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

IDENTIFIKACE ČLOVĚKA PODLE FOTOGRAFIE DLANĚ / HŘBETU RUKY

THESIS TITLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE **MIROSLAV ŠTANGA**
AUTHOR

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

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ

Zadání bakalářské práce

154400

Ústav: Ústav počítačové grafiky a multimédií (UPGM)
Student: **Štanga Miroslav**
Program: Informační technologie
Název: **Identifikace člověka podle fotografie dlaně / hřbetu ruky**
Kategorie: Zpracování obrazu
Akademický rok: 2023/24

Zadání:

1. Seznamte se s problematikou počítačového vidění pomocí hlubokého strojového učení; zaměřte se na metody self-supervised learning.
2. Pořizujte vlastní datovou sadu lidských rukou, vhodnou pro experimentování s algoritmy strojového učení v režimu self-supervised.
3. Experimentujte s vhodnými algoritmy učení, průběžně srovnávejte přístupy založené na self-supervised a supervised učení.
4. Diskutujte dosažené výsledky nad pořízenou datovou sadou a formulujte závěry a doporučení.
5. Zhodnoťte dosažené výsledky a navrhněte možnosti pokračování projektu; vytvořte plakátek a krátké video pro prezentování projektu.

Literatura:

- Richard Szeliski: Computer Vision: Algorithms and Applications, Springer, 2nd, 2022
- Goodfellow, Bengio, Courville: Deep Learning, MIT Press, 2016
- Andriy Burkov: The hundred-page machine learning book, Large Print Book, 2019
- Bharath Ramsundar, Reza Bosagh Zadeh: TensorFlow for Deep Learning: From Linear Regression to Reinforcement Learning, O'Reily Media, 2018
- Vincent Dumoulin, Francesco Visin: A guide to convolution arithmetic for deep learning, https://arxiv.org/abs/1603.07285
- Gary Bradski, Adrian Kaehler: Learning OpenCV; Computer Vision with the OpenCV Library, O'Reilly Media, 2008

Při obhajobě semestrální části projektu je požadováno:
bod 1, značné rozpracování bodů 2 a 3

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: 1.11.2023
Termín pro odevzdání: 9.5.2024
Datum schválení: 9.11.2023

Abstrakt

Táto práca je zameraná na kontrastívny prístup self-supervised učenia k tvorbe modelu hlbokého strojového učenia, na rozpoznanie osôb podľa fotografie ruky. Práca popisuje základy strojového učenia, použité nástroje a dataset. Metóda bola vypracovaná s použitím knižnice PyTorch. Návrh modelu čerpá inšpiráciu z architektúry metódy SimCLR a jej využitia princípov kontrastívneho učenia reprezentácií. Navrhnutý prístup využíva na optimalizáciu stratovú funkciu triplet loss. Následne je opísaný proces optimalizácie a je porovnaný vplyv jednotlivých hyperparametrov na presnosť modelu. Výsledný model bol trénovaný na 1696 fotografiách rúk a dosahuje presnosť 98% na validačnej sade. Presnosť dosiahnutá použitím self-supervised metódy je vyššia ako presnosť dosiahnutá použitím supervised metódy.

Abstract

This work focuses on using contrastive self-supervised learning method for creating model of deep learning intended for person recognition based on hand photographs. The paper outlines fundamentals of machine learning, utilized tools and dataset. The method was developed using PyTorch library. The proposed model draws inspiration from the SimCLR architecture and its use of contrastive representation learning. The proposed approach utilizes the triplet loss function for optimization. Then the optimization process is described and impact of individual hyperparameters on the model´s accuracy is compared. The resulting model was trained on 1696 hand photos and achieves 98% accuracy on validation set. The accuracy achieved using self-supervised methods is higher than the accuracy achieved using supervised methods.

Kľúčové slová

strojové učenie, počítačové videnie, neorónové siete, triplet loss, self-supervised učenie

Keywords

machine learning, computer vision, neural networks, triplet loss, self-supervised learning

Citácia

ŠTANGA, Miroslav. *Identifikace člověka podle fotografie dlaně / hřbetu ruky.* Brno, 2024. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce prof. Ing. Adam Herout, Ph.D.

Identifikace člověka podle fotografie dlaně / hřbetu ruky

Prehlásenie

Prehlasujem, že som túto bakalársku prácu vypracoval samostatne pod vedením pána prof. Ing. Adama Herouta, Ph.D. Uviedol som všetky literárne pramene, publikácie a ďalšie zdroje, z ktorých som čerpal.

.....
Miroslav Štanga
8. mája 2024

Poďakovanie

Chcel by som sa poďakovať prof. Ing. Adamovi Heroutovi, Ph.D. za odbornú pomoc, trpezlivosť a nasmerovanie pri tvorení mojej bakalárskej práce.

Obsah

Kapitola 1

Úvod

V poslednej dobe je veľmi populárna téma umelej inteligencie. Neustály pokrok v oblasti využitia techník umelej inteligencie nám umožnil zjednodušenie riešenia problémov, ktoré boli kedysi zložité, ale aj jednoduchých každodenných činností. Vďaka rýchlemu vývoju počítačového hardwaru sa podarilo efektívne spojazdniť neurónové siete a hlboké strojové učenie. Teoretické základy hlbokého strojového učenia objavili už v minulom storočí, ale ich efektívne využívanie na riešenie reálnych problémov sa začalo iba nedávno. Na vývoj aplikácií využívajúcich strojové učenie sa momentálne používajú hlavne dve populárne knižnice. Prvou z nich je knižnica Tensorflow, ktorá bola vyvinutá v roku 2015 spoločnosťou Google a druhá knižnica Pytorch, ktorá bola vyvinutá v roku 2016 spoločnosťou Meta.

Pri trénovaní modelov strojového učenia býva jedným z hlavných problémov zaobstaranie kvalitného datasetu. Pre efektívne trénovanie modelov strojového učenia je potrebné relatívne veľké množstvo dát, ktoré sa nám nie vždy podarí nazbierať. V takomto prípade je možné dataset umelo zväčšiť pomocou výberu vhodných dátových augmentácií. Samostatné zozbieranie dát však väčšinou nestačí a dáta treba pracne anotovať, čo zaberie veľa času. Existuje viac metód, vďaka ktorým je možné vyhnúť sa anotácii dát. Táto práca sa zameriava na self-supervised metódu hlbokého strojového učenia, ktorá je jednou z metód nevyžadujúcich anotované dáta. Mnohé techniky self-supervised učenia si sami generujú pseudo anotácie, vďaka ktorým sa následne učia dobré reprezentácie a charakteristiky vstupných dát.

Cieľom tejto práce je navrhnúť a natrénovať model hlbokého strojového učenia, ktorý bude s využitím techník self-supervised učenia vedieť na základe štvoríc fotografií rúk tieto ruky porovnať a bez informácie komu ruky patria zistiť, či sa jedná o tú istú osobu. Natrénovaný model je následne vyhodnotený a jeho výsledky sú porovnané s modelom natrénovaným v supervised režime. Výsledný model je trénovaný na datasete štvoríc fotografií rúk, ktoré sú fotené pri rôznych svetelných podmienkach a každá fotografia z jednej štvorice je fotená pod iným uhlom.

V tejto práci sú okrem základných techník a nástrojov potrebných na trénovanie modelov hlbokého strojového učenia popísané aj zložitejšie metódy ako napríklad transfer learning, self-supervised learning a niektoré pokrokové práce využívajúce tieto techniky. Ďalej sú popísané niektoré existujúce riešenia a podobná práca v tejto oblasti. Následuje podrobný popis použitého datasetu vrátane jeho zbierania, návrh riešenia a jeho implementácia. Následne pokračuje zhodnotenie výsledkov dosiahnutých počas trénovania modelov a ich porovnanie. Na záver práce sú zhrnnuté získané poznatky a postrehy vrátane návrhu možností nadviazania na prácu a jej praktických využití.

Kapitola 2

Nástroje pre trénovanie modelu hlbokého strojového učenia

V tejto kapitole sú popísané a vysvetlené dôležité techniky a nástroje, ktorých znalosť je potrebná na trénovanie modelov hlbokého učenia. Okrem základov sa táto kapitola zameriava aj na self-supervised učenie a podrobnejšie rozoberá niektoré pokrokové metódy založené na self-supervised učení.

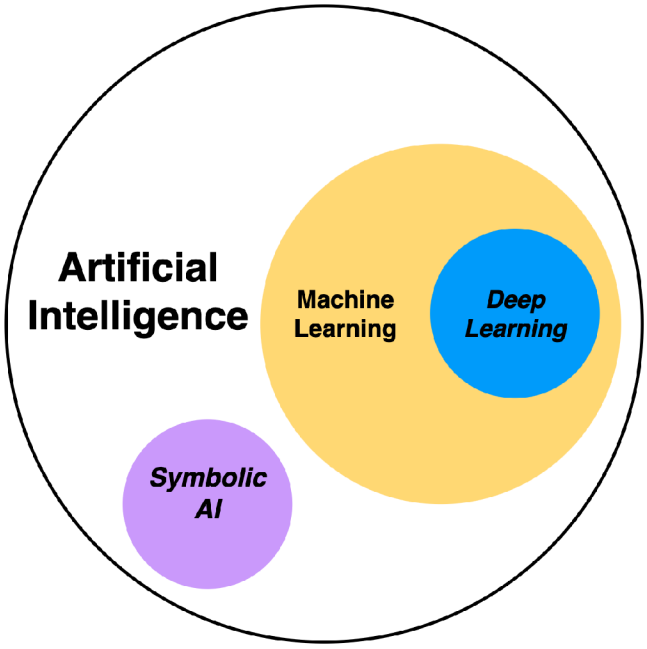
2.1 Strojové učenie

Strojové učenie je oblasť umelej inteligencie využívajúca algoritmov a techník, ktoré umožňujú počítačom učiť sa zo skúseností a dát bez nutnosti explicitného programovania. Cieľom je tvorba modelov, ktoré dokážu automaticky získavať poznatky z dát a na ich základe robiť predikcie. Strojové učenie možno rozdeliť do viacero kategórií [7]:

- Učenie s učiteľom (supervised learning) – modely sú trénované na základe párov vstupných dát a príšlušných výstupov. Bližšie popísané v podkapitole 2.8.
- Učenie bez učiteľa (unsupervised learning) – modely sú trénované na základe neoznačených dát. Cieľom je identifikovať vzory a charakteristiky vstupných dát bez znalosti výstupov.
- Posilované učenie (reinforcement learning) – agent sa učí dosiahnuť určené ciele v dynamickom prostredí. Využíva odmeny a tresty. Často používané pre učenie hrania hier.

2.2 Hlboké strojové učenie

Hlboké strojové učenie je oblasť strojového učenia zameraná na trénovanie neurónových sietí s viacerými vrstvami za účelom extrakcie zložitejších charakteristík a reprezentácií z veľkého množstva vstupných dát. Je využívané na prácu s obrázkami, videom, zvukovými súborami a neštruktúrovaným textom [13].



Obr. 2.1: Obrázok znázorňujúci členenie rôznych prístupov umelej inteligencie. Táto práca je zameraná na deep learning. Obrázok prevzatý z [12]

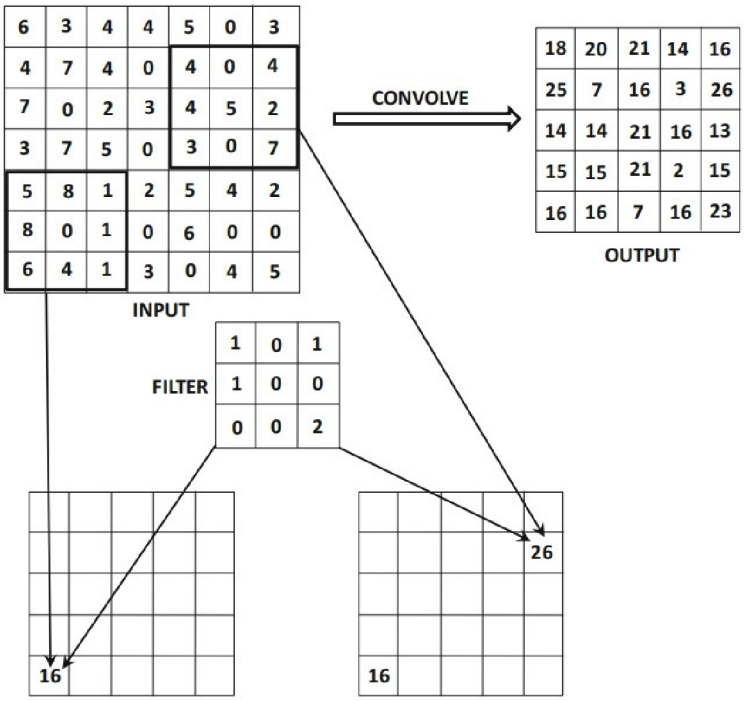
2.3 Neurónové siete

Neurónové siete sú základom moderných techník trénovania modelov hlbokého strojového učenia. Hlboké neurónové siete ako také sú ale výpočetne náročné. Práve z tohoto dôvodu sa často používajú konvolučné neurónové siete, ktoré využívajú operáciu konvolúcie na detekovanie vzorov v menších oblastiach dát a sú menej výpočetne náročné ako plne prepojené vrstvy. Príklad konvolúcie je zobrazený na obrázku 2.2. Neurónová sieť je zložená z viacero úrovní neurónov, ktoré sú medzi sebou poprepájané. Príklady niektorých vrstiev sú bližšie popísané v kapitole 2.4. Typická neurónová sieť začína vstupnou vrstvou, ktorá ako vstup berie dáta v podobe tensoru. Príkladom vstupných dát môže byť napríklad farebný obrázok o rozmeroch 224×224, ktorý bude uložený ako tensor o veľkosti $224 \times 224 \times 3$. Takýto tensor je zložený z 150 528 čísel, ktoré reprezentujú vstupný obrázok. Po prechode cez vstupnú vrstvu nasledujú skryté vrstvy. Všetky vrstvy sú medzi sebou poprepájané a každý jeden neurón má vlastné váhy pre všetky prepojenia s ostatnými neurónmi a taktiež bias. Pri trénovaní neurónových sietí sú tieto váhy a bias upravované a ich úpravou sa trénovaný model učí detekovať rôzne vzory a charakteristiky vstupných dát. Na konci siete po skrytých vrstvách sa nachádza vrstva klasifikačná. Väčšinou sa jedná o plne prepojenú vrstvu a je špecifická pre konkrétny problém, pre ktorý sa snažíme sieť trénovať [2].

