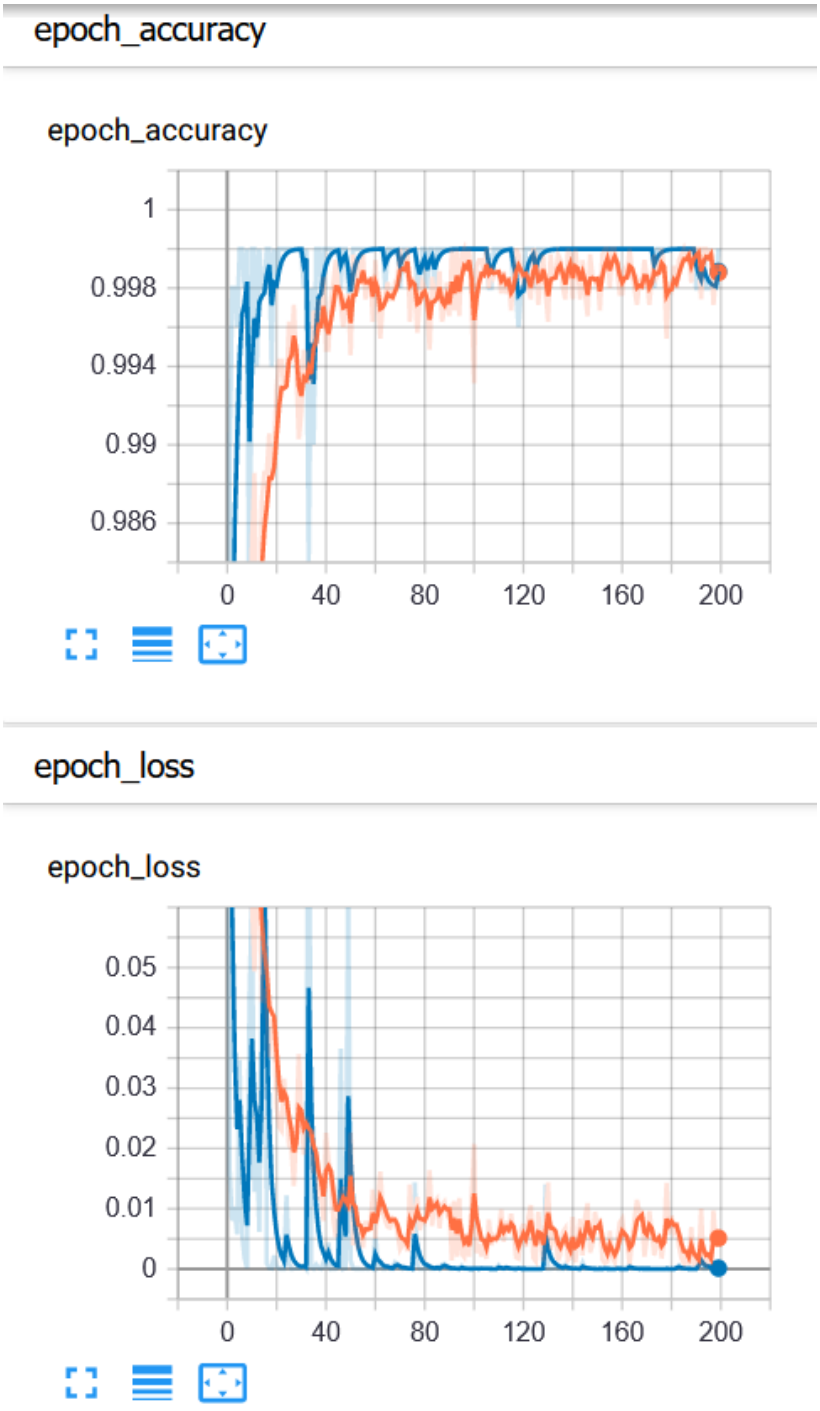
[39] KOROBCHENKO, Dmitry. How to accelerate your neural net inference with TensorRT. In: YouTube [online]. 2018 [cit. 2020-02-16]. Dostupné z: https://www.youtube.com/watch?v=6My-daDk4zE

[40] Low angle ring light. In: Dat Vision [online]. [cit. 2020-03-19]. Dostupné z: http://www.datvision.co.kr/product/product_010100.html?lcode=04

