VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

POUŽITÍ SELF-SUPERVISED LEARNING PRO ROZPOZNÁNÍ POZIC RUKOU V OBRAZE

SELF-SUPERVISED LEARNING FOR RECOGNITION OF HAND POSES IN IMAGE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **LUCIA MAKAIOVÁ**
AUTHOR

VEDOUCÍ PRÁCE **prof. Ing. ADAM HEROUT, Ph.D.**
SUPERVISOR

BRNO 2023

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

139616

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Studentka: **Makaiová Lucia**
Program: Informační technologie
Specializace: Informační technologie

Název: **Použití self-supervised learning pro rozpoznání pozic rukou v obraze**
Kategorie: Zpracování obrazu
Akademický rok: 2022/23

Zadání:

1. Seznamte se s problematikou strojového učení pro počítačové vidění a s problematikou rozpoznání sportovních pozic v obraze a videu.
2. Získejte a/nebo sestavte datovou sadu (sady) obrázků pozic rukou, vhodně imitujících sportovní pozice.
3. Experimentujte s metodami self-supervised learning nad sestavenou datovou sadou (sadami).
4. Demonstrujte použitelnost vyvinutých technik pro rozpoznání pozic.
5. Iterativně vylepšujte vyvinuté techniky i datovou sadu směrem k maximální použitelnosti.
6. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování výsledků projektu.

Literatura:

- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2016
- Bharath Ramsundar, Reza Bosagh Zadeh: TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning, O'Reily Media, 2018
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008
- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2011
- Grill J-B et al.: Bootstrap your own latent: A new approach to self-supervised Learning, NeurIPS 2020, https://arxiv.org/abs/2006.07733
- Caron M et al.: Emerging Properties in Self-Supervised Vision Transformers, https://arxiv.org/abs/2104.14294
- Sermanet et al.: Time-Contrastive Networks: Self-Supervised Learning from Video, ICRA 2018, https://arxiv.org/abs/1704.06888
- Asano et al.: Self-labelling via simultaneous clustering and representation learning, ICLR 2020, https://arxiv.org/abs/1911.05371
- L. Jing, Y. Tian, Self-supervised visual feature learning with deep neural networks: A survey, IEEE PAMI, 2020

Při obhajobě semestrální části projektu je požadováno:
Body 1 a 2, značné rozpracování bodů 3 až 5.

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

Vedoucí práce: **Herout Adam, prof. Ing., Ph.D.**
Vedoucí ústavu: Černocký Jan, prof. Dr. Ing.
Datum zadání: 1.11.2022
Termín pro odevzdání: 10.5.2023
Datum schválení: 5.4.2023

Abstrakt

Práca sa zaoberá self-supervised prístupom k strojovému učeniu pre natrénovanie rozpoznávania pozícií rúk v obraze. Použila som kontrastívnu metódu self-supervised učenia a iteratívne optimalizovala riešenie použitím techník early stopping, triplet mining, optimalizovaním hyperparametrov či experimentovaním s rôznymi modelmi. Metódu som implementovala pomocou frameworku Pytorch a pre spracovanie a vizualizáciu dát som využívala nástroj TensorBoard. Pre referenčné výsledky som natrénovala model pomocou učenia s učiteľom. Podarilo sa mi dosiahnuť úspešnosť 83 % na datasete Handz, čím sa metóda vyrovnala referenčným výsledkom. Vytvorené riešenie poskytuje informácie, ktoré je možné aplikovať pri využití self-supervised metódy pre podobné problémy, akými je napríklad rozpoznávanie športových pozícií. Hlavným výsledkom je zistenie, že self-supervised metódy sú obzvlášť vhodné pokiaľ je k dispozícii pre trénovanie len obmedzený počet označených dát, ktoré sú navyše reprezentované veľmi nerovnomerným počtom vzoriek. Na základe zistených údajov je možné vytvoriť metódu pre rozpoznávanie športových pozícii alebo ďalej optimalizovať existujúce riešenie.

Abstract

This work focuses on using self-supervised learning for the task of hand poses recognition in image. I have used contrastive method of self-supervised learning and optimized the solution iteratively, using techniques such as early stopping, triplet mining, optimization of hyperparameters or experimenting with various model architectures. The method was implemented with Pytorch framework and Tensorboard was used for data processing and visualization. I have trained the first model using a supervised method, to obtain reference values. I have successfully matched this reference result by training a self-supervised model on Handz dataset and achieving 83 % accuracy. The created solution provides findings, which can be applied to similar problems, such as recognition of sport poses. The main contribution of this work is the discovery, that self-supervised methods are particularly effective when using a labeled dataset for downstream task with just a small amount of samples, which in addition have uneven distribution of samples for individual classes. Based on these findings, it is possible to create a method for self-supervised learning for recognition of sport poses or further optimize existing solution for hand poses.

Kľúčové slová

spracovanie obrazu, strojové učenie, neurónové siete, self-supervised učenie, triplet loss, rozpoznávanie pozícií, počítačové videnie

Keywords

image recognition, machine learning, neural networks, self-supervised learning, triplet loss, recognition of poses, computer vision

Citácia

MAKAIOVÁ, Lucia. *Použití self-supervised learning pro rozpoznání pozic rukou v obraze*. Brno, 2023. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Použití self-supervised learning pro rozpoznání pozic rukou v obraze

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracovala samostatne pod vedením pána prof. Ing. Adam Herout, Ph.D. Uviedla som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpala.

.....
Lucia Makaiová
9. mája 2023

Poďakovanie

Chcela by som sa poďakovať vedúcemu práce Ing. Adam Herout, Ph.D. za podporu, nasmerovanie a ústretovosť pri vypracovaní práce.

Obsah

Kapitola 1

Úvod

Strojové učenie je súčasťou našich životov už dlhú dobu a inak to nie je ani pri rozpoznávaní pozícií v obraze. Už dlho existuje množstvo technológií a metód pre rozpoznávanie pozícií, avšak od roku 2016 sa stalo dominantným v tejto oblasti hlboké strojové učenie. To sa v podobe supervised učenia spolieha na veľké množstvo označených dát, ktoré sa získavajú veľmi pracne.

Táto práca sa zameriava na použitie self-supervised metódy, ktorej hlavnou výhodou je, že nie je závislá na veľkom množstve označených dát. Hlavná časť trénovania prebieha na neoznačených dátach, ktoré nie je ťažké získať vo veľkom množstve, či už z internetu alebo extrahovaním veľkého množstva obrázkov z videa. Dotrénovanie následne prebieha na veľmi malom počte označených dát a najčastejšie len na malej časti pôvodného modelu. Cieľom práce bolo experimentovať s touto metódou v oblasti rozpoznávania pozícií, porovnať ju s referenčným supervised prístupom a postupne optimalizovať vytvorené riešenie smerom k maximálnej použiteľnosti. Práca sa z dôvodu menšej výpočtovej náročnosti zameriava primárne na rozpoznávanie pozícií rúk, keďže pre tento problém je potrebné menšie rozlíšenie obrázku aj veľkosť modelu. Pri práci bolo však prihliadané na možnosť aplikácie metódy aj pre rozpoznávanie športových pozícií, a to hlavne pri tvorbe datasetu, ktorý obsahuje nerovnomerný počet reprezentácií rôznych pozícii. Ide teda o veľmi podobnú úlohu a princíp učenia je rovnaký, preto sú zistené informácie a vytvorená metóda ďalej aplikovateľné aj pre rozpoznávanie športových pozícií.

Konečnou motiváciou pre prácu je možnosť vytvorenia jednoduchej technológie pre rozpoznávanie športových pozícií, s ktorou by bolo získavanie spätnej väzby, napríklad pri cvičení jógy, dostupné pre každého s použitím aplikácie v mobile, bez nutnosti kupovať špeciálne zariadenia či softvér, ako to je dnes, napríklad pri herných konzolách s pohybovými senzormi. Toto riešenie by preto malo byť jednoduché a kompaktné.

Na začiatku práce je prehľad podstatných informácií z oblasti, ktorou sa práca zaoberá. Text ďalej popisuje existujúce riešenia v problematike rozpoznávania pozícií ako aj v oblasti stále populárnejšieho self-supervised prístupu k strojovému učeniu. Následne popisujem svoj postup práce, použité datasety a konkrétny spôsob, akým som implementovala self-supervised metódu strojového učenia a vyhodnocovala výsledky. Práca ďalej obsahuje vyhodnotenie výsledkov trénovania a získaných poznatkov, ako aj možnosti ďalšieho rozšírenia práce.

Kapitola 2

Strojové učenie v kontexte self-supervised prístupu k rozpoznávaniu pozícií

Pre vypracovanie práce bolo dôležité pochopiť základné princípy strojového učenia. Keďže sa táto problematika na bakalárskom stupni detailne nepreberá, prvým bodom mojej práce bolo oboznámiť sa s problematikou strojového učenia pre počítačové videnie, neurónových sietí a rozpoznávania pozícií v obraze.

Obr. 2.1: **Znázornenie kontextu práce**: Práca sa zaoberá hlbokým učením. Ide o podmnožinu strojového učenia, ktoré je jednou z oblastí umelej inteligencie. (inšpirované obrázkom z [7])

2.1 Strojové učenie

Strojové učenie je schopnosť systému umelej inteligencie získavať vlastné vedomosti na základe prvotných dát. Algoritmus strojového učenia [7] je algoritmus, ktorý je schopný učiť sa z dát. Dôležitými súčasťami algoritmu strojového učenia sú dataset, model, stratová funkcia a optimalizačný algoritmus.

2.1.1 Dataset

Dataset je súbor dát použitých pre trénovanie modelu. Delí sa na trénovací, validačný a testovací. V oblasti počítačového videnia je najčastejšie dataset zložený z obrázkov a príslušných značiek.

- **Trénovací** dataset obsahuje dáta na ktorých sa model učí.
- **Validačný** dataset môže byť použitý pre vyhodnotenie metrík a tiež vhodný výber hyperparametrov počas trénovania.
- **Testovací** dataset slúži na konečné vyhodnotenie modelu po jeho natrénovaní a mal by obsahovať len dáta ktoré model predtým ešte „nevidel“.

Dáta v datasetoch by sa nemali prelínať.

2.1.2 Model

Model strojového učenia je výsledkom vykonania algoritmu strojového učenia. Môže mať rôznu architektúru. Pre oblasť počítačového videnia sú dôležité **hlboké modely** [7]. Hlboké modely sú vhodné pre učenie na nespracovaných senzorických dátach, akými sú napríklad obrázky reprezentované ako súbor pixelov. Keďže priame spracovanie všetkých pixelov by bolo veľmi komplikované, hlboké modely tento problém dekomponujú do viacerých vrstiev, z ktorých každá vykonáva mapovanie na rôznej úrovni abstrakcie.

2.1.3 Príznak

Príznak [13] (feature) označuje špecifickú charakteristiku alebo atribút dát, ktorú model extrahuje počas učenia. Extrakcia príznakov prebieha v skrytých vrstvách modelu. Príznaky nemusia mať žiaden fyzikálny význam. Model si ich „zvolí“ výlučne v závislosti od geometrie dát. Je však možné, že sa model podarí natrénovať tak, že niektoré jeho príznaky explicitne reprezentujú napríklad hrany objektov, rohy a podobne.

2.1.4 Stratová funkcia

Funkcia ktorá vyhodnocuje presnosť predikcie modelu sa nazýva stratová funkcia [17]. Bežne rozlišujeme v strojovom učení tri typy stratových funkcií.

- **Regresná** stratová funkcia sa používa pri predikovaní spojitých hodnôt v určitom rozmedzí. Príkladom je mean squared error.
- **Klasifikačná** stratová funkcia meria rozdiel medzi pravdepodobnostným rozdelením, ktoré predikoval model a tým skutočným. V tejto práci je pre klasifikáciu použitá **krížová entropia** (cross-entrophy loss).
- **Hodnotiaca** stratová funkcia predikuje relatívnu vzdialenosť medzi hodnotami. Príkladom je **trojicová stratová funkcia** (triplet loss), ktorá sa používa hlavne pre kontrastívne učenie, a je bližšie popísaná v nasledujúcej kapitole.

2.1.5 Optimalizačný algoritmus

Optimalizácia je hľadanie hodnôt argumentov ktoré maximalizujú alebo minimalizujú funkciu. Optimalizačný algoritmus [7] teda definuje spôsob akým sa upravujú hodnoty modelu pre minimalizovanie stratovej funkcie.

Najpoužívanejším optimalizačným algoritmom pre hlboké učenie je **stochastic gradient descent** [7] (ďalej SGD). Algoritmus spočíta pre malú skupinu vzoriek (batch) výstup a stratu, následne pre ne vypočíta priemerný gradient a náležite upraví váhy modelu. Stochastický sa nazýva z dôvodu, že každý batch poskytuje iba nepresný odhad priemerného gradientu pre všetky vzorky v datasete.

2.1.6 Hyperparametre učenia

Pre úspešné natrénovanie modelu je dôležité vhodne zvoliť hyperparametre učenia. Tie vyjadrujú rýchlosť, dĺžku učenia a ďalšie dôležité vlastnosti trénovania.

- **Epocha** reprezentuje jeden prechod algoritmu cez trénovaciu sadu.
- **Batch size** vyjadruje koľko dát sa spracuje naraz predtým ako sa vykoná ďalšia optimalizácia váh modelu.
- **Rýchlosť učenia** (learning rate) definuje veľkosť jedného kroku optimalizačného algoritmu.

2.2 Self-supervised učenie

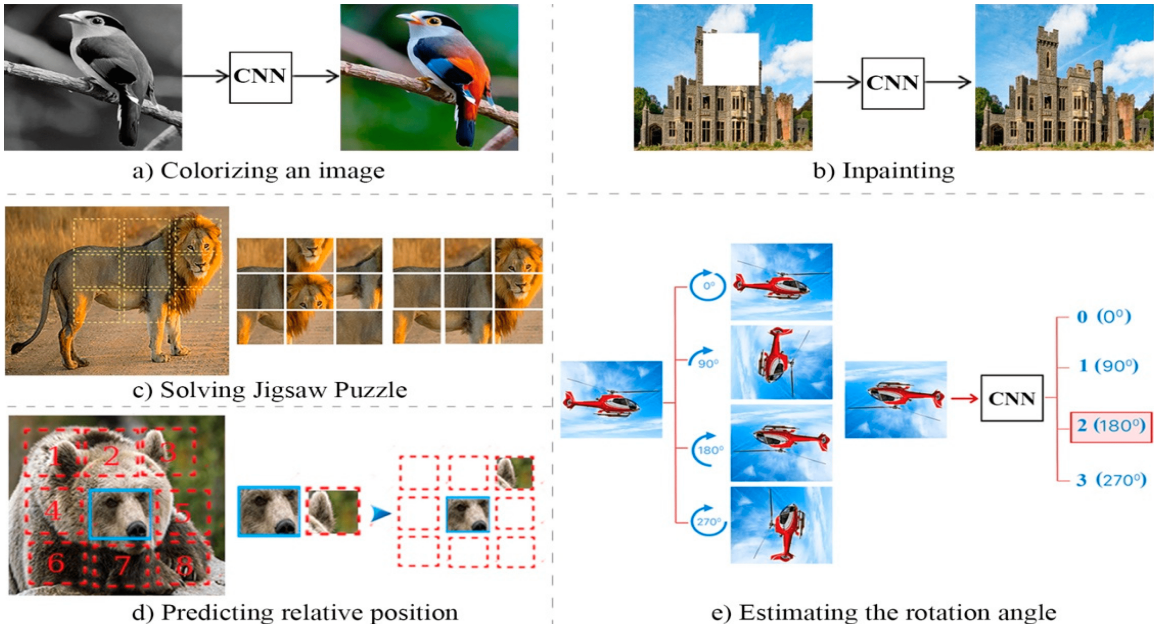
Self-supervised učenie je typ semi-supervised učenia. Má dve hlavné etapy, ktorými sú **náhradná úloha** (pretext task) a **cieľová úloha** (downstream task). Výhodou tohoto typu učenia je fakt, že zatiaľ čo získať veľké množstvo označených dát je veľmi pracné a v niektorých oblastiach nemožné, neoznačené dáta sa získavajú veľmi jednoducho. Veľké množstvo dát vieme získať z internetu, extrakciou z videí a podobne.