2.3.1 Trénovanie neurónových sietí

Pre trénovanie neurónových sietí treba najskôr pripraviť vstupné dáta do vhodného formátu. Pripravené dáta sú najskôr transformované na tensor. Ak je dostupná grafická karta, tensor je premiestnený na ňu. Trénovanie neurónových sietí pozostáva z niekoľkých krokov:

- Forward pass – vstupný tensor prechádza postupne cez všetky vrstvy siete.



Obr. 2.2: Príklad konvolúcie $7 \times 7 \times 1$ vstupu a $3 \times 3 \times 1$ filtru s krokom veľkosti 1. Obrázok prevzatý z knihy [2].

- Výpočet chyby – vypočíta sa chyba, ktorá je porovnávaná s očakávaným výstupom pomocou vhodnej stratovej funkcie. Viac informácií o stratových funkciách sa nachádza v podkapitole 2.5.
- Backpropagation – vypočíta gradient stratovej funkcie smerom od výstupnej vrstvy ku vstupnej podľa váh a biasu jednotlivých vrstiev siete [2].
- Stochastic gradient descent – je jedným s najpoužívanejších optimalizačných algoritmov pre trénovanie neurónových sietí. Jeho úlohou je minimalizácia chyby predikcie. Po spočítaní gradientov aktualizuje váhy a biasy jednotlivých vrstiev siete [13].

2.3.2 Hyperparametre

Hyperparametre neurónových sietí sú nastavenia a parametre, ktoré ovplyvňujú výkon siete [13].

- Veľkosť skrytých vrstiev – určuje koľko neurónov alebo skrytých jednotiek sa nachádza v skrytej vrstve.
- Počet skrytých vrstiev
- Počet epoch – definuje koľkokrát prejdú vstupné dáta cez neurónovú sieť počas trénovania.
- Veľkosť dávky (batch size) – udáva koľko vstupných dát sa spracováva naraz pri jednom trénovaciom alebo validačnom kroku. Väčšie dávky môžu pomôcť stabilizovať trénovanie ale taktiež využívajú viac zdrojov pri trénovaní [10].

- Rýchlosť učenia – určuje veľkosť kroku, ktorým sa aktualizujú váhy siete po každej iterácii trénovania. Dobrou praktikou je mať na začiatku trénovania rýchlosť učenia relatívne veľkú a postupne ju znižovať. Pre postupné znižovanie rýchlosti učenia sa využíva technika plánovania rýchlosti učenia.
  - Krokový plánovač – znižuje rýchlosť učenia o určitý faktor po špecifikovanom počte krokov alebo epoch.
  - Exponenciálny plánovač – znižuje rýchlosť učenia exponenciálne po špecifikovanom počte krokov alebo epoch.

Experimentovanie s rôznymi hodnotami hyperparametrov je dôležitá súčasť trénovania neurónových sietí. Optimálne hodnoty hyperparametrov môžu byť pre rozdielne úlohy odlišné, preto je dôležité s nimi experimenovať a nájsť ich vhodné hodnoty pre konkrétnu úlohu. Porozumenie vplyvu hyperparametrov na trénovanie modelu je kľúčové pre efektívnejšie ladenie a optimalizáciu.

2.4 Pytorch

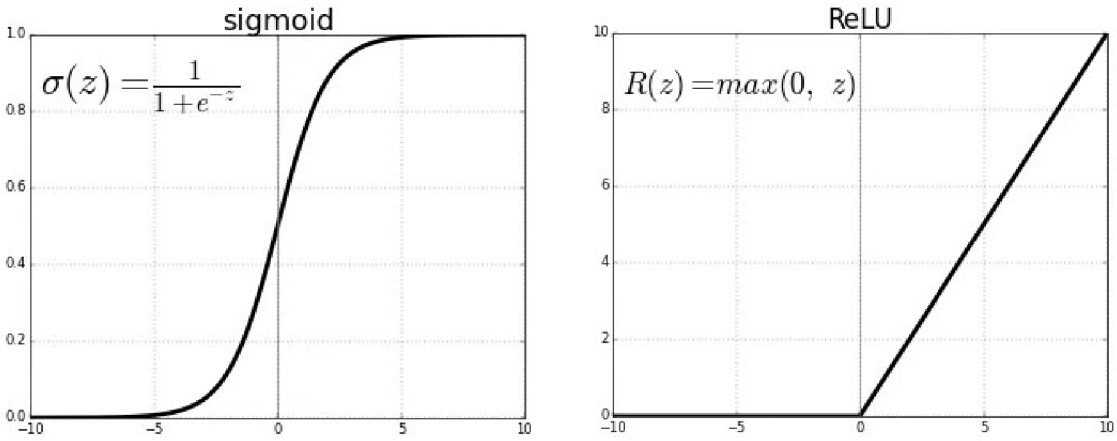
Pytorch je open–source framework pre hlboké strojové učenie. Vyznačuje sa hlavne svojou flexibilitou a jednoduchosťou používania. Je kompatibilný so skriptovacím jazykom Python a je známy jednoduchým využitím grafických kariet pre trénovanie modelov strojového učenia, čo značne zrýchli ich trénovanie. Pytorch taktiež umožňuje jednoduchú konverziu medzi numpy poľami a tensormi [25, 23]. Veľmi dôležitou dátovou štruktúrou, ktorá sa používa na reprezentáciu dát a operácie s nimi je tensor. Jedná sa o multi dimenzionálnu maticu obsahujúcu elementy jedného datového typu. Tensory sú používané ako vstupy aj výstupy jednotlivých vrstiev neurónových sietí. V rámci neurónových sietí sa využívajú rôzne typy vrstiev, ako napríklad:

- Lineárna vrstva – tiež známa aj ako plne prepojená vrstva. Každý neurón tejto vrstvy je prepojený so všetkými neurónmi predchádzajúcej vrstvy.
- Konvolučná vrstva – táto vrstva aplikuje filtre na vstupné dáta pomocou konvolúcie. Po konvolučnej vrstve zvykne nasledovať aktivačná funkcia.
- Zhlukovacia vrstva – redukuje počet parametrov a zhlukuje informácie z určitých oblastí vstupu.
  - Max–pooling – vyberie maximálnu hodnotu z malého okna vstupnej mapy
  - Average–pooling – vypočíta priemer hodnôt z malého okna vstupnej mapy
- Dropout vrstva – náhodne vypúšťa niektoré neuróny pri trénovaní, čo pomáha zamedziť pretrénovaniu
- Batch norm vrstva – normalizuje výstupy každej vrstvy podľa štatistík z celej dávky.

Každý neurón má svoje váhy a biasy, ktoré ovplyvňujú, či bude neurón aktivovaný alebo nie. Ďalej medzi vrstvami môžu byť aktivačné funkcie [29] ako napríklad:

- ReLu – ak je vstup kladný aj výstup bude kladný, ak je vstup záporný výstup bude 0.
- Sigmoid – ohraničuje výstup na interval $(0, 1)$

Aktivačné funkcie sigmoid a ReLU sú zobrazené na obrázku 2.3.



Obr. 2.3: Aktivačné funkcie sigmoid a ReLU. Prevzaté zo stránky [28].

2.4.1 Trénovanie modelu v PyTorchi

Sekvenčný model je jednoduchý typ modelu neurónovej siete tvorený postupnosťou vrstiev. Výstupy jednej vrstvy sú použité ako vstupy následujúcej vrstvy. Tu je príklad jednoduchého sekvenčného modelu tvoreného z dvoch lineárnych vrstiev, dvoch aktivačných funkcií ReLU a jednej konvolučnej vrstvy:

```
model = nn.Sequential(
    nn.Linear(in_features=224, out_features=112),
    nn.ReLU(),
    nn.Conv2d(in_channels=1, out_channels=32, kernel_size=3, stride=1,
    padding=1),
    nn.ReLU(),
    nn.Linear(in_features=112, out_features=10)
)
```

Trénovanie modelu pozostáva z dvoch fáz: trénovania a validácie. Model je najskôr nastavený do trénovacieho módu pomocou príkazu **model.train()**. Vykoná sa prechod vstupných dát cez model (forward pass) **predikcia = model(vstupné dáta)**. Vypočíta sa hodnota straty medzi predikciou a skutočnou hodnotou pomocou loss funkcie **loss = loss_fn(predikcia, skutočná hodnota)**. Následuje vynulovanie gradientov, aby sa zabránilo v akumulácii gradientov z predchádzajúcich iterácií pomocou funkcie **optimizer.zero_grad()**. V ďalšom kroku sa vykoná backpropagation, ktorá spustí spätné šírenie chyby a vypočíta nové gradienty pre jednotlivé parametre modelu **loss.backward()**. Posledná časť trénovacieho kroku pomocou optimalizačného algoritmu aktualizuje váhy modelu na základe vypočítaných gradientov **optimizer.step()**.

Validačný krok začína nastavením modelu do evaluačného módu **model.eval()**, ktorý ovplyvňuje funkciu niektorých vrstiev. Ďalej je použité **with torch.inference_mode()**, čo zabráni výpočtu gradientov. Následuje forward pass cez model a kalkukácia stratovej funkcie.

Model sa trénuje v cykloch(epoch). Spraví sa jeden trénovací krok, ktorý spracuje celú trénovaciu sadu a jeden validačný krok, ktorý spracuje celú validačnú sadu. Po vykonaní krokov sa vypočítajú metriky loss a presnosť a model pokračuje v trénovaciom cykle. Natrénovaný model je následovne možné použiť na riešenie očakávanej úlohy.

2.5 Stratová funkcia

Stratové funkcie sú v strojovom učení veľmi dôležité. Ich úlohou je meranie rozdielov medzi predikovanou hodnotou trénovaného modelu a skutočnou hodnotou v trénovacích dátach. Trénovaný model sa snaží stratovú funkciu minimalizovať, vďaka čomu sa model naučí lepšie generalizovať vstupné dáta a na ich základe predikovať výstupy [29]. Stratové funkcie sú podľa použitia delené na niekoľko typov. Tu sú príklady niektorých:

- Regresné – ich cieľom je predikovať kontinuálne hodnoty [18]. Príkladom regresných funkcií sú Mean squared error a Huber loss.
- Klasifikačné – ich cieľom je klasifikovať vstupné dáta do rôznych tried alebo kategórií. Klasifikačné stratové funkcie merajú rozdiel medzi predikovanými a skutočnými triedami a snažia sa tento rozdiel minimalizovať. Príkladom klasifikačných funkcií sú Binary cross-entropy loss a Hinge loss.
- Hodnotiace – predikujú relatívne vzdialenosti medzi hodnotami. Príkladom hodnotiacich funkcií sú Contrastive loss a triplet loss.

Na minimalizáciu stratových funkcií sa využíva optimalizátor. Príkladom veľmi populárneho optimalizačného algoritmu je napríklad optimalizátor Adam [19]. Optimalizátor Adam dokáže adaptívne upravovať rýchlosť učenia pre každý parameter zvlášť podľa jeho gradientov.

2.6 Triplet loss

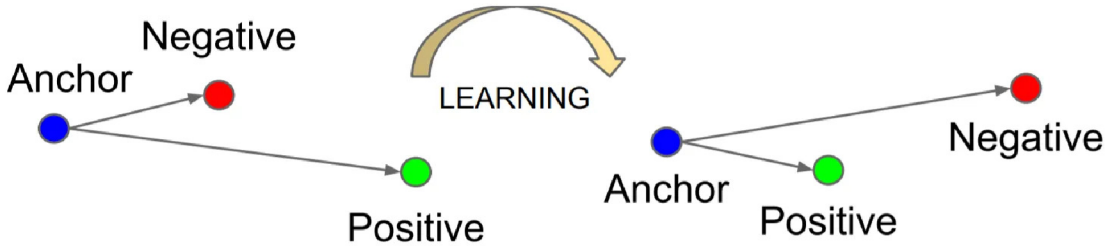
Triplet loss je hodnotiaca stratová funkcia, ktorá sa často používa pri kontrastívnom učení. Pre jej fungovanie potrebujeme vytvoriť trojice fotografií, ktoré sú označované ako bázová, pozitívna a negatívna. Bázová a pozitívna fotografia sa podobajú, napríklad fotografia toho istého človeka alebo v tejto práci fotografie z rovnakej štvorice rúk. Štvorice sú ďalej popísané v kapitole 4.1. Negatívna fotografia by mala byť od bázovej rozlišná, teda je vybraná fotografia iného človeka/štvorice ruky. Cieľom triplet loss je minimalizovať vzdialenosť medzi bázovou a pozitívnou fotografiou a maximalizovať vzdialenosť medzi bázovou a negatívnou fotografiou čo môžeme dosiahnuť minimalizovaním nasledujúceho výrazu:

$$L(A, P, N) = \max\left(||f(A) - f(P)||_2^2 - ||f(A) - f(N)||_2^2 + \alpha, 0\right)$$

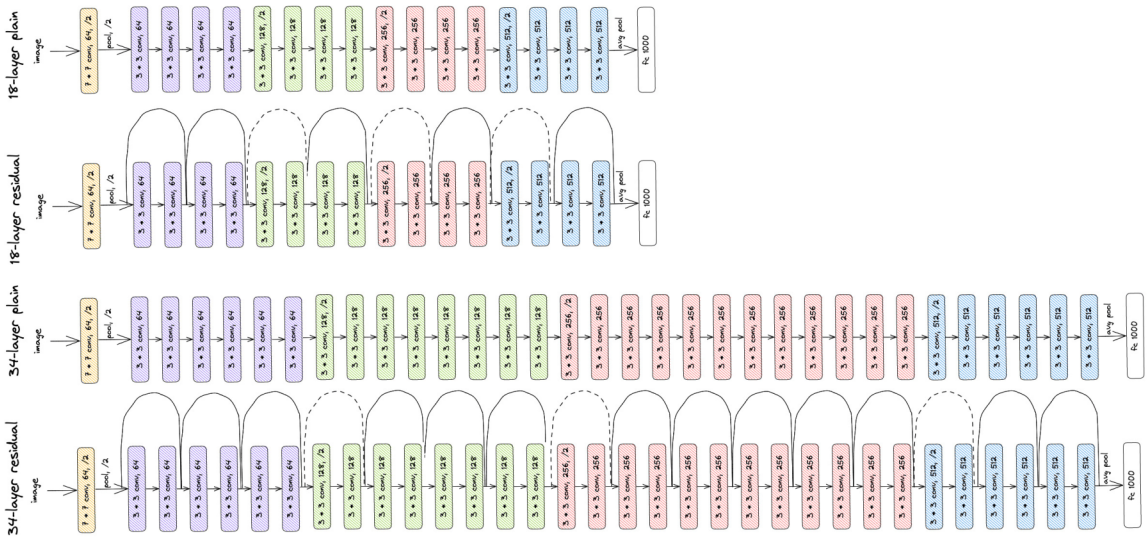
Kde A je bázová fotografia, P je pozitívna fotografia, N je negatívna fotografia a α je posunutie, ktoré zabezpečuje aby funkcia nevrátila triviálny výstup [17]. Na obrázku 2.4 vidíme ako triplet loss funkcia po natrénovaní modelu minimalizuje vzdialenosť bázového a pozitívneho obrázku a maximalizuje vzdialenosť bázového a negatívneho obrázku.