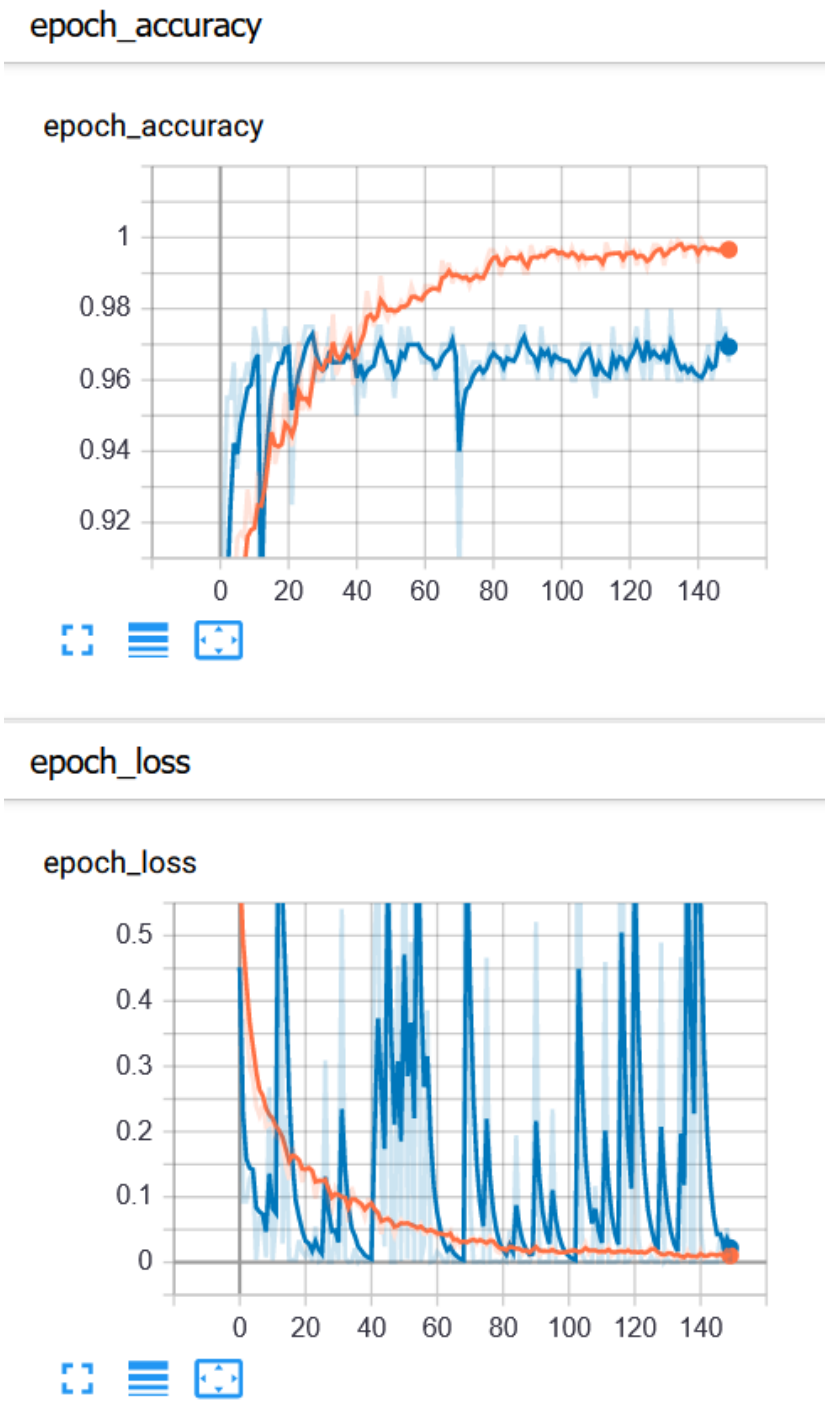
[41] Coaxial light. In: Keyence [online]. [cit. 2020-03-20]. Dostupné z: https://www.keyence.com/products/vision/vision-sys/ca-d/lineups/lineup-08.jsp

[42] MONTAVON, Grégoire, Wojciech SAMEK a Klaus-Robert MÜLLER, 2018. Methods for interpreting and understanding deep neural networks. Digital Signal Processing [online]. 73, 1-15 [cit. 2020-05-31]. DOI: 10.1016/j.dsp.2017.10.011. ISSN 10512004. Dostupné z: https://linkinghub.elsevier.com/retrieve/pii/S1051200417302385