V tejto podkapitole sú priblížené jednotlivé časti self-supervised učenia [3], stratégia triplet mining pre výber vhodných trojíc ako aj konkrétne optimizéry a stratové funkcie, ktoré boli pre self-supervised metódu v práci využité, napriek tomu že sa používajú aj iných oblastiach strojového učenia.

2.2.1 Náhradná úloha

Význam náhradnej úlohy je na veľkom množstve dát naučiť model dobrú reprezentáciu obrazu. Model dostane za úlohu predikovať podmnožinu informácií na základe ich zvyšku. Týmto spôsobom sa z podmnožiny informácií stane označenie, ktoré sa model učí predikovať, a je možné ho jednoducho zistiť z celého datasetu. Môže sa jednať napríklad o úlohu predikovať skryté slovo vo vete, v závislosti od zvyšných slov.

Pri kontrastívnom type self-supervised učenia sa najčastejšie používajú rôzne techniky rozšírenia dát (data augmentation), od ktorých sa potom odvíja typ náhradnej úlohy, ktorej príklady sú znázornené na obrázku 2.2. Ďalšou možnosťou je využiť prirodzenú informáciu obsiahnutú v datasete, akou môže byť napríklad časová následnosť snímok vo videu z ktorého boli extrahované. Taktiež môže informáciu pre náhradnú úlohu tvoriť príslušnosť do



Obr. 2.2: Príklady „umelo“ vytvorených náhradných úloh, použitím rozličných metód pre rozšírenie dát akými sú napríklad (a) odfarbenie alebo (e) rotácia. Model sa vďaka riešeniu náhradnej úlohy naučí extrakciu vhodných príznakov pre dobrú reprezentáciu obrazu. (prevzaté z [3])

skupiny snímok, zachytených paralelne v rovnakom čase, pričom tiež môže ísť o kontrastívny typ učenia [16]. Rovnaký typ prístupu k náhradnej úlohe využíva aj moja práca a je znázornený na obrázku 2.3.

2.2.2 Cieľová úloha

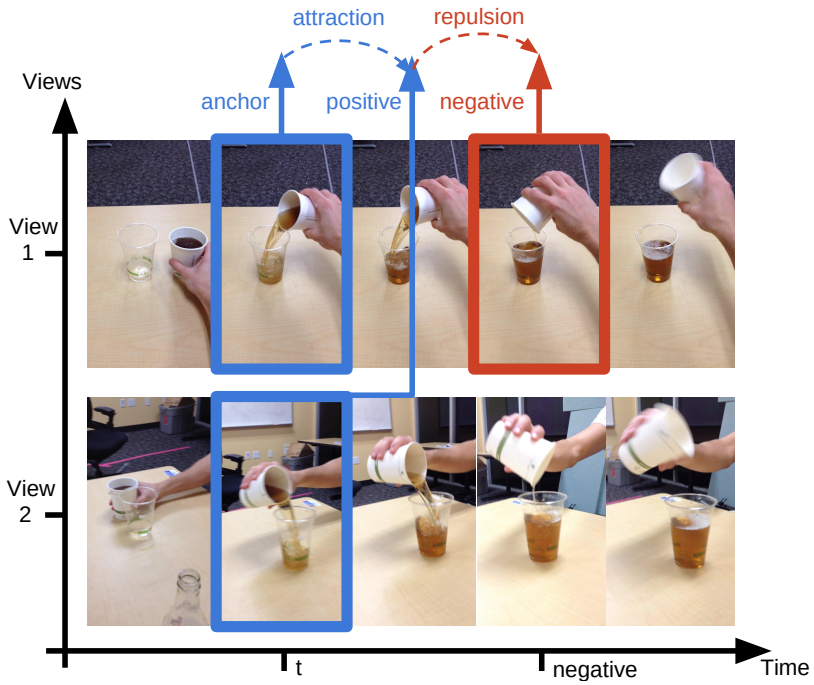
Príznaky naučené počas trénovania pre náhradnú úlohu sú v druhej časti self-supervised trénovania prenesené na model pre cieľovú úlohu. V kontexte počítačového videnia môže byť cieľovou úlohou klasifikácia, detekcia objektov, rozpoznanie pozície (pose estimation) či segmentácia obrazu.

Trénovanie môže prebiehať dvoma spôsobmi. Jednou z možností je všetky váhy pôvodnej siete dotrénovať (fine-tuning) na cieľovú úlohu. Alternatívne môžu byť ponechané pôvodné hodnoty váh siete, pričom sa namiesto poslednej, plne prepojenej vrstvy model rozšíri o lineárne vrstvy pre cieľovú úlohu. V druhom prípade sa pri tréningu optimalizujú iba váhy lineárnej vrstvy. Táto časť trénovania prebieha na malom množstve označených dát.

2.2.3 Trojicová stratová funkcia

Trojicová stratová funkcia [20] (triplet loss), sa používa pri kontrastívnom učení na trojiciach zložených z bázovej, pozitívnej a negatívnej vzorky. Trojicová stratová funkcia s ohraničením (μ), vypočítava euklidovskú vzdialenosť medzi reprezentáciami jednotlivých vzoriek. Cieľom je, aby bola vzdialenosť medzi bázovou a pozitívnou vzorkou (δ_+) čo najmenšia a naopak, vzdialenosť medzi bázovou a negatívnou vzorkou (δ_-) čo najväčšia, ako je to znázornené na obrázku 2.4. Vzorec pre trojicovú stratovú funkciu:

$$\lambda\left(\delta_+, \delta_-\right) = \max\left(0, \mu + \delta_+ - \delta_-\right) \tag{2.1}$$



Obr. 2.3: Technika kontrastívneho učenia pomocou simultánne nahratých videí. Náhradnou úlohou pre model je naučiť sa rozpoznať, či snímky reprezentujú rovnaký moment vo videu. Snímky zachytené v rovnakom čase teda tvoria pozitívny pár, pričom negatívna vzorka môže byť ľubovolná vzorka zachytená v inom čase. Model sa týmto spôsobom naučí, aké sú podobnosti medzi rôzne vyzerajúcimi snímkami v pozitívnom páre a aké sú rozdiely medzi podobne vyzerajúcou bázovou a negatívnou vzorkou. (prevzaté z [16])

2.2.4 Optimizér Adam

Optimizér Adam [12, 7] je efektívny stochastický optimizér. Je založený na algoritme SGD. Jeho výhodou je malá náročnosťou na pamäť, keďže využíva iba gradienty prvého stupňa. Optimizér Adam vykonáva adaptívnu estimáciu momentov, čo je kombinácia dvoch techník, ktorými sú adaptívna rýchlosť učenia a momentum. Tieto techniky sú hlavným rozdielom a výhodou oproti SGD.

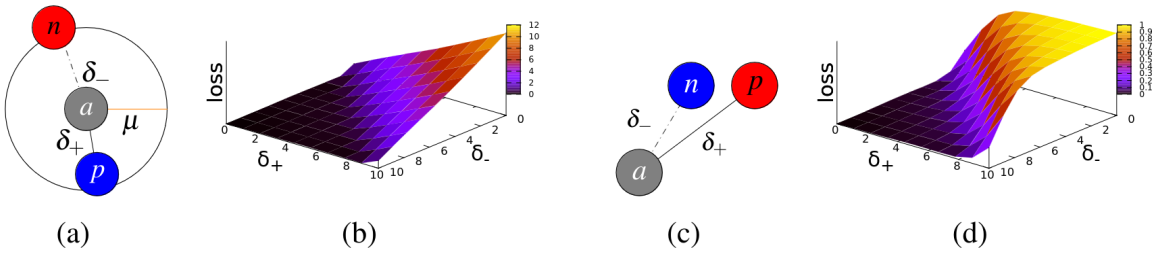
Rýchlosť učenia má veľký vplyv na úspešnosť modelu a tento hyperparameter je zložité nastaviť správne. Metóda adaptívnej rýchlosti učenia uľahčuje tento problém tým, že rýchlosť učenia pre jednotlivé parametre modelu prispôsobuje počas trénovania, najčastejšie podľa parciálnej derivácie stratovej funkcie, vzhľadom na daný parameter.

Algoritmus hybnosti (momentum) zlepšuje optimalizáciu tým, že okrem aktuálne vypočítaného gradientu berie určitý ohľad aj na tie predošlé. Vďaka tomu môže dôjsť k rýchlejšej a presnejšej optimalizácii.

2.2.5 Triplet mining

Triplet mining [10, 14] je prístup k výberu trojíc pre kontrastívne učenie, ktorého cieľom je zlepšiť efektivitu učenia.

Trojice sa delia na jednoduché, stredne ťažké a ťažké, podľa hodnoty trojicovej stratovej funkcie. Pre trénovanie sa potom vyberie len určitý typ trojíc.



Obr. 2.4: Znázornenie 2 typov trojicovej stratovej funkcie: (a) Trojicová stratová funkcia s ohraničením (μ) sa snaží o to, aby bola vzdialenosť pozitívneho páru kratšia než je dané ohraničenie a vzdialenosť medzi bázovým a negatívnym väčšia. (c) Pomerová stratová funkcia (ratio loss) funguje podobne. Avšak miesto použitia ohraničenia sa jednoduch snaží minimalizovať vzdialenosť pozitívneho a maximalizovať vzdialenosť negatívneho prvku od bázy. Obrázky (b) a (d) znázorňujú hodnoty trojicovej straty vo funkcii (δ_+), (δ_-). (prevzaté z [20])

Technika triplet mining je pri trénovaní s použitím trojicovej stratovej funkcie dôležitá, keďže pri zväčšovaní datasetu rastie počet možných trojíc kubicky. Väčšina z týchto trojíc však nie je vhodná, pretože sa jedná o ľahké trojice ktoré sú v datasete obsiahnuté v priveľkom množstve, ale ich vplyv na správne natrénovanie modelu je veľmi malý.

Mining ťažkých trojíc preto kladie dôraz na to, aby sa v datasete nachádzali iba trojice, pri ktorých je ťažké odlíšiť negatívnu vzorku od bázovej a pozitívnej. To pomôže modelu lepšie rozumieť zložitým konceptom, ktoré sa má naučiť. Na druhej strane, ak model vidí iba ťažké trojice, môže to znemožniť naučenie sa jednoduchých asociácií. Alternatívne sa preto využívajú stredne ťažkých trojíc, alebo kombinácia oboch. Toto zaradenie trojíc je znázornené na obrázku 2.5.

Triplet mining sa taktiež rozdeľuje na online a offline prístup, podľa spôsobu tvorby trojíc.

Pri **offline** prístupe sa na začiatku každej epochy vyberú konkrétne trojice z existujúceho datasetu a až potom sa vykoná jedna epocha trénovania.

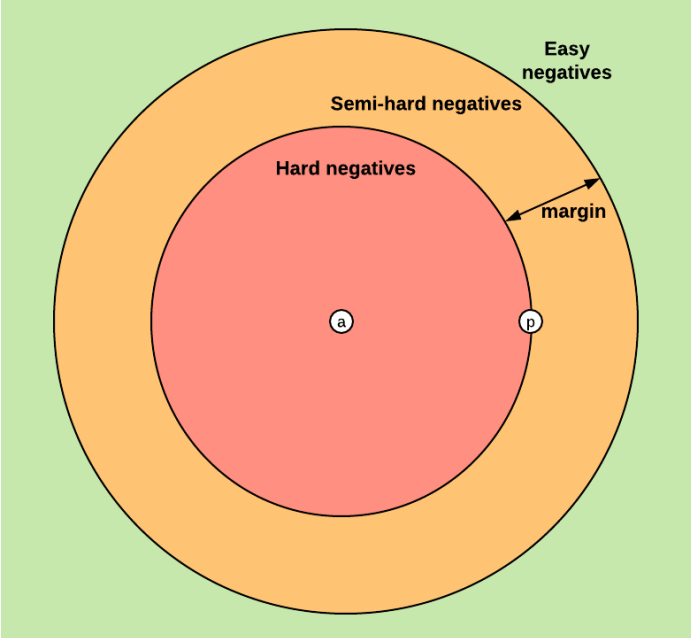
Online prístup naopak aplikuje triplet mining počas trénovania a to separátne pre každý batch. Najčastejšie sa pritom použijú stratégie batch all alebo batch hard. Stratégia batch all použije všetky trojice, ktoré možno vytvoriť, okrem tých jednoduchých. Batch hard použije výlučne ťažké trojice.

2.3 Hlboká neurónová sieť

Hlboké neurónové siete sú v posledných rokoch najpoužívanejším modelom pre strojové učenie v oblasti počítačového videnia [18]. Skladajú sa zo vstupnej vrstvy, viacero skrytých vrstiev a z výstupnej vrstvy.

2.3.1 Konvolučná neurónová sieť

Konvolučná neurónová sieť (CNN) [15] je model hlbokého učenia určený primárne pre oblasť počítačového videnia. Jeho architektúra, znázornená na obrázku 2.6, je preto navrhnutá tak, aby najviac vyhovovala tomuto použitiu a faktu, že jej vstupom sú obrázky. Jej základom sú **konvolučné vrstvy**, ktoré, ako názov napovedá, používajú konvolúciu s



Obr. 2.5: Znázornenie zadelenia trojíc podľa vzdialenosti negatívnej vzorky od bázovej (a). Ťažké negatívne vzorky sú také, ktorých vzdialenosť od bázy je menšia ako vzdialenosť pozitívnej vzorky (p). Stredne ťažké vzorky sú od pozitívnej vzorky vzdialené v rozsahu daného ohraničením (margin). Ľahké vzorky sú od bázovej aj pozitívnej vzorky tak ďaleko, že trojicová stratová funkcia s ohraničením má hodnotu 0. Tieto vzorky nie sú užitočné pre trénovanie. (prevzaté z [14])

trénovateľnými jadrami (kernels). Cieľom je redukovanie výpočtovej komplexnosti modelu pri ponechaní schopnosti naučiť sa správne príznaky pre úlohu rozpoznávania obrazu.

2.3.2 Reziduálna neurónová sieť

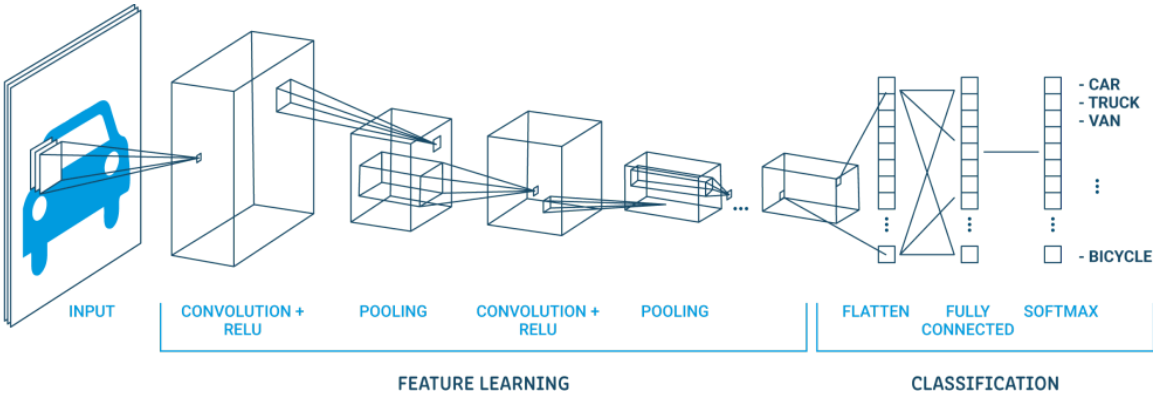
V tejto práci je použitá konvolučná reziduálna neurónová sieť ResNet [9]. Použité sú modely ResNet-18, ResNet-34 a ResNet-50.