2.7 ResNet

ResNet je typ hlbokých neurónových sietí využívajúci reziduálnych blokov. Reziduálne bloky umožňujú používanie neurónových sietí s väčším množstvom vrstiev bez toho, aby dochádzalo k problému miznúcich alebo explódujúcich gradientov. Reziduálne bloky obsahujú preskočenia spojení, vďaka ktorým je možné preskočiť jednu alebo viacero vrstiev siete. Preskočenia sieti umožňujú pridanie vstupu bloku k jeho výstupu [16]. Znázornenie architektúry sietí ResNet18 a ResNet34 aj s vyznačením preskočení je na obrázku 2.5.



Obr. 2.4: Obrázok znázorňujúci vzdialenosti embeddingov bázoveho, pozitívneho a negatívneho obrázku pred trénovaním siete a po trénovaní siete. Prevzaté zo stránky [26].



Obr. 2.5: Obrázok znázorňuje architektúru sietí ResNet18 a ResNet34 využívajúcich technológie reziduálnych blokov a rovnaké architektúry bez použitia reziduálnych blokov. Prevzaté z [5].

Sieť ResNet vyhrala súťaž ILSVRC v klasifikácii obrázkov v roku 2015 [16], kde dosiahla výrazné zlepšenie v porovnaní s predchádzajúcimi modelmi. ResNet je dodnes veľmi populárny a naväzuje na neho množstvo prác. ResNet je často používaný pri technike transfer learning, ktorá spočíva v tom, že sa zoberie predtrénovaný model, z ktorého sa odreže posledná vrstva a vloží sa namiesto nej nová klasifikačná vrstva.

Výhoda tohoto prístupu je ušetrenie zdrojov a času, kedže na trénovanie modelov „from scratch“ sú potrebné veľké datasety a ich trénovanie trvá dlho. Z tohoto dôvodu sa oplatí použiť už natrénovaný model, ktorý stačí dotrénovať na menšom množstve dát a netreba ho nechať trénovať na dobu mnohých epoch.

2.8 Supervised učenie

Supervised učenie je jedným z najpoužívanejších prístupov strojového učenia. Tento prístup využíva na učenie modelov označené dáta. Cieľom tohoto prístupu je naučiť model reprezentovať vzťahy a vzory medzi vstupnými dátami a ich označeniami. Supervised učenie sa využíva na mnohé typy úloh ako napríklad regresné alebo klasifikačné. Nevýhodou super-

vised učenia je ale fakt, že úspešnosť trénovania závisí od množstva a kvality označených dát [13].

2.9 Self-supervised učenie

Self-supervised učenie je prístup v oblasti strojového učenia, ktorého cieľom je naučiť modely dobré reprezentácie vstupných dát a ich charakteristiky bez využitia manuálne označených dát. Self-supervised učenie je veľmi výhodné z dôvodu, že manuálne označovanie dát je pre ľudí pracné a zaberie veľa času [11]. Self-supervised učenie využíva náhradnú úlohu (pretext task) na ktorej naučí model vzory a charakteristiky vstupných dát a následovne je model dotrénovaný na cieľovej úlohe (downstream task) pre ktorú bol model pôvodne určený.

2.9.1 Náhradná úloha

Náhradné úlohy majú dve hlavné charakteristiky. Prvou charakteristikou je využívanie metódy hlbokého učenia pre naučenie vlastností užitočných pre riešenie náhradnej úlohy. Druhou charakteristikou je generovanie označených dát z dostupných neoznačených dát. Tento proces sa označuje ako self-supervision [15]. Na obrázku 2.6 je zobrazených niekoľko často používaných náhraných úloh. Po dotrénovaní modelu na veľkom množstve neoznačených dát je využitá technika transfer learning a predtrénovaný model je ďalej dotrénovaný na cieľovej úlohe.

2.9.2 Cieľová úloha

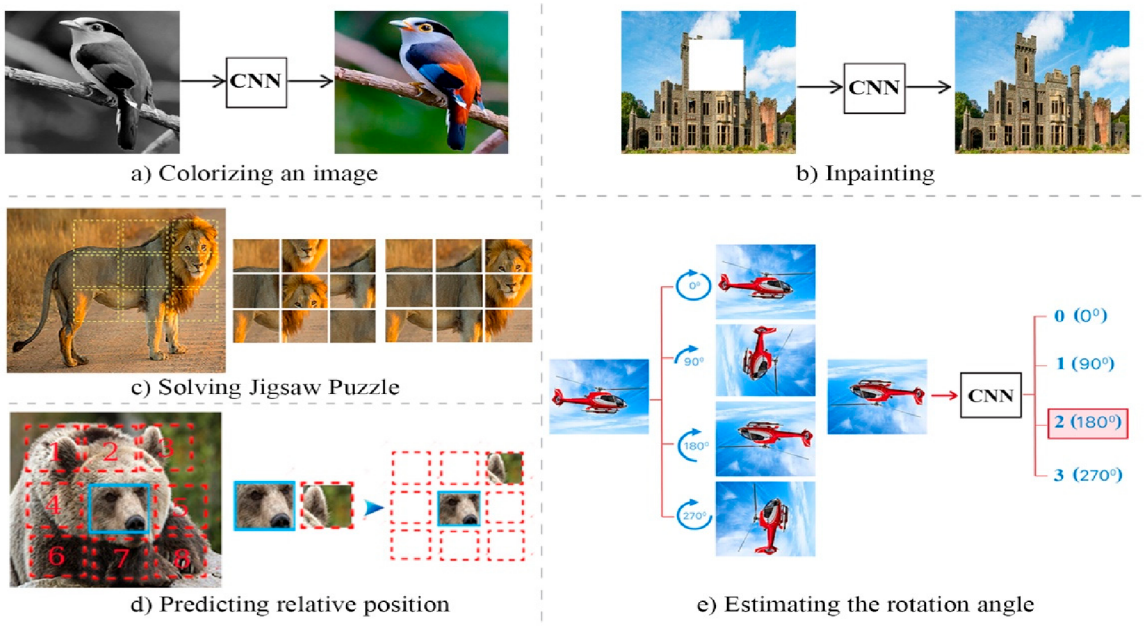
Po naučení charakteristík a reprezentácií vstupných dát na náhradnej úlohe je model dotrénovaný na cieľovej úlohe. Dotrénované sú buď všetky vrstvy modelu, alebo sa zamrazia váhy a biasy jednotlivých vrstiev a posledná vrstva sa nahradí novou lineárnou vrstvou, ktorá je dotrénovaná na malom množstve označených dát.

2.10 Metódy založené na self-supervised učení

V tejto podkapitole sú bližšie popísané niektoré moderné prístupy k self-supervised learningu. Tieto prístupy sú pokrokové v oblasti strojového učenia a umožňujú efektívne trénovanie modelov na neoznačených dátach.

2.10.1 SimCLR

SimCLR je framework, ktorý využíva kontrastívne self-supervised učenie na získanie vizuálnych reprezentácií obrázkov. Architektúra SimCLR je zobrazená na obrázku 2.7. SimCLR je citlivý na veľkosť dávok (batch size), čo si vyžaduje významné výpočtové zdroje. Avšak, počas dlhého trénovania sa táto citlivosť postupne znižuje. Dátové augmentácie sú kľúčovým prvkom pri trénovaní SimCLR a správne zvolené kombinácie môžu výrazne ovplyvniť robustnosť a presnosť modelu. Autori metódy SimCLR na základe experimentov zistili, že najlepšia kombinácia dátových augmentácií za účelom zvýšenia presnosti modelu je farebná deformácia a orezanie. Príklady dátových augmentácií, ktoré SimCLR používa sú na obrázku 2.8. Vstupom je nejaká fotografia x. Z tejto fotografie sú následovne vyrobené pomocou dvoch rôznych augmentácií upravená fotografia $\tilde{x}_i$ a fotografia $\tilde{x}_j$, ktorá je upravená iným spôsobom. Po vytvorení augmentácií obidve fotografie prejdú cez backbone $f(\cdot)$.

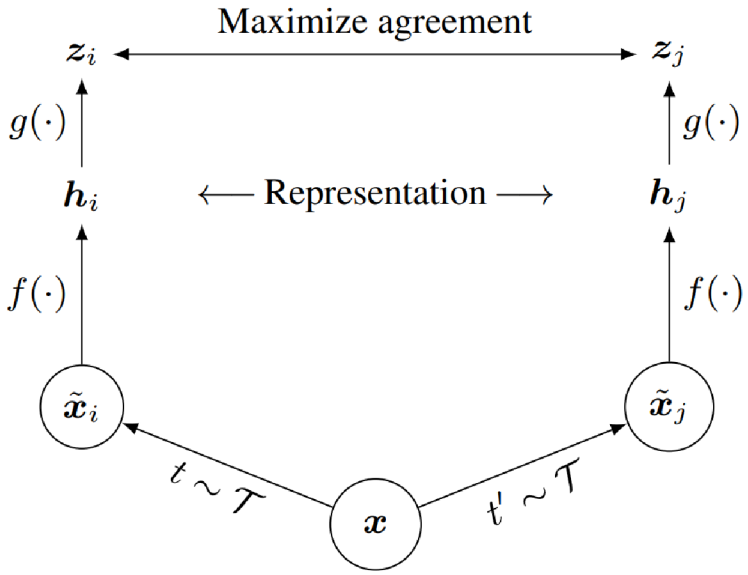


Obr. 2.6: Obrázok znázorňujúci často používané náhradné úlohy. **a)** Obrázok je prevedený z farebného na čiernobiely a trénovaný model má následne predikovať ako vyzeral originálny obrázok na základe čiernobieleho. **b)** Časť originálneho obrázku je prekrytá alebo vyrezaná a model má na jeho základe predikovať ako vyzeral originálny obrázok. **c)** Originálny obrázok je rozdelený na niekoľko menších obrázkov, ktorých pozície sú následne zamenené. Model má na základe zamenených menších obrázkov predikovať ako vyzeral originálny [15]. **d)** Z originálneho obrázku sú spravené 2 výrezy a model má predikovať relatívnu pozíciu týchto 2 výrezov. **e)** Originálny obrázok je otočený o niekoľko stupňov a model má na základe otočeného obrázku predikovať o koľko stupňov bol otočený oproti originálu. Prevzaté z [3].

Backbone je predtrénovaný model ResNetu bez poslednej vrstvy. Po prechode cez backbone vzniknú embeddingy upravených obrázkov h_i a h_j. Výsledné embeddingy ďalej prejdú cez projekciu $g(\cdot)$, ktorá má za úlohu naučiť sa dobré reprezentácie vstupných dát. Výsledkom sú dva embeddingy z_i a z_j. Cieľom algoritmu je maximalizovať zhodu podobných fotografií a minimalizovať zhodu fotografií, ktoré nie sú podobné [9].

2.10.2 BYOL

Bootstrap Your Own Latent (BYOL) [14] je nekontrastívna metóda self-supervised učenia, ktorá eliminuje potrebu generovania negatívnych príkladov pre porovnanie. Výhoda tejto metódy je fakt, že nepotrebuje príliš veľký batch size ako SimCLR a jej presnosť je menej závislá na dátových augmentáciách. Porovnanie závislosti na augmentáciách a batch size metód BYOL a SimCLR je znázornené na obrázku 2.9. BYOL využíva prístup dvoch sietí online a target, kde target je kópia online, ale jej váhy sú aktualizované na základe exponencionálneho kĺzavého priemeru online siete. Vďaka pridaniu prediktora do online siete je architektúra asymetrická. Táto asymetria spoločne s exponeniálnym kĺzavým priemerom zabraňujú kolapsu siete. Kolapsom siete rozumieme stavu, v ktorom sú výsledné reprezentácie dát rovnaké pre všetky vstupy, čo vedie k strate informácií [27].