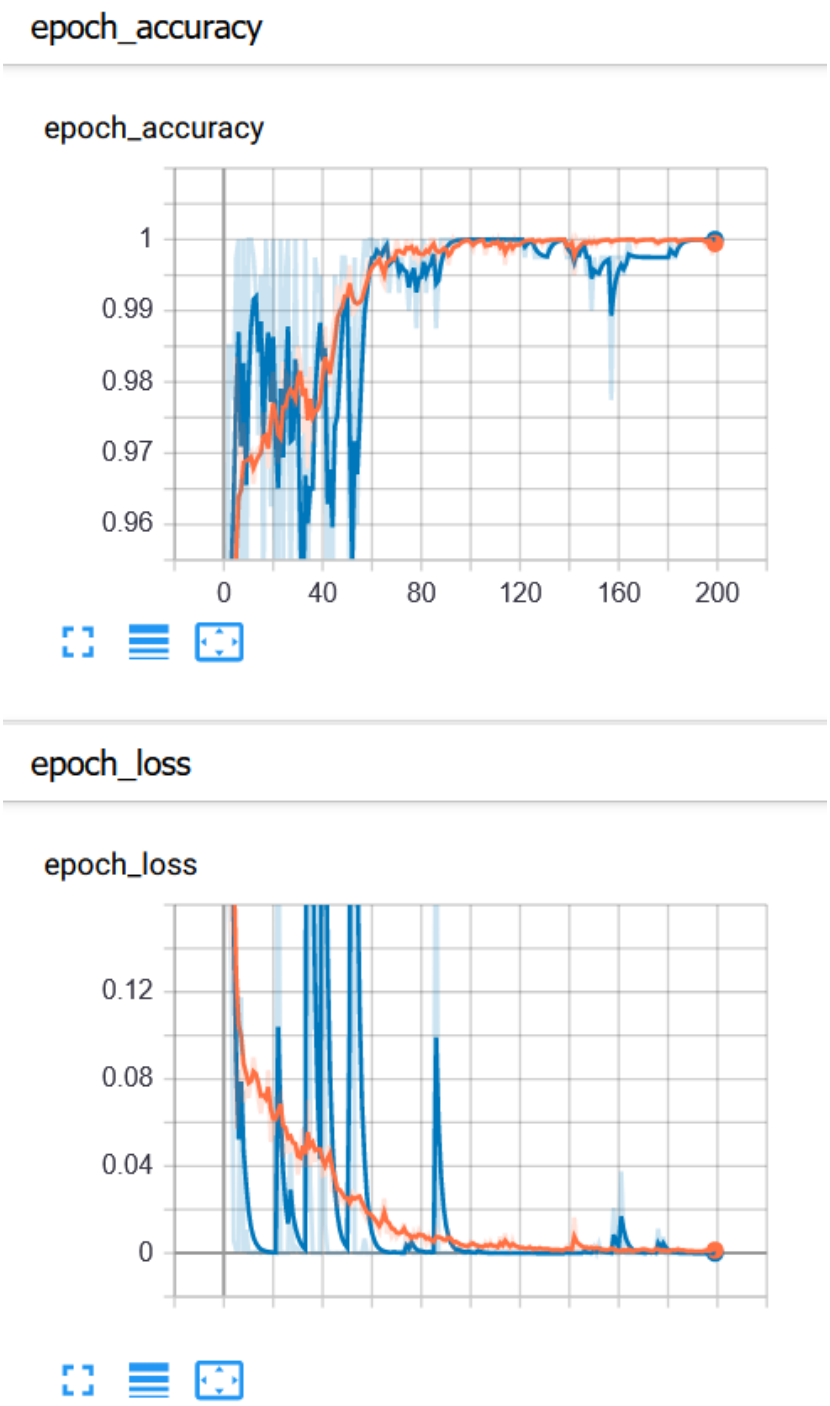
A Grafy accuracy a loss funkce pro jednotlivé úlohy



Obr. A.1: Grafy accuracy a loss funkce z učení NN pro rozpoznání typu ustřiženého kabelu



Obr. A.2: Grafy accuracy a loss funkce z učení NN pro kontrolu pinů v konektoru



Obr. A.3: Grafy accuracy a loss funkce z učení NN pro kontrolu přítomnosti plastové západky

B Obsah přiloženého datového nosiče

Vzhledem k velkému množství obsažených souborů jsou v obsahu uvedeny pouze složky v kořenovém adresáři

- `/` ..... kořenový adresář přiloženého datového nosiče
  - `Grafy` .....
  - `Kód` .....
  - `Dataset snímků` ..... pouze na SD kartě
  - `Logy` ..... pouze na SD kartě
  - `Modely` ..... pouze na SD kartě