Reziduálne siete sú riešením pre problém miznúceho gradientu (vanishing gradient), ktorý často nastáva pri klasických hlbokých modeloch. Tieto siete je preto možné optimalizovať oveľa jednoduchšie než klasické siete rovnakej hĺbky a taktiež omnoho viac profitujú z navyšovania hĺbky siete.

Základom reziduálnych sietí sú reziduálne bloky, znázornené na obrázku 2.7. Reziduálne bloky využívajú "skip connections". Sú to spojenia ktoré k výstupu daného bloku pripočítajú jeho vstup, pritom však nezvyšujú počet parametrov siete ani jej komplexnosť. Cieľom potom je, aby sa vstup bloku rovnal jeho výstupu.

2.3.3 Konvolučná vrstva

Konvolučná vrstva [18] je základom konvolučných neurónových sietí, medzi ktoré patrí aj ResNet. Táto vrstva využíva operáciu konvolúcie na vstupnej mape príznakov (feature map) s použitím jadra (kernel) určitej veľkosti. Jadro tvorí filter určitej výšky a šírky (pixel), zložený z váh ktoré sa nastavujú počas trénovania. Základný princíp konvolúcie je na



Obr. 2.6: Znázornenie architektúry konvolučnej neurónovej siete: Konvolučná neurónová sieť sa skladá z časti učenia príznakov a klasifikačnej časti. Časť učenia príznakov pozostáva z niekoľkých vrstiev zložených z konvolučnej a združovacej (pooling) vrstvy. **Konvolučná vrstva** na vstup aplikuje konvolúciu. Za každou konvolučnou vrstvou nasleduje **aktivačná funkcia** (konkrétne ReLU), ktorá slúži na nelineárnu-transformáciu a rozhoduje o aktivácii príslušného perceptrónu. **Združovacia (pooling) vrstva** slúži na ďalšie zredukovanie počtu príznakov. Klasifikačná časť predstavuje klasickú **plne-prepojenú neurónovú sieť** pre následné zaradenie obrázkov do príslušných kategórií. (prevzaté z [2])

obrázku 2.8. Obrázok 2.9 následne zobrazuje bežnú konvolučnú vrstvu s viacero vstupnými a výstupnými kanálmi.

2.3.4 Aktivačná funkcia/vrstva

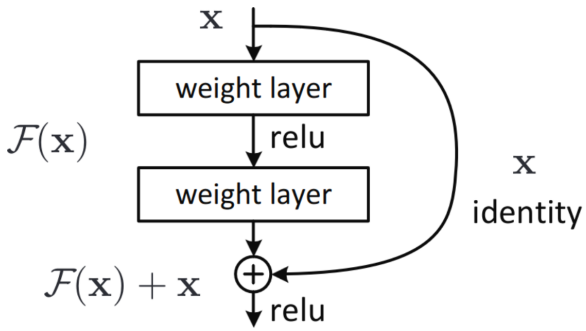
Aktivačná funkcia [7] je nelineárna funkcia slúžiaca na výpočet hodnôt skrytých vrstiev. Nelineárne zložky v neurónových sietiach sú nevyhnutné pre riešenie problémov strojového učenia, keďže väčšina z nich nemá lineárne riešenie.

V práci je využitá aktivačná funkcia ReLU. Ide o štandardne používanú funkciu, ktorá je takmer lineárna, vďaka čomu si zanecháva jednoduchosť optimalizácie typickú pre lineárne modely aj pri zabezpečení nelinearity. Znázornená je na obrázku 2.10.

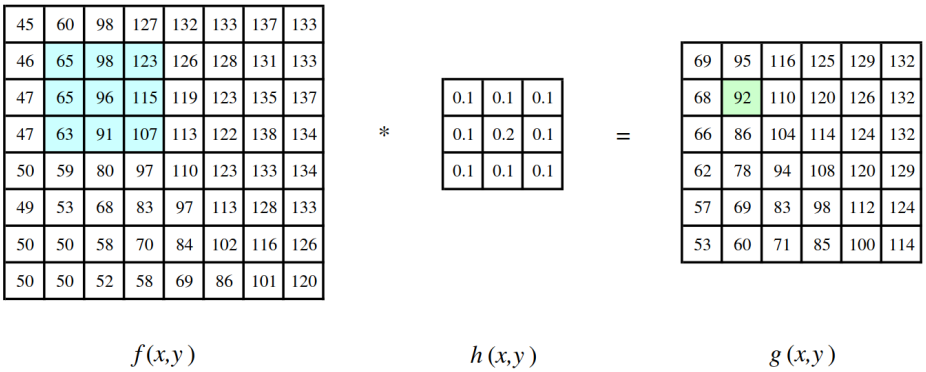
2.3.5 Združovacia (pooling) vrstva

Združovacia vrstva [7, 2] redukuje priestorovú veľkosť reprezentácie pred vstupom do ďalšej vrstvy. Vstup je rozdelený na štvorcové okolia a pre každé sa vypočíta určitá sumárna hodnota na výstup. Hodnota sa líši podľa typu združovacej vrstvy. Najčastejšie využívanými združovacími vrstvami sú max pooling a mean pooling, obrázok 2.11.

Významom združovacej vrstvy je okrem redukcie veľkosti a výpočtovej náročnosti aj zabezpečenie invariantnosti reprezentácie voči malým zmenám vstupu. To je dôležité v prípadoch, kedy nám viac záleží na prítomnosti určitého príznaku než jeho presnej pozícii. Príkladom môže byť napríklad nutnosť zistenia prítomnosti očí v správnej časti tváre pri jej rozpoznávaní, pričom ale nie je podstatné ich úplne presné umiestnenie v pixeloch.



Obr. 2.7: **Reziduálny blok:** Znázornené „skip connections“ (identity) sú v prípade reziduálneho bloku iba mapovania identity a ich výsledok (x) je pridaný k výstupu ($F(x)$). Vrstvy identity nijak neovplyvňujú výsledný počet parametrov modelu ani jeho výpočtovú komplexnosť. (prevzaté z [9])



Obr. 2.8: **Konvolúcia:** Operácia konvolúcie sa vykonáva na obrázku alebo mape príznakov ($f(x, y)$), s použitím jadra ($h(x, y)$). Výsledkom je mapa príznakov ($g(x, y)$). Na vstupe konvolúcie ($f(x, y)$) je farebne znázornené okolie, na základe ktorého bol vypočítaný konkrétny cieľový pixel, ktorý je taktiež farebne vyznačený. (prevzaté z [18])

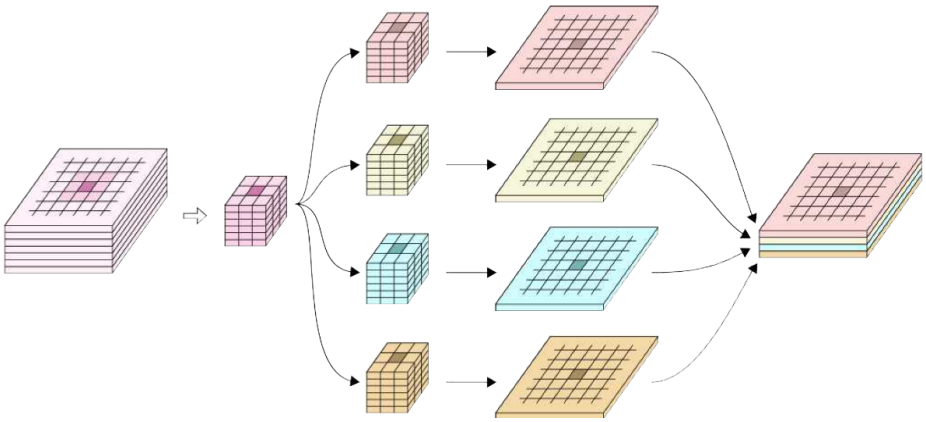
2.4 Nástroje použité pre implementáciu

Pre praktické trénovanie modelov som sa naučila používať viacero nových nástrojov, knižníc a frameworkov, ktoré sú bežne používané v oblasti strojového učenia. Taktiež som sa oboznámila s novými funkciami nástrojov, ktoré som bežne používala.

Pre implementáciu som si zvolila jazyk python, s použitím frameworku Pytorch. Pracovala som hlavne s knižnicami torch, torchvision, matplotlib a numpy. Pre vzdialený vývoj na serveri v prostredí Pycharm som použila nástroj JetBrains Gateway. Na severi som taktiež vytvorila virtuálne prostredie pomocou modulu venv. Datasety som vizualizovala a pracovala s nimi pomocou platformy Jupyter Notebook a nástroja Tensorboard, ktorý slúžil aj na vyhodnotenie výsledkov trénovania.

2.4.1 Python

Python je moderný a verzatilný programovací jazyk. Je najpopulárnejším v oblasti umelej inteligencie, hlavne vďaka svojej jednoduchosti a tomu, že ide o jeden z najefektívnejších nástrojov pre vizualizáciu.



Obr. 2.9: **Konvolučná vrstva:** V konvolučnej vrstve sa najčastejšie vykonáva 2D konvolúcia s viacerými vstupnými a výstupnými kanálmi. Každé 2D konvolučné jadro spracuje všetky vstupné vrstvy a vypočíta pre ne konvolúciu pre jeden z výstupných kanálov. Celkový počet natrénovateľných parametrov (P) je daný vzorcom, $P = V^2 + K_1 + K_2$ kde V^2 je veľkosť jadra v pixeloch, K_1 reprezentuje počet vstupných a K_2 výstupných kanálov. (prevzaté z [18])

2.4.2 Pytorch

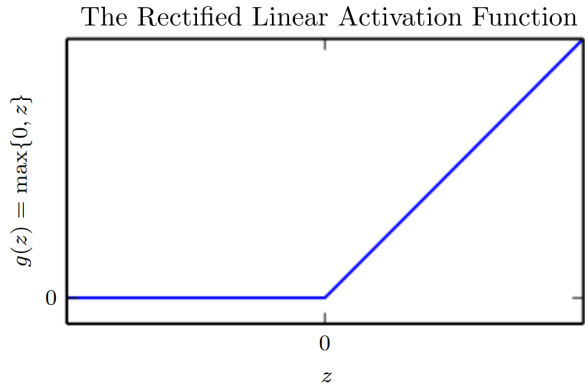
Pytorch je open source framework pre strojové učenie. Jeho základom je knižnica torch. Práca je v ňom implementovaná z dôvodu rastúcej popularity a využitia v obore strojového učenia, predovšetkým v oblasti výskumu. Jeho súčasťou je knižnica torchvision, ktorá obsahuje populárne datasety, architektúry modelov a transformácie.

2.4.3 Tensorboard

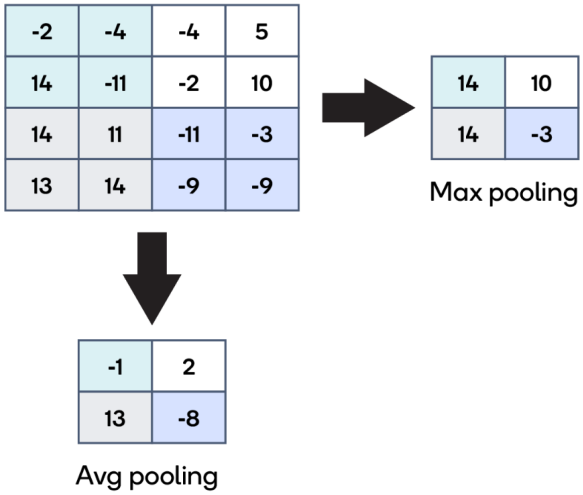
Tensorboard je nástroj od spoločnosti TensorFlow (ale je podporovaný aj v Pytorch), slúžiaci na zber dát a vizualizáciu potrebnú počas trénovania. Umožňuje vizualizovať dataset a zaznamenávať metriky učenia, napríklad presnosť alebo stratu (výsledok stratovej funkcie). Metriky je následne možné vizualizovať do grafov.

2.4.4 Jupyter notebook

Jupyter notebook [1] je interaktívna webová platforma. Je súčasťou Project Jupyter a umožňuje kombinovať zdrojový kód s textom, vizualizáciami, tabuľkami, vzorcami a inými typmi médií. Podporuje viac ako 40 rôznych programovacích jazykov.



Obr. 2.10: **ReLU:** ReLU aplikuje na vstup nelineárnu transformáciu. Táto funkcia je ale zložená z lineárnych funkcií, preto si ponecháva mnoho výhod lineárnych modelov, ktoré vedú k jednoduchšej optimalizácií. Vzorec tejto funkcie je $y(z) = \max(0, z)$. (prevzaté z [7])



Obr. 2.11: **Max pooling a mean pooling:** Max pooling je typom združovacej funkcie, ktorá z každého okolia vyberá maximálnu hodnotu. Mean pooling (avg pooling) pre každé okolie vypočíta priemernú hodnotu. (prevzaté z [2])

Kapitola 3

Existujúce riešenia a súvisiaca práca.

Pre rozpoznávanie pozícií existuje mnoho techník. V prvej časti tejto kapitoly sú zhrnuté rôzne spôsoby, akými dnes rozpoznávame pozície v obraze. V svojej práci sa zameriavam na rozpoznávanie pomocou self-supervised učenia s použitím hlbokých neurónových sietí, preto táto kapitola obsahuje len stručné zhrnutie iných prístupov. Keďže konečným cieľom práce je metóda aplikovateľná aj na rozpoznávanie športových pozícií, existujúce riešenia sú skúmané v kontexte pozícií rúk ale taktiež aj pre rozpoznávanie pozícií tela. Druhá časť poukazuje na výhody self-supervised prístupu k riešeniu tejto úlohy a popisuje niektoré existujúce self-supervised metódy.

3.1 Rozpoznávanie pozícií v obraze

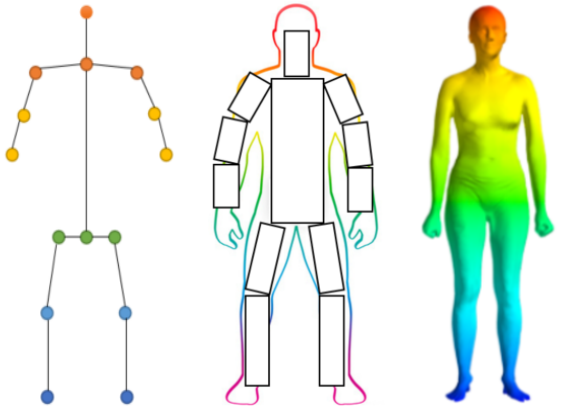
Techniky rozpoznávania pozícií [21] sa rozdeľujú na 3D a 2D. Taktiež sa líšia v modelovaní rozpoznávaného objektu. Modely môžu byť kinematické, planárne a volumetrické. Toto rozdelenie znázorňuje obrázok 3.1.

Väčšina metód pre 2D rozpoznávanie pozícií sa spolieha na kinematický model, teda lokalizuje pozície kĺbov. 2D metódy sa ďalej delia podľa počtu rozpoznávaných osôb.

- Metódy rozpoznávania jednej osoby sú **regresné**, ktoré predikujú súradnice kĺbov v obraze pred samotným rozpoznávaním pozície alebo **založené na teplotnej mape** (heatmap-based), ktoré používajú detektor častí tela pre zistenie pozície kĺbov.
- Metódy, ktoré rozpoznávajú viac osôb naraz delíme na postup **zhora dole** (top-down) a **zdola hore** (bottom-up). Zatiaľ čo prvý prístup najprv detekuje ohraničenia osôb a následne na každé ohraničenie osobitne aplikuje rozpoznávanie pozície, postup zdola hore ako prvé detekuje kĺby a až potom ich zoskupí podľa osôb.

3.1.1 DeepPose: Odhad ľudských pozícií pomocou hlbokých neurónových sietí

DeepPose [19] je metóda rozpoznávania pozícií založená na hlbokých neurónových sieťach. Ide o holistický prístup k rozpoznávaniu pozícií, ktorého výhodou je, že je podstatne jednoduchší na formuláciu ako metódy založené na grafických modeloch. Metóda nepotrebuje explicitný návrh detektorov pre časti tela, ani topológiu modelu a vzťahov medzi kĺbmi.