Obr. 2.7: Architektúra SimCLR. Prevzaté z [9]

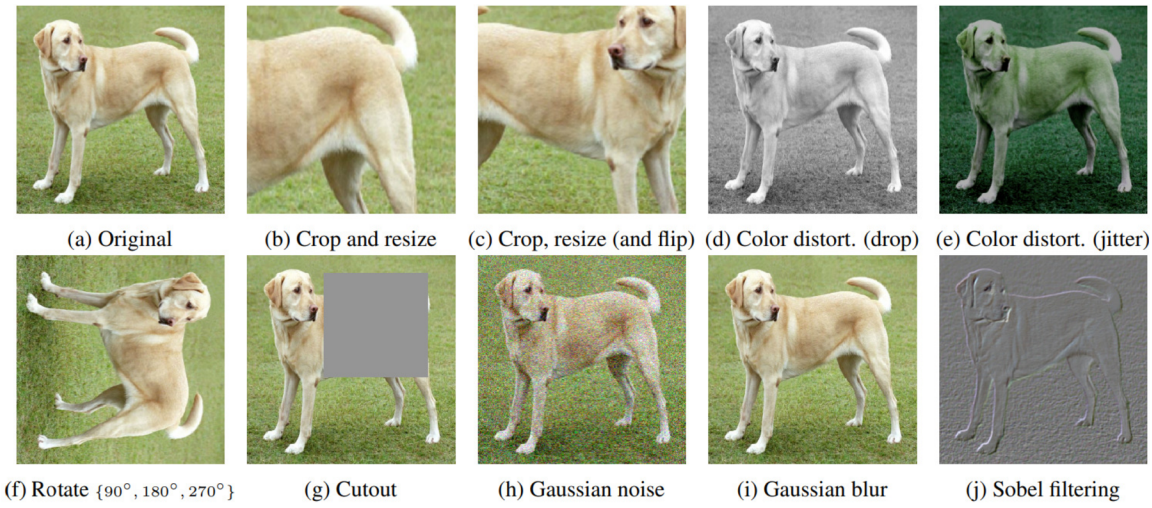
2.10.3 DINO

DINO [8] je metóda self-supervised učenia využívajúca transformátory obrazu (Vision Transformers) a architektúru učiteľ-študent pre naučenie globálnych reprezentácií obrázkov z lokálnych pohľadov. DINO trénuje dve siete učiteľ a študent s rovnakou architektúrou. Počas trénovania sú vstupné obrázky špecificky orezané. Lokálne pohľady sú malé výrezy obsahujúce menej ako 50% celého obrázku a globálne pohľady sú väčšie výrezy obsahujúce viac ako 50% obrázku. Sieť študent používa aj lokálne aj globálne pohľady zatiaľ čo sieť učiteľ používa iba globálne pohľady. Váhy v sieti učiteľ sa aktualizujú s použitím exponenciálneho kĺzavého priemeru váh siete študenta. Cieľom trénovania je aby učiteľ vedel predpovedať vysokoúrovňové charakteristiky a predikcia študenta musí zosúladiť svoje predpovedi s učiteľom pomocou využitia cross-entropy [8, 21]. Zobrazenie máp pozornosti identifikujúce dôležité časti obrazu je na obrázku 2.11.

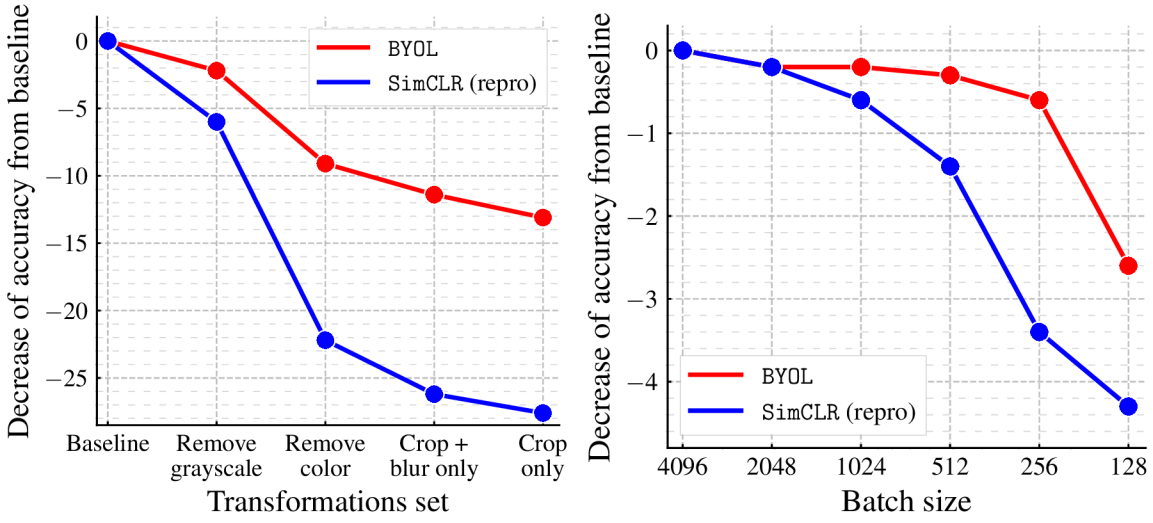
2.11 Wandb

Wandb[1] je skratka pre weights & biases a jedná sa o nástroj určený na monitorovanie, vizualizáciu a porovnávanie experimentov. Okrem logovania klasických parametrov ako loss a úspešnosť modelu dokáže monitorovať aj vyťaženie výpočetnej techniky a váhy jednotlivých vrstiev modelu. Wandb umožňuje aj vizualizáciu konvolučných filtrov a jednoduché porovnávanie metrík natrénovaných modelov. Tento nástroj taktiež umožňuje ľahkú integráciu do existujúcich projektov a umožňuje spoluprácu s ostatnými členmi tímu. Wandb je okrem iného kompatibilný aj s knižnicami PyTorch a TensorFlow.

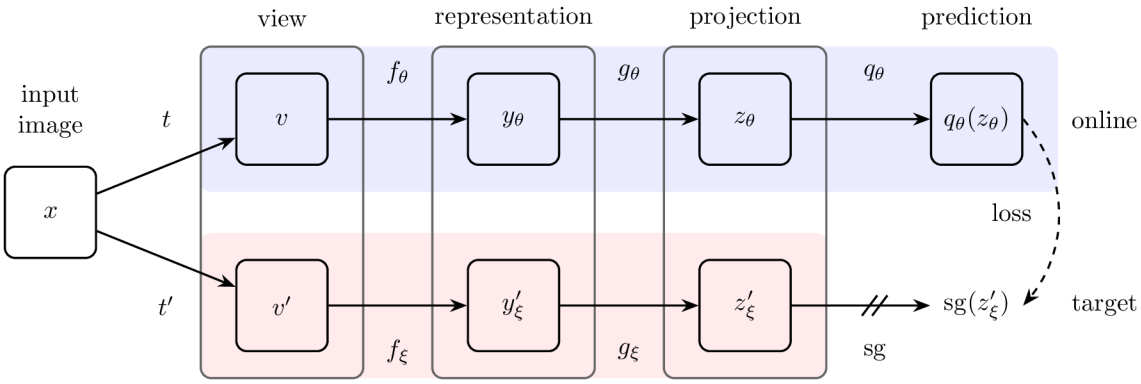
[1] Weights & Biases Documentation. Dostupné online: https://docs.wandb.ai/. [Návšteva 1. apríla 2024].



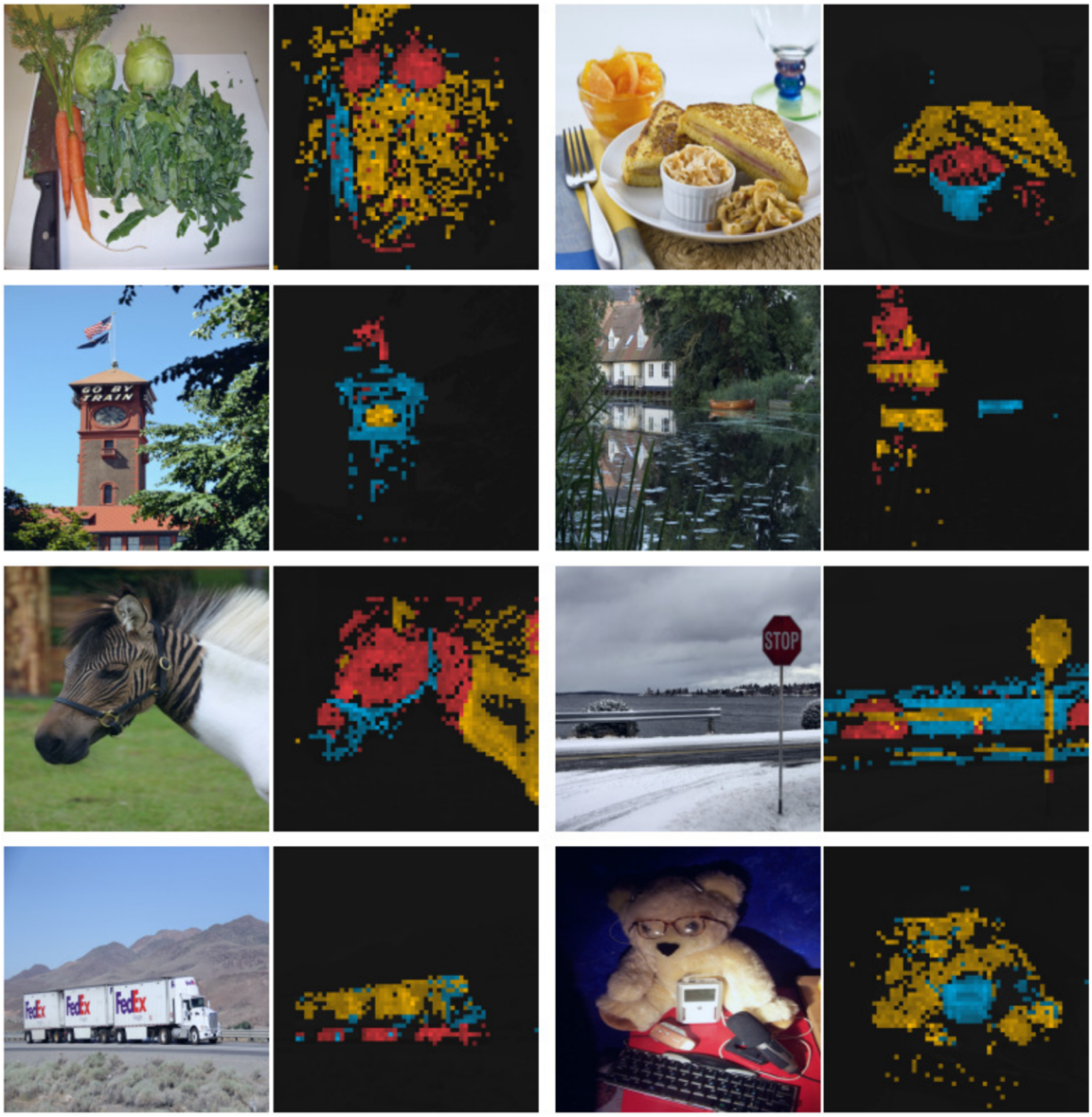
Obr. 2.8: Obrázok znázorňuje rôzne dátové augmentácie používané v SimCLR. Prevzaté z [9].



Obr. 2.9: Obrázok porovnávajúci citlivosť SimCLR a BYOL na batch size a rôzne dátové augmentácie. Prevzaté z [14].



Obr. 2.10: Obrázok znázorňujúci architektúru metódy BYOL. BYOL minimalizuje similarity loss medzi $q_\theta(z_\theta)$ a $sg(z'_\xi)$. Online sieť je voči cieľovej sieti asymetrická z dôvodu pridania prediktoru $q_\theta(z_\theta)$ čím sa snaží zabrániť kolapsu. Prevzaté z [14].



Obr. 2.11: Tento obrázok znázorňuje ako DINO využíva mechanizmus pozornosti na identifikovanie dôležitých častí obrazu. Výstupy rôznych hláv pozornosti sú znázornené rôznymi farbami. Prevzaté z [8].

Kapitola 3

Existujúce riešenia identifikácie človeka podľa rúk

V tejto kapitole je opísaných niekoľko metód identifikácie človeka podľa fotografií rúk. Rôzne metódy môžu byť zamerané na iné aspekty identifikácie napríklad:

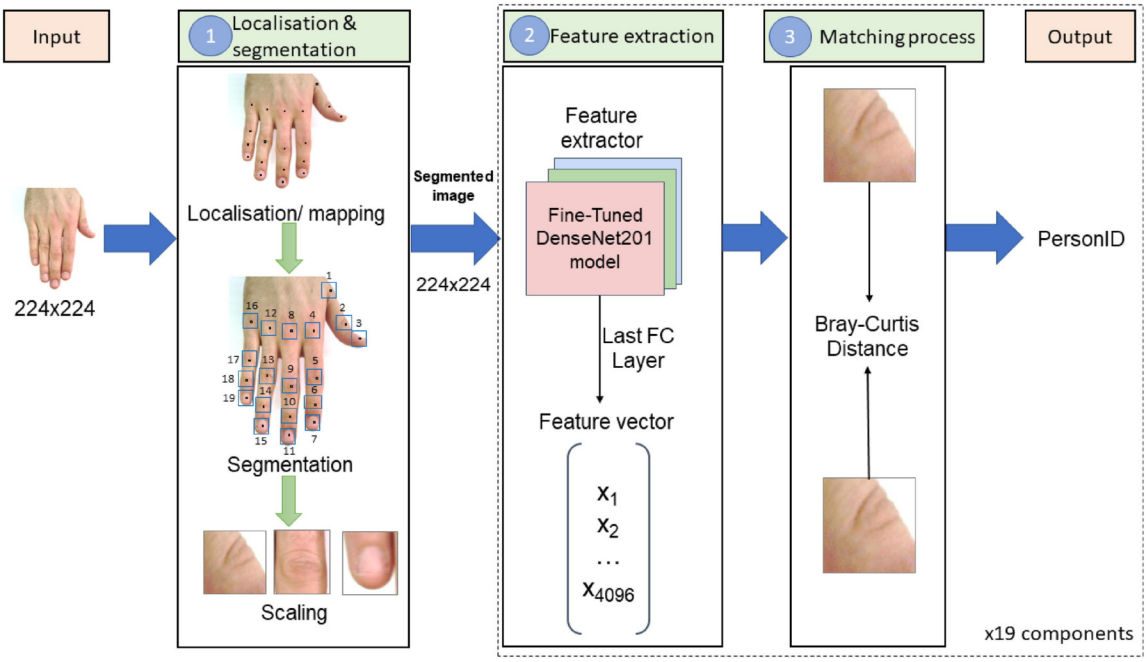
- Identifikácia podľa odtlačkov prstov – odtlačky prstov sú jedinečné pre každého jednotlivca a často sú používanie na identifikovanie osôb. Analýza odtlačkov prstov z fotografie môže poskytnúť dôležité informácie pre identifikáciu človeka.
- Analýza žíl na rukách – vzory žíl na rukách sú tiež jedinečné pre každého jednotlivca a môže slúžiť ako ďalší spôsob identifikácie.
- Analýza dĺžky a tvaru prstov
- Identifikácia podľa jaziev a iných charakteristík – rôzne kombinácie jaziev znamienok, vrások a iných charakteristík ruky môžu taktiež prispieť k identifikácii človeka.

3.1 Identifikácia osôb na základe fotografie ruky podľa nechtov a kĺbov

Person Identification from Fingernails and Knuckles Images using Deep Learning Features and the Bray-Curtis Similarity Measure [4] je práca zameraná na využitie a aplikáciu techník hlbokého učenia pre identifiáciu osôb na základe fotografií rúk s použitím Bray-Curtis podobnosti. Ako identifikačné body sú použité nechtové platničky a kĺby. Je využívaných niekoľko datasetov vrátane 11k_Hands a PolyU, ktoré obsahujú fotografie rúk z rôznych zdrojov. Taktiež sú využívané 3 rôzne predtrénované modely pre extrakciu charakteristík z nechtových platničiek.

3.2 Biometrický systém zameraný na fotografie rúk

Autori práce prezentujú prístup k biometrickej identifikácii pomocou fotografií rúk založený na konvolučných neurónovych sietiach. Navrhnutý systém využíva dataset 570 fotografií rúk, ktoré boli rozdelené na trénovaciu a testovaciu sadu v pomere 80:20. Jednotlivé obrázky transformované do rozmerov $227 \times 227 \times 3$ pre prácu s modelom AlexNet a $224 \times 224 \times 3$ pre prácu s modelmi GoogLeNet a ResNet. Na trénovanie bola využitá metóda Mini-batch. Dosiahnutá testovacia presnosť bola výrazne lepšia ako dovtedy existujúce systémy [24].



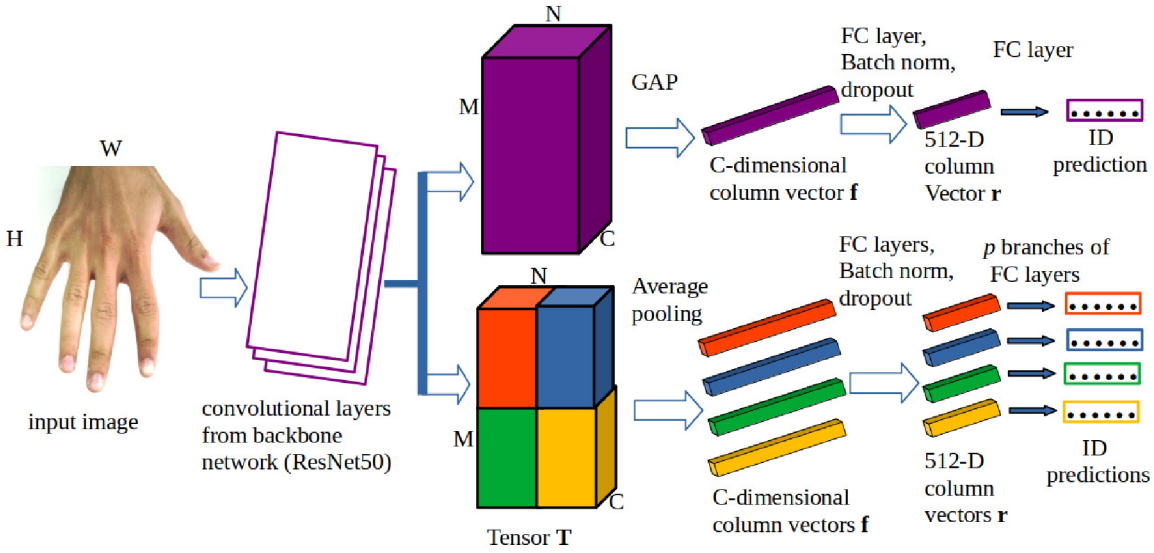
Obr. 3.1: Obrázok popisujúci fungovanie modelu. Na vstupnom obrázku sú najskôr lokalizované dôležité časti (kĺby a nechty). Tieto časti sú potom segmentované a spracované pomocou doladeného extraktora charakteristík. Výsledný vektor charakteristík je následovne porovnaný pomocou Bray-Curtis distance podľa ktorej je možné osobu identifikovať. Prevzaté z [4]

3.3 Identifikácia osôb podľa žíl na chrbte ruky

Dorsal Hand Veins Based Biometric Identification System Using Deep Learning je práca využívajúca techník hlbokého učenia pre biometrickú identifikáciu podľa žíl na chrbte ruky. Podobne ako odtlačok prsta tvar žíl na rukách je jedinečný aj pre identické dvojičky [31]. Pre detekciu žíl na ruke je využité infračervené svetlo. Zozbieraný dataset je upravený s cieľom zlepšiť jeho kvalitu a odstrániť šum. Na upravenom datasete je následne trénovaná konvolučná neurónová sieť, ktorej úlohou je naučiť sa dobré reprezentácie a charakteristiky vstupných obrázkov. Následne je natrénovaná sieť použitá pre klasifikáciu nových obrázkov. Identifikácia podľa žíl na chrbte ruky s využitím konvolučnej neurónovej siete dosahuje úspešnosť vyššiu ako 99% [31].

3.4 Identifikácia osoby na základe fotografie chrbtu ruky

Global and Part-Aware Network (GPA-Net) [6] je model, ktorý slúži na identifikáciu osôb na základe unikátnych charakteristík ich rúk ako napríklad tvar a veľkosť prstov alebo vzory žíl na chrbte ruky. Jedná sa o jedinečné charakteristiky každej osoby, ktoré by mohli byť použité pre účely biometrickej identifikácie s veľkou presnosťou. Model využíva ako backbone sieť ResNet50 predtrénovanú na datasete ImageNet. Originálny ResNet je využívaný iba po global average pooling a potom sa sieť delí na 2 vetvy. Globálna vetva sumarizuje vstupný vektor celý. Lokálna vetva vstupný vektor rozdelí rovnako vertikálne a horizontálne na viacero tensorov, z ktorých sa učí čiastočné reprezentácie. Model bol trénovaný na data-



Obr. 3.2: Obrázok znázorňuje architektúru navrhnutého modelu GPA-Net. Prevzaté z [6].

setoch PolyU a 11k_Hands. GPA-Net je implementovaný pomocou knižnice PyTorch a pri trénovaní využíva Cross-Entropy loss funkciu a mini-batch Stochastic Gradient Descent. Veľkosť vstupných obrázkov je zmenená na 384×384 a taktiež sú použité dátové augmentácie random horizontal flip, normalizácia a color jitter. Presnosť natrénovaného modelu je približne 95%.

3.5 Existujúce datasety fotografií rúk

Medzi existujúce datasety fotografií rúk patria napríklad datasety 11k_Hands [1] a Hong Kong Polytechnic University Hand Dorsal dataset [20]. Tieto datasety obsahujú snímky dlaní aj zadných častí rúk. Existujúce datasety fotografií rúk však neboli vhodné pre potreby trénovania modelu strojového učenia v self-supervised režime s využitím štvoríc obrázkov.

Kapitola 4

Návrh modelu hlbokého strojového učenia pre rozpoznanie rúk

Strojové učenie bola pre mňa zaujímavá, ale nová téma. Začal som preštudovaním základov a nástrojov strojového učenia potrebných k implementácii neurónových sietí. Po preštudovaní základov som premýšľal nad výberom vhodného frameworku. Medzi najpoužívanejšie frameworky patria TensorFlow a PyTorch. Obidva frameworky majú svoje výhody a nevýhody, ale nakoniec som sa rozhodol pre použitie PyTorchu. Našiel som množstov tutoriálov, vďaka ktorým som sa rýchlo naučil princípom strojového učenia a používaniu PyTorchu. Väčšinu základov som sa naučil vďaka stránke Learn PyTorch[1]. Popri študovaní základov strojového učenia som začal so zbieraním datasetu, ktorý som postupne rozširoval posledné dva semestre. Po pochopení základov som vytvoril prvý model určený na rozpoznávanie fotografií rúk, ktorý bol založený na supervised učení. Jednalo sa o jednoduchý klasifikačný model, ktorý používal ako backbone ResNet50 a dokázal s úspešnosťou 80% klasifikovať fotografie rúk 11-tich ľudí.

4.1 Dataset fotografií rúk

Dataset použitý na trénovanie modelu je zložený zo štvoríc fotografií rúk. Fotografie sú fotené vždy pod iným uhlom. Na obrázku 4.1 sa nachádza ilustračná štvorica fotiek rúk, ktorá znázorňuje akým spôsobom bola väčšina štvoríc fotená.

So zberom fotografií mi pomohla rodina, priatelia aj spolužiaci, ktorí mi dovolili odfotiť svoje ruky. Niektorí ľudia mi taktiež nafotili fotografie sami a poslali mi ich. Celkovo sa mi podarilo zozbierať 1696 fotografií rúk, čo predstavuje 424 štvoríc. Na obrázku 4.2 je vidieť koľko fotografií rúk jednotlivých osôb sa nachádza v použitom datasete.

Fotografie boli fotené v rôznych svetelných podmienkach a na rôznych pozadiach. Dataset je ďalej rozdelený na dve sady, trénovaciu a validačnú. Trénovacia sada je zložená z 80% fotografií a model ju využíva pre naučenie charatkeristík vstupných dát. Validačnú sadu tvorí zvyšných 20% fotografií, ktoré model pri trénovaní nevidel a je využitá na vyhodnotenie úspešnosti trénovaného modelu.
Na obrázku 4.3 vidíme ako môže vyzerať časť batchu použitého pri trénovaní modelu. V každom stĺpci sa nachádzajú fotografie ruky patriace inej osobe a v každom riadku je fotografia rovnakej ruky fotená z iného uhlu. Na tento obrázok ďalej odkazuje kapitola 4.4, ktorá obrázok používa na vysvetlenie tvorby trojíc.

[1] https://www.learnpytorch.io/

Obr. 4.1: Štyri fotografie rovnakej ruky z definovaných uhlov. **úplne vľavo:** fotografia z vrchu, **vľavo:** fotografia z ľavej strany, **vpravo:** fotografia z pravej strany, **úplne vpravo:** fotografia z vrchu fotená pod vyšším uhlom.

4.2 Datové augmentácie

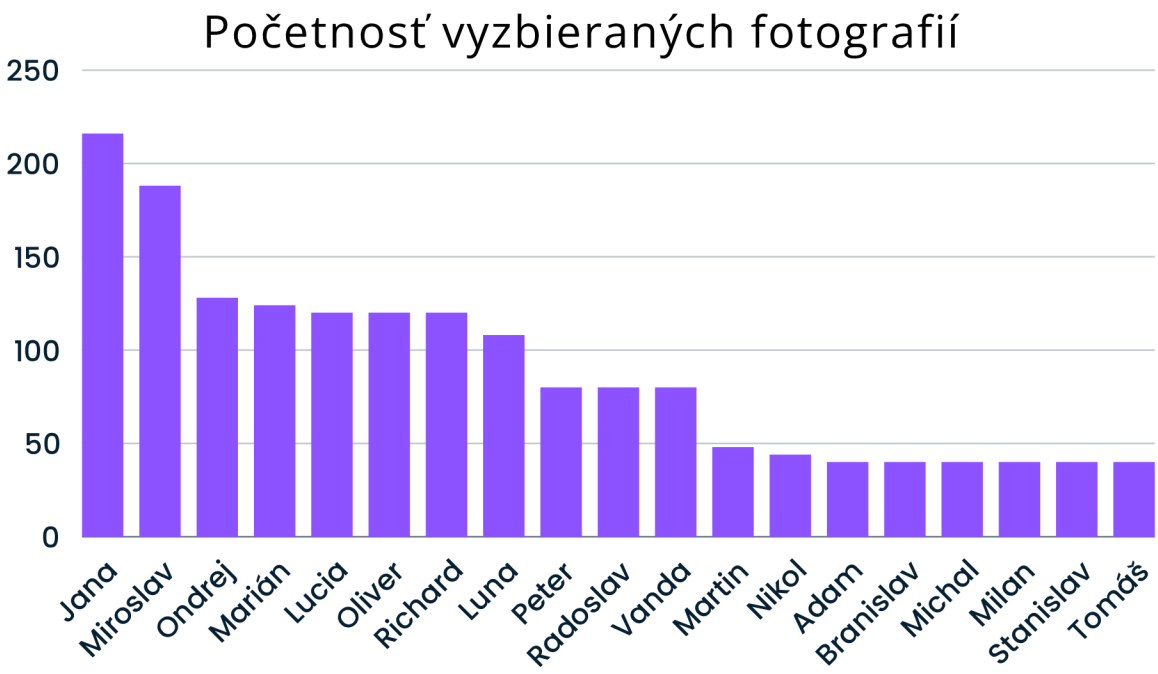
Datové augmentácie bývajú zvyčajne použité za účelom umelého zväčšenia datasetu v prípadoch, že použitý dataset nie je dostatočne veľký alebo pre zlepšenie robustnosti modelu [30]. Pri nedostatočnej veľkosti datasetu zvykne dochádzať k pretrénovaniu, model sa veľmi dobre naučí trénovacie dáta, ale nedokáže generalizovať charakteristiky vstupných dát, ktoré by mal neskôr použiť na predikciu dát, ktoré ešte nevidel. Množsto príkladov často používaných dátových augmentácií je zobrazených na obrázku 2.8. Dátové augmentácie som sa rozhodol použiť z dôvodu, že bez ich použitia už v piatej epoche dochádzalo k pretrénovaniu modelu. Po viacerých testoch som sa rozhodol konkrétne pre použitie TrivialAugmentWide [22], ktorý je dostupný v knižnici Pytorch. Moje testy ukázali, že TrivialAugment bol najefektívnejší proti pretrénovaniu a nespôsobil výrazné zníženie presnosti natrénovaných modelov. Nastavenie agresívnejších augmentácií malo lepší efekt proti pretrénovaniu, ale taktiež zapríčinilo väčšiu variabilitu pri trénovaní. Pre dosiahnutie najlepších výsledkov som každý experiment spustil minimálne 3-krát a vybral najlepší z nich.