Obr. 3.1: Typy modelov pre rozpoznávanie pozícií:
Kinematický model reprezentuje ľudské telo (alebo ruku) ako skupinu kĺbov v určitých pozíciách. Je jednoduchý a intuitívny, nezachytáva však informácie o tvare a textúre.
Planárny model najčastejšie využíva pre aproximáciu kontúr ľudského tela obdĺžniky, ktoré rozdeľujú model na rôzne časti tela, podobne ako je to pri kinematickom modeli.
Volumetrický model je určený na 3D rozpoznávanie pozície a mal by byť schopný zachytiť dokonca aj deformácie tela zodpovedajúce určitým pozíciám. Ide o najpodrobnejší z modelov. (prevzaté z [21])

3.1.2 OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields

OpenPose [4] je 2D metóda rozpoznávania pozícií viacero ľudí v obraze, použitím prístupu zdola hore. Jej základom sú skupiny 2D vektorových polí (PAF: Part Affinity Fields), ktoré zachytávajú polohu a orientáciu končatín v obraze. Metóda je znázornená na obrázku 3.2.

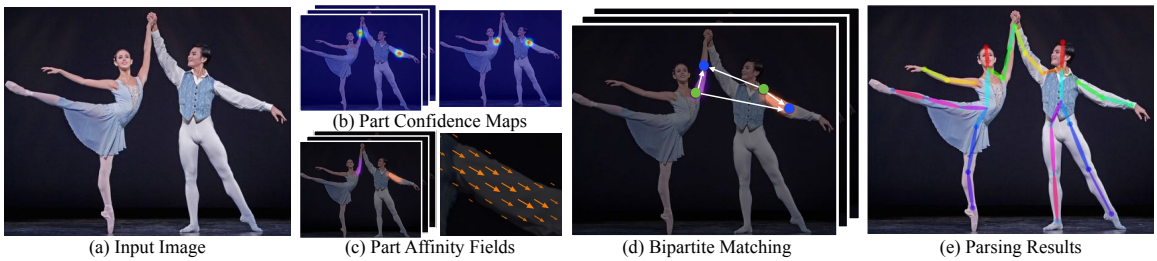
Výhodou oproti prístupu zhora dolu je hlavne menšia časová náročnosť. Trvanie rozpoznania použitím prístupu zhora dolu je totiž proporčné k počtu osôb, keďže nad každou pozíciou v obraze prebieha rozpoznávanie individuálne. To znemožňuje aj prípadnú korekciu chýb, hlavne pokiaľ sa dve osoby v obraze čiastočne prekrývajú, tento prístup je neúspešný.

3.2 Self-supervised učenie

Self-supervised učenie [18] je založené na transfer učení a doménovej adaptácii. Podstatou transfer učenia je natrénovanie modelu na určitú úlohu, a prenesenie tejto vedomosti (knowledge transfer) na inú úlohu. Doménová adaptácia pritom predstavuje samotné upravenie cieľovej neurónovej siete na zamýšlanú úlohu.

Základom self-supervised učenia je vytvorenie náhradnej úlohy, ktorá môže byť automaticky odvodená od charakteristiky neoznačeného datasetu alebo umelo vytvorená rozširovaním dát (data augmentation).

Úlohou modelu je potom predikovať podmnožinu informácii z ich zvyšku. Náhradná úloha pritom môže mať rôznu formu. Výhodou tohoto prístupu je, že nie je potrebné veľké množstvo dát a taktiež ani manuálna tvorba modelu rozpoznávaného objektu. Metóda je taktiež jedinečná v tom, že po natrénovaní modelu na náhradnej úlohe je možné model



Obr. 3.2: (a) Metóda OpenPose ako prvé použije celý obrázok ako vstup pre konvolučnú neurónovú sieť. Výstupom sú (b) mapy odhadu umiestnenia detekovaných častí tela a (c) skupiny 2D vektorových polí (PAF) zachytávajúce polohu a orientáciu končatín v obraze. (d) V ďalšom kroku sa časti tela porovnajú a (e) nakoniec sa z nich zostavia výsledné pozície osôb v obraze. (prevzaté z [4])

dotrénovať aj na rôzne ďalšie cieľové úlohy, teda jeho využitie nie je fixné len na jednu konkrétnu úlohu.

3.2.1 SimCLR: A Simple Framework for Contrastive Learning of Visual Representations

SimCLR [6] je jednoduchá metóda, ktorá uľahčuje predošlé kontrastívne self-supervised metódy tým, že nepotrebuje špeciálnu architektúru ani pamäťové banky. Autori zdôrazňujú dôležitosť správneho výberu transformácií pre rozšírenie dát. Taktiež udávajú, že pre tento typ učenia je vhodný väčší batch size, konkrétne používajú batch size od 256 do 8192.

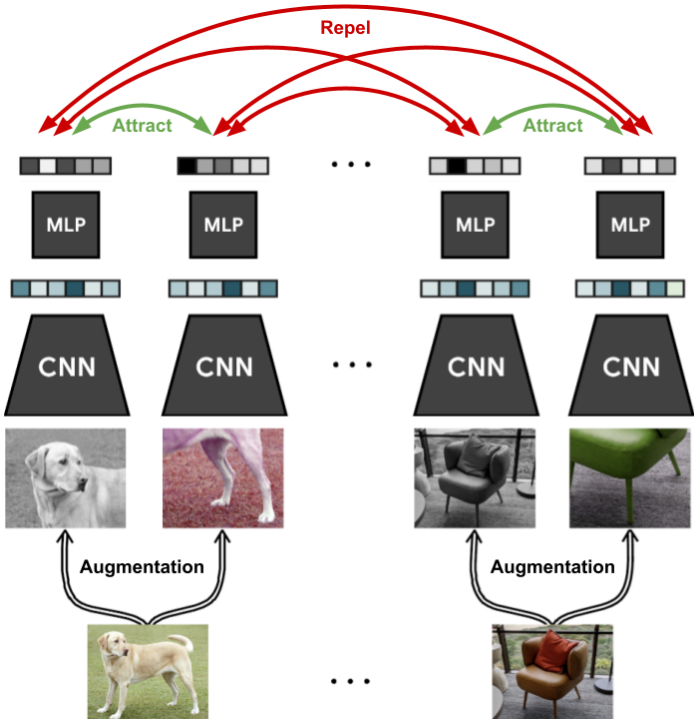
Metóda SimCLR spočíva v aplikovaní rôznych transformácií, pre vytvorenie pozitívneho páru. Z transformácií je najdôležitejšia kombinácia náhodného orezania a skreslenia farby, ktorá je kľúčová pre dosiahnutie dobrých výsledkov. Negatívne vzorky nie sú vytvorené explicitne, ale všetky vzorky v rámci batch, okrem daného pozitívneho páru, sú považované za negatívne vzorky k nemu. Metóda využíva trojicovú stratovú funkciu a snaží sa minimalizovať stratu, teda maximalizovať zhodu medzi pozitívnymi pármi. Hlavná myšlienka tejto metódy je zobrazená na obrázku 3.3.

3.2.2 BYOL: Bootstrap Your Own Latent

Ide o nekontrastívnu metódu, používajúcu iba pozitívne páry bez nutnosti vytvárať a priraďovať negatívne. Jednou z jej výhod je, že nepotrebuje veľký batch size. BYOL [8] sa spolieha na dve neurónové siete, označované aktívna (online) a cieľová (target), ktoré majú rovnakú architektúru a učia sa od seba. BYOL trénuje aktívnu sieť aby predikovala reprezentáciu iného rozšíreného pohľadu rovnakého obrázku, vytvoreného cieľovou sieťou. Cieľová sieť je optimalizovaná len skrz výpočet exponenciálneho kĺzavého priemeru (EMA) váh aktívnej siete. Architektúru oboch sietí možno vidieť na obrázku 3.4.

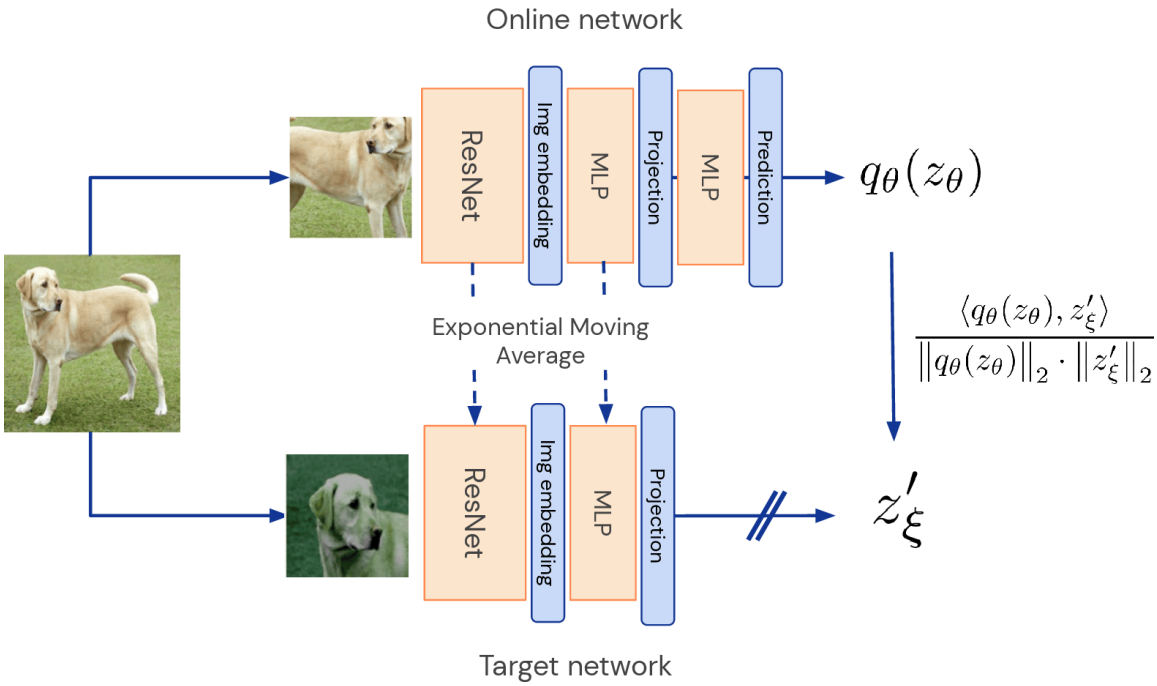
3.2.3 DINO: self-distillation with no labels

Alternatívou ku konvolučným neurónovým sieťam sa v poslednej dobe stávajú transformátory obrazu (ViT: Vision Transformer), ktoré sú aktuálne veľmi úspešné v oblasti self-supervised strojového učenia. Metóda s názvom DINO [5] nepoužíva ako model konvolučné neurónové siete, ale spomínané transformátory obrazu a je znázornená na obrázku 3.6.

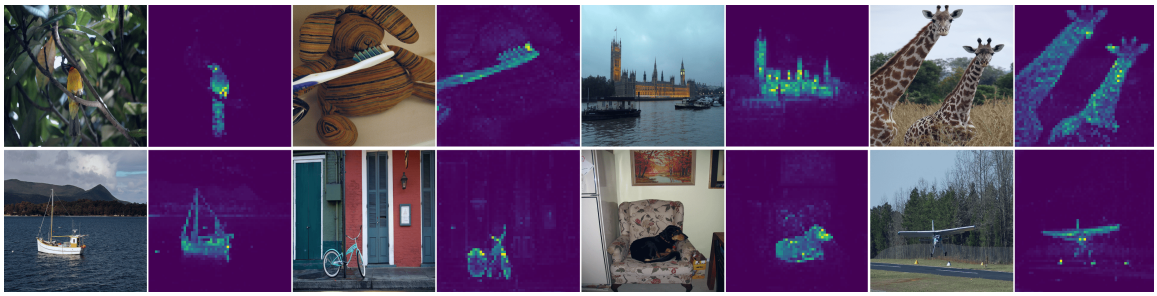


Obr. 3.3: Metóda SimCLR: Metóda spočíva vo vytvorení dvoch rôznych rozšírení každého obrázku v datasete. Tieto dve rozšírenia sú považované za pozitívny pár. Rozšírenia iného obrázku predstavujú pre tento pár negatívne vzorky. Sieť je následne trénovaná na vytváranie podobnej vektorovej reprezentácie pre pozitívny pár a čo najodlišnejšej pre dve rozšírenia, prislúchajúce rôznym objektom. (prevzaté z [6])

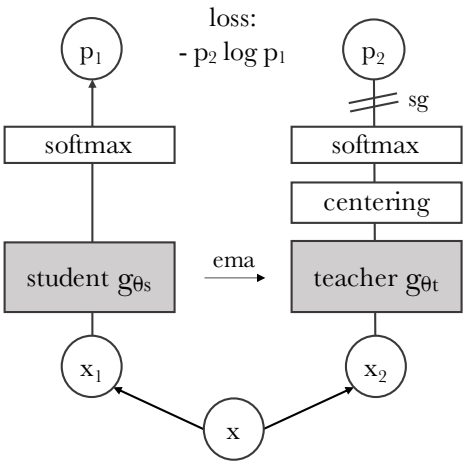
V supervised metódach potrebujú transformátory viac dát a kvalitou rozpoznávania a príznakmi sa špecificky nelíšia. Pri použití ViT pre self-supervised učenie však naučené príznaky explicitne obsahujú rozloženie scény, hlavne hranice objektov, čo možno vidieť na obrázku 3.5. Ich nevýhodou je však potreba oveľa väčšieho množstva neoznačených dát.



Obr. 3.4: Znázornenie metódy BYOL: Architektúra oboch sietí, sa skladá z rozšírenia pohľadu, reprezentácie a projekcie. Aktívna sieť je rozšírená o ďalší viac-vrstvový perceptrón (MLP), ktorý slúži na predikovanie projekcie cieľovej siete. BYOL minimalizuje podobnostnú stratu medzi touto predikciou a ozajstnou projekciou. Na konci trénovania je využitá len časť reprezentácie aktívnej siete, ktorej architektúra je založená na modeli ResNet50. (prevzaté z [8])



Obr. 3.5: Self-attention DINO: Mapy vedľa obrázkov z použitého datasetu znázorňujú príznaky ktoré sa model naučil pomocou učenia bez učiteľa.Možno na nich vidieť, že model sa automaticky naučil rozlišovať príznaky špecifické pre konkrétne triedy obrázkov. (prevzaté z [5])



Obr. 3.6: Metóda DINO pre jeden pár pohľadov (x_1, x_2). Študentskej a učiteľskej sieti sú poskytnuté dve rôzne náhodne transformácie vstupného obrázka. Obe siete majú rovnakú architektúru ale iné parametre. Výstup učiteľskej siete je centrovaný priemerom vypočítaným pre batch. Výstupom oboch sietí je K-rozmerný príznak normalizovaný teplotnou funkciou softmax voči dimenzii príznaku. Podobnosť príznakov je potom vypočítaná použitím krížovej entropie. Aplikovaním operátora stop-gradient (sg) na učiteľa je zabezpečená propagáciu gradientu len cez študenta. Učiteľské parametre sú aktualizované pomocou exponenciálneho kĺzavého priemeru (EMA) parametrov študentskej siete. (prevzaté z [5])

Kapitola 4

Postup tvorby datasetov a návrhu self-supervised metódy

4.1 Postup práce

Keďže bola tématika strojového učenia pre mňa úplne nová, prácu som začala osvojením si základných znalostí z tejto oblasti a následným hlbším štúdiom konkrétnych podoblastí strojového učenia súvisiacich s mojou prácou.