4.3 Návrh modelu pre rozpoznanie rúk

Po preštudovaní rôznych techník a metód self-supervised učenia som sa rozhodol pri návrhu inšpirovať metódou SimCLR. SimCLR využíva datové augmentácie na generovanie dvojíc podobných obrázkov. Môj dataset obsahuje štvorice rúk, ktoré využívam namiesto generovania dvojíc. Augmentácie som v mojom riešení použil pre zväčenie datasetu za účelom prevencie pretrénovania. Architektúra modelu, ktorý som navrhol využíva ako backbone predtrénovaný model ResNet50, za ktorým nasleduje projekčná hlava zložená z dvoch lineárnych vrstiev a ReLU aktivačnej funkcie. Pre vypočítanie straty využívam triplet loss funkciu, ktorá je dostupná vo frameworku PyTorch. Architektúra aj s použitím triplet loss funkcie je znázornená na obrázku 4.4.

4.4 Využitie triplet loss pre trénovanie modelu

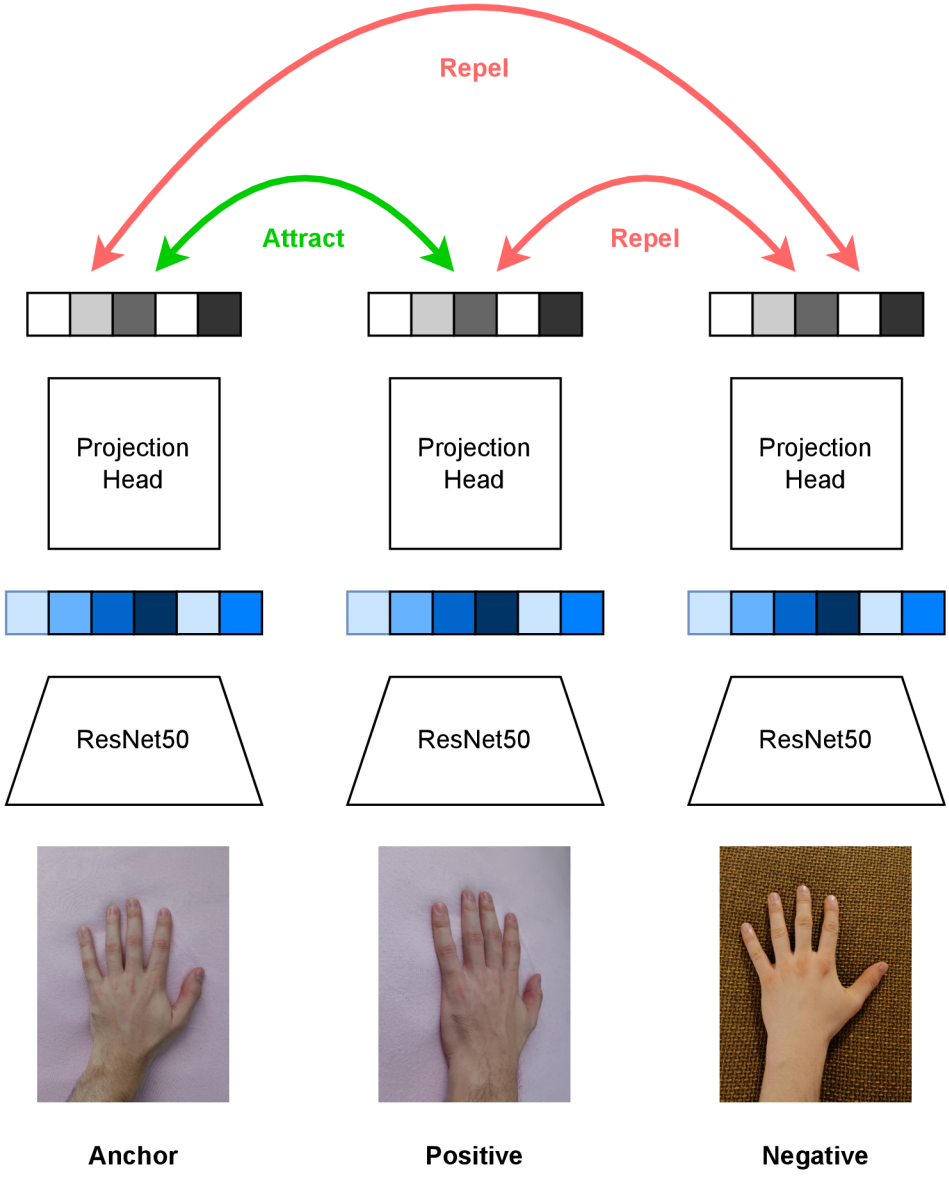
Pre výpočet triplet loss funkcie potrebujeme najskôr vybrať trojice fotografií. Jedna trojica je tvorená z bázovej fotografie, pozitívnej fotografie a negatívnej fotografie. Pre lepšiu



Obr. 4.2: Graf znázorňujúci rozloženie počtu fotografií, ktoré tvoria dataset štvoríc rúk

ilustráciu výberu trojíc použijeme ilustračný minibatch 4.3. Jednotlivé fotografie budem odkazovať v poradí riadok, stĺpec, [1,5] bude fotografia z prvého riadku a piateho stĺpca. Pre vytvorenie prvej trojice zoberieme ako bázovú fotografiu [1,1]. Ako pozitívnu fotografiu vyberieme rovnakú ruku fotenú z iného uhlu [2,1]. Negatívna bude fotografia inej ruky, napríklad [1,2]. Takýmto spôsobom vytvoríme všetky možné trojice, pre ktoré sa následovne spočíta výsledok stratovej funkcie a zistíme či danú trojicu náš model odhadol správne. Ak je vzdialenosť bázovej a pozitívnej dvojice menšia ako vzdialenosť bázovej a negatívnej dvojice, model ruky porovnal správne. Pre výpočet metrík straty a presnosti spočítame zvlášť sumu pre stratu a sumu pre presnosť. Výslednú sumu delím celkovým počtom trojíc.

Obr. 4.3: Na obrázku je znázornené, akým spôsobom môže vyzerať časť batch pri trénovaní modelu. V datasete sú fotografie ľavej aj pravej ruky fotené pri rôznych svetelných podmienkach, na rôznych pozadiach a z iných uhlov.



Obr. 4.4: Obrázok znázorňuje architektúru navrhnutého modelu a použitie tripletloss stratovej funkcie. Anchor a Positive sú fotografie rovnakej ruky fotené z iných uhlov. Fotografia Negative je fotografia z inej štvorice fotografií ruk. Anchor a positive znázorňujú rovnakú ruku, takže by sa mali na seba podobať viac ako dvojice anchor negative alebo positive negative.

Kapitola 5

Implementácia a vyhodnotenie natrénovaného modelu

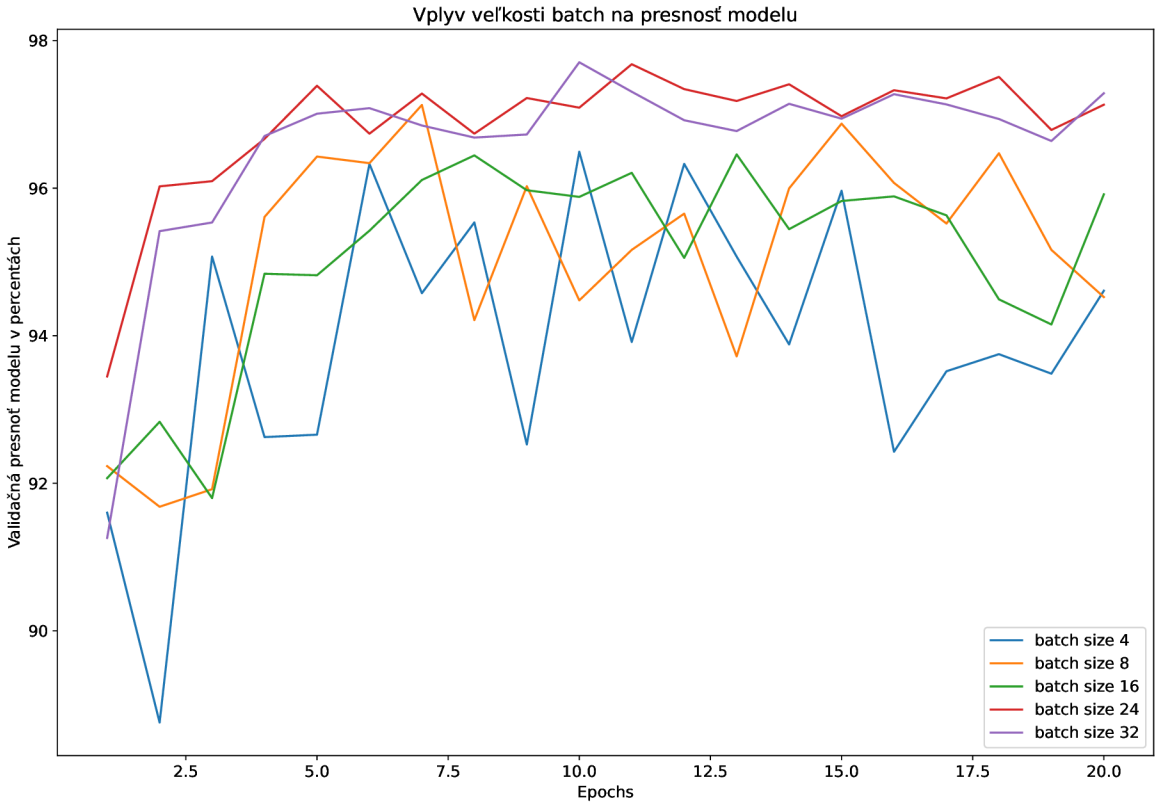
Výsledné modely sú implementované v jazyku Python pomocou frameworku pre trénovanie modelov hlbokého strojového učenia, PyTorch. Rozhodol som sa použiť PyTorch predovšetkým pre jeho flexibilitu a jednoduchosť použitia. Ďalším dôležitým aspektom, ktorý ovplyvnil moje rozhodnutie, je široká akceptácia a popularita PyTorchu v odbornej a priemyselnej komunite zameranej na strojové učenie. Na trénovanie vytvorených modelov som používal grafickú kartu Nvidia GeForce RTX 4060Ti s 16GB pamäte. Na monitorovanie priebehu, logovanie, vizualizáciu a porovnanie výsledkov rôznych experimentov som používal nástroj Wandb. Grafy použité v tejto kapitole som vykreslil pomocou nástroja Pyplot, ktorý je súčasťou knižnice Matplotlib.

5.1 Nahrávanie datasetu

Načítanie datasetu sa v PyTorchi robí s použitím triedy ImageDataset, ktorá ale vie načítať dáta iba po jednom. Z tohoto dôvodu som sa rozhodol implementovať vlastnú triedu CustomImageDataset použitú na načítanie trénovacej sady a CustomImageDatasetValidation na načítanie validačnej sady. Implementované triedy majú podobnú funkcionalitu ako trieda ImageDataset, ale narozdiel od nej načítajú celú štvoricu fotografií. Tieto triedy sú rovnako kompatibilné aj s triedou DataLoader dostupnou v PyTorchi, ktorá slúži na načítanie dát počas trénovania a validácie modelu.

5.2 Porovnanie parametrov

Táto podkapitola je venovaná porovnaniu vplyvu rôznych hodnôt hyperparametrov na validačnú presnosť trénovaného modelu. Ako som spomínal už v podkapitole 4.2, ktorá sa zaoberá použitím dátových augmentácií, nazbieraný dataset nie je dostatočne veľký na to, aby pri trénovaní modelu nedošlo k jeho pretrénovaniu. Dátové augmentácie použité v mojich experimentoch sú dosť agresívne, preto som každý experiment spúšťal minimálne 3-krát, aby som zaistil lepšie výsledky.



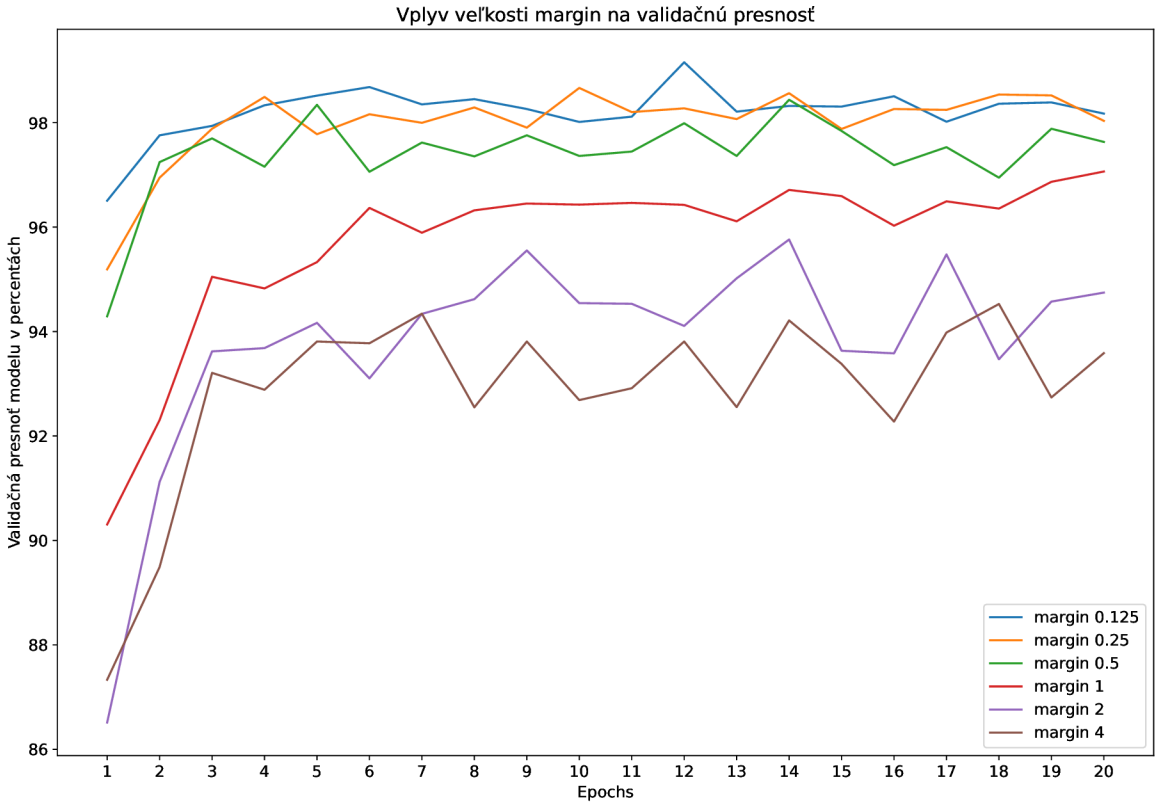
Obr. 5.1: Na grafe je porovnanie vplyvu veľkosti batch na validačnú presnosť modelu.