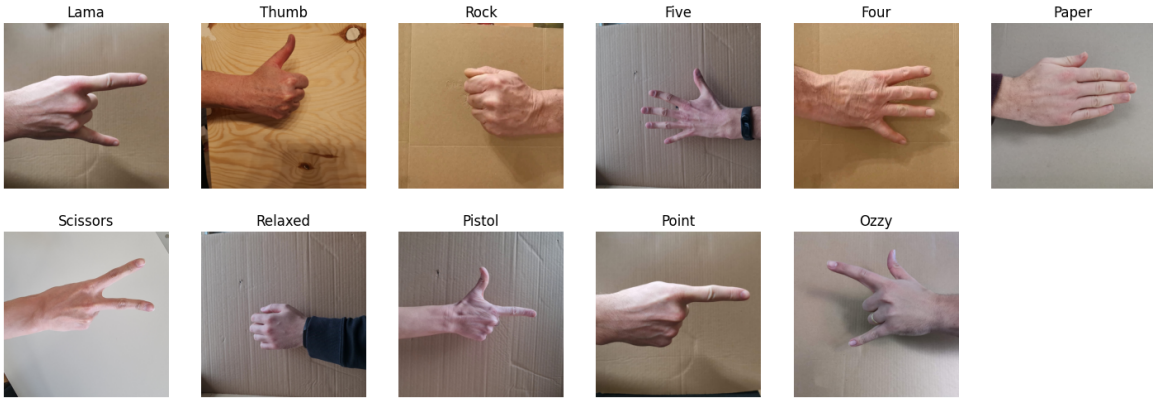
Na začiatok som sa naučila, ako sa trénuje konvolučná neurónová sieť pre úlohu klasifikácie objektov na datasete CIFAR10, s pomocou návodov na internete. Pre vykonanie rovnakej úlohy na datasete CIFAR100 som už musela, kvôli nedostatku výpočtovej pamäte na svojom zariadení, využiť vzdialenú prácu na serveri sophie1. Ide o školský server so štyrmi grafickými kartami NVIDIA RTX A5000 s operačnou pamäťou 24GB. Na serveri som si vytvorila virtuálne prostredie pomocou modulu *venv* a využila nástroj JetBrains Gateway pre vzdialený vývoj v prostredí Pycharm.

Následne som sa naučila pracovať s nástrojom Tensorboard pre vyhodnocovanie trénovania. Musela som taktiež zistiť ako používať Tensorboard na vzdialenom serveri, keďže na ňom prebiehali trénovania ktoré som chcela vizualizovať. Takisto som sa naučila základy práce s datasetmi a spôsoby ich vizualizácie. Túto časť práce som implementovala v prostredí jupyter notebook a využila pri tom knižnicu matplotlib.pyplot pre tvorbu grafov.

Po získaní datasetov pozícií rúk od vedúceho som začala supervised učenie klasifikácie pozícií pre referenčné výsledky. Toto učenie dosiahlo na datasete Rochambeau (kapitola 4.2.1) úspešnosť 87 %, a na datasete Handz (kapitola 4.2.2) 83 %.

Po preštudovaní rôznych metód self-supervised učenia som sa rozhodla pre použitie kontrastívnej metódy. Následne som implementovala prvú časť trénovania (kapitola 4.5). Pri tomto trénovaní som zistila, že znižovanie stratovej funkcie nie je vhodná metrika pre posudzovanie kontrastívneho učenia, preto som pre úspešnosť učenia použila iný spôsob vyhodnotenia. Kritériom pre úspešnú trojicu podľa novej techniky vyhodnocovania bola väčšia podobnosť pozitívneho páru než negatívneho. Vyhodnocovanie je bližšie popísané v kapitole 4.5.4.

Trénovanie modelu pre cieľovú úlohu (kapitola 4.6) bolo veľmi podobné ako pri tvorbe referenčného modelu, keďže išlo o klasickú klasifikačnú úlohu. Hlavným rozdielom bolo použitie modelu natrénovaného na náhradnú úlohu a jeho rozšírenie o lineárne vrstvy. Trénované boli iba novo pridané vrstvy a váhy pôvodnej časti modelu sa nemenili.



Obr. 4.1: Dataset Rochambeau obsahujúci 11 rôznych pozícií rúk.

Poslednou časťou práce bola optimalizácia, kedy som okrem zmeny hyperparametrov učenia a modelu experimentovala aj s technikou triplet mining, pre vhodnejší výber trojíc a lepšie výsledky.

4.2 Datasety

Pre prácu boli použité celkovo tri datasety, ktoré mi boli poskytnuté mojím vedúcim práce. Ide o datasety Rochambeau, Handz a dataset trojíc extrahovaných z troch súbežne nahraných videí. Datasety sú súčasťou práce, ktorá je popísaná v článku [11], ktorý je aktuálne vo fáze revízie.

4.2.1 Rochambeau

Dataset Rochambeau obsahuje 1895 obrázkov zachytávajúcich 11 rôznych pozícií rúk. Je inšpirovaný hrou kameň-papier-nožnice a pri jeho tvorbe bolo prihliadané na jeho využitie pre tento typ učenia. Dataset je vytvorený z pozícií poskytnutých len od malého počtu osôb, pretože aj pri rozšírení na športové pozície by bol množstvo nahratých videí a osôb v nich veľmi malé. Dataset obsahuje pre každú pozíciu podobný počet vzoriek. Jednotlivé pozície vidieť na obrázku 4.1.

4.2.2 Handz

Dataset Handz je o niečo rozsiahlejší. Obsahuje až 20 pozícií a celkovo má 2382 obrázkov. Dataset vznikol zo súbežne nahraných videí. Obsahuje rovnaké pozície ako dataset Rochambeau a je rozšírený o pozície, ktoré boli zachytené vo videách a nepatrili do pôvodného datasetu. Rozdelenie zastúpenia týchto pozícií je nerovnomerné, čo tiež približuje problematiku športových pozícií, keďže niektoré sú známejšie než iné. Obrázok 4.2 zobrazuje jednotlivé pozície a obrázok 4.3 znázorňuje ich distribúciu.

4.2.3 Dataset trojíc

Dataset obsahuje až 135 000 obrázkov, čiže 45 000 trojíc extrahovaných z videí, zachytávajúcich rôzne pozície rúk. Tento dataset neobsahuje žiadne označenia pozícií. Trojice obrázkov

Obr. 4.2: Dataset Handz obsahuje pôvodné pozície z datasetu Rochambeau a 9 ďalších, ktoré boli zachytené vo videách a nepatrili do pôvodného datasetu. Pozície sú reprezentované rôznym počtom vzoriek, čo simuluje možný dataset športových pozícií, kedy by známejšie alebo jednoduchšie pozície boli taktiež zachytené vo väčšom počte.

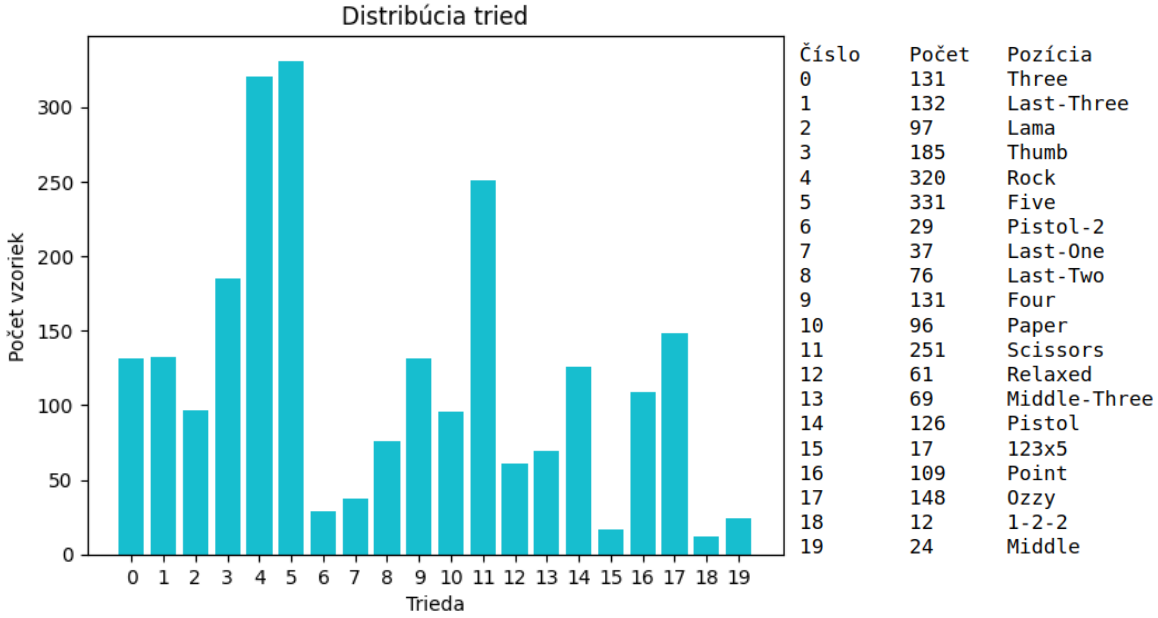
obsahujú bázový prvok (anchor) a k nemu pozitívny prvok zobrazujúci rovnakú pozíciu a negatívny zobrazujúci inú pozíciu. Príklady týchto trojíc možno nájsť na obrázku 4.4.

Skript pre tvorbu tohoto datasetu mi poskytol vedúci práce. Tento skript umožňuje extrahovať daný počet obrázkov z videa. Určenie správnych momentov pre extrakciu snímok je zabezpečené výpočtom celkového a koncentrovaného množstva pohybu.

4.3 Výsledná self-supervised metóda

Výhodou self-supervised prístupu pre rozpoznávanie pozícií je jeho nenáročnosť na zdroje. Nepotrebovala som teda veľké množstvo označených dát a pre trénovanie postačila trojica súbežných videí obsahujúca rôzne pozície rúk. Ďalej už bol potrebný len malý obnos označených dát na krátke dotrénovanie modelu na cieľovú úlohu, teda klasifikáciu.

Pre každú časť trénovania bol použitý rovnaký základný prístup k trénovaniu. Po každej epoche sa vyhodnotila aktuálna úspešnosť modelu na validačnom datasete a v prípade zlepšenia úspešnosti sa nový model uložil. Trénovanie bolo ukončené po stanovenom maxi-



Číslo	Počet	Pozícia
0	131	Three
1	132	Last-Three
2	97	Lama
3	185	Thumb
4	320	Rock
5	331	Five
6	29	Pistol-2
7	37	Last-One
8	76	Last-Two
9	131	Four
10	96	Paper
11	251	Scissors
12	61	Relaxed
13	69	Middle-Three
14	126	Pistol
15	17	123x5
16	109	Point
17	148	Ozzy
18	12	1-2-2
19	24	Middle

Obr. 4.3: Dataset Handz je zložený z 20 rôznych pozícií. Dataset obsahuje veľmi nerovnomerný počet reprezentácií pre jednotlivé pozície, v závislosti od toho aká je daná pozícia bežná. Podobná situácia by sa dala očakávať aj pri tvorbe datasetu športových pozícií.

málnom počte epoch alebo predčasne, pokiaľ sa za daný počet epoch úspešnosť nezlepšila. Ide o implementáciu optimalizačnej stratégie early stopping, ktorá bude detailnejšie popísaná v kapitole 4.7.

4.4 Supervised učenie pre referenciu

Aby som výsledky učenia mala s čím porovnať, na začiatku som vytvorila skript *reference.py* pre natrénovanie siete ResNet50 použitím metódy učenia s učiteľom. Tento skript sa nachádza medzi zdrojovými textmi programov, ktoré sú priložené k práci. Táto príloha tiež obsahuje aj ďalšie skripty, ktoré sú spomenuté v nasledujúcich kapitolách.

Trénovanie som vykonala pre datasety Handz a Rochambeau. Na oboch datasetoch sa sieť natrénovala podobne dobre. Do 50-tich epoch dosiahol model na validačnom datasete úspešnosť okolo 83 % pre dataset Handz. Keďže dataset Rochambeau je asi o 20 % menší, nastavila som preň maximálny počet epoch na 70. Model trénovaný na datasete Rochambeau takisto dosiahlo vysokú úspešnosť, konkrétne 88 %. Následne obe siete dosiahli veľmi podobnú úspešnosť aj na testovacom datasete. Celkovo sieť obsahovala 25 miliónov parametrov.

Zo začiatku som pri testovaní použila obrázky veľkosti 128×128 pixelov. V tomto prípade bola úspešnosť iba okolo 70 %. Po zdvojnásobení veľkosti obrázkov sa výsledky značne zlepšili. Okrem veľkosti obrázkov som experimentovala aj so zmenou ostatných hyperparametrov. Po viacerých pokusoch sa ako optimálne ukázalo použiť batch o veľkosti 16, počet epoch 50 a pre rýchlosť učenia zvoliť hodnotu 0.004.

Vyhodnocovanie presnosti na validačnom datasete prebiehalo v každom kroku a uložená bola vždy len nová najlepšia úspešnosť. Vyhodnotenie na trénovacom datasete prebehlo až

Obr. 4.4: Dataset trojíc sa skladá z trojíc obrázkov, ktoré obsahujú bázovú (anchor), pozitívnu a negatívnu vzorku. Pozitívne vzorky reprezentujú tú istú pozíciu. Negatívna vzorka by mala reprezentovať inú, ideálne podobnú pozíciu.

po dotrénovaní a to na poslednej uloženej, teda najúspešnejšej sieti. Podobnosť úspešnosti na validačných a testovacích datasetoch dokazuje, že nedošlo k pretrénovaniu sietí.

4.5 Náhradná úloha

Cieľom náhradnej úlohy (pretext task) je naučiť model na veľkom množstve neoznačených dát dobré pochopenie obrazu. Táto vedomosť sa potom prenesie na cieľovú sieť a množstvo dát potrebné pre následné dotrénovanie je veľmi malé. Náhradná úloha je vytvorená na základe charakteru neoznačených dát. Často sa používajú rôzne techniky augmentácie dát, naznačené na obrázku 2.2. Vďaka charakteristike datasetu trojíc však nebolo nutné používať žiadnu z týchto techník. Namiesto toho som využila prirodzenú informáciu obsiahnutú v datasete, ktorou je príslušnosť do trojice obrázkov, ktoré súbežne zachytávajú v rôznych uhloch tú istú pozíciu.

Toto trénovanie prebiehalo pomocou skriptu *pretext_train.py*. Skript obsahuje aj test trojíc, avšak pre možnosť kontroly výsledkov som vytvorila taktiež skript *pretext_test.py*, ktorý umožňuje načítať zvolený model a overiť jeho úspešnosť v teste trojíc na testovacích dátach.

4.5.1 Knižnica pre trojice obrázkov

Pre použitie datasetu som vytvorila vlastnú triedu, ktorá umožňuje načítať bázovú vzorku a k nej príslušnú pozitívnu a negatívnu vzorku naraz ako jednu trojicu. Jej základom je bázová trieda pre vlastné datasety ImageFolder, ktorá je vstavaná v module torchvision. K tejto triede som dodefinovala vlastnú metódu pre zistenie dĺžky datasetu a pre získanie konkrétnej položky v datasete.

4.5.2 Transformácie

Vďaka charakteristike datasetu a zvolenej metóde učenia nebolo nutné používať veľké množstvo transformácií. Pre väčšinu trénovaní som preto použila iba normalizáciu a náhodné vodorovné pretočenie. Vplyv tranformácií som však otestovala pri niektorých trénovaniach

aplikovaním celej skupiny náhodných transformácií, ktoré dávali pre charakteristiku úlohy zmysel. Zistila som ale, že veľké množstvo použitých transformácií nemalo vplyv na kvalitu učenia.

4.5.3 Model použitý pre učenie

Torchvision.models je súčasťou knižnice torchvision. Obsahuje definície rôznych modelov, ktoré sú najčastejšie využívané pre rôzne úlohy v oblasti počítačového videnia. Pre trénovanie som použila modely ResNet, založené na práci He et al. [9], pričom v torchvision je implementovaná mierne optimalizovaná verzia tohoto modelu. Na začiatku prebiehalo trénovanie na modeli ResNet50 a pri optimalizovaní som využila aj modely ResNet34 a ResNet18, ktoré obsahujú menej skrytých vrstiev a teda aj menej parametrov. Sieť ResNet50 má približne 25,5M a ResNet34 skoro 22M parametrov. Najväčší rozdiel je však pri modeli ResNet18, ktorý ich má iba 11,7M.

4.5.4 Spôsob vyhodnotenia trojíc

Pre zhodnotenie trénovania sa najčastejšie využíva stratová funkcia. Model sa učí pokým sa stratová funkcia znižuje. Pri trojicovej stratovej funkcii však táto metrika nie je vhodná na vyhodnotenie. Je to z dôvodu, že namiesto minimalizácie je cieľom, aby euklidovská vzdialenosť vektorovej reprezentácie medzi bázovou a pozitívnou vzorkou bola stále menšia a vzdialenosť medzi bázovou a negatívnou výrazne väčšia. Preto je vhodnejším kritériom pre vyhodnocovanie splnenie tejto podmienky. Z definície trojicovej stratovej funkcie 2.2.3 vyplýva, že toto kritérium je splnené pokiaľ je strata menšia ako zvolené ohraničenie.

Túto podmienku kontrolujem pre celý validačný dataset po každej epoche a následne vyhodnotím úspešnosť. Pri každom zlepšení úspešnosti sa model uloží a pokiaľ je úspešnosť 100 %, trénovanie je ukončené.

Pre overenie správnosti tohoto kritéria a zabránenie pretrénovaniu som použila pre cieľovú úlohu dva modely z toho istého trénovania. Jeden z nich bol uložený v momente, keď sieť prvýkrát dosiahla viac ako 99-percentnú úspešnosť a druhý pri dosiahnutí úspešnosti 100%. Ukázalo sa, že aj tento rozdiel je dôležitý. Pri prvej dvojici modelov sieť dosiahla lepšiu úspešnosť, pri druhej bola úspešnosť podobná, ale sieť s lepšou úspešnosťou pri náhradnej úlohe potrebovala na dosiahnutie rovnakej úspešnosti klasifikácie len polovicu epoch. Tieto výsledky sú znázornené na obrázku 5.3 v nasledujúcej kapitole 5.

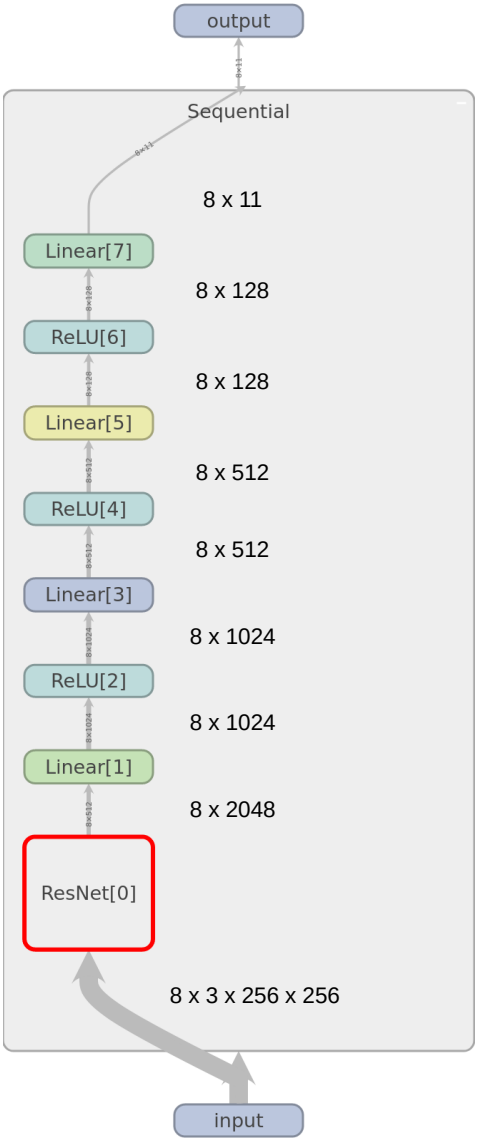
4.6 Cieľová úloha

Cieľovou úlohou pre model je klasifikácia, teda schopnosť rozpoznať konkrétnu pozíciu ruky z obrazu. Keďže model bol v prvej časti trénovaný iba na neoznačených dátach, cieľom náhradnej úlohy je správne rozlíšiť vektorovú reprezentáciu rôznych pozícií a každú priradiť k príslušnej triede. Toto trénovanie prebieha pomocou skriptu *downstream_train.py* na malej časti celkového modelu s použitím datasetov Handz a Rochambeau.

4.6.1 Model pre dotrénovanie

Cieľová úloha dotrénováva iba určitú časť modelu. Pri trénovaní na cieľovú úlohu sa načítajú váhy modelu trénovaného na náhradnej úlohe, ktorý mal najlepšiu úspešnosť v teste trojíc. Následne sa pre všetky parametre modelu zastaví výpočet gradientu, čím sa zamedzí optimalizácii. Na pôvodnú sieť, bez jej poslednej vrstvy sa potom napoja ďalšie lineárne vrstvy

s aktivačnými funkciami. Pôvodný počet výstupných príznakov pre predposlednú vrstvu je vstupom pre lineárnu časť modelu a výstupom tejto časti je počet tried označeného datasetu, použitého pre trénovanie na cieľovú úlohu. Počet parametrov ktoré sa dotrénovávajú je potom pri ResNet50 2,7M a pre zvyšné dva modely 173 780. Model pre cieľovú úlohu založený na ResNet50 je znázornený na obrázku 4.5.



Obr. 4.5: Model pre cieľovú úlohu pri použití siete ResNet50 a datasetu Rochambeau: Vstup pre sieť predstavuje batch z 8 obrázkov, s 3 farebnými kanálmi a 256×256 pixelmi. Po prechode ResNet časťou siete vznikne z obrázkov vektorová reprezentácia s 2048 príznakmi. Pomocou lineárnych vrstiev sa veľkost vektora postupne zmenšuje na veľkosť odpovedajúcu počtu tried v datasete.

4.6.2 Spôsob vyhodnotenia klasifikácie

Vyhodnotenie presnosti prebieha po každej epoche na validačnom datasete a rovnako ako v prvej časti trénovania sa vždy pri zlepšení výsledku model uloží. Okrem celkovej úspešnosti sa zakaždým vypočíta aj čiastková úspešnosť pre jednotlivé triedy. Po každých piatich krokoch sa navyše zaznamenáva aj výsledok stratovej funkcie, ktorý slúži na kontrolu učenia. Na zhodnotenie úspešnosti modelu na trénovacom datasete slúži skript *downstream_test.py*.

4.7 Použité techniky pre optimalizáciu riešenia

Pre optimalizáciu som hľadala optimálne hodnoty hyperparametrov učenia. Následne som experimentovala aj s hĺbkou modelu ResNet, či rôznou architektúrou rozšírenia siete ResNet o lineárne vrstvy pre ich dotrénovanie na cieľovú úlohu. Okrem toho som pri trénovaní použila algoritmus pre predčasné zastavenie trénovania a experimentovala som aj s technikou triplet mining pre vhodný výber trojíc.

4.7.1 Algoritmus pre predčasné zastavenie trénovania

Predčasné zastavenie trénovania (early stopping)[7] je jednoduchá a často používaná stratégia pre optimalizáciu dĺžky učenia modelu. Je založená na predpoklade, že aj keď pre trénovací dataset hodnota stratovej funkcie stále klesá, po nejakom čase táto hodnota pre validačný dataset začne znova stúpať. Tento jav je známy ako pretrénovanie a predčasné zastavenie učenia je najjednoduchšou prevenciou proti nemu. Pre obe časti self-supervised metódy, ako aj pre referenčné učenie som implementovala túto stratégiu, preto pokiaľ nenastane zlepšenie úspešnosti modelu po určenom počte epoch, testovanie sa predčasne ukončí. Vzhľadom k pozorovaniam som ako ďalšie možnosti ukončenia trénovania pridala úspešnosť 100% na validačnom datasete a prekročenie maximálneho počtu epoch.

4.7.2 Triplet mining

Okrem optimalizácie pomocou vhodného výberu hyperparametrov učenia a transformácií datasetu som sa pokúsila o optimalizáciu pomocou techniky triplet-mining. Zistila som, že pre kontrastívne učenie je dôležitý správny výber negatívnych vzoriek v trojici, ako aj to, aby bol počet týchto vzoriek v rámci batch čo najväčší. Pri zvyšovaní batch size som však narazila na problém s nedostatkom pamäte. Alternatívnym riešením k väčšiemu počtu negatívnych vzoriek bol ich vhodný výber. Toho sa dá dosiahnuť použitím stratégie triplet mining, ktorá rozdeľuje negatívne vzorky na ľahké, polo-ťažké a ťažké, podľa výsledku trojicovej stratovej funkcie pre trojicu, v ktorej sa nachádzajú. Myšlienkou je teda eliminovať z várky tie vzorky, ktoré sú pre trénovanie zbytočné. Kvalitnejšie vzorky tak umožnia obsiahnuť, napríklad pre batch o veľkosti 64, rovnako veľa informácií pre učenie, ako v batchi dvojnásobnej veľkosti bez použitia tejto techniky.

Podarilo sa mi implementovať offline verziu tejto optimalizačnej stratégie, ktorá po každej epoche musí prejsť celý dataset a vyhodnotiť, ktoré vzorky použiť v ďalšej epoche. To násobne zvýšilo dĺžku testovania. Implementovať online verziu nebolo možné, kvôli charakteristike datasetu, kde sú trojice už explicitne vytvorené a ich miešaním by mohli vzniknúť trojice kde je negatívny prvok rovnaký ako bázový a pozitívny.

Zo začiatku som používala iba ťažké trojice (hard triplets), avšak to sa neukázalo až také úspešné. Miesto očakávaného znižovania počtu ťažkých trojíc sa tieto trojice iba menili ale ich počet bol vždy podobný. Ďalej som teda skúsila použiť aj nejakú časť polo-ťažkých

trojíc. Experimentovala som aj s použitím techniky triplet mining iba v niektorých epochách a použitím celého pôvodného datasetu v ďalších. Výsledky týchto experimentov sú v kapitole 5.

Kapitola 5

Výsledky trénovania a experimentov

Výsledky trénovania som vyhodnocovala pomocou nástroja Tensorboard. Zaznamenané je supervised trénovanie pre referenciu aj následné trénovania použitím self-supervised metódy.

Self-supervised metóda má dve časti, no podstatnejšie boli výsledky cieľovej úlohy. Preto som pre prvú časť overovala iba relevantnosť testu trojíc pre následné trénovanie na cieľovú úlohu a keď bolo úspešné, zameriavala som sa primárne na porovnanie výsledkov z trénovania na cieľovej úlohe.

Grafy boli vyhladené pomocou funkcie v nástroji Tensorboard preto sa môže stať, že výsledky občas nebudú na prvý pohlad očividné. V legende je uvedená aj úspešnosť modelu na testovacom datasete, ktorá sa v grafe nenachádza ale je najrelevantnejším hodnotením modelu.

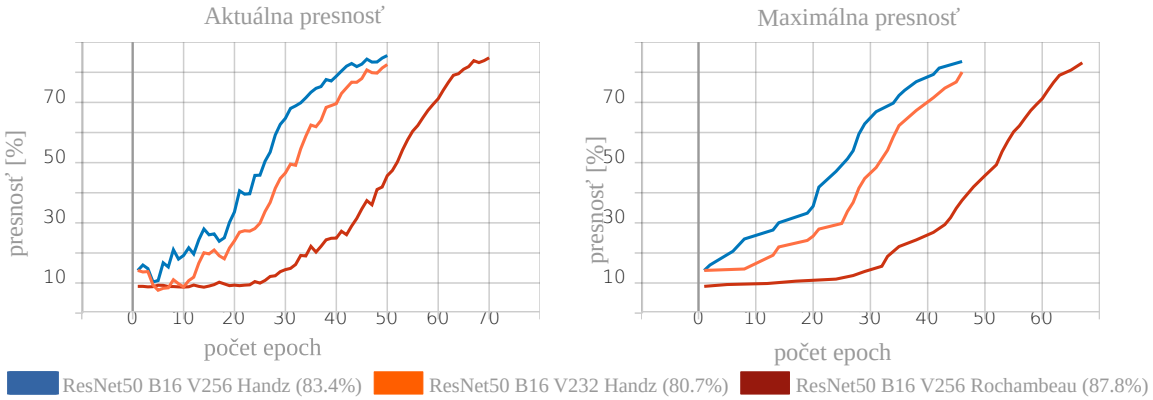
5.1 Referenčné supervised učenie

Referenčné supervised učenie prebiehalo na oboch označených datasetoch. Pôvodná veľkosť obrázka použitá pre trénovanie bola 128×128, no zväčšenie obrázka značne zvýšilo úspešnosť učenia, preto bola veľkosť nakoniec zdvojnásobená. Napriek tomu, že datasety neobsahovali veľa snímok, bolo možné natrénovať model až na úspešnosť 83 % pre dataset Handz a 88 % pre dataset Rochambeau. Oba výsledky boli veľmi podobné úspešnosti na validačnom datasete z čoho vyplýva, že nedošlo k pretrénovaniu.

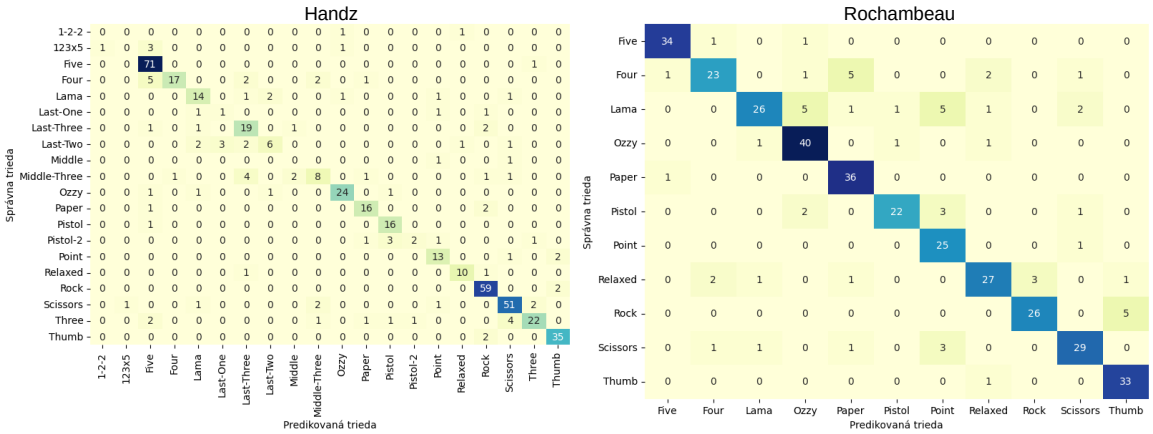
Výsledky sú na obrázku 5.1 a pre znázornenie, ktoré pozície si model najčastejšie zamieňal som vytvorila maticu zámen zobrazenú pre oba modely na obrázku 5.2.

5.2 Self-supervised učenie

Pre úspešnosť self-supervised učenia bolo dôležité, aby sa model pred trénovaním na cieľovú úlohu naučil dobrú reprezentáciu obrazu. Dosiahnutie najvyššej úspešnosti modelu síce väčšinou trvalo aj 100 epoch, prvotný rast úspešnosti modelu bol však veľmi rýchly. Model v prvých desiatkach epoch zvyšoval svoju úspešnosť niekoľkonásobne rýchlejšie než referenčný model, z čoho možno odvodiť, že sa úspešne naučil dobrú reprezentáciu obrazu.