5.2.1 Porovnanie veľkostí batch

Pri experimentovaní z veľkosťou batch boli pre každý beh experimentu použité rovnaké hyperparametre na trénovanie modelu. Použité parametre:

- Počet epoch – 20
- Learning rate – 0.001
- Veľkosť kroku pre plánovač rýchlosti učenia – 3
- Parameter vynásobenia rýchlosti učenia – 0.25
- Triplet loss margin – 1
- Veľkosť vstupného obrázku – 224×224
- Veľkosť výstupu projekcie – 128

Na grafe 5.1 vidíme, že podobne ako pri metóde SimCLR je výhodnejšie používať väčší batch. Pri použití nižších hodnôt veľkosti batch vidíme veľkú stratu validačnej presnosti výsledného modelu. Zväčšenie použitej veľkosti batch pri trénovaní modelu pomáha stabilizovať proces trénovania. Spracovávanie väčšieho množstva dát naraz počas trénovania modelu pomáha minimalizovať pravdepodobnosť výberu horších vzoriek. Avšak spracovanie veľkého objemu dát naraz spôsobuje vyššie využitie zdrojov. Z tohoto dôvodu som nebol schopný otestovať väčšie hodnoty batch.



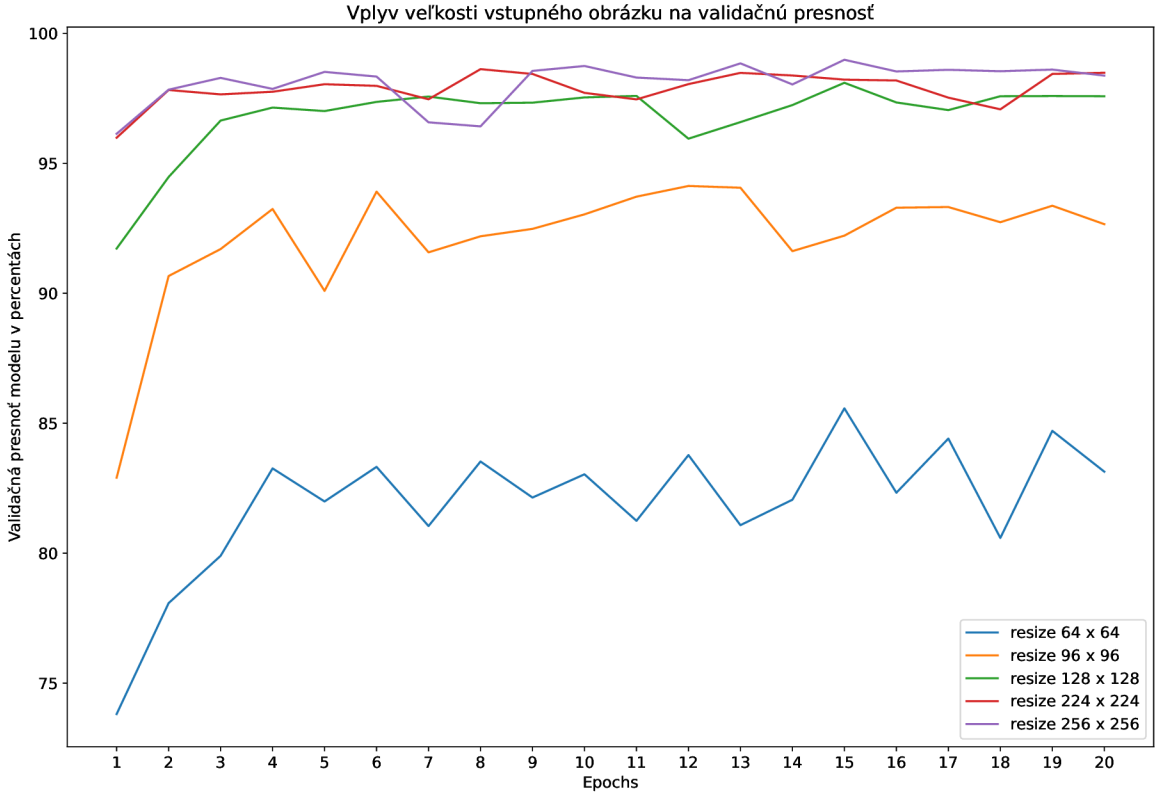
Obr. 5.2: Graf znázorňuje vplyv triplet loss margin na validačnú presnosť modelu

5.2.2 Porovnanie triplet loss margin

Parametre použité pre experimentovanie s triplet loss margin:

- Počet epoch – 20
- Veľkosť batch – 32
- Learning rate – 0.001
- Veľkosť kroku pre plánovač rýchlosti učenia – 2
- Parameter vynásobenia rýchlosti učenia – 0.25
- Veľkosť vstupného obrázku – 224×224
- Veľkosť výstupu projekcie – 128

Margin je veľmi dôležitý parameter pre triplet loss funkciu. V základe býva väčšinou nastavený na hodnotu 1. Zníženie margin môže viesť k zložitejšiemu trénovaniu a spomaliť proces trénovania. Pri nižšom margin je model citlivejší na malé rozdiely medzi pozitívnymi a negatívnymi pármi. Ako vidíme na grafe 5.2 nižšia hodnota margin zlepší presnosť modelu. Pri nižších hodnotách margin je však vyššie riziko pretrénovania a preto treba túto hodnotu dobre nastaviť pre optimálny výkon modelu. Vyššie hodnoty margin umožňujú lepšiu separáciu medzi podobnými a nepodobnými dátami. Model je v takomto prípade menej citlivý na malé rozdiely medzi dátami. Po vykonaní viacerých experimentov som usúdil, že pre



Obr. 5.3: Obrázok znázorňuje vplyv veľkosti vstupného obrázku na validačnú presnosť modelu

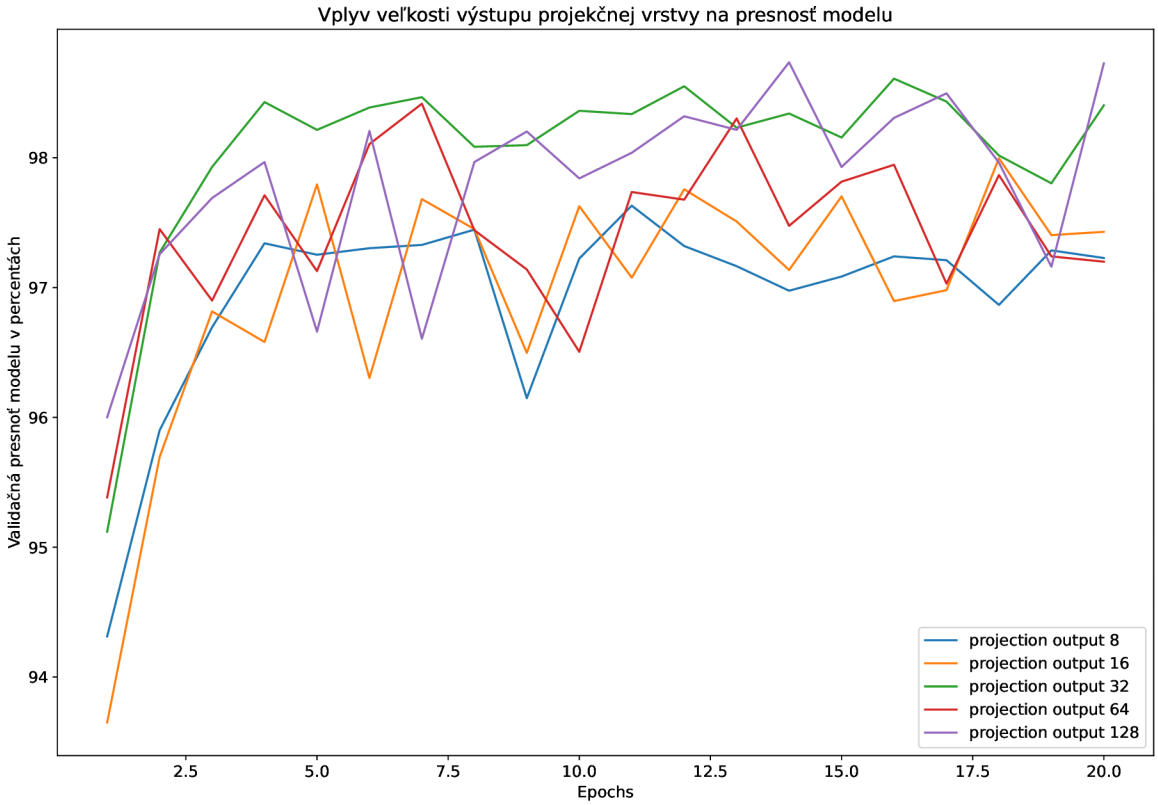
potreby trénovania modelu na rozpoznávanie fotografií rúk je výhodnejšie použitie menšej hodnoty margin.

5.2.3 Vplyv veľkosti vstupného obrázku na validačnú presnosť

Parametre použité pre experimentovanie s veľkosťou vstupného obrázku:

- Počet epoch – 20
- Veľkosť batch – 32
- Learning rate – 0.001
- Veľkosť kroku pre plánovač rýchlosti učenia – 2
- Parameter vynásobenia rýchlosti učenia – 0.25
- Triplet loss margin – 0.25
- Veľkosť výstupu projekcie – 128

Veľkosť vstupného obrázku je jedným z dôležitých hyperparametrov pre trénovanie modelov hlbokého učenia. Na obrázku 5.3 vidíme, žr pri nižších rozmeroch ako napríklad 64×64 model nemá dostatok parametrov podľa ktorých by vedel od seba jednotlivé ruky odlíšiť a z toho dôvodu klesá výsledná presnosť. Rozdiely medzi vyššími rozmermi vstupného



Obr. 5.4: Obrázok znázorňuje vplyv veľkosti výstupu projekcie na validačnú presnosť modelu

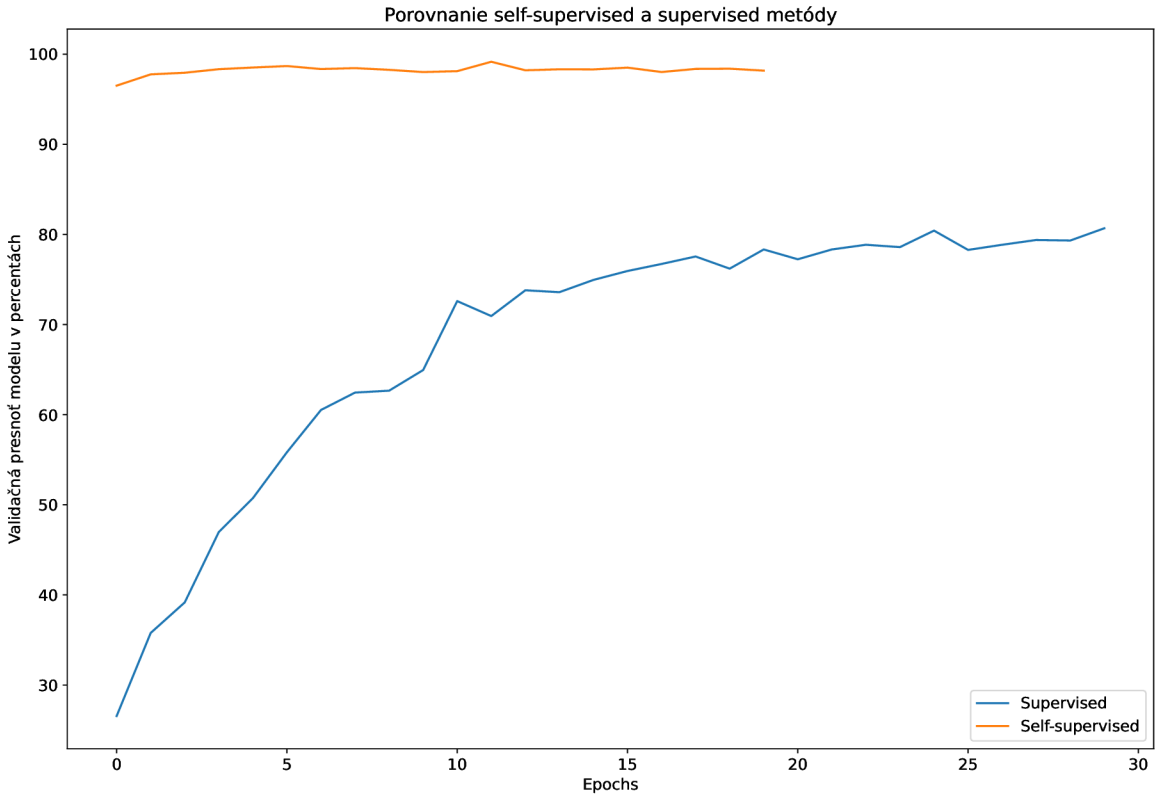
obrázku ako sú 224×224 a 256×256 sú minimálne. Používanie väčších vstupných obrázkov je výpočetne náročné a zlepšenie presnosti výsledného modelu nie je výrazne lepšie. Z tohoto dôvodu je bežne používaná veľkosť 224×224 postačujúca pre veľké množstvo úloh riešených pomocou neurónových sietí.