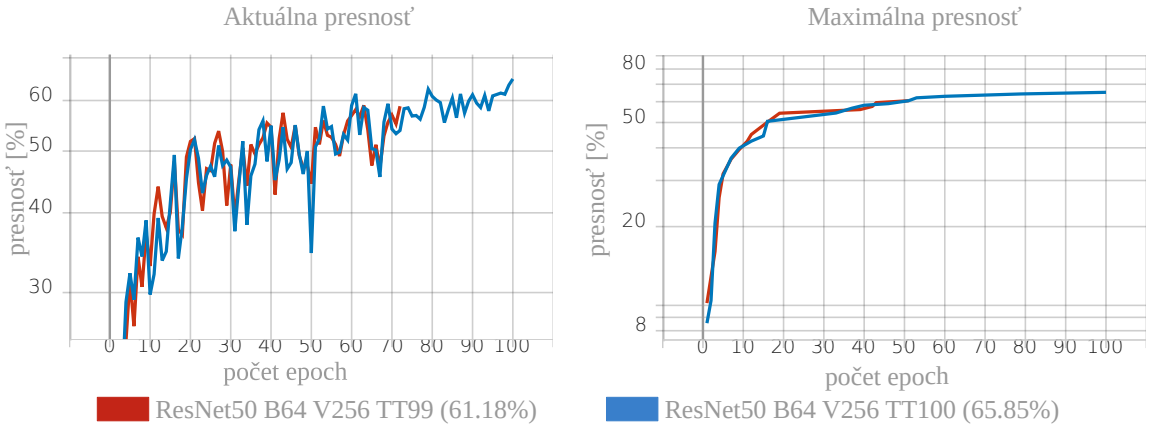
Obr. 5.1: ResNet50 supervised učenie: Referenčné supervised učenie malo vysokú úspešnosť pre oba datasety. Napriek tomu, že dataset Handz obsahuje pozície v nerovnomernom zastúpení, dosiahol model na tomto datasete podobnú úspešnosť ako na datasete Rochambeau. Maximálny počet epoch pre dataset Rochambeau bol z dôvodu menšieho počtu vzoriek vyšší ako pre dataset Handz. Veľmi podstatným sa ukázalo byť rozlíšenie obrázkov v datasete. Aj pri malej zmene veľkosti obrázku z 232×232 na 256×256 sa úspešnosť testovania viditeľne zlepšila.



Obr. 5.2: Matice zámen pre modely trénované na datasetoch Handz a Rochambeau znázorňujú, pri ktorých pozíciách model najčastejšie spravil chybu v predikcii a taktiež, ktoré pozície predikoval namiesto tej správnej. Pri datasete Rochambeau vo väčšine nesprávnych predikcií išlo o podobné pozície. Pri datasete Handz výsledky nie sú natoľko očividné, hlavne kvôli nerovnomernej reprezentácii jednotlivých pozícií. Možno však vidieť, že niektoré pozície model nepredikoval správne ani raz. Ide o tie pozície, ktoré boli reprezentované vo veľmi malom počte.

5.2.1 Overenie výsledkov testu trojíc

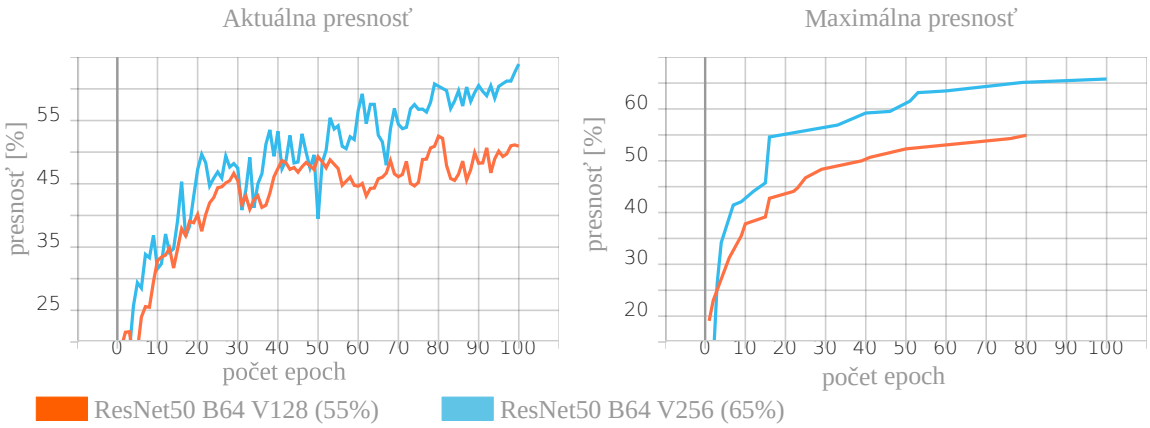
Po úspešnom overení relevantnosti testu trojíc, ktoré je znázornené na obrázku 5.3, som pre ďalšie trénovania použila vždy model s najvyššou úspešnosťou v teste trojíc. Nasledujúce experimenty som potom vyhodnocovala už len podľa úspešnosti na cieľovej úlohe.



Obr. 5.3: Na grafoch možno vidieť porovnanie modelov z rovnakého trénovania na náhradnej úlohe s rovnakou architektúrou sietí a hyperparametrami. Rozdiel medzi týmito modelmi bol iba v tom, kedy boli počas tréningu uložené a akú úspešnost mali v teste trojíc (TT99 a TT100 reprezentujú percentá v teste trojíc). Napriek tomu, že sa pri trénovaní na cieľovej úlohe výsledky veľmi nelíšia, výsledná úspešnosť modelu bola pri každej dvojici lepšia pre model s lepšou úspešnosťou v teste trojíc. Na grafe tiež možno vidieť, že došlo k predčasnému zastaveniu učenia pre druhý model, keďže sa daný počet epoch ďalej nezlepšovala jeho úspešnosť.

5.2.2 Závislosť úspešnosti od veľkosti obrázku

Napriek tomu, že už pri referenčnej úlohe sa ukázalo, že zväčšením rozlíšenia sa úspešnosť zlepšuje, rozhodla som sa overiť či táto skutočnosť platí rovnako aj pri self-supervised prístupe. Zistila som, že väčšie rozlíšenie obrázka takisto zlepšuje úspešnosť, tá však bola stále nízka. Na obrázku 5.4 možno vidieť, že najvyššia dosiahnutá úspešnosť bola približne 65 %.

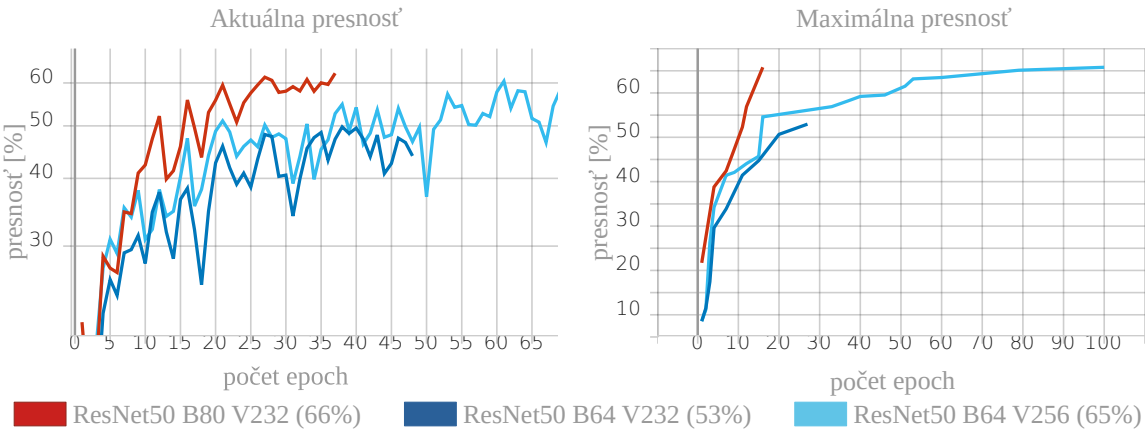


Obr. 5.4: Závislosť úspešnosti od rozlíšenia obrazu: Rovnako ako pri supervised metóde, zdvojnásobenie veľkosti obrázku z 128 × 128 na 256 × 256 výrazne pomohlo trénovaniu modelu a výslednú úspešnosť táto jediná zmena zvýšila až o 10 %.

5.2.3 Vplyv batch size na úspešnosť

Ako bolo spomenuté pri metóde SimCLR 3.3, pre kontrastívne učenie je vhodné použiť väčší batch size, keďže sa v ňom nachádza viac negatívnych vzoriek v rámci jedného kroku učenia. Pri rozlíšení 256×256 bolo možné použiť maximálne batch o veľkosti 64. Preto som rozlíšenie znížila na 232×232, čo umožnilo zvýšiť batch size na 80.

Z obrázku 5.5 vyplýva, že batch size ovplyvňuje učenie výraznejšie ako veľkosť obrázka, preto je vhodné jeho zvýšenie aj na úkor mierneho zníženia rozlíšenia vzoriek.



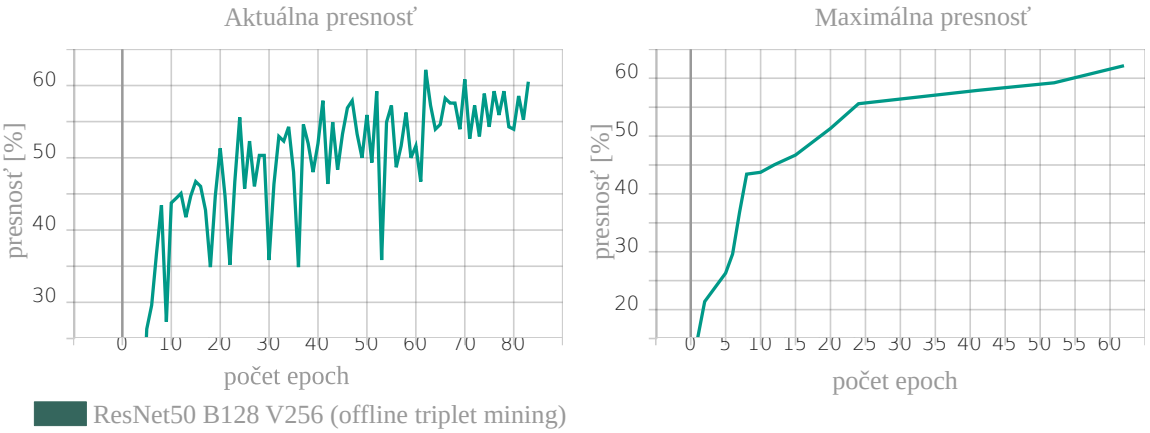
Obr. 5.5: Overenie vplyvu batch size na trénovanie ukázalo, že väčší batch kladne vplýva na úspešnosť trénovania a to vo väčšej miere ako rozlíšenie obrázku. Preto bolo v nasledujúcich trénovania cieľom tento hyperparameter maximalizovať. Zvýšenie batch o 16 vzoriek totiž aj pri miernom znížení rozlíšenia dokázalo už v 16 epochách prekonať ostatné testované modely. Všetky tri modely boli trénované na datasete Rochambeau.

5.2.4 Technika triplet mining

Z predchádzajúcich experimentov vyplynulo, že self-supervised učenie benefituje zo zvyšovania rozlíšenia obrázkov a batch size. Pri zvyšovaní batch size som ale narazila na hranicu pre nedostatok operačnej pamäte, preto som ako alternatívne riešenie implementovala triplet mining. Ten by sa mal vhodnejším výberom trojíc pre testovanie vyrovnať väčšiemu batch size v obsahu informácií a vhodných negatívnych vzoriek.

Implementovala som offline verziu tejto metódy, kedy bol pred každou epochou dataset aktualizovaný a z pôvodného datasetu boli použité iba ťažké negatívne vzorky (hard negatives). Očakávala som, že počet týchto nesprávne predikovaných trojíc, kde je negatívna vzorka vyhodnotená ako podobnejšia k bázovej než pozitívna, bude postupne klesať. Namiesto toho však počet vzoriek ostával podobný a učenie stagnovalo.

Pre ďalšie trénovanie som preto použila aj istú časť polo-ťažkých negatívnych vzoriek, ktoré mali zabrániť aby sa model optimalizoval na úkor správne predikovaných trojíc. Úspešnosť sa tým zlepšila, no výsledky sa stále nezlepšili dostatočne aby sa táto metóda ukázala ako užitočná. Najlepšiu úspešnosť s technikou triplet mining som dosiahla pri jej kombinácii s pôvodnou metódou využívajúcou celý dataset, ktorá je zobrazená na obrázku 5.6.



Obr. 5.6: **Výsledky použitia techniky triplet mining**: Metóda triplet mining sa pre tento prípad neukázala ako vhodná. Aplikovať bolo možné iba offline verziu tejto techniky, ktorá nezvyšuje počet možných trojíc, iba limituje použité trojice z pôvodného datasetu. Cieľom bolo vyhnúť sa použitiu trojíc nazývaných aj jednoduché, ktoré neprispievajú k optimalizácii modelu, a prenechať tým miesto v rámci batch pre vhodnejšie trojice. To malo viesť k rovnakému efektu, ako navýšenie batch size, ktoré nebolo pre nedostatočný výpočtový výkon možné. Ako však výsledky naznačujú, tento zámer nebol úspešný. Úspešnosť modelu bola iba 58.5 % a trénovanie sa len predĺžilo kvôli prechodu celého datasetu po každej epoche, pre výber vhodných trojíc.

5.2.5 Experimentovanie s hĺbkou modelu

Keďže technika triplet mining učeniu nepomohla a stále nebolo možné pre neurónovú sieť ResNet50 zvýšiť batch size, rozhodla som sa experimentovať s modelmi ResNet18 a ResNet34, pri ktorých bolo možné, vďaka ich menšiemu počtu parametrov, podstatne navýšiť batch size. Vďaka tomu som dosiahla zatiaľ najlepšiu úspešnosť až 74 % na datasete Rochambeau a 78 % na datasete Handz, s použitím modelu ResNet18. Výsledky sú znázornené na obrázku 5.7.

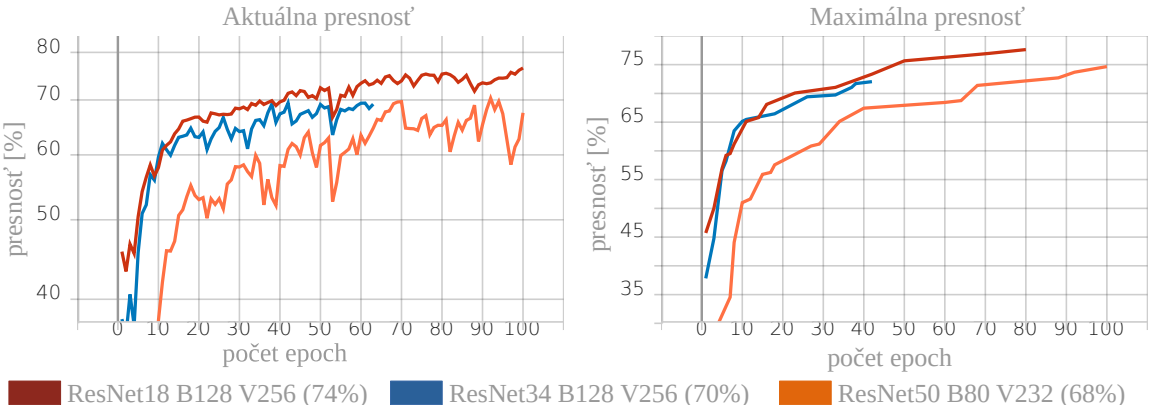
5.2.6 Optimálna architektúra lineárnej vrstvy

Po experimentovaní s architektúrami modelu pre trénovanie na náhradnej úlohe som sa rozhodla optimalizovať architektúru lineárnej vrstvy, o ktorú je model ResNet rozšírený pre cieľovú úlohu. Rôzne architektúry som testovala pre bázový model ResNet34 pri konštantných hyperparametroch. Výsledky testovania možno vidieť na obrázku 5.8.

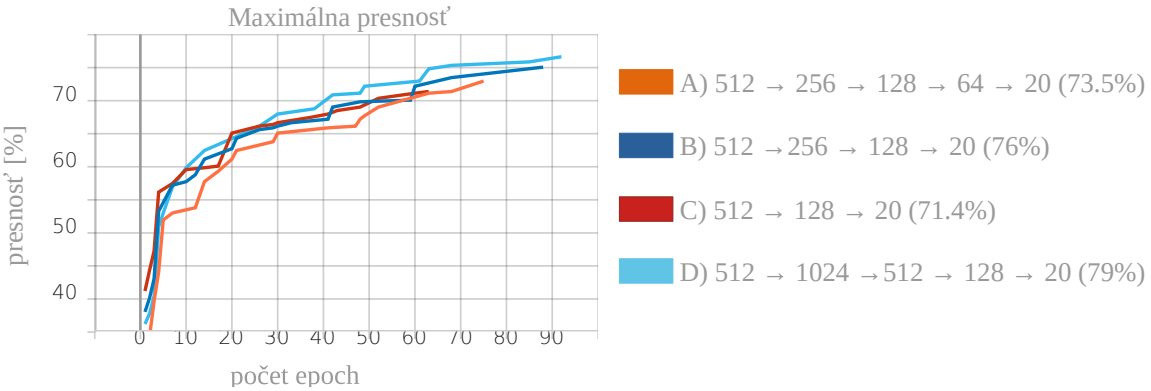
5.2.7 Optimalizovaný model

Na konci práce som využila všetky získané poznatky ohľadom trénovania a pre konečné trénovanie použila model ResNet18 s batch size o veľkosti 128, veľkosťou obrázku 256×256 a štvorvrstvovú architektúru lineárnej vrstvy pre trénovanie s najvyššou úspešnosťou.

Experimentovala som s ďalším navýšením batch size, úspešnosť sa však nezlepšila. Pre sieť ResNet18 je teda pri týchto podmienkach nastavenie batch size na 128 optimálnym. Taktiež som vyskúšala použiť pre trénovanie hyperparametre supervised trénovania, čo sa však neukázalo ako lepšie. Vo výsledku teda tréning na cieľovú úlohu prebiehal s pôvodným nastavením hyperparametrov a to s batch size 8 a s hodnotou 0.01 pre rýchlosť učenia.

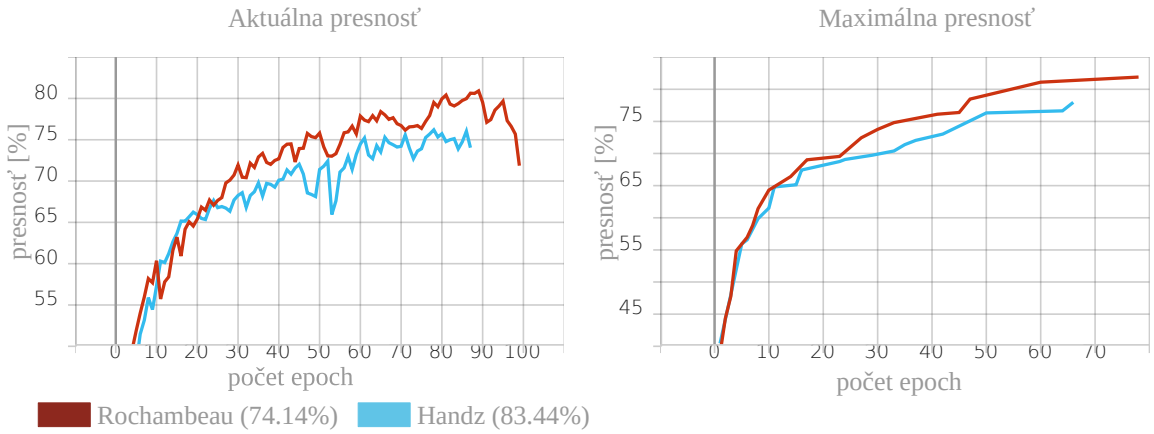


Obr. 5.7: Presnosť pri použití alternatívnych modelov: Najúspešnejším sa ukázalo použitie modelu ResNet18. Zlepšenie je možno vidieť aj pri modeli ResNet34, ktorý síce na validačnom datasete dosiahol nižšiu maximálnu úspešnosť. To bolo však spôsobené aplikovaním early stopping stratégie ako aj pretrénovania modelu ResNet50, čo dokazuje podstatne nižšia úspešnosť modelu ResNet50 pri finálnom teste na testovacom datasete.

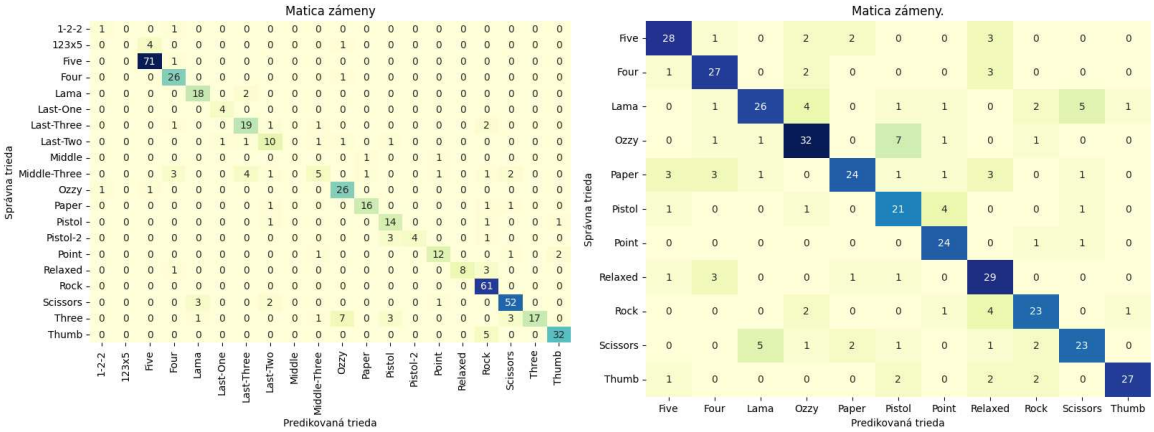


Obr. 5.8: Testovanie rôznych architektúr lineárnej vrstvy prebiehalo na datasete Handz. Pri testovaní som postupne znižovala počet lineárnych vrstiev avšak ako posledné som skúsila nie úplne intuitívnu architektúru. Tento experiment som vykonala z dôvodu, že som túto architektúru používala pre rozšírenie siete ResNet50, avšak ukázala sa ako najúspešnejšia aj keď na vstupe lineárnej časti je len 512 príznakov namiesto 2048.

Zvyšovanie úspešnosti prebiehalo zo začiatku veľmi rýchlo, keďže behom približne 10-15 epoch dosiahol model úspešnosť nad 65 %. Na konci trénovania mal model Handz úspešnosť 83,4 % a Rochambeau 74 %. Výsledky sú znázornené na obrázku 5.9. Pre ďalšie informácie o modeloch som taktiež vytvorila matice zámen, ktoré sú zobrazené na obrázku 5.10.



Obr. 5.9: Najúspešnejšie trénovanie: Po veľkom množstve experimentov sa ukázalo ako optimálne použitie modelu ResNet18, s batch size 128 a obrázkami o veľkosti 256 × 256. Toto trénovanie s optimálnou architektúrou lineárnej vrstvy dosiahlo úspešnosť nad 83 % na datasete Handz, čo výrazne prekonáva výsledok pri použití druhého datasetu.



Obr. 5.10: Matica zámen pre najúspešnejšie modely trénované na datasete Handz (vľavo) a Rochambeau (vpravo). Na ľavej strane možno vidieť, že model sa naučil rozpoznávať aj pozície, ktoré boli v trénovacom datasete reprezentované menej než 30 vzorkami. Najlepšie to možno vidieť pri pozícii Last-One. V datasete bolo pre túto pozíciu len 24 vzoriek, ale pri testovaní model správne zaradil všetky 4 obrázky reprezentujúce túto pozíciu. Na pravej strane možno vidieť model, s ktorým sa nepodarilo dosiahnuť ani 75% úspešnosť, pričom viacero často zamieňaných pozícií ani nie je veľmi podobných.

Kapitola 6

Záver

Cieľom práce bolo navrhnúť metódu self-supervised učenia pre rozpoznanie pozícií rúk v obraze a optimalizovať túto metódu smerom k maximálnej použiteľnosti.

Na začiatku som sa oboznámila s problematikou strojového učenia a rozpoznávania športových pozícií a následne, na základe získaných informácii zvolila vhodnú metódu pre implementáciu. Trénovanie prebiehalo na datasetoch zložených z pozícií rúk, ktoré som získala od svojho vedúceho práce. Podarilo sa mi kontrastívnou self-supervised metódou natrénovať modely, ktoré sa úspešnosťou priblížili k supervised výsledkom. Vyhodnotením výsledkov som následne demonštrovala použiteľnosť modelov. Najúspešnejšie boli dva modely s rovnakou architektúrou a hyperparametrami učenia (ResNet18 rozšírený o 4 lineárne vrstvy s batch size 128, rozlíšením obrázka 256×256). Model trénovaný pre cieľovú úlohu na datasete Handz dosiahol úspešnosť 82 % a model, ktorý pre trénovanie využil dataset Rochambeau mal 74 % úspešnosť. Zaujímavým zistením je, že zatiaľ čo učenie s učiteľom bolo výrazne úspešnejšie pri trénovaní na datasete Rochambeau s rovnomernou reprezentáciou pozícií, self-supervised metóda dosiahla značne lepší výsledok pri datasete Handz. To môže znamenať, že pre datasety tohoto typu by mohol byť self-supervised prístup ešte vhodnejší a pri správnej optimalizácii by mohol prekonať supervised metódu. O optimalizáciu modelu som sa snažila skúmaním rôznych nastavení hyperparametrov učenia. Výsledky som sa snažila zlepšiť aj pomocou optimalizácie datasetu, a to použitím techniky offline triplet mining pre výber vhodných trojíc po každej epoche. Experimentovala som aj s použitím rôznych architektúr siete v oboch etapách self-supervised učenia.

Vďaka práci som získala prehľad o postupe práce pri trénovaní modelov strojového učenia. Naučila som sa veľa aj o strojovom učení, od základov až po konkrétne metódy rozpoznávania pozícií pomocou strojového učenia a mala možnosť vyskúšať zaujímavú a relatívne novú metódu strojového učenia pre oblasť počítačového videnia.

V práci by som ďalej pokračovala upravením dátovej sady pre možnosť využitia techniky online triplet mining. Výhodou tejto techniky je väčšie množstvo trojíc v rámci datasetu a ich vhodnejší výber. Taktiež by som sa inšpirovala prácou BYOL a vyskúšala implementovať nekontrastívnu metódu, používajúcu pre trénovanie iba pozitívne páry. Ďalšou možnosťou pokračovania by bolo vyskúšať použiť ako model pre trénovanie vision transformer, čo som sa však rozhodla neskúsiť, pretože podľa existujúcich prác sú tieto modely vhodné len pri veľmi veľkom množstve dát a uprednostnila som použitie menšieho množstva kvalitných trojíc. Keďže konečným cieľom je rozpoznávanie športových pozícií, po optimalizácii spomenutými spôsobmi by som túto metódu aplikovala na dataset športových pozícií.

Literatúra

[1] *Jupyter Notebook* [online]. Project Jupyter, 2015 [cit. 2023-17-04]. Dostupné z: http://cameronoelsen.github.io/jupytersite/.

[2] *Deep Learning and Convolutional Neural Networks for Computer Vision, Applications, Code* [online]. Qualcomm Technologies, Inc., 2023 [cit. 2023-23-04]. Dostupné z: https://developer.qualcomm.com/software/qualcomm-neural-processing-sdk/learning-resources/cnn-architectures/deep-learning-convolutional-neural-networks-computer-vision.

[3] Albelwi, S. Survey on Self-Supervised Learning: Auxiliary Pretext Tasks and Contrastive Learning Methods in Imaging. *Entropy*. 2022, zv. 24, č. 4, s. 551. Dostupné z: https://www.mdpi.com/1099-4300/24/4/551.

[4] Cao, Z., Hidalgo, G., Simon, T., Wei, S.-E. a Sheikh, Y. OpenPose: Realtime Multi-Person 2D Pose Estimation Using Part Affinity Fields. *IEEE Trans. Pattern Anal. Mach. Intell.* 2021, zv. 43, č. 1, s. 172–186. Dostupné z: https://doi.org/10.1109/TPAMI.2019.2929257.

[5] Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J. et al. Emerging Properties in Self-Supervised Vision Transformers. In: *2021 IEEE/CVF International Conference on Computer Vision, ICCV 2021, Montreal, QC, Canada, October 10-17, 2021*. IEEE, 2021, s. 9630–9640. Dostupné z: https://doi.org/10.1109/ICCV48922.2021.00951.

[6] Chen, T., Kornblith, S., Norouzi, M. a Hinton, G. E. A Simple Framework for Contrastive Learning of Visual Representations. In: *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*. PMLR, 2020, sv. 119, s. 1597–1607. Proceedings of Machine Learning Research. Dostupné z: http://proceedings.mlr.press/v119/chen20j.html.

[7] Goodfellow, I., Bengio, Y. a Courville, A. *Deep Learning*. MIT Press, 2016. Dostupné z: http://www.deeplearningbook.org.

[8] grill, J., Strub, F., Altché, F., Tallec, C., Richemond, P. H. et al. Bootstrap Your Own Latent - A New Approach to Self-Supervised Learning. In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Dostupné z: https://proceedings.neurips.cc/paper/2020/hash/f3ada80d5c4ee70142b17b8192b2958e-Abstract.html.

[9] He, K., Zhang, X., Ren, S. a Sun, J. Deep Residual Learning for Image Recognition. In: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. IEEE Computer Society, 2016, s. 770–778. Dostupné z: http://doi.ieeecomputersociety.org/10.1109/CVPR.2016.90.

[10] Hermans, A., Beyer, L. a Leibe, B. In Defense of the Triplet Loss for Person Re-Identification. *ArXiv*. 2017. Dostupné z: http://arxiv.org/abs/1703.07737.

[11] Herout, A. a Tesařová, A. Rochambeau – Learning Pose Recognition on Unlabeled Concurrent Videos. *V procese revízie*. 2022.

[12] Kingma, D. P. a Ba, J. Adam: A Method for Stochastic Optimization. *ArXiv*. 2017. ICLR 2015. Dostupné z: http://arxiv.org/abs/1412.6980.

[13] Kvasnička, V., Beňušková, L., Pospíchal, J., Farkaš, I., Tiňo, P. et al. *Úvod do teórie neurónových sietí*. Iris, 1997. ISBN 978-8-088-77830-1. Dostupné z: https://books.google.sk/books?id=0lI6AAAACAAJ.

[14] Moindrot, O. *Triplet Loss and Online Triplet Mining in TensorFlow* [online]. GitHub, Inc, marec 2018 [cit. 2023-17-04]. Dostupné z: https://omoindrot.github.io/triplet-loss.

[15] O'Shea, K. a Nash, R. An Introduction to Convolutional Neural Networks. *ArXiv*. 2015. Dostupné z: http://arxiv.org/abs/1511.08458.

[16] Sermanet, P., Lynch, C., Chebotar, Y., Hsu, J., Jang, E. et al. Time-Contrastive Networks: Self-Supervised Learning from Video. In: *2018 IEEE International Conference on Robotics and Automation, ICRA 2018, Brisbane, Australia, May 21-25, 2018*. IEEE, 2018, s. 1134–1141. Dostupné z: https://doi.org/10.1109/ICRA.2018.8462891.

[17] Shah, D. *Cross Entropy Loss: Intro, Applications, Code* [online]. V7Lab, 2023 [cit. 2023-17-04]. Dostupné z: https://www.v7labs.com/blog/cross-entropy-loss-guide.

[18] Szeliski, R. *Computer Vision*. Springer Cham, 2022. ISBN 978-3-030-34372-9. Dostupné z: https://szeliski.org/RichardSzeliski.htm.

[19] Toshev, A. a Szegedy, C. DeepPose: Human Pose Estimation via Deep Neural Networks. In: *2014 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2014, Columbus, OH, USA, June 23-28, 2014*. IEEE, 2014, s. 1653–1660. Dostupné z: http://dx.doi.org/10.1109/CVPR.2014.214.

[20] Vassileios Balntas, D. P. a Mikolajczyk, K. Learning local feature descriptors with triplets and shallow convolutional neural networks. In: *Proceedings of the British Machine Vision Conference 2016, BMVC 2016, York, UK, September 19-22, 2016*. BMVA Press, 2016. Dostupné z: http://www.bmva.org/bmvc/2016/papers/paper119/index.html.

[21] Zheng, C., Wu, W., Yang, T., Zhu, S., Chen, C. et al. Deep Learning-Based Human Pose Estimation: A Survey. *ArXiv*. 2022. Dostupné z: https://arxiv.org/abs/2012.13392.