5.2.4 Vplyv veľkosti výstupu projekcie na validačnú presnosť

Parametre použité pre experimentovanie s veľkosťou výstupu projekcie:

- Počet epoch – 20
- Veľkosť batch – 32
- Learning rate – 0.001
- Veľkosť kroku pre plánovač rýchlosti učenia – 2
- Parameter vynásobenia rýchlosti učenia – 0.25
- Triplet loss margin – 0.25
- Veľkosť vstupného obrázku – 224×224

Príliš nízke veľkosti výstupu projekčnej vrstvy môžu viesť k strate informácií a zníženiu presnosti. Použitie vyšších hodnôt výstupu projekčnej vrstvy umožňuje modelu naučiť sa

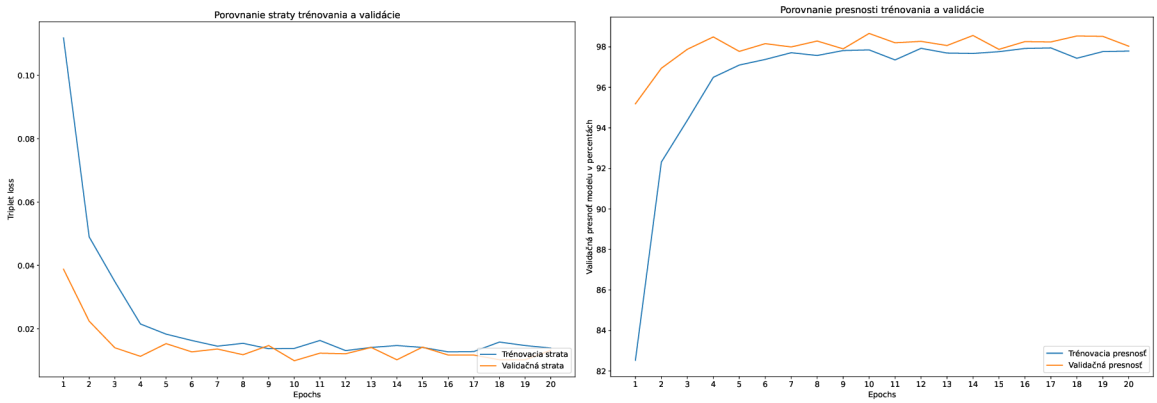


Obr. 5.5: Graf zobrazujúci porovnanie supervised a self-supervised metódy.

viac charakteristík vstupných dát, čo nie vždy musí mať pozitívny vplyv na validačnú presnosť. Ako vidíme na grafe 5.4 nastavenie projekčnej vrstvy na hodnotu 32 vyzerá byť najefektívnejšie, avšak z výsledného grafu je ťažké povedať či vyššie hodnoty v tomto prípade škodia. Hodnoty vyzerajú byť výrazne rozkmitané z dôvodu použitia agresívnych dátových augmentácií. Predpokladám, že v prípade použitia výrazne väčšieho datasetu by sa dalo predísť pretrénovaniu a lepšie porovnať vplyv veľkosti výstupu projekčnej vrstvy na výslednú presnosť.

5.3 Porovnanie supervised a self-supervised metódy

V tejto podkapitole porovnávam výsledky metód supervised a self-supervised prístupov k trénovaní neurónových sietí. Ako vidíme na grafe 5.5 výsledná validačná presnosť modelu trénovaného s použitím self-supervised metódy, približne 98% je výrazne lepšia ako testovacia presnosť modelu trénovaného s využitím supervised metódy cca 80%. Supervised model som trénoval na klasifikácii fotografií rúk 19-tich ľudí. Ako backbone modelu som použil predtrénovaný ResNet50, z ktorého som odstránil poslednú vrstvu a nahradil ju plne prepojenou vrstvou. Model trénovaný pomocou self-supervised metódy sa už v prvej epoche po trénovaciom kroku naučí veľmi dobré reprezentácie vstupných dát. Predpokladám, že toto je spôsobené veľkou podobnosťou, ktorú majú jednotlivé fotografie rúk rovnakej štvorice. Trénovanie modelu pomocou supervised metódy má ale výrazne nižšie nároky na výpočetné zdroje a trénovanie jednej epochy je oproti použitému self-supervised prístupu 3-krát rýchlejšie.



Obr. 5.6: Grafy znázorňujúce validačnú a trénovaciu stratu a presnosť

5.4 Najúspešnejší model

V tejto podkapitole je predstavený najúspešnejšie natrénovaný model. Výsledkom tejto práce je model natrénovaný pomocou kontrastívneho self-supervised učenia. Validačná presnosť natrénovaného modelu je približne 98%. Model bol natrénovaný s použitím nasledovných hyperparametrov:

- Počet epoch – 20
- Veľkosť batch – 32
- Learning rate – 0.001
- Veľkosť kroku pre plánovač rýchlosti učenia – 2
- Parameter vynásobenia rýchlosti učenia – 0.25
- Triplet loss margin – 0.25
- Veľkosť vstupného obrázku – 224×224
- Veľkosť výstupu projekcie – 128

5.5 Zhodnotenie výsledkov a možné nadviazanie na prácu

S využitím moderných techník self-supervised učenia sa mi podarilo natrénovať model, ktorý dosahuje až 98% úspešnosť rozpoznania rúk na validačnej sade. 98 ale samozrejme nie je 100 a je tu ešte priestor na zlepšenie. Ako možné pokračovanie práce by som navrhoval rozšírenie datasetu a preskúmanie vplyvu rôznych projekčných hláv, napríklad aj nelineárnych na presnosť modelu. Zaujímavou témou by mohol byť aj rozbor vplyvu rôznych dočasných alebo trvalých faktorov na presnosť rozpoznania. Mohlo by sa jednať napríklad o rôzne malé rany alebo odreniny, prítomnosť náplastí, tetovaní, prsteňov alebo hodiniek na ruke.

Doladený model by mohol byť využitý v rámci bezpečnosti napríklad pre dvojkrokovú biometrickú autentizáciu, ktorá by brala v úvahu chrbtovú časť ruky aj dlaň alebo prípadne odtlačok prsta a chrbtovú časť ruky. Ďalším z možných využití by mohol byť systém, ktorý by zbieral vo väčších firmách štatistiky, ktorý zamestnanec ako často prechádza cez konkrétne dvere, kde by boli nainštalované kamery smerujúce na kľučku. Taktiež by sa takýto model mohol dať využiť aj v systéme inteligentnej domácnosti.

Kapitola 6

Záver

Cieľom tejto práce bolo navrhnúť a implementovať model hlbokého strojového učenia na identifikáciu človeka pomocou fotografií rúk.

V rámci mojej bakalárskej práce som si preštudoval problematiku počítačového videnia a hlbokého strojového učenia. Oboznámenie s rôznymi technikami a metódami trénovania modelov mi umožnilo porozumieť ich fungovaniu a možnostiam aplikácie v praxi.

Počas posledných dvoch semestrov som s pomocou rodiny a priateľov aktívne zbieral dátovú sadu štvoríc fotografií rúk fotených z iných uhlov. Celkovo sa mi podarilo vyzbierať 1696 fotografií.

Na vyzbieranej dátovej sade som sa učil technikám a prístupom k trénovaniu modelov hlbokého strojového učenia. Začal som trénovaním modelu pre klasifikáciu fotografií rúk pomocou supervised učenia. S využitím techniky transfer learning a použitím siete ResNet50 sa mi podarilo natrénovať model s 80% úspešnosťou na validačnej sade. Naučil som sa o základných prístupoch optimalizácie, ale aj o pokročilých technikách ako je a self-supervised learning a metódach na ňom založených. Pokračoval som návrhom modelu založeného na kontrastívnom self-supervised učení, inšpirovaným architektúrou SimCLR. Výsledný model bol trénovaný na nazbieranom datasete s použitím triplet loss stratovej funkcie. Trénovanie modelu som postupne optimalizoval a porovnával vplyv hodnôt jednotlivých hyperparametrov na výslednú validačnú presnosť. Podarilo sa mi natrénovať model s výrazne lepšou validačnou presnosťou (98%) ako bola presnosť, ktorú dosahoval model trénovaný supervised metódou.

Práca by ďalej mohla pokračovať rozšírením datasetu rúk. Použitie väčšieho datasetu by mohlo zlepšiť presnosť aj robustnosť trénovanej siete. Odporúčal by som ďalej vyskúšať experimenty s rôznymi projekčnými hlavami. Výsledná práca by mohla mať rôzne praktické využitia, napríklad pri dvojkrokovej biometrickej autentizácii v kombinácii s modelom, ktorý by identifikoval dlaň alebo čítačkou odtlačkov prsta.

Literatúra

[1] Afifi, M. 11K Hands: gender recognition and biometric identification using a large dataset of hand images. *Multimedia Tools and Applications*, 2019. Dostupné z: https://doi.org/10.1007/s11042-019-7424-8.

[2] Aggarwal, C. C. *Neural Networks and Deep Learning: A Textbook*. 1. vyd. Springer Cham, 2018. XXIII, 497 s. ISBN 978-3-319-94463-0 (eBook), 978-3-030-06856-1 (Softcover). Dostupné z: https://doi.org/10.1007/978-3-319-94463-0. Published: 25 August 2018 (eBook), 31 January 2019 (Softcover).

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

[4] Alghamdi, M.; Angelov, P. a Alvaro, L. P. Person identification from fingernails and knuckles images using deep learning features and the Bray-Curtis similarity measure. *Neurocomputing*, 2022, zv. 513, s. 83–93. ISSN 0925-2312. Dostupné z: https://www.sciencedirect.com/science/article/pii/S0925231222012176.

[5] Antoniadis, P. Residual Networks, 2024. Dostupné z: https://www.baeldung.com/cs/residual-networks.

[6] Baisa, N. L.; Williams, B.; Rahmani, H.; Angelov, P. a Black, S. *Hand-Based Person Identification using Global and Part-Aware Deep Feature Representation Learning*. 2022.

[7] Bishop, C. M. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 978-0-387-31073-2.

[8] Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J. et al. *Emerging Properties in Self-Supervised Vision Transformers*. 2021.

[9] Chen, T.; Kornblith, S.; Norouzi, M. a Hinton, G. *A Simple Framework for Contrastive Learning of Visual Representations*. 2020.

[10] Chollet, F. *Deep Learning with Python*. Manning, november 2017. ISBN 9781617294433.

[11] Ericsson, L.; Gouk, H.; Loy, C. C. a Hospedales, T. M. Self-Supervised Representation Learning: Introduction, advances, and challenges. *IEEE Signal Processing Magazine*. Institute of Electrical and Electronics Engineers (IEEE), máj 2022, zv. 39, č. 3, s. 42–62. ISSN 1558-0792. Dostupné z: http://dx.doi.org/10.1109/MSP.2021.3134634.

[12] Fuentes, E. Introduction to Artificial Intelligence and Machine Learning, 2023. Dostupné z: https://community.aws/content/2drbbXokwrIXivItJ8ZeCk3gT5F/introduction-to-artificial-intelligence-and-machine-learning.

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

[14] Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P. H. et al. *Bootstrap your own latent: A new approach to self-supervised Learning*. 2020.

[15] Gui, J.; Chen, T.; Zhang, J.; Cao, Q.; Sun, Z. et al. *A Survey on Self-supervised Learning: Algorithms, Applications, and Future Trends*. 2023.

[16] He, K.; Zhang, X.; Ren, S. a Sun, J. *Deep Residual Learning for Image Recognition*. 2015.

[17] Hoffer, E. a Ailon, N. *Deep metric learning using Triplet network*. 2018.

[18] James, G.; Witten, D.; Hastie, T. a Tibshirani, R. *An Introduction to Statistical Learning with Applications in Python*. 2nd. New York, NY: Springer, 2021. ISBN 978-3-031-38747-0. Dostupné z: https://doi.org/10.1007/978-3-031-38747-0.

[19] Kingma, D. P. a Ba, J. *Adam: A Method for Stochastic Optimization*. 2017.

[20] Kumar, A. a Xu, Z. Personal Identification Using Minor Knuckle Patterns From Palm Dorsal Surface. *IEEE Transactions on Information Forensics and Security*, 2016, zv. 11, č. 10, s. 2338–2348.

[21] Mehra, A. *Unlocking the Power of Self-Supervised Learning in Computer Vision with DINO* Blog post. 2023. Dostupné z: https://www.labellerr.com/blog/unlocking-the-power-of-self-supervised-learning-in-computer-vision-with-dino/.

[22] Müller, S. G. a Hutter, F. *TrivialAugment: Tuning-free Yet State-of-the-Art Data Augmentation*. 2021.

[23] Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J. et al. *PyTorch: An Imperative Style, High-Performance Deep Learning Library*. 2019.

[24] Prihodova, K. a Hub, M. Hand-Based Biometric System Using Convolutional Neural Networks. *Acta Informatica Pragensia*, 2020, zv. 9, č. 1, s. 48–57. Dostupné z: https://aip.vse.cz/artkey/aip-202001-0004.php.

[25] Raschka, S.; Liu, Y.; Mirjalili, V. a Dzhulgakov, D. *Machine Learning with PyTorch and Scikit-Learn: Develop Machine Learning and Deep Learning Models with Python*. Packt Publishing, 2022. Expert insight. ISBN 9781801819312. Dostupné z: https://books.google.sk/books?id=UHbNzgEACAAJ.

[26] Reddy, S. The intuition of Triplet Loss. *Medium*, 2019. Dostupné z: https://medium.com/analytics-vidhya/triplet-loss-b9da35be21b8.

[27] Richemond, P. H.; Grill, J.-B.; Altché, F.; Tallec, C.; Strub, F. et al. *BYOL works even without batch statistics*. 2020.

[28] SHARMA, S. Activation Functions in Neural Networks. *Medium*, 2017. Dostupné z: https://towardsdatascience.com/activation-functions-neural-networks-1cbd9f8d91d6.

[29] Stevens, E.; Antiga, L. a Viehmann, T. *Deep Learning with PyTorch*. 2020. ISBN 9781617295263.

[30] Yang, S.; Xiao, W.; Zhang, M.; Guo, S.; Zhao, J. et al. *Image Data Augmentation for Deep Learning: A Survey*. 2023.

[31] Zehtab Nayebi, M. a Turgut, Z. Dorsal Hand Veins Based Biometric Identification System Using Deep Learning. *Erzincan University Journal of Science and Technology*. Erzincan Binali Yildirim University, 2021, zv. 14, č. 1, s. 1–15.